YAYASAN PRIMA AGUS TEKNIK

PHP FRAMEWORK

Iman Saufik Suasana, M.Kom

PHP FRAMEWORK

Oleh :
Iman Saufik Suasana, M.Kom

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

PHP FRAMEWORK

Penulis :
Iman Saufik Suasana, M.Kom

ISBN : 978-623-8120-48-2 (PDF)

Editor :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Penyunting :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas terselesaikannya buku "***PHP Framework***" dengan baik. Saat ini, kerangka kerja memudahkan untuk membangun situs web. Kerangka kerja ini hadir dengan banyak fitur *out-of-the-box* yang dapat sangat mempercepat pengembangan situs web Anda tanpa harus memulai dari awal. Kerangka kerja yang banyak digunakan adalah CodeIgniter.

Buku ini menjelaskan beberapa fitur utama CI. Itu tidak mencakup semuanya, atau mencakup salah satu dari mereka dengan detail penuh. CI hadir dengan Panduan Pengguna on-line yang sangat baik yang menjelaskan banyak hal. Ini diunduh dengan file CI. Buku ini tidak mencoba menduplikasi Panduan Pengguna. Alih-alih, ini mencoba mempermudah Anda untuk memahami cara kerja kerangka kerja CI, sehingga Anda dapat memutuskan apakah itu tepat untuk Anda, dan mulai menggunakannya dengan cepat. Di beberapa tempat, buku ini melampaui Panduan Pengguna, ketika mencoba menjelaskan cara kerja CI. (Panduan Pengguna lebih berorientasi praktis.) Ini berarti bahwa ada beberapa bab yang cukup teoretis di antara halaman "ini caranya". Penulis telah menemukan bahwa itu membantu untuk memahami apa yang dilakukan CI di bawah tenda; jika tidak, Anda terkadang mendapatkan pesan kesalahan membingungkan yang tidak mudah diselesaikan.

CodeIgniter adalah framework PHP open source gratis, ringan. Framework PHP ini sangat efektif untuk mengembangkan website dan aplikasi sederhana. CodeIgniter sebagai framework memiliki keunggulan memiliki library dan package yang cukup lengkap. Ini berarti Anda dapat duduk dan bersantai sambil mendesain situs web Anda tanpa harus memprogram ulang semuanya. Cukup gunakan perpustakaan yang disediakan. Dan karena ini open source, Anda dapat mengembangkan kerangka kerja ini sesuai keinginan Anda. Ini juga mengapa semakin banyak pengembang menggunakan CodeIgniter. Codeigniter menganut pola desain atau arsitektur Model-View-Controller (MVC). Pisahkan bagian kode yang menangani proses bisnis dari bagian kode yang menangani tujuan presentasi (tampilan). Pola desain ini memungkinkan pengembang web untuk berkolaborasi (teamwork) pada aplikasi berbasis web. Ini memungkinkan pengembang web untuk fokus pada setiap bagian tanpa mengganggu bagian lain. Aplikasi dapat diselesaikan lebih cepat.

Contoh-contoh dalam buku ini tidak menunjukkannya secara rinci, tentu saja: tetapi mereka, penulis harap, menunjukkan bahwa Anda dapat menggunakan CI untuk membuat pengkodean umum menjadi lebih sederhana, dan beberapa hal yang tidak biasa juga. Buku ini memandu Anda melalui fitur-fitur utama CodeIgniter secara sistematis, menjelaskannya dengan jelas dengan contoh kode ilustratif. Semoga buku ini bisa memberikan manfaat bagi pembaca, akhir kata penulis ucapkan terima kasih.

Penulis, Juli 2023

Iman Saufik, M.Kom.

DAFTAR ISI

BAB 1
PENGENALAN CODEIGNITER

Sebagian besar dari kita hanya ingin menulis aplikasi yang berfungsi dengan baik, dan melakukannya sesederhana dan semudah mungkin. Buku ini tentang CodeIgniter, alat untuk membuat PHP lebih mudah digunakan. Jika Anda perlu menghasilkan hasil, jika Anda berpikir bahwa detail dan seluk-beluk pengkodean adalah untuk geeks, maka Anda harus melihat CodeIgniter (CI ke teman-temannya).

CI gratis, ringan, dan mudah dipasang, dan itu benar-benar membuat hidup Anda lebih berarti lebih mudah. Baca saja bab ini untuk mengetahui caranya:

- Apa yang dapat dilakukan CI untuk Anda
- Apa yang dimaksud dengan 'kerangka kerja' dan bagaimana CI cocok?
- Model bisnis sumber terbuka
- Beberapa kelemahan CI (tidak, tidak sempurna)

1.1 APA YANG DAPAT CODEIGNITER LAKUKAN?

Jika Anda sudah menulis kode dalam PHP, CodeIgniter akan membantu Anda melakukannya dengan lebih baik, dan lebih mudah. Ini akan mengurangi jumlah kode yang sebenarnya Anda ketik. Skrip Anda akan lebih mudah dibaca dan diperbarui. Ini akan membantu Anda memberi situs web besar struktur yang koheren. Ini akan mendisiplinkan pengkodean Anda dan membuatnya lebih kuat, dalam beberapa kasus tanpa Anda sadari. Itu klaim yang cukup besar. Anda telah menghabiskan beberapa waktu untuk mempelajari PHP, HTML, CSS, database, dan beberapa akronim lainnya dari bahasa geek. Anda memerlukan pengetahuan dasar, tetapi tidak harus ahli, tentang PHP untuk mendapatkan manfaat dari CI. CodeIgniter bukan untuk Anda jika:

- Anda tidak memiliki pengetahuan yang memadai tentang PHP dan HTML.
- Anda ingin menulis Sistem Manajemen Konten (CMS) dasar dengan cepat dan sederhana, dengan pengkodean minimum. (Lihat produk seperti
- Mesin Ekspresi.)
- Anda hanya ingin menulis situs web sederhana dengan beberapa fitur standar.

Menghemat waktu

CI tidak butuh waktu lama untuk belajar, dan dengan cepat membayar usaha Anda dalam waktu yang dihemat nanti. Mari kita lihat ukuran sederhana:

Bagaimana CI mengurangi jumlah kode yang perlu Anda ketik. Ini tidak hanya baik untuk orang yang malas. Semakin sedikit Anda mengetik, semakin sedikit kesalahan yang Anda buat, dan semakin sedikit waktu yang Anda habiskan untuk men-debug kode Anda. Semakin kecil kode Anda, semakin cepat memuat dan semakin sedikit ruang yang digunakan. Berikut adalah dua contoh (yang akan dijelaskan nanti dalam buku ini, jadi jangan khawatir sekarang tentang cara kerjanya!). Bayangkan Anda sedang menulis kueri basis data. Ini adalah bagaimana Anda dapat menulis fungsi dalam program PHP Anda untuk menanyakan database MySQL:

```
$connection = mysql_connect("localhost","fred","12345");
mysql_select_db("websites", $connection);
$result = mysql_query ("SELECT * FROM  sites", $connection);while ($row =
mysql_fetch_array($result, MYSQL_NUM))
{
    foreach ($row as $attribute)print
        "{$attribute[1]} ";
}
```

Sekarang lihat bagaimana fungsi CI akan menangani kueri serupa:

```
$this->load->database('websites');
$query = $this->db->get('sites'); foreach ($query-
    >result() as $row)
    {
        print $row->url
    }
```

Bandingkan jumlah karakter: 244 untuk sintaks tradisional; 112 untuk CI.

Sekarang bayangkan Anda sedang menulis formulir entri data dalam HTML, dan Anda menginginkan kotak kueri drop-down. Katakanlah kotak kueri tarik-turun ini menampilkan tiga opsi dan memungkinkan pengguna untuk memilih salah satunya. Dalam HTML, kotak drop-down dapat dibuat seperti ini:

```
<select name="type">
<option value="1">www.this.com</option>
<option value="2">www.that.com</option>
<option value="3" selected>www.theother.com</option>
</select>
```

Versi CI lebih pendek dan, karena bekerja dari array, lebih disesuaikan dengan pemrosesan PHP:

```
$urlarray = array(
                    '1'   => 'www.this.com',
                    '2'   => 'www.that.com',
                    '3'   => 'www.theother.com',
                );

$variable .= form_dropdown('url', $urlarray, '1');
```

Dalam HTML, Anda perlu mengetikkan 154 karakter; di CI, 128.

Jadikan Situs Lebih Kuat

Meskipun Anda tidak perlu menulis banyak kode, CI menyediakan banyak fungsi standar untuk Anda, dan mengingat semua keanehan dan keanehan tersebut. Itu melacak hal-hal yang mungkin telah Anda lupakan semua. (Sentuhan-sentuhan kecil yang membedakan situs amatir dari yang profesional…)

Perbarui Tautan Secara Otomatis

Bayangkan Anda baru saja menulis halaman menu, dengan banyak hyperlink ke halaman lain di situs Anda. Semuanya dalam format HTML tradisional:

```
<a href="http://www.mysite.com/index.php/start/hello/fred">say hello to Fred</a>
```

Kemudian, Anda memutuskan untuk memindahkan situs ke URL lain. Itu berarti Anda harus menelusuri kode Anda dengan susah payah, mencari setiap URL, dan menulis ulang, atau tidak ada tautan Anda yang akan berfungsi.

CI memberi Anda fungsi sederhana untuk menulis hyperlink seperti ini:

```
echo anchor(start/hello/fred, Say hello to Fred);
```

CI juga mendorong Anda untuk meletakkan URL situs Anda dalam file konfigurasi yang dapat diakses oleh situs Anda lainnya. Fungsi jangkar CI yang kami gunakan di sini secara otomatis merujuk ke file konfigurasi itu. Jadi, ketika Anda datang untuk memindahkan situs Anda, Anda hanya perlu mengubah satu entri di file konfigurasi, dan semua hyperlink Anda diperbarui secara otomatis.

Simpan Kerusakan Basis Data: 'persiapkan' Formulir Entri Data

Entri data penuh dengan masalah. Karena keterbatasan HTML dan database, data yang berisi simbol tertentu—misalnya, apostrof dan tanda kutip—dapat menyebabkan database Anda mogok atau memberikan hasil yang tidak Anda harapkan. Jawabannya adalah menyiapkan atau 'mempersiapkan' data Anda di formulir entri data Anda, sebelum diserahkan ke database. Semua ini membutuhkan waktu dan sejumlah pengkodean tambahan. Pembantu formulir CI melakukan ini, secara otomatis. Jadi, ketika Anda membuat kotak input dengan mengetik:

```
echo form_input('username', 'johndoe');
```

Anda juga mendapatkan manfaat tersembunyi dari:

```
function form_prep($str = '')
{
    if ($str === '')
    {
        return '';
    }
    $temp = '__TEMP_AMPERSANDS__';

    // Replace entities to temporary markers so that
    // htmlspecialchars won't mess them up
    $str = preg_replace("/&#(\d+);/", "$temp\\1;", $str);
    $str = preg_replace("/&(\w+);/",  "$temp\\1;", $str);
    $str = htmlspecialchars($str);

    // In case htmlspecialchars misses these.
    $str = str_replace(array("'", '"'), array("&#39;",
    "&quot;"), $str);
```

```
// Decode the temp markers back to entities
$str = preg_replace("/$temp(\d+);/","&#\\1;",$str);
$str = preg_replace("/$temp(\w+);/","&\\1;",$str);
return $str;
}
```

Ini adalah kode yang menangani karakter khusus seperti '&'; sehingga tidak menimbulkan kebingungan saat formulir Anda dikirimkan. Seperti yang Anda lihat, ada beberapa kode regex yang cukup rumit di sana.

Mungkin Anda suka mengetik regex. Beberapa orang suka berbaring di tempat tidur paku, beberapa suka mendengarkan ABBA; itu negara bebas. (Yah, di sinilah saya menulis ini.) Tetapi jika Anda tidak menyukai hal-hal ini, Anda dapat membiarkan CI melakukannya untuk Anda (regex, maksud saya, bukan ABBA), dan Anda bahkan tidak perlu menyadarinya dari kode yang bekerja di latar belakang untuk Anda, setiap kali Anda menulis satu baris kode sederhana itu:

```
echo form_input('username', 'johndoe');
```

Jadikan Kode Anda Lebih Baik dan Dapat Diandalkan

CI juga memudahkan Anda melakukan hal-hal yang mungkin belum pernah Anda coba sebelumnya. Tentu saja, pengguna PHP selalu dapat mengintegrasikan pustaka dari PEAR dan sumber lain, tetapi ini tidak selalu mudah untuk diintegrasikan, atau digunakan, dan sintaks serta standarnya sangat berbeda. CI memiliki seperangkat standar yang sama, dan setelah Anda menguasai sintaksnya, semua bagiannya bekerja sama tanpa komplikasi. Semua kodenya ditulis dengan baik dan dapat diandalkan, dan diuji oleh komunitas penggunanya. Ini menempatkan lebih banyak kecanggihan di tangan Anda. Mari kita lihat dua contoh untuk mengilustrasikan hal ini.

Kirim Lampiran Email tanpa Kerepotan

Mengirim email adalah bisnis yang kompleks. Kode CI untuk melakukannya terlihat mudah diikuti:

```
$this->load->library('email');
$this->email->from('your@your-site.com', 'Your Name');
$this->email->subject('Email Test');
$this->email->message('Testing the email class.');
$this->email->send();
```

Ada sejumlah masalah yang terlibat dalam pengiriman email: mengatur pembungkusan kata (dan menghindarinya agar URL tidak terbungkus dan putus) misalnya, atau mengirim lampiran. Fungsi standar PHP bisa menjadi sangat kompleks di sini, dan hasilnya adalah banyak penulis kode tergoda untuk menghindari penggunaan fungsi ini jika memungkinkan.

Kelas email CI mempermudah pengiriman lampiran. Anda menulis:

```
$this->email->attach('/path/to/photo1.jpg');
```

CI melakukan sisanya. Bekerja di belakang layar, misalnya, adalah fungsi yang memilah jenis MIME untuk hampir ratusan jenis lampiran yang berbeda. Jadi ia tahu bahwa foto Anda, photo1.jpg, adalah tipe MIME 'image/jpeg'. Ia ingat untuk menghasilkan pembatas batas di tempat yang tepat di sekitar lampiran Anda. Ini menangani pembungkusan teks Anda, dan memungkinkan Anda untuk dengan mudah menandai potongan teks yang tidak ingin Anda bungkus.

Hemat Bandwidth dengan Zipping File yang Perlu Diunduh Pengguna

Untuk menghemat bandwidth, adalah praktik yang cukup umum untuk mengompresi atau 'ZIP' file sebelum Anda mengunduhnya. Itu bukan sesuatu yang pernah saya lakukan, dan saya tidak tahu bagaimana melakukannya. Di sisi lain, CI memiliki fasilitas bagus yang memungkinkan Anda menghasilkan file zip dengan empat baris kode:

```
$name = 'mydata1.txt';
$data = 'the contents of my file….....';
$this->zip->add_data($name, $data);
$this->zip->archive('c:/my_backup.zip');
```

Jalankan ini, dan Anda menemukan arsip ZIP di drive C Anda yang berisi satu file. Pembaca file ZIP Anda akan membuka ritsletingnya dan menghasilkan data asli untuk Anda.

Orang-orang yang menggunakan situs Anda tidak akan tahu bahwa Anda telah menghasilkan hasil yang mengesankan ini dengan begitu mudah. Mereka akan terkesan! Situs Anda akan menghemat bandwidth. Anda melakukannya dalam hitungan menit, bukan jam.

1.2 APA ITU FRAMEWORK CODEIGNITER?

Tak lama setelah pemrograman ditemukan, seseorang memperhatikan bahwa itu melibatkan banyak operasi berulang. Dan tak lama setelah itu, orang lain—mungkin Ada Lovelace, kunci pas di tangan, menyesuaikan mesin diferensial Babbage, atau mungkin Alan Turing di Bletchley Park—memutuskan untuk memodulasi kode, jadi Anda hanya perlu menulis potongan tertentu sekali, dan bisa kemudian menggunakannya kembali. Pemrogram PHP digunakan untuk menulis potongan kode yang terpisah dalam fungsi, dan kemudian menyimpan fungsi tersebut dalam file yang disertakan. Pada satu tingkat, kerangka hanya itu: banyak potongan kode, disimpan dalam file terpisah, yang menyederhanakan pengkodean operasi berulang.

Dalam contoh di atas, menghubungkan ke database atau membangun elemen formulir HTML diabstraksikan dan disederhanakan untuk Anda. Anda memanggil fungsi dalam kerangka kerja, yang lebih mudah ditangani daripada kode aslinya. Ini lebih dari itu. Menulis kode melibatkan pilihan terus menerus antara banyak cara untuk mengatasi masalah yang sama; jadi sebagian besar kerangka kerja juga memaksakan serangkaian pilihan pada Anda. Mereka sudah mulai menangani masalah dengan satu cara, jadi Anda juga harus melakukannya. Jika ini adalah pilihan yang masuk akal, ini juga membuat hidup Anda lebih sederhana. (Jika tidak, itu seperti mencoba menulis brosur penjualan menggunakan Excel, atau membuat proyeksi arus kas menggunakan Word. Keduanya mungkin dapat dilakukan, tetapi tidak juga menggunakan waktu Anda dengan sebaik-baiknya.)

Keputusan desain yang masuk akal memastikan bahwa hal-hal yang Anda butuhkan dapat diakses, tetapi mencegahnya tumpah satu sama lain. Kerangka kerja yang baik membuat keputusan itu untuk Anda, memulai Anda dengan dasar yang masuk akal untuk program Anda dan membimbing Anda melalui langkah selanjutnya. Sebutkan kerangka kerja saat ini, dan orang-orang berpikir tentang Ruby on Rails.

Rails telah menjadi kisah sukses sekitar setahun terakhir, karena tampaknya menawarkan pengembangan situs web yang mudah dan cepat, dengan jumlah pengkodean yang minimum. Pada dasarnya, ini adalah struktur dan seperangkat alat, yang dibuat untuk digunakan dengan bahasa Ruby, yang memungkinkan Anda membangun jenis program Ruby tertentu dengan lebih cepat. Ini bukan satu-satunya kerangka kerja untuk Ruby, tetapi sangat efektif dan, sepatutnya, sangat populer. Di sisi lain, jika Anda telah meluangkan waktu dan usaha untuk mempelajari PHP, memulai dari awal lagi di Ruby adalah waktu yang lama. Ada beberapa kerangka kerja yang tersedia untuk PHP juga. CI hanya salah satu dari sekitar 40. Mereka termasuk framework Zend, Cake, Trax, dan lain-lain. Ada bagan praktis di http://www.phpit.net/article/ten-different-php-frameworks/ yang membandingkan sepuluh yang paling populer.

Jika Anda melihatnya, Anda akan melihat bahwa posting di forum pengguna mereka menjadi sangat panas tentang kerangka kerja mana yang terbaik. Kebenaran tampaknya bahwa masing-masing memiliki kekuatannya, dan tidak ada yang tanpa kelemahannya sendiri. Batu ujian saya adalah bahwa saya sibuk; jadi kerangka kerja harus menghemat waktu saya, dan setelah menemukan yang cocok untuk saya, saya berpegang teguh pada itu. Itu sebabnya buku ini hanya tentang CI.

1.3 SIAPAKAH PENGEMBANG CODEIGNITER PERTAMA KALI?

CI ditulis oleh Rick Ellis, musisi rock yang menjadi programmer. Rick mencari nafkah sebagai CEO pMachine, yang menjual sistem manajemen konten luar biasa yang disebut Expression Engine. Pada Januari 2006, ia menulis di blognya, http://www.ellislab.com:

Gambar 1.1 Foro Rick Ellis penemu Framework CI

"... Saya menghabiskan beberapa minggu untuk meneliti dan menginstal kerangka kerja PHP, benar-benar menggedor beberapa dari mereka, dan saya benar-benar kecewa. Saya menemukan bahwa sebagian besar kerangka kerja mengalami masalah ini:

- Mereka memiliki dokumentasi yang mengerikan dan mengerikan, jika memang ada.
- Mereka membuat banyak asumsi tentang pengetahuan Anda
- dan tingkat keterampilan, dan umumnya mengharapkan Anda untuk mengetahui semuanya.
- Mereka ditulis untuk orang yang memiliki hak root server dan dapat mengubah pengaturan sistem.
- Mereka berasumsi bahwa Anda memiliki akses ke baris perintah. Faktanya, banyak yang tidak berfungsi jika Anda tidak dapat melakukan bash out perintah.
- Mereka cenderung membutuhkan banyak dependensi, seperti perpustakaan PEAR atau berbagai sumber terbuka.
- Mereka cenderung tidak perlu rumit untuk digunakan, dengan sintaks tumpul, template berbasis XML, dan fitur lain yang sama sekali tidak diperlukan untuk sebagian besar aplikasi web.
- Mereka terlalu besar, atau terlalu minimalis untuk berguna.
- Kerangka kerja terbaru hanya berjalan di PHP 5, yang saat ini hanya memiliki tingkat adopsi 5%.

Saya belum menemukan satu kerangka kerja PHP yang benar-benar, sangat, sangat sederhana untuk digunakan, didokumentasikan secara menyeluruh dari atas ke bawah, secara asli mencakup semua alat yang diperlukan untuk membangun aplikasi yang kuat, memiliki antarmuka berbasis browser, dan dirancang untuk rata-rata pembuat kode PHP Anda, tanpa hak admin, yang menggunakan akun hosting standar. Tidak satu. Yang membuat saya berpikir ada pasar untuk produk seperti itu. ..."

Hasilnya adalah CI, yang ditulis sebagai proyek waktu luang. Rick dengan murah hati memutuskan untuk menyediakannya, gratis. Di sela-sela menjalankan bisnisnya, ia juga mengupdate CI dari waktu ke waktu. Dia juga menciptakan forum yang luar biasa, di mana pengguna CI dapat mengangkat masalah dan berbagi kiat, serta menemukan (dan terkadang memecahkan) bug dalam kodenya. Semua ini tersedia di situs web CI di http://www.codeigniter.com/. Apakah dia memenuhi tujuannya sendiri? Baca dan nilai sendiri...

1.4 MODEL BISNIS 'SUMBER TERBUKA'

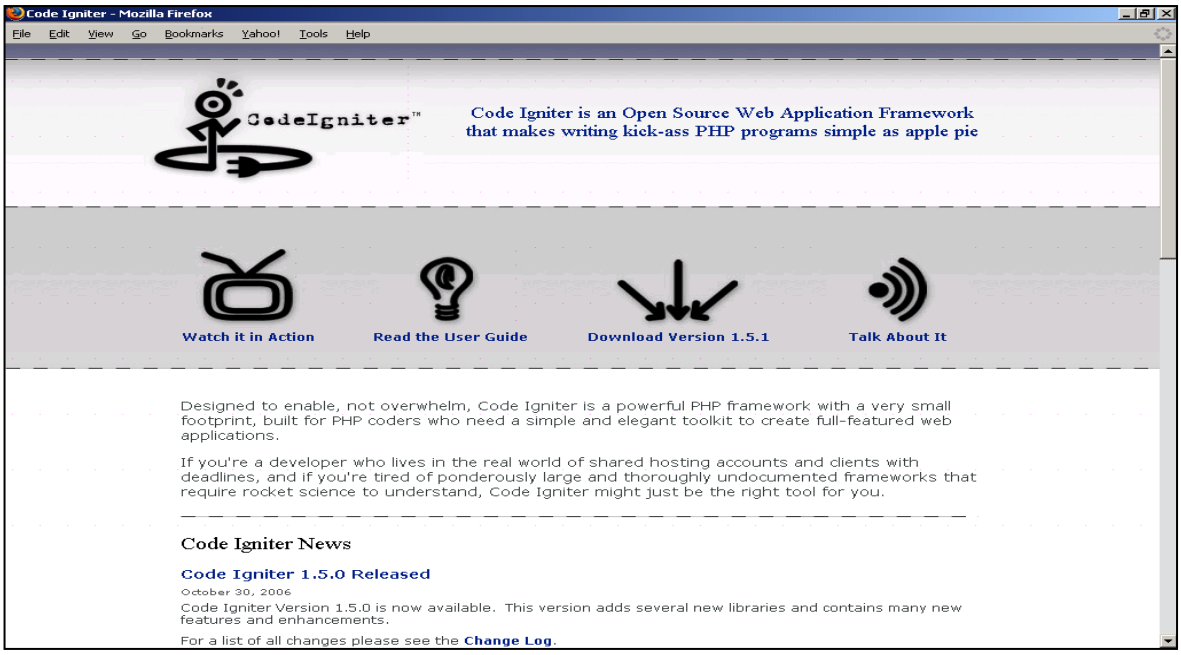
Mungkin ada sesuatu yang membingungkan tentang perangkat lunak semacam ini. Jika Anda menyukai perangkat lunak Anda dengan kontrak dukungan yang mahal dan nama 'perusahaan besar', maka CI bukan untuk Anda. (Namun, apa yang Anda lakukan dengan PHP? Pengguna PHP tahu bahwa dukungan, dan pengembangan perangkat lunak PHP, sebagian bergantung pada upaya 'komunitas' yang tidak dibayar—ratusan atau ribuan pengguna.)

Ada beberapa masalah dengan dukungan masyarakat. Konsistensi dan kualitas tinggi tidak 'dijamin'—siapa pun dapat memposting ke forum, dan terkadang postingan ini benar-benar salah. (Perhatikan bahwa jika Anda membaca cetakan kecil pada lisensi untuk perangkat

lunak komersial yang mahal, kualitas juga tidak dijamin di sana.) Tetapi dengan produk 'sumber terbuka', Anda harus mengambil minat yang cerdas daripada menerima semua yang Anda baca di forum pada nilai nominal. CI adalah kerangka kerja bagi orang-orang yang mampu mengambil minat yang cerdas.

Namun, setiap pengembang yang masuk akal harus bertanya-tanya apakah bijaksana untuk menginvestasikan waktu dan energi dalam produk yang merupakan 'band satu orang'. Rick Ellis menulisnya sebagai proyek waktu luang, dengan bantuan dari rekan pMachine-nya, Paul Burdick. Gratis. Dia tidak membuat komitmen untuk memelihara atau mengembangkannya. Dia mungkin kembali menjadi musisi rock.

Di sisi lain, setelah Anda mengunduhnya, versi yang Anda unduh akan terus berfungsi. Anda tidak harus bergantung pada peningkatan dan tambalan. Pengkodean Rick sangat bagus dan ada beberapa bug serius di dalamnya. Jika itu berhasil untuk Anda, maka tidak ada alasan mengapa itu tidak terus bekerja. Sejauh ini saya hanya menemukan dua kasus di mana kode saya gagal berfungsi, dan kesalahannya adalah bug dalam kerangka kerja daripada dalam pengkodean saya sendiri di atasnya. (Kedua bug telah dipecahkan.)



Gambar 1.2 Situs web CI adalah pintu gerbang ke komunitas dan forum.

1.5 APA YANG TIDAK DILAKUKAN CI

Ada beberapa hal yang tidak dilakukan CI. Rick bermaksud CI menjadi kerangka kerja yang kecil dan 'ringan'. (Unduhan zip untuk versi 1.5 hanya 737 KB dan unduhan dalam hitungan detik. Kerangka kerja Zend adalah 10 megabita.) Ini bukan jawaban untuk semua masalah yang pernah Anda alami. Tapi itu:

- Mempermudah dan mempercepat pemrograman dalam PHP
- Struktur situs Anda dan membantu Anda melalui keputusan arsitektur.

Salah satu hasil dari menjadi 'ringan' adalah tidak memiliki banyak fitur seperti beberapa pesaingnya. Rel telah menjadi terkenal sebagian karena mengandung 'scaffolding' dan 'generator'. Ini adalah alat yang secara otomatis menulis skrip dasar tertentu untuk Anda. Jadi, misalnya, setelah Anda menyiapkan database, Rails membuat halaman web 'out-of-the-box' untuk melakukan operasi dasar Buat, Baca, Perbarui, dan Hapus (CRUD) pada tabel database. Selain itu, Rails memungkinkan Anda untuk menulis 'generator'—potongan kode yang secara otomatis menulis skrip dasar lainnya. Komunitas Rails telah menciptakan cukup banyak dari ini; sehingga Anda dapat secara otomatis menghasilkan skrip yang melakukan segala macam hal pintar.

CI tidak melakukan ini. (Ada 'scaffolding' yang belum sempurna—scaffolding adalah template yang menjelaskan bagaimana database aplikasi dapat digunakan— dalam CI, tetapi seperti yang dikatakan oleh manual online: "Perancah dimaksudkan untuk penggunaan pengembangan saja. Ini memberikan keamanan yang sangat kecil .... Jika Anda menggunakan scaffolding pastikan Anda menonaktifkannya segera setelah Anda selesai menggunakannya. JANGAN biarkan diaktifkan di situs langsung." Cukup kata.) Sebaliknya CI berkonsentrasi untuk membuat hal-hal dasar menjadi mudah. Beberapa hal yang ditanganinya adalah:

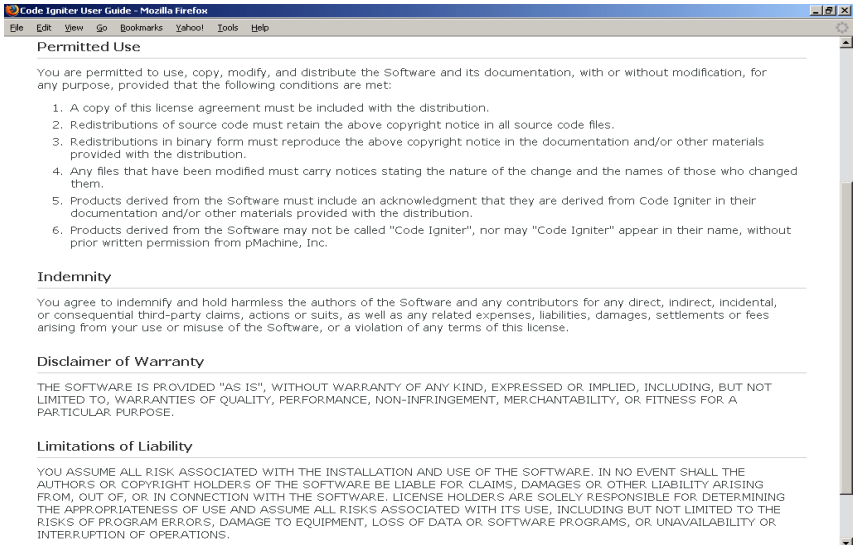
- Manajemen sesi dan cookie (lihat Bab 6)
- Akses database dan kueri (lihat Bab 4)
- Membangun hal-hal HTML, seperti halaman dan formulir, dan memvalidasi entri formulir (lihat Bab 5)
- Pengujian (Bab 8)
- Berkomunikasi di Internet, menggunakan FTP atau XMLRPC (Bab 9)

Terdengar akrab? Semua ini adalah proses dasar, yang harus Anda lalui jika Anda membangun situs web dinamis. CI membuat proses ini lebih mudah, dan membuat kode Anda lebih mungkin berfungsi.

1.6 LISENSI

Jika Anda membuat aplikasi komersial, persyaratan lisensi untuk perangkat lunak apa pun yang Anda gunakan menjadi penting. (Jika Anda meningkatkan modal ventura, harap pengacara VC membahasnya secara rinci.) Tidak ada masalah dengan CI. Ini memiliki lisensi yang sangat murah hati yang diunduh dengan file Anda.

Tidak seperti beberapa perangkat lunak komersial yang dapat saya pikirkan, lisensi CI bahkan muat di satu layar. Ini dia, pada tangkapan layar berikut:



Gambar 1.3 Lisensi Framework CI

1.7 RINGKASAN

Jika Anda sudah mengetahui beberapa PHP dan sedang menulis situs web cerdas, kerangka kerja CodeIgniter adalah tentang membuat hidup Anda lebih mudah. Ini membantu Anda

- Menghemat waktu
- Jadikan situs Anda lebih tangguh
- Mencapai pengkodean yang lebih canggih

Itu membuat pengkodean menyenangkan lagi, bukan tugas.

Ada beberapa kerangka kerja, dan tidak hanya untuk bahasa PHP. Semuanya menawarkan potongan kode pra-tertulis yang membuat proses pengkodean yang berulang atau kompleks menjadi lebih mudah, dan memaksakan struktur yang bermanfaat pada pengembangan situs Anda. Buku ini tidak membuat perbandingan antara kerangka kerja. Saya telah menemukan CI bekerja untuk saya, dan saya ingin menjelaskan bagaimana dan mengapa. Saya harap itu berguna bagi Anda, dan Anda akan dapat menghemat waktu sebanyak yang saya lakukan, dan sebagai hasilnya, lebih menikmati proses pengkodean.

Buku ini membawa Anda melalui beberapa fitur utama kerangka kerja, dan mencoba menjelaskan beberapa dari apa yang terjadi 'di bawah tenda'. Saya telah menggunakan contoh dunia nyata untuk ilustrasi kode dalam buku ini untuk mencoba menunjukkan bahwa CI adalah alat yang serius yang dapat dengan cepat dan mudah digunakan dalam lingkungan yang menuntut.

BAB 2
KERJA DUA MENIT
MENYIAPKAN SITUS CODEIGNITER

Menyiapkan paket CI di server web Anda mudah. Bab kecil ini menjelaskan apa yang terjadi ketika Anda menginstal situs, dan file mana yang akan dibuat. Mari lihat:

- Perangkat lunak apa yang Anda perlukan untuk situs pengembangan Anda
- Menginstal file CI: unduhan sederhana dan operasi unzip
- Konfigurasi dasar CI: apa itu folder dan bagaimana caranya terorganisir
- Pengontrol awal dan tampilan yang diinstal CI
- Beberapa modifikasi dasar untuk menunjukkan cara kerjanya

2.1 PRASYARAT

CodeIgniter sangat fleksibel. Ini akan bekerja sama baiknya dengan PHP 4.3.2 ke atas, atau PHP 5. Karena sebagian besar ISP masih belum mendukung PHP 5, ini berguna, dan menghemat biaya hosting. Anda juga akan membutuhkan database. Panduan pengguna online CI mengatakan: "Database yang didukung adalah MySQL, MySQLi, MS SQL, Postgre, Oracle, SQLite, dan ODBC." Untuk mengembangkan dan menguji situs web dinamis, Anda memerlukan server web. Biasanya, Anda akan mengembangkan dan menguji situs Anda di server lokal, yaitu server yang berjalan di mesin Anda sendiri (dengan alamat loopback 127.0.0.1 atau localhost) daripada di situs jarak jauh di Internet.

Jika Anda tidak terbiasa dengan proses pengaturan server web, paling mudah untuk menginstal paket seperti Xampplite, yang menginstal Apache, PHP, dan MySQL ke mesin Windows dengan konfigurasi minimum oleh Anda. Xampplite gratis, dilengkapi dengan instruksi lengkap, dan hampir selalu mudah dipasang. Atau, beberapa versi Windows dilengkapi dengan server web mereka sendiri. Ini juga membantu untuk memiliki editor PHP yang baik di sistem Anda. Anda dapat melakukan semuanya pada editor teks, tetapi saya menemukan bahwa fitur penyorotan sintaks dari editor yang baik menyelamatkan saya dari membuat banyak kesalahan sederhana dengan tanda kurung tertutup atau tanda kutip yang tidak cocok. Setelah Anda mencapai sejauh ini, saya memperkirakan Anda akan membutuhkan waktu dua menit untuk menjalankan CI di sistem Anda.

2.2 MENGINSTAL CODEIGNITER

Satu hal yang tidak Anda perlukan adalah kartu kredit Anda: CI benar-benar gratis! Setelah server Anda diatur, buka situs CodeIgniter di http://www.codeigniter.com/ dan unduh kerangka kerja versi terbaru. Versi 1.5.3, yang terbaru, hanya 737KB saat di-zip, jadi unduhannya tidak terlalu lama. Buka zip folder, dan instal file CodeIgniter di folder root web Anda. Jika Anda menggunakan Xampplite, ini biasanya folder htdocs di dalam folder Xampplite. File index.php CodeIgniter harus berada di direktori root. Folder root adalah folder yang akan Anda tunjuk jika Anda menavigasi ke situs—dalam hal ini, dengan mengakses

http://127.0.0.1. Dari dua menit yang kita perlukan untuk menyiapkan situs, satu menit sudah habis!

Termasuk dengan CI adalah panduan pengguna yang komprehensif (dalam folder user_guide). Anda akan sering menggunakan ini. Biasanya jelas, dan sering kali lebih detail daripada yang bisa dilakukan buku ini. Jadi, cobalah jika Anda terjebak.

Saat file-file ini ada di mesin Anda, Anda dapat mengaksesnya dengan dua cara:

- Sebagai URL—misalnya, http://127.0.0.1
- Melalui jalur direktori normal: mis., C:/xampplite/htdocs/index.php

Anda seharusnya dapat melihat layar selamat datang CI hanya dengan menavigasi ke URL Anda dengan browser. Sesederhana itu! Halaman selamat datang memberi tahu Anda bahwa apa yang Anda lihat dibangun oleh dua file, tampilan dan pengontrol.

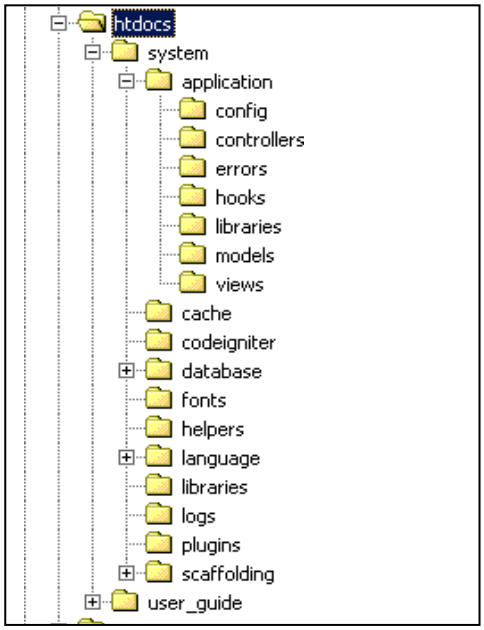
2.3 MENJELAJAHI STRUKTUR FILE

Setelah Anda menginstal file CI, lihat direktori baru yang telah dibuat. Memahami apa yang dilakukan berbagai jenis file sangat penting. Folder root Anda sekarang akan terlihat seperti diagram di bawah ini. Jika Anda pernah melihat Rails, struktur ini akan terlihat cukup familiar. Anda dapat membagi folder ini menjadi tiga grup:

- Yang akan Anda isi (misalnya, pengontrol, model, dan tampilan: semua ada di folder application). Terlepas dari tampilan selamat datang dan pengontrol yang baru saja Anda lihat, folder-folder ini kosong.
- File dalam folder sistem adalah kode sistem untuk CI (system/libraries,system/codeigniter, system/drivers, dll.). Anda dapat membacanya, dan mengubahnya jika diinginkan—tetapi jangan lakukan ini sampai Anda memahami cara kerja CI. Dan jika Anda mengubah kode dasar, ingatlah bahwa Anda mungkin harus mengubahnya lagi saat mengunduh pembaruan CodeIgniter, karena versi baru dapat menimpa perubahan Anda. Anda mungkin juga menemukan bahwa kode baru tidak lagi berfungsi dengan amandemen Anda. Terakhir, Anda mungkin menemukan bahwa apa yang ditulis Rick cukup bagus.
- Yang sudah setengah ditulis, tetapi mungkin memerlukan tambahan atau perubahan (language, config, errors.) Folder-folder ini disetel ke default, tetapi Anda harus segera mengubah file konfigurasi Anda; jadi mari kita selesaikan itu.

2.4 FILE KONFIGURASI

Ingat kami akan mengambil dua menit untuk mengatur situs kami? Menit kedua dihabiskan untuk melakukan beberapa konfigurasi dasar. Folder config berisi sekelompok file yang mengatur konfigurasi dasar untuk situs Anda.



Gambar 2.1 Tampilan Konfigurasi file

Buka file config/config.php dan beri tahu situs di mana menemukannya. Beberapa baris pertama file harus mengatakan sesuatu seperti:

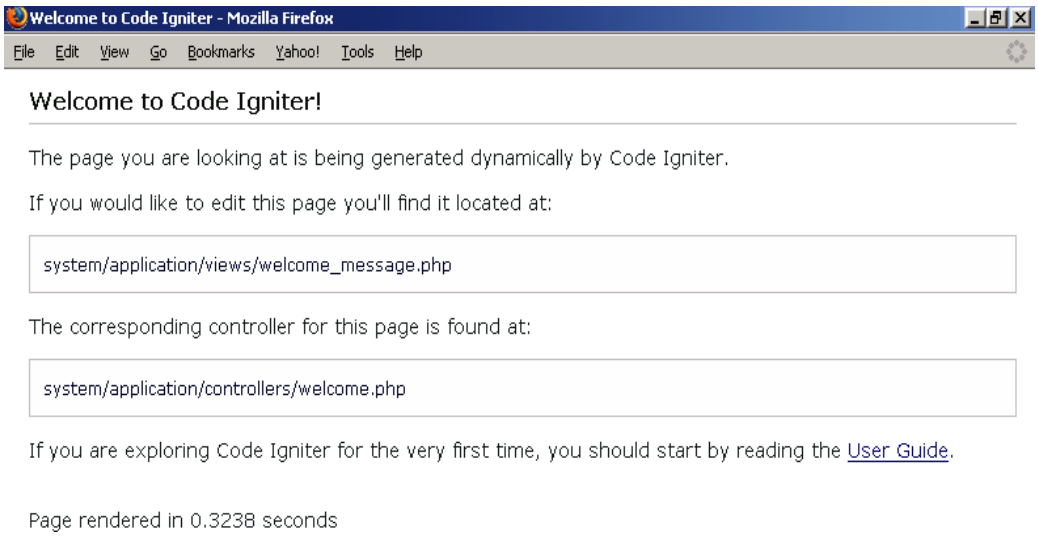
```
/*
|---------------------------------------------------
| Base Site URL
|---------------------------------------------------
|
| URL to your Code Igniter root. Typically this
| will be your base URL, WITH a trailing slash:
|
|    http://www.your-site.com/
|
*/
$config['base_url'] = "http://127.0.0.1/";
/*
```

Perhatikan seberapa baik file CI dikomentari!

Ubah nilai dalam tanda kutip agar sesuai dengan root web Anda sendiri. Jika Anda memiliki masalah, instruksi pengaturan yang lebih rinci diberikan dalam manual online. Sebagai prinsip dasar, gunakan file config.php untuk menyimpan informasi tentang situs Anda daripada menyebarkannya di sekitar file Anda. Pertama, lebih mudah untuk memperbarui jika semuanya ada di satu tempat. Kedua, saat Anda mentransfer situs Anda dari server pengembangan ke server produksi, Anda hanya perlu membuat satu set perubahan. Terakhir, banyak fungsi CI berasumsi bahwa informasi tertentu dapat ditemukan di sana. Ada file konfigurasi lain di folder konfigurasi, tetapi Anda dapat dengan aman membiarkannya di pengaturan default untuk saat ini. Dari dua menit kami perlu menyiapkan situs yang kedua. Di sisa bab ini, kita akan bermain-main dengan situs kita.

2.5 APAKAH BEKERJA?

Cara mudah untuk melihat apakah situs Anda berfungsi adalah dengan menavigasinya menggunakan browser Anda. Dengan asumsi Anda menjalankannya di folder root server lokal, ketikkan http://127.0.0.1 dan Anda akan melihat ini:



Gambar 2.2 Hasil tampilan http://127.0.0.1

Itu berarti CI aktif dan berjalan. Apakah itu membawa Anda lebih dari dua menit?

2.6 RINGKASAN

Dalam bab ini, kita telah melihat betapa mudahnya menginstal CI. Setelah Anda menyiapkan server web pengembangan, yang perlu Anda lakukan hanyalah mengunduh kode CI, membuka ritsletingnya, dan menyalinnya. Kemudian, kami melihat dengan cepat bentuk file yang telah kami instal dan melakukan beberapa konfigurasi dasar, dan di sanalah kami: situs CI yang berfungsi. Jika bab ini sangat pendek, itu karena CI mudah dipasang. Seperti semua hal lain dalam buku ini, ini tentang menghemat waktu dan membuat hidup lebih mudah.

BAB 3
MENAVIGASI SITUS

Sekarang setelah kita menginstal CI, kita perlu memahami cara kerjanya. Pembaca yang akrab dengan pola desain akan mengenali sekarang bahwa CI mengimplementasikan pola Model—View—Controller (MVC). Ini adalah metode mengatur file yang membentuk situs web, atau, jika Anda suka, membagi situs menjadi bagian-bagian yang masuk akal daripada memiliki satu gumpalan besar kode. Dalam bab ini, kita akan melihat secara singkat teori di balik MVC, dan kemudian cara CI mengatur dirinya sendiri secara internal. Secara khusus, apa yang ada di folder yang berbeda itu dan bagaimana mereka berkomunikasi? Bagaimana sebuah situs terstruktur? Dan bagaimana CI menavigasi di sekitarnya?
Bab ini melihat:

- Bagaimana MVC membantu mengatur situs web dinamis
- Proses di mana CI menganalisis permintaan Internet yang masuk dan memutuskan bagian mana dari kode Anda yang akan menanganinya
- Apa yang dilakukan kode itu?
- Aturan sintaks CodeIgniter
- Berbagai jenis file atau kelas yang dapat Anda temukan—atau tulis sendiri—di situs CI
- Cara meneruskan parameter ke pengontrol menggunakan URL
- Cara menulis tampilan yang lebih baik dan meneruskan data dinamis ke tampilan tersebut
- Bagaimana balasan dikembalikan ke surfer
- Bagaimana file atau kelas menyampaikan informasi dan kontrol satu sama lain
- Seberapa berguna kode disimpan di dalam file pembantu dan perpustakaan
- Beberapa petunjuk praktis tentang desain situs.

3.1 MVC—HANYA SINGKATAN LAIN?

MVC adalah sarana untuk mengatur situs web dinamis. Pola desain telah ada sejak 1979 ketika pertama kali dijelaskan oleh orang Norwegia, Trygve Reenskaug. Berikut adalah garis besar dari berbagai jenis file:

- Model adalah objek, yang mewakili data yang mendasarinya. Mereka mengarahkan kursor ke atas database dan mengaksesnya sesuai kebutuhan. Mereka juga dapat melakukan operasi pada data untuk menambah maknanya.
- Tampilan menunjukkan status model. Mereka bertanggung jawab untuk menampilkan informasi kepada pengguna akhir. (Meskipun biasanya tampilan HMTL, mereka mungkin bentuk antarmuka apa pun. Mereka mungkin tampilan yang disesuaikan secara khusus untuk layar PDA kecil atau telepon WAP, misalnya.)
- Kontroler menawarkan opsi untuk mengubah status model. Mereka bertanggung jawab untuk model konsultasi. Mereka memberikan data dinamis ke tampilan.

CI memiliki subfolder untuk model, tampilan, dan pengontrol. Setiap file di dalamnya adalah file .php, biasanya dalam bentuk kelas yang mengikuti konvensi penamaan tertentu. CI

membantu Anda mengikuti pola MVC, dan sebagai hasilnya membuatnya lebih mudah untuk meletakkan kode Anda. CI memungkinkan Anda banyak fleksibilitas, dan Anda mendapatkan semua keuntungan dari struktur MVC. Cobalah untuk berpikir dalam istilah MVC saat Anda menulis. Sejauh mungkin, cobalah untuk menjaga 'tampilan' Anda hanya terfokus pada presentasi, dan 'pengontrol' Anda murni pada pengendalian aliran aplikasi. Simpan logika aplikasi dalam model data dan database.

Dengan cara ini, jika Anda memutuskan untuk membuat kumpulan tampilan baru untuk metode tampilan baru, Anda tidak perlu mengubah banyak kode di pengontrol atau model mana pun. Jika Anda memperbarui beberapa 'logika bisnis', Anda hanya perlu mengubah kode di model.

Di sisi lain, meskipun ini adalah divisi yang sangat menarik dan berguna, penting untuk tidak menganggapnya terlalu serius. MVC dimaksudkan untuk membantu Anda dan bukan untuk menjadi pengekang. Program dan kerangka kerja yang berbeda mengimplementasikan MVC dengan cara yang sedikit berbeda. Forum CI berisi banyak pertanyaan sedih tentang cara yang 'benar' untuk mengimplementasikan MVC. (Haruskah saya melakukan kueri basis data dari pengontrol, atau haruskah ini hanya dilakukan dalam model? Dapatkah saya mengembalikan tampilan langsung dari model, atau haruskah saya melalui pengontrol terlebih dahulu?)

Daripada mencoba mencapai hasil yang 'benar' secara teoretis, ingatlah dua prinsip yang berguna. Ini ditetapkan di bagian Tujuan Desain dan Arsitektur dari Panduan Pengguna CI:

- Kopling Lepas: Kopling adalah sejauh mana komponen sistem saling bergantung. Semakin sedikit komponen yang bergantung satu sama lain, semakin dapat digunakan kembali dan fleksibel sistem tersebut. Tujuan kami adalah sistem yang digabungkan dengan sangat longgar.
- Komponen Singularitas: Singularitas adalah sejauh mana komponen memiliki tujuan yang terfokus secara sempit. Di CodeIgniter, setiap kelas dan fungsinya sangat otonom untuk memungkinkan kegunaan yang maksimal.

Ini adalah tujuan Rick Ellis dalam membangun CI, dan itu juga merupakan tujuan yang baik untuk situs Anda sendiri. Asalkan Anda memenuhi tujuan ini, tidak masalah apa nama bagian kode Anda. Itu tidak bekerja. Pengalaman saya sendiri adalah bahwa helper atau library yang 'terpasang longgar' yang ditulis untuk satu situs dapat dengan mudah dipindahkan ke situs lain, menghemat waktu pengembangan berjam-jam. Jadi, jika pengontrol Anda menanyakan database secara langsung, atau model Anda memanggil tampilan, kode CI akan berfungsi dengan baik—biasanya tidak ada masalah teknis—tetapi interpretasi MVC Anda mungkin tidak 'benar'.

3.2 STRUKTUR SITUS CI: PENGONTROL DAN VIEWS

Seluruh situs CI Anda dinamis. Artinya, mungkin tidak ada halaman 'statis' yang dapat Anda lihat sebagai kode HTML sederhana. (Anda dapat menambahkan beberapa jika Anda mau, tetapi mereka akan berada di luar kerangka CI.) Jadi, di mana situs Anda?

Ketika kami menginstal CI, kami melihat bahwa folder aplikasi menyertakan subfolder yang disebut model, tampilan, dan pengontrol. Setiap respons yang dihasilkan situs CI disusun oleh ketiga jenis file ini.

Mari kita lihat prosesnya secara detail.

Untuk menekankan poin bahwa kita tidak berurusan dengan halaman statis, masing-masing dengan URL sendiri, kami akan menunjukkan permintaan 'URL' dan menunjukkan bagaimana CI menafsirkannya. Pertama-tama, pertimbangkan permintaan Internet normal. Sambungan dibuat ke URL situs Anda, www.example.com, dan kemudian melalui soket muncul permintaan HTTP seperti:

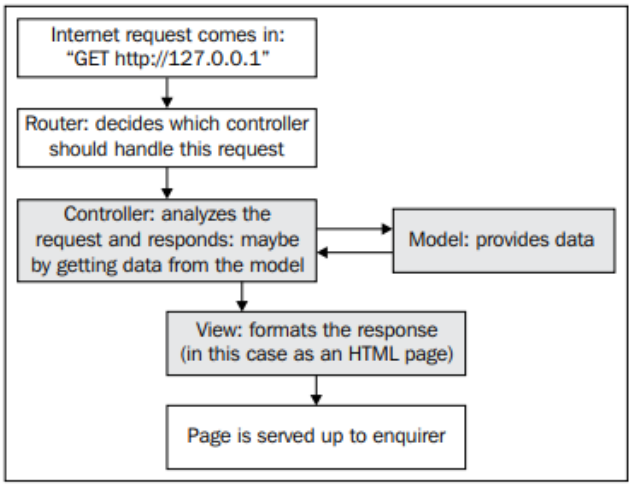
```
GET /folder/file.html HTTP/1.0
```

GET adalah jenis permintaan, HTTP/1.0 adalah versi HTTP yang digunakan, dan semua yang ada di antaranya adalah jalur relatif, dan nama file Anda. Namun di situs Anda, tidak ada file HTML statis sederhana yang dapat ditemukan. Sebaliknya, semua permintaan yang masuk dicegat oleh file index.php.

Jika pengguna meminta halaman di situs Anda dengan URL yang benar—misalnya dengan mengklik hyperlink di salah satu halaman Anda—permintaan akan terlihat lebih seperti ini:

```
GET /index.php/tests/showall HTTP/1.0
```

Jika pengguna tidak mengetahui URL persisnya dan hanya meminta www.example.com, maka CI memiliki sistem untuk menyetel alamat default (Kita akan melihat cara melakukannya sebentar lagi.). Dalam kedua kasus, langkah-langkahnya adalah:



Gambar 3.1 Langkah-langkah request situs CI

Permintaan yang masuk dari Internet ke root web Anda dicegat oleh index. php, yang bertindak sebagai 'router'. Artinya, ia memanggil 'pengontrol', yang kemudian mengembalikan 'tampilan'.

Bagaimana router mengetahui pengontrol mana yang harus dipanggil? Seperti yang telah kita lihat, terkadang permintaan yang masuk memberitahukannya. Misalnya, jika permintaan mengatakan:

```
GET http://127.0.0.1/index.php/welcome/index
```

dan jika Anda memiliki pengontrol bernama welcome, di situlah permintaannya.

Welcome Controller

Jadi, mari kita lihat pengontrol sambutan. Itu adalah satu-satunya pengontrol yang telah ditulis dan kodenya ada di system/application/controllers/welcome.php. Inilah yang dikatakannya:

```
<?php
  class  Welcome extends Controller
  {
          function Welcome()
          {
                        parent::Controller();
            }
          function index()
          {
                  $this->load->view('welcome_message');
            }
  }
?>
```

Anda akan melihat dari baris kedua bahwa file ini adalah sebuah kelas. Setiap pengontrol mewarisi dari kelas Controller asli—karenanya extends Controller. Di dalam kelas ada dua fungsi atau metode—Welcome() dan index().

CI membutuhkan nama pengontrol untuk memulai dengan huruf besar (kelas Selamat Datang), meskipun file disimpan sebagai /system/application/controllers/welcome.php —nama yang sama tetapi dengan huruf kecil.

Tiga baris berikutnya membentuk fungsi konstruktor. Perhatikan bahwa CI menggunakan konvensi PHP 4 yang lebih lama untuk penamaan fungsi konstruktor, yang juga dapat diterima di PHP 5—ini tidak mengharuskan Anda menggunakan PHP 5 dan senang dengan salah satu versi bahasa tersebut. Fungsi konstruktor digunakan untuk mengatur kelas setiap kali Anda membuat instance. Di sini, Anda meletakkan instruksi apa pun untuk memuat pustaka atau model lain, atau definisi variabel kelas apa pun.

Sejauh ini satu-satunya hal di dalam konstruktor adalah pernyataan parent::Controller(). Ini hanyalah cara untuk memastikan bahwa Anda mewarisi fungsionalitas kelas Controller. Jika Anda ingin memahami kelas parent CI Controller secara detail, Anda dapat melihat file /system/libraries/controller.php.

(Salah satu hal yang meyakinkan tentang CI adalah bahwa semua kode ada untuk Anda periksa, meskipun Anda tidak sering perlu melakukannya.)

Bekerja dengan Views

Mari kita kembali ke permintaan yang masuk sejenak. Router perlu mengetahui, tidak hanya pengontrol mana yang harus menangani permintaan, tetapi juga fungsi mana di dalam pengontrol itu. Itulah sebabnya permintaan ditentukan GET http://127.0.0.1/welcome/ index. Jadi router mencari fungsi di dalam pengontrol sambutan yang disebut index. Dan ini dia! Kemudian muncul fungsi index(). Fungsi ini hanya memuat tampilan ('welcome_view') menggunakan fungsi pemuat CI (this->load->view). Pada tahap ini, ia tidak melakukan sesuatu yang keren dengan tampilan, seperti meneruskannya informasi dinamis. Itu datang kemudian. 'welcome_view' yang ingin dimuat ada di folder views yang baru saja Anda instal: system/application/views/welcome_view.php. Tampilan khusus ini hanya halaman HTML sederhana, tetapi disimpan sebagai file PHP karena sebagian besar tampilan memiliki kode PHP di dalamnya. (Tidak ada gunanya melakukan semua ini jika kita hanya akan menyajikan HTML statis biasa.)
Berikut kode (sedikit disingkat) untuk tampilan:

```
<html>
<head>
	<title>Welcome to Code Igniter</title>
	<style type="text/css">
		body
		{
			background-color: #fff;
			margin: 40px;
			font-family: Lucida Grande, Verdana, Sans-serif; font-size: 14px;
			color: #4F5155;
		}
. . . . . more style information here . . . .
	</style>
</head>
<body>
	<h1>Welcome to Code Igniter!</h1>
	<p>The page you are looking at is being generated dynamically by Code Igniter.</p>
	<p>If you would like to edit this page you'll find it located at:</p>
	<code>system/application/views/welcome_message.php</code>
	<p>The corresponding controller for this page is found at:</p>
	<code>system/application/controllers/welcome.php</code>
	<p>If you are exploring Code Igniter for the very first time, you should start by reading
	the <a href="user_guide/">User Guide</a>.</p>
</body>
</html>
```

Seperti yang Anda lihat, ini seluruhnya terdiri dari HTML, dengan stylesheet CSS yang disematkan. Dalam contoh sederhana ini, pengontrol belum meneruskan variabel apa pun ke tampilan.

Default Controller

Saya sebutkan sebelumnya bahwa CI merutekan permintaan ke pengontrol default jika permintaan tidak menentukan ke mana ia ingin pergi. Anda menyetel pengontrol default dari file konfigurasi—dalam hal ini adalah /system/application/config/routes. Ini berisi:

```
$route['default_controller'] = "welcome";
```

Jika Anda tidak menyetel default, pengguna situs yang tidak tahu persis URL yang diminta—itu kebanyakan dari mereka, jika dipikir-pikir—akan mendapatkan halaman '404 tidak ditemukan'.

Dalam hal ini, rute default adalah ke pengontrol sambutan Anda. Jika tidak ada fungsi yang ditentukan, rute default ke fungsi /index dari pengontrol apa pun yang dipilih, jadi pastikan Anda menyertakan fungsi indeks, jika hanya untuk mencegah halaman '404'. Harap dicatat bahwa fungsi indeks tidak sama dengan fungsi konstruktor.

Anda dapat mengubah default ini jika Anda mau, dengan memasukkan dalam pengontrol yang ingin Anda ubah, sebuah fungsi yang disebut _remap($function), di mana $function adalah fungsi yang ingin Anda intersep dan redirect. _remap selalu dipanggil terlebih dahulu, apa pun yang dikatakan URL.

3.3 ATURAN SINTAKS CODEIGNITER

Sebelum kita mulai, mari kita rangkum aturan sintaks yang digunakan CI. Kerangka kerja mengharapkan file diatur dengan cara tertentu, jika tidak, mungkin mengalami kesulitan mengidentifikasi file Anda dengan benar, atau menggunakannya.

Controller

Ini adalah kelas (yaitu kode OO). Itu dipanggil langsung oleh URL, mis., 'www.example.com/index.php/start/hello'. Controller digunakan untuk memanggil fungsi dengan nama, misalnya, mainpage(); namun, Anda tidak dapat memanggil fungsi di dalam pengontrol lain.

Sintaks: Pengendali dimulai dengan class Start extends Controller (di mana Namanya dari controller memiliki huruf pertama dalam huruf besar) dan disimpan sebagai file .php di /system/application/controllers folder. Saat disimpan, mereka tidak boleh memiliki huruf pertama dalam huruf besar; seperti di start.php dan bukan Start.php. Juga, mereka harus menyertakan konstruktor yang mengandung setidaknya:

```
function display()
    {parent::Controller();}
```

Semua kode lain harus ditulis sebagai fungsi terpisah di dalam kelas, mis., fungsi hello()

View

Tampilan adalah file HTML yang dapat berisi 'pulau' PHP. Mereka dimuat oleh $this->load->view('testview', $data). Memuat dan menggunakan tampilan dilakukan dalam tindakan yang sama.

Sintaks: Tampilan ditulis dalam HTML. Kode PHP disertakan dalam <?php ?> tag seperti file HTML lainnya. Itu disimpan sebagai file .php di folder tampilan.

3.4 JENIS FILE ATAU KELAS DI SITUS CI

Ada beberapa sub-folder berbeda di dalam folder aplikasi. Kami telah melihat folder controller, config, dan views. Tapi apa itu perpustakaan, model, dan skrip? Ini adalah salah satu area di mana CI tampaknya agak membingungkan. (Jika Anda telah menggunakan versi CI sebelum versi 1.5, Anda akan menyadari alasannya. Rick Ellis tidak senang dengan versi sebelumnya dan telah banyak mengubah strukturnya. Namun, untuk alasan kompatibilitas, beberapa anomali tetap ada.)

Secara teknis, folder-folder ini diperlakukan dengan cara yang hampir sama. Tidak ada alasan mengapa Anda tidak boleh meletakkan kode Anda di salah satu folder ini, meskipun Anda harus membuatnya sedikit berbeda di masing-masing folder. Katakanlah Anda telah menulis sebuah blok kode yang disebut tampilan, misalnya, yang berisi fungsi yang disebut halaman utama. Ada empat cara yang mungkin Anda lakukan: sebagai model, library, helper, atau plug-in. Tabel berikut merangkum perbedaan antara setiap pendekatan, dan menunjukkan cara memuat dan menggunakan setiap jenis.

Tabel 3.1 Solusi Jenis File yang ditawarkan

Jenis berkas	**Bagaimana cara menggunakannya**
model	Ini adalah kelas (yaitu berorientasi objek atau kode OO) Muat seperti ini: $this->load->model('display'); Gunakan seperti ini: $ this->display->mainpage(); Catatan tentang sintaks: Itu harus dimulai class Display extends Model Itu harus mencakup konstruktor yang mengandung setidaknya: function display() {parent::Model();} dan berisi fungsi mainpage() terpisah.Conceptually: The User Guide mengatakan, "Model adalah kelas PHP yang dirancang untuk bekerja dengan informasi dalam database Anda."
Library	Itu ada di sistem dan folder aplikasi. Sekali lagi, ini adalah kelas. (Catatan: pustaka Anda sendiri tidak secara otomatis disertakan dalam objek super CI, jadi Anda perlu memanggil sumber daya CI dengan cara yang berbeda. Lihat Bab 7 untuk detailnya) Muat seperti ini: $this->load->library('display'); Gunakan seperti ini: $this->display->mainpage(); Catatan tentang sintaks: Tidak perlu memperluas kelas dasar, atau untuk fungsi konstruktor. Ini cukup: class Display() { function mainpage()

	{ //code here } } Secara konseptual: Dimaksudkan untuk menyimpan kode Anda sendiri untuk memperluas CI fungsionalitas, atau untuk membuat fungsionalitas khusus situs.
helper	Itu bisa di folder system/helpers atau di folder application/ helpers. Ini adalah skrip (kode prosedural, bukan kelas OO) Muat seperti ini: *$this->load->helper('display');* Gunakan fungsi darinya seperti ini: mainpage(); Catatan tentang sintaks: File harus disimpan sebagai display_helper.php—yaitu, add _helperke nama file. mainpage() harus berupa fungsi yang disertakan dalam file, yang hanya merupakan kumpulan fungsi terpisah, bukan kelas. Akibatnya, Anda tidak dapat lagi mengakses sumber daya CI lainnya secara langsung. Secara konseptual: 'pembantu' dimaksudkan sebagai kumpulan tingkat rendah berfungsi untuk membantu Anda melakukan tugas tertentu.
plug-in	Itu ada di folder system/plugins tetapi juga dapat dibuat di folder applications/plugins. Ini adalah skrip (bukan kelas OO) Muat seperti ini: $this->load->plugin('display'); Gunakan fungsi darinya seperti ini: mainpage(); Catatan tentang sintaks: File harus disimpan sebagai display_pi.php—yaitu. tambahkan _pi ke akhir dari nama file. mainpage() harus berupa fungsi yang disertakan dalam file, yang hanya merupakan kumpulan fungsi terpisah, bukan kelas. Akibatnya, Anda tidak dapat lagi mengakses sumber daya CI lainnya secara langsung. Secara konseptual: Panduan Pengguna mengatakan, "… perbedaan utamanya adalah bahwa plug-in biasanya menyediakan satu fungsi, sedangkan Helper biasanya merupakan kumpulan fungsi….. plug-in dimaksudkan untuk dibuat dan dibagikan oleh komunitas kami. " (Lihat Bab 15 untuk contoh plug-in.)

Anda dapat meletakkan potongan kode baru Anda di salah satu folder ini, meskipun Anda harus menulisnya sebagai kelas berorientasi objek di dua folder pertama, dan sebagai skrip

prosedural di folder kedua, dan dalam kasus terakhir, Anda akan tidak dapat menggambar langsung di kelas CI lainnya. Jika tidak, perbedaan antara jenis folder sebagian besar adalah konseptual.

Anda akan melihat bahwa CI dapat memiliki dua set pembantu, plug-in, dan perpustakaan, meskipun bukan model. Mungkin ada satu set masing-masing di folder aplikasi, dan set lainnya di folder sistem. Perbedaannya, sekali lagi, sebagian besar bersifat konseptual.

- Yang ada di folder sistem dimaksudkan untuk menjadi bagian dari kode CI inti dan untuk digunakan bersama oleh semua aplikasi. Jika Anda memperbarui ke versi CI yang lebih baru, Anda akan menimpa folder sistem dan file-file ini dapat dimodifikasi.
- Yang ada di folder aplikasi hanya akan tersedia untuk satu aplikasi itu. Jika Anda memperbarui ke versi baru CI, folder aplikasi tidak akan ditimpa.
- Saat Anda mencoba memuat helper, plug-in, atau library, CI dengan bijaksana mencari di kedua jalur. Jika Anda mencoba memuat pustaka yang disebut tampilan, misalnya, CI akan melihat terlebih dahulu di direktori sistem/aplikasi/perpustakaan Anda. Jika direktori tidak ada atau pustaka tampilan tidak ada, CI kemudian akan mencari di folder sistem/perpustakaan.
- Ini berarti dimungkinkan untuk secara efektif menimpa pustaka inti, pembantu, dan plug-in CI dengan memperkenalkan milik Anda sendiri dengan nama yang sama di folder aplikasi. Jangan lakukan ini secara tidak sengaja. Namun, fleksibilitas ini merupakan keuntungan besar bagi pengguna CI berpengalaman jika Anda ingin memperluas kelas dasar dan skrip yang disertakan dengan CI—lihat Bab 13.

application	config	File konfigurasi: menyimpan informasi dasar tentang situs Anda yang tetap ada di antara sesi
	controllers	Pengendali
	errors	Berisi template untuk pengumuman kesalahan. Anda mungkin tidak perlu menyentuh ini.
	hooks	Kosongkan saat pertama kali diinstal, gunakan ini untuk 'kait' yang Anda buat. Hooks adalah cara untuk mengontrol cara file lain dimuat.
	libraries	Koleksi kode Anda, dimaksudkan untuk bekerja dengan aplikasi khusus ini
	models	Koleksi kode Anda, sekali lagi dimaksudkan untuk bekerja dengan aplikasi khusus ini
	views	Template untuk menampilkan informasi kepada pengguna
cache	Kosong saat pertama kali diinstal: jika Anda mengaktifkan caching (lihat Bab 10) data disimpan di sini	
codeigniter	File sistem dasar.	
database	File library untuk kelas database CI.	
fonts	Tidak dijelaskan dalam panduan pengguna, kecuali sebagai tempat menyimpan font untuk gambar watermark	

helpers	'Pembantu' tingkat sistem
language	Anda dapat menyimpan daftar frasa kunci Anda sendiri di sini—lihat Bab 11
libraries	Pustaka tingkat sistem
logs	Jika Anda mengatur sistem untuk mencatat kesalahan, file log dibuat di sini secara default
plugins	Lebih banyak blok kode tingkat sistem
scaffolding	Pustaka tingkat sistem untuk mengaktifkan 'scaffolding' yang belum sempurna.

3.5 MERANCANG VIEW YANG LEBIH BAIK

Pada tahap ini, Anda mungkin bertanya mengapa kami berusaha keras untuk menyajikan halaman HTML sederhana. Mengapa tidak memasukkan semuanya ke dalam satu file? Untuk situs sederhana, itu poin yang valid—tetapi siapa yang pernah mendengar tentang situs sederhana? Salah satu hal paling keren tentang CI adalah caranya membantu kami mengembangkan struktur yang konsisten, sehingga saat kami menambahkan dan mengembangkan situs kami, itu konsisten secara internal, ditata dengan baik, dan mudah dirawat.

Pada awalnya, kita membutuhkan tiga langkah umum:

- Tulis halaman tampilan
- Tulis lembar gaya
- Perbarui file konfigurasi kami untuk menentukan di mana stylesheet berada

Setelah ini selesai, kita perlu memperbarui pengontrol kita untuk menerima parameter dari URL, dan meneruskan variabel ke tampilan.

Pertama, mari kita mendesain ulang tampilan kita dan menyimpannya sebagai:

system/application/views/testview.php.

```
<html>
<head>
   <!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0
   Strict//EN'http:\Wwww.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>
   <html xmlns='http:\Wwww.w3.org/1999/xhtml'>
   <title>Web test Site</title>
   <base href= <?php echo "$base"; ?> >
   <link rel="stylesheet" type="text/css" href="<?php echo"$base/$css";?>">
</head>
<body>
   <h1><?php echo $mytitle; ?> </h1>
   <p class='test'> <?php echo $mytext; ?> </p>
</body>
</html>
```

Sebagian besar masih HTML, tetapi perhatikan 'pulau kode' PHP di baris yang disorot.

Anda akan melihat bahwa bit pertama kode PHP dibangun dalam tautan ke stylesheet. Mari simpan stylesheet sederhana sebagai mystyles.css, di folder root situs. Itu hanya mengatakan:

```
h1 {
margin:  5px;  padding-left
10px;
padding-right:  10px;
background: #ffffff;color: blue;
width: 100%; font-size:
36px;
}
.test{ margin: 5px;
padding-left: 10px; padding-right: 10px;
background: #ffffff;color: red;
width: 100%; font-size:
36px;
}
```

Itu memberi kami dua gaya untuk dimainkan, dan Anda akan melihat bahwa kami telah menggunakan keduanya dalam tampilan. Pertama, mari tambahkan entri ke file konfigurasi:

```
$config['css'] = "mystyles.css";
```

Ini hanya untuk memberi tahu situs nama dan alamat file CSS yang baru saja kita tulis.
Tetapi perhatikan bahwa tautan ke stylesheet direferensikan di $base/$css—di mana variabel $base dan $css itu, mendapatkan nilainya? Dan kalau dipikir-pikir, variabel-variabel itu $mytitle dan $mytext di akhir kode? Kami membutuhkan pengontrol baru!

3.6 MERANCANG CONTROLLER YANG LEBIH BAIK

Sekarang, kita membutuhkan pengontrol baru. Kami akan menyebutnya Mulai dan simpan sebagai

```
/system/ application/controllers/start.php.
```

Kontroler ini harus melakukan beberapa hal:

- Panggil tampilan
- Berikan tampilan dengan URL dasar dan lokasi file css yang baru saja kita tulis
- Berikan tampilan dengan beberapa data: mengharapkan judul ($mytitle) dan beberapa teks ($mytext)
- Terakhir, kami ingin pengontrol menerima parameter dari pengguna (yaitu melalui permintaan URL)

Dengan kata lain, kita harus mengisi variabel dalam tampilan. Jadi mari kita mulai dengan kita Mulai pengontrol. Ini adalah kelas OO:

```
<?php
class Start extends Controller {var
            $base;
    var $css;
```

Perhatikan bahwa di sini kita telah mendeklarasikan $base (alamat root web), dan $css (nama file css) sebagai variabel atau properti kelas. Ini menyelamatkan kita dari keharusan mendeklarasikan ulang jika kita menulis lebih dari satu fungsi di setiap kelas. Tetapi Anda dapat mendefinisikan dan menggunakannya sebagai variabel lokal dalam satu fungsi, jika Anda mau. Fungsi konstruktor sekarang mendefinisikan properti yang telah kita deklarasikan, dengan mencarinya di file konfigurasi. Untuk melakukan ini, kami menggunakan sintaks:

```
$this->config->item('name_of_config_variable');
```

seperti dalam:

```
function Start()
{
parent::Controller();
$this->base = $this->config->item('base_url');
$this->css = $this->config->item('css');
}
```

dan CI memulihkan apa pun yang kami masukkan dalam file konfigurasi dengan nama itu.
Dengan menggunakan sistem ini, bagaimanapun banyak pengontrol dan fungsi yang kita tulis, kita hanya perlu mengubah variabel fundamental ini sekali, bahkan jika situs kita menjadi sangat populer sehingga kita harus memindahkannya ke server yang lebih besar.

Mendapatkan Parameter ke Fungsi

Sekarang, di dalam kelas pengontrol Mulai, mari kita definisikan fungsi yang benar-benar akan melakukan pekerjaan.

```
function hello($name)
{
$data['css'] = $this->css;
$data['base'] = $this->base;
$data['mytitle'] = 'Welcome to this site';
$data['mytext'] = "Hello, $name, now we're getting dynamic!";
$this->load->view('testview', $data);
}
```

Fungsi ini mengharapkan parameter, $name, (tetapi Anda dapat menetapkan nilai default—myfunction($myvariable = 0)), yang digunakan untuk membangun string yang ditetapkan ke variabel $mytext. Nah, seperti yang baru saja kami tanyakan, dari mana asalnya?
Dalam hal ini, itu harus berasal dari permintaan URL, di mana itu akan menjadi parameter ketiga. Jadi, itu datang melalui permintaan HTTP:

```
GET /index.php/start/hello/fred HTTP/1.0
```

Atau dengan kata lain, saat Anda mengetikkan URL:
http://www.mysite.com/index.php/start/hello/fred
Perhatikan bahwa kode contoh ini tidak 'membersihkan' variabel yang diteruskan fred, atau memeriksanya dengan cara apa pun. Anda mungkin ingin melakukan ini dalam kode produksi.

Kita akan melihat cara memeriksa input formulir di Bab 7. Biasanya, variabel yang dilewatkan oleh hyperlink dengan cara ini dihasilkan oleh situs Anda sendiri. Pengguna jahat dapat dengan mudah menambahkan miliknya sendiri, hanya dengan mengirimkan URL seperti: http://www.mysite.com/index.php/start/hello/my_malicious_variable.
Jadi, Anda mungkin ingin memeriksa apakah variabel yang Anda terima berada dalam kisaran yang Anda harapkan sebelum menanganinya.

Segmen terakhir dari URL diteruskan ke fungsi sebagai parameter. Bahkan, Anda dapat menambahkan lebih banyak segmen parameter tambahan jika Anda mau, sesuai dengan batasan praktis yang diberlakukan oleh browser Anda.

Mari kita rekap tentang bagaimana CI menangani URL, karena kita telah membahas semuanya sekarang:

Tabel 3.2 Fungsi URL

Segmen URL	**Apa Fungsinya?**
http://www.mysite.com	URL dasar yang menemukan situs Anda.
/index.php	Menemukan router CI yang mengatur tentang membaca sisa URL dan memilih rute yang benar ke situs Anda.
/start	Nama pengontrol CI akan dipanggil. (Jika tidak ada nama yang ditetapkan, CI akan memanggil pengontrol default mana pun yang Anda tentukan.)
/hello	Nama fungsi yang akan dipanggil CI, di dalam pengontrol yang dipilih. (Jika tidak ada fungsi yang ditentukan, default ke fungsi indeks, kecuali Anda telah menggunakan _remap.)
/fred	CI meneruskan ini ke fungsi sebagai variabel.
Jika ada segmen URL lebih lanjut , e.g. /bert	CI meneruskan ini ke fungsi sebagai variabel kedua.
More variables	CI akan meneruskan segmen URL lebih lanjut sebagai variabel lebih lanjut.

Melewati Data ke View

Mari kembali ke fungsi hello:

```
function hello($name)
{
$data['css'] = $this->css;
$data['base'] = $this->base;
$data['mytitle'] = 'Welcome to this site';
$data['mytext'] = "Hello, $name, now we're getting dynamic!";
$this->load->view('testview', $data);
}
```

Perhatikan bagaimana fungsi hello() pertama kali membuat array yang disebut $data, mengambil campuran properti objek yang diatur oleh konstruktor dan teks. Kemudian memuat tampilan berdasarkan nama, dengan array yang baru saja dibuat sebagai parameter kedua.

Di balik layar, CI memanfaatkan fungsi PHP lain dengan baik: extract(). Ini mengambil setiap nilai dalam larik $data dan mengubahnya menjadi variabel baru dengan caranya sendiri—jadi larik $data yang baru saja kita definisikan diterima oleh tampilan sebagai rangkaian variabel terpisah: $teks (sama dengan " Halo, $name, sekarang kita mulai dinamis"), $css (sama dengan nilai dari file konfigurasi), dan seterusnya.

Dengan kata lain, ketika dibangun, array $data terlihat seperti ini:

```
Array (
[css] => mystyles.css
[base] => http://127.0.0.1/packt [mytitle] => Welcome
to this site
[mytext] => Hello, fred, now we're getting dynamic!
)
```

Namun dalam perjalanan ke tampilan, itu dibongkar, dan variabel berikut dibuat dalam tampilan agar sesuai dengan setiap pasangan nilai kunci dalam larik:

```
$css      = 'mystyles.css';
$base     = 'http://127.0.0.1/packt';
$mytitle = 'Welcome to this site';
$mytext   = 'Hello, fred, now we're getting dynamic!';
)
```

Meskipun Anda hanya dapat meneruskan satu variabel ke tampilan, Anda dapat mengemas banyak informasi ke dalam satu variabel tersebut. Setiap nilai dalam array $data itu sendiri dapat berupa array lain, dan seterusnya, sehingga Anda dapat meneruskan potongan informasi ke tampilan dengan cara yang terstruktur secara ketat.

Sekarang navigasikan ke http://127.0.0.1/packt/index.php/index/start/fred (perhatikan bahwa URL-nya berbeda—ini mencari fungsi start yang kami tulis di pengontrol indeks) dan Anda akan melihat hasil: halaman dinamis yang ditulis menggunakan arsitektur MVC. (Yah, setidaknya VC! Kami belum benar-benar menggunakan M.) Inilah yang seharusnya terlihat seperti sekarang:

Gambar 3.2 tampilan fungsi start pada hasil

Anda dapat melihat (setidaknya saya harap Anda bisa!) bahwa parameter fred adalah segmen terakhir dari URL. Itu telah diteruskan ke fungsi, dan kemudian melalui tampilan.

Harap diingat bahwa tampilan Anda harus ditulis secara paralel dengan pengontrol Anda. Jika tampilan tidak mengharapkan dan membuat tempat untuk variabel, itu tidak akan ditampilkan. Jika tampilan mengharapkan variabel untuk disetel dan ternyata tidak, Anda mungkin mendapatkan pesan kesalahan. (Tampilan Anda tentu saja dapat menerima variabel secara kondisional.)

Bagaimana Kelas CI Saling Menyampaikan Informasi dan Kontrol

Saat Anda menulis pengontrol, model, dll., Anda harus melewati kontrol dan data di antara mereka. Mari kita lihat beberapa cara di mana kita dapat melakukan ini.

Memanggil Views

Kami telah melihat bagaimana pengontrol memanggil tampilan dan meneruskan data ke sana:

Pertama, ia membuat larik data ($data) untuk diteruskan ke tampilan; kemudian memuat dan memanggil tampilan dalam ekspresi yang sama:

```
$this->load->view('testview', $data);
```

Memanggil Fungsi Secara Langsung

Jika Anda ingin menggunakan kode dari library, model, plug-in, atau helper, tentu Anda harus memuatnya terlebih dahulu, lalu memanggilnya seperti yang dijelaskan pada tabel sebelumnya. Jadi, jika 'tampilan' adalah model dan saya ingin menggunakan fungsi halaman utama, pengontrol saya mungkin memanggil:

```
$this->display->mainpage();
```

Jika fungsi membutuhkan parameter, kita dapat meneruskannya ke fungsi seperti ini:

```
$this->display->mainpage('parameter1', $parameter2);
```

Berinteraksi dengan Controller

Anda dapat memanggil perpustakaan, model, plug-in, atau pembantu dari dalam pengontrol apa pun, dan model dan perpustakaan juga dapat saling memanggil serta plug-in dan pembantu.

Namun, Anda tidak dapat memanggil satu pengontrol dari yang lain, atau memanggil pengontrol dari model atau pustaka. Hanya ada dua cara agar model atau pustaka dapat merujuk kembali ke pengontrol:

Pertama, dapat mengembalikan data. Jika controller memberikan nilai seperti ini:

```
$fred = $this->mymodel->myfunction();
```

dan fungsi diatur untuk mengembalikan nilai, maka nilai itu akan diteruskan ke variabel $fred di dalam pengontrol.

Kedua, model atau pustaka Anda dapat membuat (dan mengirim ke tampilan) URL, yang memungkinkan pengguna manusia memanggil fungsi pengontrol. Pengendali ada untuk menerima interaksi manusia. Anda tidak bisa, tentu saja, hyperlink langsung ke model atau perpustakaan. Pengguna selalu berbicara dengan pengontrol, tidak pernah ke hal lain—tetapi Anda dapat menulis fungsi panggilan di pengontrol. Dengan kata lain, tampilan Anda mungkin berisi hyperlink ke fungsi pengontrol:

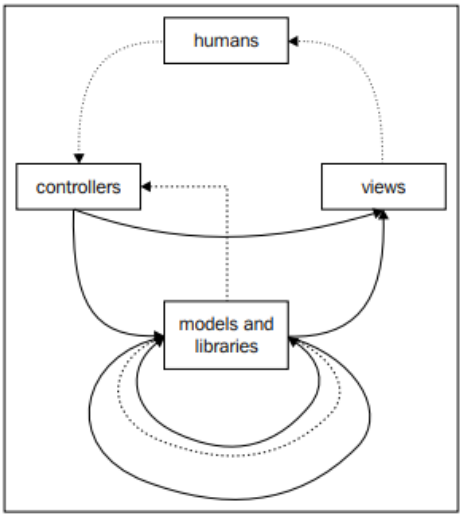
```
echo anchor(start/callmodel, Do something with a model);
```

tetapi fungsi callmodel akan keluar hanya untuk memanggil fungsi dalam model:

```
function callmodel()

{
$this->load->model(mymodel);
$this->mymodel->myfunction();
}
```

Ini Seperti Egg-Cup



Gambar 3.3 Diagram Komponen Egg-Cup

Diagram ini menunjukkan cara yang berbeda di mana komponen dapat mengatasi satu sama lain. Garis tak terputus mewakili panggilan fungsi langsung seperti:

```
$this->mymodel->myfunction();
```

Ini dapat terjadi dari pengontrol ke tampilan, dan dari pengontrol ke pustaka atau model. (Model juga dapat memanggil tampilan, tetapi mungkin tidak.) Mereka tidak dapat terjadi secara terbalik: tampilan dll. tidak dapat memanggil pengontrol. Namun, perpustakaan dan model dapat saling memanggil dan dipanggil satu sama lain. Garis putus-putus mewakili informasi yang lewat dengan mengembalikan nilai. Model dan pustaka dapat melakukan ini ke pengontrol, atau satu sama lain. Tampilan tidak mengembalikan nilai. Garis putus-putus

mewakili informasi atau kontrol yang lewat melalui pengguna manusia—dengan kata lain, tampilan akan menunjukkan sesuatu kepada pengguna di layar dan dapat mengundang pengguna untuk mengklik hyperlink (yang memicu pengontrol). Kemiripan apa pun dengan cangkir telur adalah murni kebetulan. Itu baru saja keluar seperti itu.

Contoh CI Helper: The URL Helper

Sebagai contoh anda dapat membagi kode Anda menjadi potongan-potongan yang rapi, terfokus, URL helper CI berisi serangkaian fungsi yang membantu Anda memanipulasi URL. Anda memuatnya seperti ini:

```
this->load->helper('url');
```

Dan kemudian, Anda dapat menggunakannya untuk menemukan dan mengembalikan situs dan/atau URL dasar yang Anda atur di file konfigurasi Anda:

```
echo site_url();
echo base_url();
```

Dan kemudian, Anda dapat menggunakannya untuk menemukan dan mengembalikan situs dan/atau URL dasar yang Anda atur di file konfigurasi Anda:

http://www.mysite.com/index.php/start/hello/fred

Jika Anda ingin kode Anda sendiri untuk membuat hyperlink ke URL seperti ini, Anda dapat menggunakan bantuan URL untuk melakukannya. Sintaksnya adalah:

```
echo anchor(start/hello/fred, Say hello to Fred);
```

Ini menghasilkan hyperlink ke URL yang sama, dan menampilkan kata-kata Say hello to Fred kepada pengguna yang akan diklik. Dengan kata lain, ini setara dengan:

```
<a href="http://www.mysite.com/index.php/start/hello/fred ">say helloto Fred</a>
```

Ingat, ada dua keuntungan menggunakan pembantu CI. Pertama, lebih sedikit mengetik (49 karakter sebagai lawan 82, keduanya termasuk spasi. Jika Anda menyertakan memuat pembantu URL—27 karakter lain, yang hanya perlu Anda lakukan sekali per pengontrol—masih menjadi 76 daripada 82.) Kedua, pembantu URL secara otomatis mencari URL situs di file konfigurasi (dan nama file indeks). Ini berarti bahwa jika Anda mengubah lokasi situs Anda, Anda hanya perlu mengubah file konfigurasi sekali, Anda tidak perlu menelusuri kode Anda untuk hyperlink yang tidak berfungsi lagi.

Pembantu URL memiliki fungsi berguna lainnya. Misalnya, ia dapat membuat hyperlink 'mailto'.

```
echo mailto('me@example.com', 'Click Here to Email Me');
```

memiliki efek yang sama dengan mengetik HTML ini:

```
<a href="mailto:me@example.com">click here to email me</a>
```

Jika Anda khawatir robot akan mengambil alamat email dari situs web Anda dan menggunakannya untuk spamming, ubah mailto dalam kode CI menjadi safe_mailto. Apa yang muncul di layar pemirsa Anda persis sama, dan bekerja dengan cara yang sama. Namun, jika Anda memeriksa kode HTML yang sebenarnya, ini sekarang telah menjadi tumpukan JavaScript yang kompleks, yang tidak dapat (dengan mudah) dibaca oleh robot:

```
<script type="text/javascript">
//<![CDATA[
var l=new Array();
l[0]='>';l[1]='a';l[2]='/
';l[3]='<';l[4]='|101';l[5]='|109';l[6]='|32';l[7]='|108';l[8]='|105'
;l[9]='|97';l[10]='|109';l[11]='|101';l[12]='|32';l[13]='|111';l[14]=
'|116';l[15]='|32';l[16]='|101';l[17]='|114';l[18]='|101';l[19]='|72'
;l[20]='|32';l[21]='|107';l[22]='|99';l[23]='|105';l[24]='|108';l[25]
='|67';l[26]='>';l[27]='"';l[28]='|109';l[29]='|111';l[30]='|99';l[31
]='|46';l[32]='|101';l[33]='|108';l[34]='|112';l[35]='|109';l[36]='|9
7';l[37]='|120';l[38]='|101';l[39]='|64';l[40]='|101';l[41]='|109';l
[42]=':';l[43]='o';l[44]='t';l[45]='l';l[46]='i';l[47]='a';l[48]='m'
;l[49]='"';l[50]='=';l[51]='f';l[52]='e';l[53]='r';l[54]='h';l[55]='  ';l[56]='a';l[57]='<';
for (var i = l.length-1; i >= 0; i=i-1){
if (l[i].substring(0, 1) == '|') document.write("&#"+unescape(l[i].substring(1))+";");
else document.write(unescape(l[i]));}
//]]>
</script>
```

Anda, dan pengguna Anda, tidak perlu melihat kode ini. Itu hanya untuk membingungkan robot dan menjaga alamat email Anda aman dari spam. Anda meletakkannya di sana dengan menambahkan empat huruf dan garis bawah: Anda menulis safe_mailto alih-alih mailto, dan CI melakukan sisanya.

Ada beberapa fungsi berguna lainnya di URL helper. Lihat Panduan Pengguna untuk mengetahui lebih lanjut tentang mereka dan cara menggunakannya.

Pertimbangkan saja pembantu URL secara keseluruhan. Mari kembali ke batu ujian untuk pengkodean yang telah kita bahas sebelumnya dalam bab ini:

- Kode ini memiliki 'singularitas komponen' yang tinggi. Itu melakukan berbagai hal terbatas, dan jelas apa itu.
- Ini adalah 'loosely coupled'—memiliki antarmuka yang sederhana dan tidak ada ketergantungan lain pada kode apa pun yang memanggilnya. Anda dapat menggunakan pembantu URL dalam proyek CI apa pun yang Anda tulis. Sebagian besar proyek Anda akan membutuhkan semacam hyperlink, misalnya. Anda dapat menggunakan helper ini berulang kali untuk membuatnya.

Jika Anda melihat kode URL helper (di system/application/helpers/ url_helper.php) Anda akan melihat bahwa itu adalah kode prosedural—yaitu, ini hanyalah sekumpulan fungsi,

bukan kelas OO. Itu tidak memuat kelas atau pembantu CI lainnya. (Tidak menjadi objek, tidak bisa melakukan ini.)

Contoh Pustaka Sederhana: Membuat Menu

Sekarang mari kita lihat beberapa kode yang menggunakan kelas CI.
Misalnya, berikut adalah file library sederhana yang membuat menu dengan tiga pilihan:

```
<?php
class Menu{
function show_menu()
{
$obj =& get_instance();
$obj->load->helper('url');
$menu  = anchor("start/hello/fred","Say hello to Fred |");
$menu  .= anchor("start/hello/bert","Say hello to Bert |");
$menu  .= anchor("start/another_function","Do something else |");
return $menu;
}
}
?>
```

(Untuk saat ini, jangan khawatir tentang sintaksis yang tidak biasa — $obj-> $this-> in line 6. Ini diusir dalam Bab 7.)

Perhatikan bahwa kode ini sekarang menjadi kode OO, di mana fungsi show_menu () terkandung dalam satu kelas, 'menu'. Ini dapat mengakses kelas dan pembantu CI lainnya: dalam hal ini menggunakan helper URL, yang baru saja kami periksa.

Pertama, memuat helper URL, dan kemudian membuat string ($ menu), yang terdiri dari kode HTML untuk hyperlink ke tiga pengontrol dan fungsi yang ditentukan untuk itu. Kemudian mengembalikan string menu $.

Anda mungkin menyebutnya dari pengontrol seperti ini:

```
$mymenu  = $this->menu->show_menu();
```

Dan kemudian pengontrol dapat menggunakan variabel $ menu untuk memanggil tampilan:

```
$data['menu'] = $mymenu;
$this->load->view('myview', $data);
```

Kelas ini menghasilkan menu, yang spesifik situs. Untuk alasan ini, saya akan menyimpannya di /system/application/libraries, daripada sistem/pustaka, folder. Ini tidak secara longgar digabungkan seperti penolong URL, yang dapat saya gunakan di situs mana pun. Memang memiliki singularitas tinggi: itu menciptakan menu, dan hanya itu yang terjadi. Saya dapat memanggilnya dari pengontrol mana pun di situs saya dan tahu bahwa itu akan menampilkan menu standar dalam model saya.

3.7 RINGKASAN

Kerangka kerja MVC adalah cara yang banyak digunakan dan sangat efektif untuk mengatur situs web yang kompleks. CI menggunakannya untuk membantu Anda memilah kode Anda sendiri, tetapi juga cukup fleksibel tentang bagaimana hal itu. Cobalah untuk diingat dua prinsip 'kopling longgar' dan 'singularitas komponen' saat Anda menulis kode Anda sendiri. Jangan terlalu khawatir apakah aplikasi Anda sesuai dengan teori ketat MVC atau tidak. Yang penting adalah memahami apa berbagai jenis file, dan bagaimana mereka saling berhubungan.

Kemudian, Anda dapat memutuskan apakah akan menulis kode Anda sendiri di perpustakaan atau file model, atau sebagai pembantu atau plug-in. Kami telah melihat struktur file CI, dan melihat bagaimana Anda bisa, jika Anda mau, memeriksa semua kode CI, tetapi (untungnya!) Anda tidak perlu melakukannya.

Kami mengotak -atik salah satu file asli: file konfigurasi, yang menyimpan informasi situs penting di satu tempat untuk memudahkan kami untuk meningkatkan atau mengubah nanti. Kami telah melihat struktur objek dasar pengontrol, dan menggunakan konstruktor sederhana untuk mendapatkan beberapa data dari file konfigurasi kami dan memasukkannya ke properti kelas. Dan kami telah secara dinamis memberikan informasi dari pengontrol baru yang kami tulis, ke tampilan baru. Sejauh ini, hal utama yang telah dilakukan

CI bagi kami adalah mendorong kami untuk menggunakan struktur dasar saat kami mulai mendefinisikan situs kami. Saat kita melanjutkan, akan menjadi jelas betapa pentingnya struktur itu.

Juga, kami melihat cara komponen CI melewati data dan kontrol di antara mereka. Sangat berguna untuk memahami ini ketika Anda mulai menulis kode Anda. Terakhir, kami melihat Helper URL CI sendiri sebagai contoh yang baik dari sepotong kode, dan kami menulis kelas perpustakaan 'menu' menu 'menu kami sendiri.

BAB 4
MENGGUNAKAN CI UNTUK MENYEDERHANAKAN DATABASE

Anda melihat CI karena Anda ingin membuat pengkodean lebih mudah dan lebih produktif. Bab ini adalah tentang kelas rekaman aktif CI. Jika CI menawarkan tidak lebih dari kelas rekaman aktifnya, itu masih akan bernilai setiap sen dari harga pembelian. Baiklah, gratis. Saya akan mengulanginya - masih akan menjadi alat utama untuk meningkat produktivitas Anda.

Catatan aktif memungkinkan Anda untuk menangani basis data dengan keributan minimum dan kejelasan maksimum. Mudah digunakan dan dipelihara. Bab ini melihat bagaimana Anda mengatur database untuk bekerja dengan CI, dan kemudian bagaimana Anda menggunakan kelas rekaman aktif untuk memanipulasi database. Anda akan melihat:

- Bagaimana kode catatan aktif dibandingkan dengan kode antarmuka 'klasik' php mysql
- Cara menulis pertanyaan 'baca', dan tampilkan hasilnya
- Cara membuat, memperbarui, dan menghapus kueri

CI memungkinkan Anda untuk menulis pertanyaan dalam gaya PHP 'klasik' tradisional juga, tetapi saya tidak akan membahasnya. Ini sepenuhnya tercakup dalam panduan pengguna online. Saya mulai melakukannya dengan cara lama, tetapi begitu saya mencoba catatan aktif, saya tidak pernah melihat ke belakang.

4.1 PENGATURAN KONFIGURASI

Anda mungkin memperhatikan bahwa sebagian besar bab dalam buku ini terus kembali ke folder Sistem/Aplikasi/Konfigurasi dan file konfigurasi di dalamnya. Ini sangat penting untuk mengendalikan cara kerja CI. Dan sementara Anda dapat meninggalkan sebagian besar dari mereka dengan aman diatur pada default, file konfigurasi basis data memang perlu diubah sebelum apa pun akan berfungsi sama sekali.

Pada dasarnya, Anda hanya perlu mengatakan di mana database Anda berada, dan jenis apa itu. File default hanya mengatakan:

```
$active_group = "default";
$db['default']['hostname'] = "";
$db['default']['username'] = "";
$db['default']['password'] = "";
$db['default']['database'] = "";
$db['default']['dbdriver'] = "";
```

bersama dengan beberapa opsi lain yang dapat Anda tinggalkan di default, untuk saat ini. Opsi yang harus Anda isi adalah:

- nama host: lokasi basis data Anda, mis., 'Localhost' atau alamat IP
- Nama pengguna dan kata sandi: Nama pengguna dan kata sandi pengguna basis data dengan izin yang cukup untuk melakukan apa pun yang Anda ingin situs Anda lakukan. Ini

bukan (biasanya) nama pengguna dan kata sandi yang sama dengan situs Anda atau panel kontrol ISP Anda.

- database: Nama basis data Anda, mis., 'Situs web'
- dbdriver: Jenis database yang Anda gunakan - pada saat penulisan, opsi yang ditawarkan CI adalah MySQL, MySQLI, Postgre SQL, ODBC, dan MS SQL.

Dalam pengalaman saya, salah satu hal paling sulit untuk diatur di situs CI baru dapat menjadi tautan ke database. Anda mungkin perlu berkonsultasi dengan ISP Anda jika ragu - kadang - kadang basis data mereka berjalan di alamat yang berbeda ke server web mereka. Jika Anda menggunakan MySQL, mereka mungkin menawarkan phpMyadmin, yang biasanya memberi tahu Anda nama host - ini mungkin 'localhost' atau mungkin alamat IP.

Anda akan mencatat bahwa bagian dari file konfigurasi ini sebenarnya adalah array multi-dimensi. Dalam $ db adalah array yang disebut default, dan Anda menambahkan pasangan kunci Variabel seperti hostname = 127.0.0.1 ke array itu. Ini agar Anda dapat mengatur database lain, sebagai array sekunder lainnya, dan bertukar di antara mereka dengan mudah dengan hanya mengubah $active_groupPengaturan ke nama array lain. Hal ini memungkinkan untuk menjalankan situs dengan beberapa opsi basis data - misalnya, database uji dan database produksi - dan untuk bertukar di antara mereka dengan mudah. Atau Anda mungkin perlu menggambar informasi dari dua database terpisah.

4.2 MERANCANG BASIS DATA UNTUK SITUS

Saya ingin menunjukkan bahwa CI dapat digunakan untuk mengembangkan situs web yang serius dengan tujuan yang serius. Saat ini saya sedang menjalankan beberapa situs web untuk klien, dan saya ingin program yang akan memantau mereka, mengujinya dengan cara yang saya tentukan, menyimpan database tentang apa yang telah dilakukan, dan izinkan saya memiliki laporan ketika saya menginginkannya. Jadi mari kita coba membangunnya. Pertama mari kita atur beberapa tujuan. Ini adalah:

1. Untuk mengelola satu atau lebih situs web jarak jauh dengan intervensi manusia minimum
2. Untuk menjalankan tes reguler di situs jarak jauh
3. Untuk menghasilkan laporan tentang permintaan, memberikan rincian situs dan tes yang dilakukan

Jadi, hal pertama yang kita butuhkan adalah database situs web untuk diperiksa. Siapkan database Disebut situs web di MySQL atau RDBM apa pun yang Anda gunakan.

Sekarang, kita perlu menambahkan beberapa tabel untuk menampung berbagai jenis data. Mari kita tambahkan ke database situs web kami tabel untuk situs, yang mencakup bidang untuk URL mereka, nama dan kata sandi mereka - nama pengguna, dan jenisnya. Kami juga akan menyertakan bidang ID untuk setiap situs-dan setidaknya di MySQL, yang dapat diatur untuk menghasilkan ID baru yang unik untuk setiap entri, menggunakan tipe bidang Increment Auto.

Setiap situs dapat di -host dengan host yang berbeda, atau mesin host, dan kami membutuhkan tabel host lain untuk menyimpan data tentang ini. Kemungkinan besar memiliki domain yang terkait dengannya, jadi kami membutuhkan tabel domain untuk

melacak data tentang domain, seperti ketika itu akan diperbarui, pendaftar, dan nama pengguna kami di situs pendaftar.

Maka tentu saja, kami memiliki orang -orang yang melelahkan, klien, beberapa di antaranya mungkin memiliki lebih dari satu situs, jadi kami memerlukan tabel orang terpisah untuk menyimpan nama mereka, alamat email, alamat surat siput, nomor ponsel, ditambah nama hewan peliharaan, Dan semua hal lain yang sangat vital untuk CRM yang baik.

Jadi tabel situs kami perlu memasukkan bidang untuk ID domain, ID host, dan mungkin beberapa orang ID, satu untuk pemilik situs atau klien dan satu untuk manajer situs. (Itu Anda, atau salah satu staf yang harus Anda pekerjakan untuk mengimbangi ketika aplikasi ini menyentuh pasar.)

Seperti yang Anda lihat, ini adalah database relasional penuh, dan kami baru saja memulai! (Rincian fuller dari database ini ditetapkan sebagai lampiran bab ini, dalam bentuk kueri MySQL, jika Anda ingin mengaturnya sendiri.) Kami akan menginginkan cara fleksibel sederhana untuk mengakses semua ini. Jadi, mari kita beralih ke apa yang dapat ditawarkan CI, dan khususnya ke kelas rekaman aktifnya.

4.3 ACTIVE RECORD

'Rekaman Aktif' adalah 'pola desain' - yang lain dari sistem yang sangat abstrak seperti MVC, yang menyediakan templat untuk menyelesaikan masalah pengkodean umum dan juga menghasilkan beberapa buku paling membosankan di planet ini. Dalam dirinya sendiri, itu bukan kode, hanya pola kode. Ada beberapa interpretasi yang berbeda tentang itu. Pada intinya adalah penciptaan hubungan antara database Anda dan objek, setiap kali Anda melakukan pertanyaan.

Biasanya, setiap tabel adalah kelas, dan setiap baris menjadi objek. Semua hal yang mungkin ingin Anda lakukan dengan baris tabel - buat, membacanya, memperbaruinya, atau menghapusnya, misalnya - 'metode', yang diwarisi oleh objek itu dari kelasnya. Ruby on Rails dibangun di sekitar pola catatan aktif, dan begitu pula CI - meskipun implementasi yang tepat dalam dua kerangka kerja tampaknya memiliki perbedaan yang halus. Teori yang cukup - apa artinya? Nah, pernyataan kode sederhana dan jelas, jika Anda tidak keberatan memasukkan panah di dalamnya.

Keuntungan Menggunakan Class Active Record

Rekaman aktif menghemat waktu Anda, membawa fungsi otomatis yang tidak perlu Anda pikirkan, dan membuat pernyataan SQL mudah dipahami.

Menghemat waktu

Saat Anda menulis kueri basis data normal di PHP, Anda harus menulis koneksi ke database setiap kali. Dengan CI, Anda terhubung sekali ke database, dengan meletakkan baris berikut dalam fungsi konstruktor masing -masing pengontrol atau model:

```
$this->load->database();
```

Setelah Anda melakukan ini, Anda tidak perlu mengulangi koneksi, berapa banyak pertanyaan yang kemudian Anda buat dalam pengontrol atau model itu. Anda mengatur detail database

dalam file konfigurasi seperti yang kita lihat sebelumnya di bab ini. Sekali lagi, ini memudahkan untuk memperbarui situs Anda, jika Anda pernah mengubah nama database, kata sandi, atau lokasi.

Fungsi Otomatis

Setelah Anda terhubung ke database, sintaks catatan aktif CI membawa kode tersembunyi dengannya. Misalnya, jika Anda memasukkan kueri 'Sisipkan' berikut:

```
$data = array
        (
        'title' => $title,'name' => $name, 'date' => $date
        );
$this->db->insert('mytable', $data);
```

Nilai yang Anda masukkan telah melarikan diri di belakang layar dengan kode ini:

```
function escape($str)
{
switch (gettype($str))
{case 'string':
$str = "'".$this->escape_str($str)."'";break;
case 'boolean':         $str = ($str === FALSE) ? 0 : 1;break;
default         :       $str = ($str === NULL) ? 'NULL' : $str;
break;
}
return $str;
}
```

Dengan kata lain, kerangka kerja CI membuat kode Anda lebih kuat. Sekarang, mari kita lihat cara kerjanya. Pertama, menghubungkan ke database sangat sederhana. Dalam PHP klasik, Anda mungkin mengatakan sesuatu seperti ini:

```
$connection = mysql_connect("localhost","fred","12345");
mysql_select_db("websites", $connection);
$result = mysql_query ("SELECT * FROM sites", $connection);while ($row =
mysql_fetch_array($result, MYSQL_NUM))
{
foreach ($row as $attribute)print
"{$attribute[1]} ";
}
```

Dengan kata lain, Anda harus menyatakan kembali host, nama pengguna, dan kata sandi, membuat koneksi, lalu pilih database dari koneksi itu. Anda harus melakukan ini setiap kali. Hanya dengan begitu, Anda melanjutkan ke kueri yang sebenarnya. CI menggantikan hal -hal koneksi dengan satu baris:

```
$this->load->database();
```

Yang Anda masukkan sekali, di setiap pengontrol atau model atau konstruktor kelas yang Anda tulis. Setelah itu, dalam setiap fungsi dalam pengontrol itu, dll., Anda langsung masuk ke dalam kueri Anda. Informasi koneksi disimpan dalam file konfigurasi database Anda, dan CI pergi dan mencarinya di sana setiap kali. Jadi, di setiap fungsi CI, Anda langsung ke pertanyaan Anda. Kueri di atas yang ditulis dalam CI keluar sebagai:

```
$query = $this->db->get('sites'); foreach ($query-
>result() as $row)
{
print $row->url
}
```

Sederhana, bukan?
Sisa bab ini menetapkan cara membuat pertanyaan yang berbeda, membuatnya lebih spesifik.

4.4 MEMBACA QUERIES

Kueri paling umum yang akan kami tulis hanya mengambil informasi dari database sesuai dengan kriteria kami. Instruksi dasar untuk melakukan kueri baca adalah:

```
$query = $this->db->get('sites');
```

Ini adalah kueri 'Pilih *' di tabel situs - dengan kata lain, ia mengambil semua bidang. Jika Anda lebih suka menentukan tabel target (situs) di baris yang terpisah, Anda dapat melakukannya dengan cara ini:

```
$this->db->from('sites');
$query = $this->db->get();
```

Jika Anda ingin 'memilih' atau membatasi jumlah bidang yang diambil, daripada mendapatkan semuanya, gunakan instruksi ini:

```
$this->db->select('url','name','clientid');
$query = $this->db->get('sites');
```

Anda mungkin ingin menyajikan hasil dalam urutan tertentu-katakan dengan nama situs-dalam hal ini Anda masukkan (sebelum `$this->db->get` line):

```
$this->db->orderby("name", "desc");
```

Desc berarti dalam urutan menurun. Anda juga dapat memilih ASC (Ascending) atau Rand (Random). Anda mungkin juga ingin membatasi jumlah hasil yang ditampilkan permintaan Anda; Katakanlah Anda hanya menginginkan lima hasil pertama. Dalam hal ini masukkan:

```
$this->db->limit(5);
```

Tentu saja, dalam sebagian besar pertanyaan, Anda tidak akan meminta setiap catatan dalam tabel. Kekuatan database tergantung pada kemampuan mereka untuk memilih - untuk memilih satu bagian data yang Anda inginkan dari tumpukan barang yang tidak Anda lakukan. Ini biasanya dilakukan oleh pernyataan di mana yang diungkapkan CI dengan cara ini:

```
$this->db->where('clientid', '1');
```

Pernyataan ini akan menemukan semua situs web yang ditautkan ke klien yang IDnya adalah 1. Tapi itu tidak banyak membantu kami. Kami tidak ingin mengingat semua ID di tabel orang kami. Sebagai manusia, kami lebih suka mengingat nama manusia. Jadi kita perlu menautkan di tabel orang:

```
$this->db->from('sites');
$this->db->join('people', 'sites.peopleid = people.id');
```

Untuk setiap orang ID di tabel situs, lihat informasi terhadap ID itu di tabel orang juga.
Perhatikan Konvensi SQL bahwa jika nama bidang mungkin ambigu antara dua tabel, Anda merujuknya dengan nama tabel terlebih dahulu, maka suatu periode, maka nama bidang. Jadi situs. Peopleid berarti ladang orang di tabel situs. Faktanya, tidak ada bidang yang disebut PeopleId di kedua tabel, tetapi ada bidang ID di kedua situs dan orang, sehingga RDBM akan memprotes jika Anda mencoba menjalankan kueri tanpa menyelesaikan ambiguitas untuk itu. Bagaimanapun, itu adalah kebiasaan yang baik untuk membuat makna Anda eksplisit, dan sintaks CI dengan senang hati menerima nama yang lebih lengkap. Anda dapat bermain -main dengan sintaks di mana pernyataan. Misalnya, tambahkan operator negasi:

```
$this->db->where('url !=','www.mysite.com' );
```

atau operator perbandingan:

```
$this->db->where('id >','3' );
```

atau menggabungkan pernyataan ("WHERE… AND…"):

```
$this->db->where('url !=','www.mysite.com');
$this->db->where('id >', '3');
```

or use $this->db->orwhere()untuk mencari alternatif ("WHERE … OR"):

```
$this->db->where('url !=','www.mysite.com' );
$this->db->orwhere('url !=','www.anothersite.com' );
```

Jadi katakanlah kita telah membangun kueri seperti ini:

```
$this->db->select('url','name','clientid','people.surname AS client');
$this->db->where('clientid', '3');
$this->db->limit(5);
```

```
$this->db->from('sites');
$this->db->join('people', 'sites.clientid = people.id');
$this->db->orderby("name", "desc");
$query = $this->db->get();
```

Ini harus memberi kami lima situs web pertama (dipesan dengan nama) milik klien nomor 3, dan mengambil nama keluarga klien serta nomor ID -nya!

Mafaat tersembunyi dari menggunakan catatan aktif adalah bahwa data yang mungkin datang dari pengguna secara otomatis melarikan diri, jadi Anda tidak perlu khawatir tentang menempatkan kutipan di sekitarnya. Ini berlaku untuk fungsi seperti $ this-> db-> where (), dan juga untuk pernyataan pembuatan data dan pembaruan yang dijelaskan di bagian berikutnya. (PERINGATAN KEAMANAN: Ini bukan hal yang sama dengan mencegah serangan silang-karena Anda memerlukan fungsi XSS_CLEAN () CI. Ini juga tidak sama dengan memvalidasi data Anda-untuk ini Anda memerlukan kelas validasi CI. Lihat Bab 5.)

Menampilkan Hasil Query

Menampilkan hasil kueri database dalam CI cukup sederhana. Kami mendefinisikan kueri kami seperti di atas,

Berakhir di:

```
$query = $this->db->get();
```

Kemudian, jika ada beberapa hasil, mereka dapat dikembalikan sebagai objek $ ROW yang melaluinya Anda mengulangi dengan loop foreach:

```
foreach ($query->result()  as $row)
{
print $row->url;  print $row-
>name;  print $row->client;
}
```

atau jika kita hanya menginginkan satu hasil, itu dapat dikembalikan sebaga i objek, atau di sini sebagai $ row array:

```
if ($query->num_rows()  > 0)
{
$row = $query->row_array();

print $row['url']; print
$row['name']; print $row['client'];
}
```

Secara pribadi, saya lebih suka sintaks objek daripada array - kurang mengetik!

Saat Anda mengikuti pola MVC, Anda biasanya ingin menyimpan pertanyaan dan interaksi database dalam model, dan menampilkan informasi melalui tampilan.

4.5 MEMBUAT DAN UPDATE QUERY

Active Record memiliki tiga fungsi yang membantu Anda membuat entri baru di database Anda. Mereka adalah $this->db->insert(), $this->db->update(), and $this->db->set(). Perbedaan antara kueri 'buat' dan 'pembaruan' adalah bahwa ketika Anda membuat catatan baru, tidak ada referensi untuk catatan yang ada, Anda menulis yang baru. Saat Anda memperbarui, ada catatan yang ada, dan Anda mengubahnya. Jadi dalam kasus kedua, Anda harus menentukan catatan mana yang Anda ubah. Dalam kedua kasus, Anda harus mengatur nilai yang ingin Anda tinggalkan dalam database setelah kueri Anda. Nilai yang tidak Anda tetapkan akan dibiarkan tidak berubah - atau, jika tidak ada sebelumnya, mereka masih akan 'nol' setelah kueri Anda.

CI memungkinkan Anda untuk mengatur nilai-nilai Anda dengan array, atau dengan $this->db-set(); Perbedaannya hanya satu dari sintaksis. Jadi, mari kita tambahkan baris ke tabel situs kami di database situs web. Kami sudah terhubung ke database ini di pengontrol kami. Fungsi konstruktor pengontrol termasuk garis:

```
$this->load->database();
```

Kami ingin menambahkan situs baru, yang memiliki URL, nama, jenis, dan nomor ID klien. Sebagai array, ini mungkin:

```
$data = array(
'url' => 'www.mynewclient.com','name' => 'BigCo Inc',
'clientid' => '33',
'type' => 'dynamic'
);
```

Untuk menambahkannya ke tabel situs, kami mengikutinya dengan:

```
$this->db->insert('sites', $data);
```

Atau, kami dapat mengatur setiap nilai menggunakan $ this-> db-> set ():

```
$this->db->set('url', 'www.mynewclinet.com');
$this->db->set('name', 'BigCo Inc');
$this->db->set('clientid', '33');
$this->db->set('type', 'dynamic');
$this->db->insert('sites');
```

Jika kami memperbarui catatan yang ada, sekali lagi kami dapat membuat array, atau menggunakan $this->db->set(). Tetapi ada dua perbedaan.

Pertama, kami harus menentukan catatan yang ingin kami perbarui; dan kedua, kita perlu menggunakan $this->db->update() Jika saya ingin memperbarui catatan (katakanlah catatan dengan bidang 'ID' yang diatur ke 1) di tabel Situs saya, menggunakan data yang ditetapkan dalam array $ data saya di atas, sintaksnya adalah:

```
$this->db->where('id', '1');
$this->db->update('sites', $data);
```

Atau saya dapat mengatur informasi menggunakan $this->db->set(), seperti di atas.

CI memberi Anda beberapa fungsi untuk memeriksa apa yang telah dilakukan database. Paling berguna:

```
$this->db->affected_rows();
```

harus mengembalikan '1' setelah pernyataan insert atau pembaruan saya - tetapi mungkin menunjukkan lebih banyak baris jika saya mengubah beberapa baris data sekaligus. Anda dapat menggunakannya untuk memeriksa bahwa operasi telah melakukan apa yang Anda harapkan.

Anda perhatikan bahwa saya tidak mengatur bidang ID ketika saya membuat catatan baru. Itu karena kami mengatur database untuk mengisi bidang ID secara otomatis ketika catatan baru ditambahkan. Tetapi saya harus menentukan ID ketika saya memperbarui catatan yang ada, jika tidak database tidak tahu mana yang akan diubah.

Namun, jika saya membuat catatan baru, saya tidak tahu nomor ID sampai saya membuatnya. Jika saya perlu merujuk ke catatan baru, saya bisa mendapatkan nomor ID baru dengan:

```
$new_id_number  = $this->db->insert_id();
```

(Kode ini harus pergi, sesegera mungkin, setelah operasi yang menghasilkan catatan, atau mungkin memberikan hasil yang menyesatkan.)

Untuk sedikit lebih banyak kedamaian pikiran, ingatlah bahwa CI Active Record Function, termasuk $this->db->insert()dan $this->db->update() secara otomatis melarikan diri dari data yang diteruskan kepada mereka sebagai input.

Dari versi 1.5, CI juga mencakup dukungan untuk transaksi - menghubungkan dua atau lebih tindakan basis data bersama -sama sehingga semuanya berhasil, atau semua gagal. Ini sangat penting dalam aplikasi pemeliharaan pembukuan entri ganda dan banyak situs komersial. Contohnya. Katakanlah Anda menjual tiket teater. Anda mencatat menerima pembayaran dalam satu transaksi, dan kemudian mengalokasikan kursi kepada pelanggan di yang lain. Jika sistem Anda gagal setelah melakukan operasi basis data pertama, tetapi sebelum melakukan yang kedua, Anda mungkin berakhir dengan pelanggan yang marah - yang telah ditagih, tetapi belum memiliki kursi yang dipesan. CI sekarang membuatnya lebih sederhana untuk menghubungkan dua atau lebih operasi basis data ke dalam satu transaksi, sehingga jika semuanya berhasil, transaksi 'berkomitmen', dan jika satu atau lebih gagal, transaksi 'digulung kembali'. Kami tidak perlu menggunakan ini di situs contoh kami, tetapi jika Anda ingin informasi lebih lanjut, lihat Panduan Pengguna Online CI.

4.6 HAPUS QUERY

Hapus kueri mungkin yang paling sederhana untuk dijelaskan. Yang Anda butuhkan hanyalah nama tabel dan nomor ID catatan yang akan dihapus. Katakanlah saya ingin menghapus catatan di tabel situs saya dengan nomor ID 2:

```
$this->db->where('id', '2');
$this->db->delete('sites');
```

Saya sedikit gugup dengan pertanyaan 'hapus' karena mereka sangat kuat. Harap ingat untuk memastikan bahwa ada nilai yang valid dalam klausa 'di mana', atau Anda dapat menghapus seluruh tabel Anda! Baik penulis maupun Packt Publishing tidak akan bertanggung jawab jika….

Mencampur Active Record dan Gaya 'Klasik'

CI tidak memaksa Anda untuk menggunakan Rekaman Aktif. Anda juga dapat menggunakan CI untuk mengeluarkan kueri SQL langsung. Misalnya, dengan asumsi Anda memuat database di konstruktor Anda, Anda masih dapat menulis kueri seperti ini:

```
$this->db->query("SELECT id, name, url FROM sites WHERE 'type' ='dynamic'");
```

Secara pribadi, saya menemukan Rekaman Aktif lebih mudah digunakan. Secara konseptual, menetapkan kueri saya dalam array membuatnya lebih mudah untuk dilihat dan dimanipulasi sebagai entitas daripada menulisnya dalam sintaks SQL. Ini sedikit lebih bertele-tele, tetapi terstruktur dengan jelas; itu secara otomatis lolos data; dan mungkin lebih portabel. Ini juga meminimalkan kesalahan pengetikan dengan koma dan tanda kutip.

Namun, ada beberapa kasus, di mana Anda mungkin harus menggunakan SQL asli. Anda mungkin ingin melakukan penggabungan kompleks, atau contoh lain adalah jika Anda perlu menggunakan beberapa kondisi 'di mana'. Jika Anda ingin menemukan situs web yang terkait dengan klien 3, tetapi hanya situs web dari dua jenis tertentu, Anda mungkin perlu memberi tanda kurung di sekitar SQL untuk memastikan kueri diinterpretasikan dengan benar.

Dalam kasus seperti ini, Anda selalu dapat menulis SQL sebagai string, memasukkannya ke dalam variabel, dan menggunakan variabel dalam fungsi `$this->db->where()` CI, sebagai berikut:

```
$condition = "client ='3' AND (type ='dynamic' OR type='static')";
$this->db->where($condition);.
```

Tanpa tanda kurung, ini ambigu. Maksud kamu:

```
(client='3' AND type = 'dynamic') OR type = 'static'
```

atau

```
client='3' AND (type = 'dynamic' OR type = 'static')
```

Ya, tentu saja, itu jelas, tetapi mesin biasanya salah menebak. Kebetulan, hati-hati dengan sintaks $condition. Kueri SQL yang sebenarnya adalah:

```
client='3' AND (type = 'dynamic' OR type = 'static'
```

Kutipan ganda berasal dari penugasan variabel:

```
$condition = "    ";
```

Sangat mudah untuk membuat tanda kutip tunggal dan ganda Anda bingung.
Beberapa ekspresi CI yang saya kutip di atas, seperti $this->db->affected_rows(), bukan merupakan bagian dari model Rekaman Aktifnya. Tapi mereka bisa dicampur dengan mudah.

Satu-satunya saat Anda mungkin mengalami masalah adalah jika Anda mencoba menggabungkan Rekaman Aktif dan SQL langsung dalam kueri yang sama. (Saya belum mencoba ini. Jika Anda memiliki banyak waktu, Anda dapat mengujinya, tetapi terus terang, saya pikir itu akan menunjukkan gaya hidup yang menyedihkan.
Cobalah melihat kereta sebagai gantinya. Setidaknya itu membuat Anda keluar ke udara segar! Saya menggunakan CI karena saya terlalu sibuk untuk tidak melakukannya!)

4.7 Ringkasan

Kami telah melihat kelas Rekaman Aktif CI dan melihat betapa mudahnya untuk:

- Atur koneksi ke satu atau lebih database
- Apakah SQL standar membaca, memperbarui, membuat, dan menghapus kueri
- Melakukan fungsi lain yang kita perlukan, untuk menggunakan database dengan benar

Fungsi Rekaman Aktif CI bersih dan mudah digunakan, dan membuat pengkodean lebih jelas untuk dibaca. Ini mengotomatiskan koneksi database, memungkinkan Anda untuk mengabstraksi informasi koneksi ke satu file konfigurasi. Itu dapat melakukan dengan cukup baik apa pun yang dapat Anda lakukan dengan SQL 'klasik'—lebih dari yang saya miliki ruang untuk menjelaskan di sini. Lihat Panduan Pengguna online untuk perincian lebih lengkap.

BAB 5
MENYEDERHANAKAN HALAMAN DAN FORMULIR HTML

Bab ini membahas cara lain di mana CI membantu menghemat waktu Anda dan membuat pengkodean Anda lebih ketat dan logis.

Pertama, kami akan membahas berbagai cara membangun tampilan—halaman yang mengontrol cara Anda melihat hasil yang disiapkan oleh pengontrol dan model Anda. Selanjutnya, Anda akan melihat cara membuat formulir HTML dengan cepat, dan dengan perlindungan bawaan; dan Anda juga akan melihat cara memvalidasi formulir Anda. Saya berasumsi bahwa pembaca buku ini sudah familiar dengan HTML dan CSS. Contoh berikut sangat disederhanakan, jadi kita bisa fokus pada kode CI. Dan saya berasumsi bahwa kami telah menulis file CSS dan menyimpannya di suatu tempat di situs kami.

5.1 MENULIS VIEW

Tampilan mengontrol bagaimana pengguna melihat situs web Anda. Mereka memudahkan Anda untuk menyajikan antarmuka yang konsisten, dan untuk mengubahnya jika perlu. Salah satu keuntungan dari MVC adalah Anda memisahkan presentasi dari logika, menjaga semuanya jauh lebih bersih. Sejauh ini, semua yang telah kita lakukan adalah melihat tampilan 'Selamat Datang' yang sangat sederhana yang dipasang di luar kotak saat Anda pertama kali memuat CI. (Lihat Bab 3.) Sekarang mari kita lihat bagaimana membuatnya lebih rumit. Tampilan hanyalah sekumpulan rak HTML untuk menampung konten Anda. Rak mungkin satu warna atau lain; mungkin ada banyak anak kecil, atau hanya beberapa yang elegan. Tapi tampilan tidak tahu atau peduli data apa yang ada di rak itu. Seperti politisi tertentu, hanya tertarik pada presentasi.

Untuk membuat tampilan, pertama-tama Anda perlu membuat halaman web HTML kerangka sebagai file PHP. Sebut saja basic_view.php. Simpan di folder application/views Anda. (Alasan untuk menyimpannya di folder ini hanya karena file loader mencarinya di sana.)

```
<html>
<head>
</head>
<body>
<p>Hello world!</p>
</body>
</html>
```

Kemudian Anda tinggal memuatnya dari pengontrol ketika Anda ingin menggunakannya, menggunakan $this->load ->view() di dalam fungsi yang sesuai:

```
function index()
{
        $this->load->view('basic_view');
}
```

Perhatikan bahwa jika ini adalah model atau pembantu, Anda akan memuatnya terlebih dahulu, lalu memanggilnya secara terpisah saat Anda ingin menggunakannya. Dengan tampilan, memanggilnya juga memuatnya, jadi Anda hanya perlu satu baris kode.

Tentu saja, itu pandangan kosong. Untuk membuatnya berguna, kita membutuhkan konten. Katakan kita ingin tambahkan judul dan beberapa teks. Pertama kita mendefinisikannya di controller:

```
function index()
{
$data['mytitle']        = "A website monitoring tool";
$data['mytext']         = "This website helps you to keep track of the
                           other websites you control.";
}
```

Perhatikan bagaimana kita mendefinisikannya bukan sebagai variabel skalar terpisah, tetapi sebagai elemen array $data (atau nama lain yang ingin kita berikan.). Untuk entri array pertama, 'kunci' adalah 'mytitle' dan 'nilai' adalah "Alat pemantauan situs web".
Selanjutnya, kita memanggil fungsi pemuatan tampilan:

```
function index()
    {
    $data['mytitle']        = "A website monitoring tool";
    $data['mytext']         = "This website helps you to keep track of the
                               other websites you control.";
    $this->load->view('basic_view',  $data);
    }
```

Kami telah membuat array $data menjadi parameter kedua dari $this->load->view() fungsi, setelah nama tampilan itu sendiri. Sekali $dataarray mencapai tampilan, CI menggunakan fungsi extract() PHP untuk mengubah masing-masing elemen array $data menjadi variabel terpisah, dengan 'kunci' sebagai nama variabel, dan 'nilai' sebagai nilai variabel. Variabel-variabel ini kemudian dapat direferensikan secara langsung dalam pandangan kami:

```
<html>
<head>
</head>
<body>
   <h1 class='test'><?php  echo $mytitle; ?> </h1>
   <p class='test'><?php  echo $mytext;  ?> </p>
</body>
</html>
```

Anda hanya dapat meneruskan satu variabel data ke tampilan, tetapi dengan membangun larik larik, Anda dapat mengemas sejumlah besar informasi dengan rapi ke dalam satu variabel tersebut. Tampaknya rumit, tetapi sebenarnya merupakan cara yang terstruktur dan elegan untuk menyampaikan informasi.

5.2 SINTAKS PHP PANJANG DAN PENDEK

Sebelum kita melanjutkan, catatan tentang berbagai bentuk sintaks PHP.
Cara normal untuk memasukkan 'pulau kode' PHP di tengah kode HTML adalah seperti ini:

```
<?php echo $somevariable?>
```

Namun, jika Anda tidak menyukai ini, CI juga mendukung versi yang lebih pendek:

```
<?=$somevariable?>
```

Dalam hal ini, tanda kurung luar yang membatasi pulau kode telah kehilangan huruf PHP (hanya <? ?>); dan echo telah diganti dengan =. Anda juga dapat menggunakan sintaks yang lebih pendek untuk loop if, for, foreach, dan while. Instruksi lengkap ada di Panduan Pengguna online.

Secara pribadi, saya lebih suka menggunakan format standar karena saya sudah terbiasa. Jika Anda menggunakan format pendek, perhatikan bahwa beberapa server tidak akan menafsirkan format yang disingkat benar. Jika Anda masih ingin menggunakan tag pendek, buka file konfigurasi Anda, dan ubah garis:

```
$config['rewrite_short_tags'] = FALSE;
```

CI kemudian akan menulis ulang tag pendek ke bentuk normal sebelum mengirimkannya ke server. Namun, jika ada kesalahan PHP, menggunakan fungsi penulisan ulang ini membuat pesan kesalahan menjadi kurang bermakna, sehingga debugging mungkin lebih sulit. Seperti yang saya katakan, saya lebih memilih untuk tetap menggunakan format standar.

Sebagai catatan, CI juga memiliki kelas 'template parser', yang memungkinkan Anda untuk menempatkan variabel dalam kode HTML Anda tanpa kode PHP yang sebenarnya sama sekali. Saya belum membahas ini. Dia sangat berguna jika Anda bekerja dengan desainer HTML yang mungkin bingung dengan kode PHP. Rincian kelas ini tersedia di Panduan Pengguna.

5.3 VIEW BERSARANG

Sejauh ini, apakah kita menggunakan format PHP panjang atau pendek, ini adalah HTML yang cukup kasar. Akan lebih baik, misalnya, untuk meletakkan lebih banyak informasi di bagian <head> halaman. Lebih baik lagi jika ini bisa menjadi potongan standar setiap halaman. Sekali lagi, sesuatu yang hanya perlu kita tulis (atau ubah) sekali, dan kemudian dapat digunakan kembali, menyarangkan tampilan ini di dalam tampilan lain kapan pun kita membutuhkan hal-hal header HTML yang membosankan.

Mari buat 'tampilan' header halaman untuk situs kita, yang menampilkan header halaman standar, serta deklarasi HTML dan informasi meta.
Pertama, kami mengetikkan kode untuk tampilan header 'bersarang' kami:

```
<!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Strict//EN'http://www.
w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'><html xmlns='http://www.
w3.org/1999/xhtml'>
<title><?php echo $mywebtitle ?></title>
<base href= <?php echo "$base"; ?> >
<?php echo $myrobots ?>
<link rel="stylesheet" type="text/css" href="<?php echo "$base/
$css;?>">
```

Simpan ini sebagai views/header_view. Ini memperkenalkan variabel baru:

- $mywebtitle, yang merupakan judul halaman (tag meta; ini tidak akan muncul di layar, tetapi mesin pencari akan membacanya. Ini mungkin berbeda dari halaman ke halaman, jadi saya membuatnya menjadi variabel.)
- $myrobots, yang saya gunakan untuk instruksi standar kepada robots bahwa situs ini tidak boleh diindeks.
- $base and $css, yang menjelaskan URL dasar dan URL tambahan untuk kami .css, stylesheet yang kita anggap sudah kita tulis, yang memungkinkan kita menerapkan pemformatan secara konsisten. Variabel-variabel ini akan dihasilkan dari data yang telah kita simpan di file CI.config. (Saya juga bisa menggunakan variabel konfigurasi CI site_url alih-alih basis.)

Yang perlu kita ketahui sekarang adalah:

- Bagaimana kita menyebut tampilan 'bersarang' kedua?
- Bagaimana kita memberikan nilai pada variabelnya?

Ada dua pilihan. Pertama, panggilan ke tampilan dapat dilakukan dari dalam tampilan lain. Jadi tampilan utama kami, basic_view, hanya membutuhkan baris baru:

```
<html><head>
<?php $this->load->view('header_view'); ?>
</head><body>
<?php echo $mytitle; ?>
<?php echo $mytext; ?>
</body>
</html>
```

Adapun variabel, mereka dapat diberikan oleh dua baris baru di controller asli:

```
function index()
    {
    $data['mytitle']          = "A website monitoring tool";
    $data['mytext']           = "This website helps you to keep track
                                 of the other websites you control.";
    $data['myrobots']         = '<meta name="robots" content="noindex
                                 ,nofollow">';
    $data['mywebtitle']  = 'Web monitoring tool';
    $data['base']             = $this->config->item('base_url');
    $data['css']              = $this->config->item('css');
    $this->load->view('basic_view', $data);
  }
```

Di sini variabel baru $myrobots, $css, $base, dan $mywebtitle dibuat sebagai elemen baru dari array $data yang ada, diteruskan ke basic_view, yang membongkarnya, dan kemudian tersedia ke header_view saat ini dipanggil oleh basic_view. (Ingatlah untuk tidak menggunakan nama variabel yang sama dalam dua tampilan yang Anda susun, atau yang satu akan menimpa yang lain.)
Cara kedua adalah menambahkan tampilan dari dalam pengontrol, dengan menetapkannya ke variabel:

```
function index()
    {
    $data['mytitle']            = "A website monitoring tool";
    $data['mytext']             = "This website helps you to keep track of the
                                                    other websites you control.";
    $data['myrobots']           = '<meta name="robots" content="noindex, nofollow">';
    $data['mywebtitle'] = 'Web monitoring tool';
    $data['base']               = $this->config->item('base_url');
    $data['css']                = $this->config->item('css');
    $data['header']             = $this->load->view('header_view',   '', TRUE);
    $this->load->view('basic_view', $data);
```

Ini mungkin lebih benar dari perspektif MVC yang ketat.
Sebenarnya ada tiga parameter yang dapat Anda lewati dengan fungsi load->view.

- Yang pertama, header_view, adalah nama tampilan yang akan dimuat. Ini penting.
- Yang kedua, yang bersifat opsional, adalah data yang akan dimuat ke dalamnya.
- Yang ketiga adalah nilai Boolean.

Jika Anda tidak menentukannya, defaultnya adalah FALSE, dan tampilan dikirim ke browser. Namun, jika Anda menyarangkan tampilan dengan cara ini, Anda ingin tampilan tersebut dikembalikan sebagai string untuk bersarang di dalam variabel yang Anda teruskan ke tampilan host. Mengatur parameter ketiga ke TRUE mencapai ini.

Sekarang kami memiliki referensi ke stylesheet bawaan, kami dapat memperbarui tampilan untuk digunakan menampilkan kelas yang mungkin telah kita definisikan di sana:

```
<html><head>
<?php $this->load->view('header_view');  ?>
</head><body>
   <h1 class='test'><?php  echo $mytitle; ?> </h1>
   <p class='test'><?php  echo $mytext;  ?> </p>
</body>
</html>
```

Perhatikan lagi bagaimana sistem MVC CI memungkinkan Anda untuk memisahkan tampilan dari konten. Tampilan hanya menyediakan 'rak' untuk konten, dan bahkan gaya rak tersebut berasal dari lembar gaya .css. Tampilan tidak peduli apa yang dikatakan $mytext, itu hanya menampilkannya di rak kanan dengan format yang tepat. Pengontrol yang mendefinisikan $mytext bahkan tidak tahu (atau peduli) bagaimana informasi yang dihasilkannya ditampilkan.

Jadi, jika kita perlu mengubah tampilan halaman kita, atau menampilkannya di sistem yang berbeda (misalnya WAP), maka kita hanya perlu mengubah satu tampilan dan satu stylesheet CSS. Kita tidak perlu dipusingkan dengan kode beberapa controller.

Dan jika kita ingin mengubah informasi yang ditampilkan pada halaman, kita tidak perlu menyentuh tampilan, dan mengingatkan diri kita sendiri untuk mengubah beberapa variabel di setiap halaman yang telah kita tulis. Kami hanya mengubah apa yang didorong oleh pengontrol.

Ingat prinsip 'kopling longgar'? Sekali lagi, ini memudahkan untuk mendesain, meningkatkan, dan memelihara situs Anda.

5.4 MASALAH PRAKTIS ARSITEKTUR SITUS

Tunggu sebentar, Anda mengatakan di header_view kami, kami membuat alamat stylesheet CSS secara dinamis:

```
<link rel="stylesheet" type="text/css" href="<?php echo "$base/$css";?>">
```

Ini berarti bahwa pengontrol harus menghasilkan data ini, yang hanya relevan dengan bagaimana informasi ditampilkan, dan kami baru saja mengatakan bahwa pengontrol tidak boleh tahu atau peduli tentang itu. Bukankah itu bertentangan dengan prinsip 'kopel longgar' yang baru saja kita buat? Terlebih lagi, menghasilkan informasi ini secara dinamis memerlukan beberapa operasi: Pertama, pengontrol harus mencarinya di file konfigurasi, kemudian pengontrol harus mengemasnya dalam array $data dan meneruskannya ke tampilan, lalu tampilan harus mengekstrak variabel tunggal $base dan $css dan cari nilainya.

Sepertinya cara memutar dalam melakukan sesuatu. Mengapa tidak menyematkan data secara statis saja dalam tampilan?

```
<link rel="stylesheet" type="text/css" href="http://www.mysite.com/mystylesheet.css";">
```

Keuntungan membangun variabel ini secara dinamis, meskipun melanggar 'aturan' MVC, dan terlepas dari biaya pembuatan variabel dan meneruskannya, adalah bahwa kode tersebut kemudian jauh lebih portabel. Jika Anda memindahkan situs, atau memindahkan file CSS Anda, Anda hanya perlu mengubah kode sekali dalam file konfigurasi, dan setiap pengontrol dan tampilan akan mencerminkan perubahan sekaligus. Jika Anda melakukan hard-code alamat ke setiap tampilan, Anda harus meluangkan waktu untuk mencari semua URI absolut yang Anda tulis beberapa bulan lalu. Jadi mana yang terbaik?

Tidak ada jawaban yang benar. Itu tergantung pada apa prioritas Anda. Kuncinya adalah menerapkan prinsip-prinsip MVC dengan bijaksana—sebagai alat dan bukan pengekang. CI memberi Anda banyak kebebasan untuk melakukan ini.

Opsi ketiga—yang saya gunakan—adalah membuat model 'tampilan' khusus. Ini ada untuk membangun halaman standar. Itu menempatkan di header dan referensi file CSS dan seterusnya, dan menerima sebagai parameter informasi unik yang dibutuhkan file. Saya akan menunjukkan ini nanti di bab ini.

5.5 CI'S FORM HELPER: MEMASUKKAN DATA

Mari kita beralih ke cara Anda menggunakan halaman HTML Anda. Salah satu bagian terpenting dari setiap situs dinamis adalah interaksi dengan pengguna, dan ini biasanya berarti bentuk HTML. Bentuk penulisannya berulang-ulang dan membosankan. Pembantu formulir CI adalah bagian kode yang sangat berguna. Ini memperkenalkan sintaks yang sedikit berbeda, yang membuat pembuatan formulir lebih mudah. Mari kita buat formulir untuk memungkinkan diri kita memasukkan data di situs kita tentang situs web baru. Di tabel situs, kami ingin memasukkan nama, jenis, dan URL situs web, dan tanggal pembaruannya.

Anda dapat membuat formulir sebagai file HTML sederhana, atau Anda dapat membuatnya di dalam pengontrol, mengemasnya ke dalam variabel, lalu memanggil tampilan, dan meneruskan variabel ke tampilan. Saya melakukan yang kedua. Pertama, kita harus memuat form helper ke controller di mana kita perlu menggunakannya. Kemudian, kami menempatkan baris berikut dalam fungsi konstruktor pengontrol:

```
$this->load->helper('form');
```

dan tentu saja, kita harus memulai formulir.

Sekarang, untuk bidang formulir, alih-alih mengetik:

```
$variable .= <input type='text' name='name' value=''>
```

CI memungkinkan Anda memasukkan:

```
$variable .= form_input('name', '');
```

(Ingat bahwa 'nama' adalah judul bidang, 'nilai' adalah apa yang ingin Anda masukkan ke dalamnya. Menempatkan sesuatu di sini memberi Anda nilai default, atau Anda dapat secara dinamis mengambil nilai yang ada dari tabel.)
Hmm, katamu. 33 karakter bukannya 48, tidak banyak penghematan, terutama karena saya harus memuat pembantu terlebih dahulu (28 karakter lainnya). Kenapa mengganggu? Sehat:

Keuntungan Form Helper Satu: Kejelasan

Keuntungan pertama menggunakan penolong formulir CI adalah kejelasan dalam kode Anda. Jika kamu menginginkan kotak input yang lebih rumit, lalu dalam HTML Anda akan mengetik:

```
$variable = 'input type="text" name="url" id="url" value="www.mysite.com" maxlength="100"
size="50" style="yellow" />';
```

(Ingat jenis itu adalah jenis kotak yang Anda inginkan—teks, tersembunyi, apa pun.
name adalah nama yang akan digunakan sebagai kunci untuk nilai ini dalam larik POST.
id adalah agar kotak dapat direferensikan pada halaman, jika Anda melakukan hal-hal yang rapi dengan JavaScript.

value adalah nilai yang sudah ada atau default yang Anda inginkan agar kotak tersebut ditampilkan saat muncul di halaman.

Maxlength dan ukuran yang jelas; style dapat berupa kumpulan pemformatan HTML atau referensi ke gaya .css yang ditentukan di lembar gaya Anda.
CI menggunakan array sebagai gantinya:

```
$data = array(
        'name'   => 'url',
        'id'     => 'url',
        'value'  => 'www.mysite.com',
        'maxlength'     => '100',
        'size'   => '50',
        'style'  => 'yellow',
);
$variable = form_input($data);
```

Terlihat lebih panjang, dan sebenarnya: 145 karakter dibandingkan dengan 110 untuk HTML sederhana. Namun, jauh lebih jelas, dan lebih mudah untuk memahami dan memelihara. Ini menjadi lebih jelas jika Anda mulai menghasilkan beberapa nilai secara dinamis.

Bidang formulir tersembunyi sangat sederhana. Katakanlah kita ingin merekam secara otomatis
tanggal database kami diperbarui. Kami menempatkan tanggal dalam variabel $date, lalu:

```
form_hidden('updated', $date);
```

Jika Anda menginginkan kotak 'area teks', untuk memberi pengguna Anda lebih dari satu baris untuk memasukkan data, misalnya untuk URL, yang mungkin cukup panjang, gunakan fungsi form_textarea() CI. Jika Anda senang dengan ukuran default, ini hanya akan membaca:

```
$data = array(
        'name'          => 'url',
        'id'            => 'url',
        'value'         => 'www.mysite.com',
        );
$variable = form_textarea($data);
```

Pembantu formulir CI sangat membantu ketika Anda menulis dropdown dan kotak centang atau kotak radio. Katakanlah kita ingin mengubah bidang 'url' kita menjadi kotak drop-down, untuk memungkinkan pembaca memilih satu URL dari daftar beberapa. Pertama, buat daftar opsi dalam array, lalu gunakan fungsi form_dropdown() :

```
 $urlarray  = array(
                        '1'   => 'www.this.com',
                        '2'   => 'www.that.com',
                        '3'   => 'www.theother.com',
```

```
                                    );
$variable = form_dropdown('url', $urlarray, '1');
```

Nilai pertama yang diteruskan ke formulir, url, adalah nama bidang di tabel situs yang ingin kami perbarui; yang kedua adalah array opsi, yang ketiga adalah kunci dari opsi yang ingin Anda tetapkan sebagai default. Dengan kata lain, jika pengguna menerima nilai default, Anda Array $_POST akan berisi nilai 'url => 1', tetapi pengguna Anda akan melihat opsi *'www.this.com'.*
Bandingkan ini dengan HTML Vanilla biasa, jika tidak, Anda harus menulis:

```
<select name="type">
<option value="1" selected>www.this.com</option>
<option value="2">www.that.com</option>
<option value="3" >www.theother.com</option>
</select>
```

Sekarang kode CI sebenarnya lebih pendek (128 karakter dibandingkan dengan 154), serta lebih mudah diikuti.

Jika Anda menyimpan daftar kemungkinan URL dalam tabel database terpisah (katakanlah itu disebut 'url'), maka membuat kotak tarik-turun dinamis itu mudah. Pertama-tama buat array dari semua nilai yang mungkin:

```
$urlarray  = array();
$this->db->select('id, url');
$query = $this->db->get('urls');if ($query->num_rows()
> 0)
{
foreach ($query->result()  as $row)
{
$urlarray[$row->id]  = $row->url;
}
}
```

lalu ulangi fungsi CI form_dropdown() yang kita gunakan sebelumnya:

```
echo form_dropdown('type', $urlarray, '1');
```

Hanya konten $urlarray yang berubah; baris kode ini tetap sama.

Jika Anda memperbarui entri daripada membuat yang baru, Anda tidak ingin menunjukkan nilai default kepada pengguna Anda. Anda ingin menunjukkan nilai yang sudah ada untuk entri itu. Anda seharusnya sudah mengetahui nomor id dari entri yang ingin Anda perbarui, jadi Anda perlu melakukan pencarian database dari file 'sites' terlebih dahulu. Pastikan Anda menggunakan nama variabel yang berbeda untuk kueri kedua dan variabel 'baris' kedua, atau mereka dapat menimpa set pertama yang Anda tulis:

```
$this->db->select('id, url, name');
$this->db->where('id','$id')
$sitequery = $this->db->get('sites');
$siterow = $sitequery->row();
```

Kemudian fungsi drop-down formulir CI Anda akan membaca:

```
echo form_dropdown('url', $urlarray, $siterow->url);
```

Tidak ada ruang di sini untuk menelusuri semua opsi yang ditawarkan oleh penolong formulir. Itu dapat menangani kotak centang, bidang tersembunyi, kotak radio, dan lainnya. Ini dijelaskan sepenuhnya dalam Panduan Pengguna online CI.

Keuntungan Form Helper Dua: Otomatisasi

Keuntungan kedua menggunakan penolong formulir untuk membuat formulir HTML Anda adalah ia mengotomatiskan beberapa hal yang seharusnya Anda skrip sendiri. Pertama, ia memotong HTML dan karakter seperti tanda kutip, yang dapat dimasukkan pengguna, dan menghindarinya untuk menghentikannya agar tidak merusak formulir. Kedua, itu mengotomatiskan tautan. Saat Anda membuka formulir, Anda harus menentukan halaman target, yang akan menerima data formulir dan memprosesnya. (Dalam CI, ini adalah fungsi di dalam pengontrol daripada halaman statis yang sebenarnya. Katakanlah fungsi pembaruan pengontrol situs web.) Jadi, jika Anda menggunakan kode HTML biasa, Anda akan menulis:

```
<form method="post" action="http:/www.mysite.com/index.php/websites/update" />
```

Padahal, jika Anda membuka formulir dengan CI, Anda hanya perlu menggunakan:

```
form_open(websites/update)
```

CI secara otomatis mengerjakan URL dasar dari nilai dalam file konfigurasi Anda dan mengarahkan formulir ke sana. Sekali lagi, jika Anda memindahkan situs Anda, Anda tidak akan menemukan bahwa formulir Anda rusak, dan harus mencari URL hard-code untuk diperbarui.

Perhatikan secara kebetulan, bahwa CI mengasumsikan formulir Anda akan selalu mengirim data POST daripada GET data. CI memanfaatkan URL itu sendiri secara ekstensif, jadi ini menghindari kebingungan.

5.6 MODEL TAMPILAN

Seperti yang dijanjikan (dan sedikit disederhanakan) inilah model tampilan saya:

```
<?php
class Display extends Model {

/*create the array to pass to the views*/
        var $data = array();
/*two other class variables*/
```

```
        var $base;
        var $status = '';
/*the constructor function: this calls the 'model' parent class, loads other CI libraries and
helpers it requires, and dynamically sets variables*/
        function Display()
        {
                parent::Model();
                $this->load->helper('form');
                $this->load->library('user_agent');
                $this->load->library('errors');
                $this->load->library('menu');
                $this->load->library('session');
/*now set the standard parts of the array*/
                $this->data['css'] = $this->config->item('css');
                $this->data['base'] = $this->config->item('base_url');
                $this->base       = $this->config->item('base_url');
                $this->data['myrobots'] = '<meta name="robots"
                content="noindex,nofollow">';
/*note that CI's session stuff doesn't automatically recall the extra variables you have added,
so you have to look up the user's status in the ci_sessions table*/
                $sessionid = $this->session->userdata('session_id');
                $this->db->select('status');
                $this->db->where('session_id', $sessionid);
                $query = $this->db->get('ci_sessions');
                        if ($query->num_rows()  > 0)
                                {
                                $row = $query->row();
                                $this->status = $row->status;
                                }
        }
/*function to assemble a standard page. Any controller can call this. Just supply as $mydata
an array, of key/value pairs for the contents you want the view to display. Available variables
in this view are: mytitle. menu, mytext, diagnostic
*/
        function mainpage($mydata)
                {
                $this->data['mytitle'] = 'Monitoring website';
                $this->data['diagnostic'] = $diagnostic; foreach($mydata as $key => $variable)
                {$this->data[$key] = $variable;}
/*here's the menu class we looked at in Chapter 3*/
                $fred = new menu;
                $this->load->library('session');
                $mysess = $this->session->userdata('session_id'); if(isset($this->status) &&
                $this->status > 0)
                {$this->data['menu']=
                $fred->show_menu($this->status);}
                $this->load->view('basic_view', $this->data);

                }

}
?>
```

Saya dapat memanggil halaman utama dari pengontrol apa pun dengan baris:

```
$this->load->model('display');
$this->display->mainpage($data);
```

dan saya kemudian tahu bahwa pandangan saya sedang dirakit secara dinamis, persis seperti yang saya inginkan.

5.7 CI'S VALIDATION KELAS: MEMERIKSA DATA DENGAN MUDAH

Salah satu tugas saat Anda menulis formulir HTML adalah memvalidasi input pengguna. Kita semua tahu bahwa kita harus melakukannya, tapi… Sejauh ini kita telah menulis sebuah formulir sederhana, yang dengan percaya diri akan menerima data apa pun yang dimasukkan pengguna. Anda harus selalu berasumsi bahwa sebagian kecil pengguna Anda jahat, dan yang lainnya bodoh. (Jangan beri tahu mereka.) Jika mereka bisa membuat kesalahan sederhana, mereka akan melakukannya. Memvalidasi formulir Anda menguji informasi yang mereka kirimkan, untuk memastikannya sesuai dengan aturan yang Anda tentukan.

Anda dapat melakukannya di sisi klien, menggunakan JavaScript tetapi ini memiliki nilai terbatas sebagai tindakan pencegahan keamanan, karena pengguna dapat dengan mudah menghindarinya. Memvalidasi data di sisi server berarti perjalanan pulang pergi ekstra ke server, tetapi itu sepadan untuk menambah ketenangan pikiran.

Menulis kodenya juga cukup rumit, tetapi—Anda dapat menebaknya—CI memiliki validasi kelas yang bekerja erat dengan bantuan formulir untuk mempermudah validasi.
Mari kita ubah proses pengiriman formulir kita sendiri untuk mencerminkan hal ini. Anda perlu membuat beberapa perubahan dalam bentuk, dan juga dalam fungsi yang ditunjuknya.

Jika formulir Anda dimulai form_open('sites/update'), fungsi yang perlu Anda ubah adalah fungsi 'update' di pengontrol 'sites'. Jika Anda tidak menggunakan pembantu formulir CI, padanan HTML adalah:

```
<form method="post" action="http:/www.mysite.com/index.php/sites/update" />
```

Anda perlu melakukan tiga hal:

1. Siapkan validasi
2. Siapkan pengontrol
3. Siapkan formulir

Atur Validasi

Dalam fungsi yang ditunjuk oleh formulir Anda, muat pustaka validasi dan nyatakan aturan validasi Anda:

```
$this->load->library('validation');
$rules['url']                   = "required";
$rules['name']                  = "required";
$this->validation->set_rules($rules);
```

Setiap entri harus memiliki sesuatu di bidang 'url' dan 'nama'. CI memberi kita berbagai opsi untuk menentukan sesuatu itu seharusnya, dan Panduan Pengguna menjelaskannya secara lengkap. Mereka cukup jelas: min_length[6] jelas berarti entri yang valid di bidang harus memiliki 6 karakter atau lebih, numerik berarti tidak boleh mengandung huruf, dll. Anda juga dapat menggabungkan aturan, menggunakan karakter 'pipa' untuk pisahkan mereka:

```
$rules['name'] = "required|alpha|max_length[12]";
```

akan membutuhkan nama yang terdiri dari 12 karakter alfabet atau kurang. Anda bahkan dapat menulis aturan Anda sendiri.

Siapkan Controller

Masih dalam fungsi yang sama, buat loop 'if else':

```
if ($this->validation->run() == FALSE)
{
        $this->load->view('myform');
}
        else
{
        $this->load->view('success');
}
```

Anda menjalankan tes validasi, dan jika entri tidak lulus tes, Anda kembali ke formulir entri. (Jika Anda membuat tampilan Anda dalam suatu fungsi di dalam pengontrol, katakan karena ia memiliki bidang tarik-turun dinamis, lalu arahkan kembali ke fungsi itu sebagai gantinya:

```
$this->myfunction  daripada $this->load->view('myform');
```

Jika proses validasi Anda berhasil, buat tampilan lain ("berhasil") untuk memberi tahu pengguna bahwa entri telah diterima, dan untuk memberikan opsi navigasi apa pun yang Anda inginkan agar dia melanjutkan.

Mengatur Formulir

Formulir entri yang telah kami buat juga perlu diubah. Cukup mengembalikan pengguna ke formulir setiap kali entri tidak lulus tes validasi akan membuat pengguna Anda benar-benar menyeberang! Anda harus mengatakan bidang mana yang gagal, dan mengapa. Jadi, Anda harus menggemakan baris tambahan di suatu tempat dalam bentuk:

```
$this->validation->error_string;
```

Ini mencetak pesan yang sesuai, dan menghemat banyak frustrasi pengguna.

Anda juga perlu mengatur agar bidang formulir yang dimasukkan dengan benar diisi ulang dengan nilai yang benar—jika tidak, pengguna harus memasukkan kembali semua bidang

setiap kali dia membuat kesalahan di salah satu bidang tersebut. Cara lain untuk membuatnya benar-benar marah!

Pertama, Anda harus kembali ke controller dan menambahkan beberapa kode lagi. Segera setelah aturan validasi yang Anda siapkan, buat larik dari setiap bidang yang ingin Anda isi ulang. Kunci array adalah nama bidang yang sebenarnya digunakan dalam tabel Anda; nilai array adalah apa yang Anda inginkan untuk memanggil bidang dalam pesan kesalahan:

```
$fields['url'] = 'The URL of your site';
```

Setelah itu, tambahkan baris:

```
$this->validation->set_fields($fields);
```

Sekarang Anda telah menyiapkan larik di pengontrol untuk menyimpan nilai yang ada, Anda hanya perlu menambahkan baris di formulir Anda untuk menggemakannya kembali ke pengguna. Untuk baris teks sederhana, ini akan menjadi:

```
<input type="text" name="url" value="<?php echo $this->validation ->url; ?>" />
```

atau, jika Anda menggunakan pembantu formulir CI:

```
$variable .= form_input('url', "$this->validation->url");
```

Jika ini adalah formulir entri baru, itu sudah cukup. Jika Anda menggunakan formulir untuk memperbarui entri yang ada, maka, saat pertama kali formulir muncul, nilainya harus apa pun yang telah disetel sebelumnya. (Ingat contoh kode di atas, tempat kami mencarinya dan menyebutnya $siterow->url?

Tetapi katakanlah Anda menggunakan formulir Anda untuk memperbarui entri yang ada, dan itu memantul kembali karena satu bidang tidak memvalidasi, Anda ingin nilai setiap bidang lain di formulir Anda menjadi apa pun yang telah Anda ubah. Jika tidak, Anda harus mengetik ulang seluruh formulir karena satu kesalahan validasi.

Ini mudah dicapai dengan loop 'salah satu'. Sesuatu seperti:

```
if(isset($_POST['url']))
        {$myvalue = $this->validation->url;}
else{$myvalue  = $siterow->url;}
```

Pertama kali formulir muncul, tidak akan ada apa pun di larik pos; jadi Anda mengambil nilai dari tabel yang mendasarinya. Tetapi setelah Anda mengirimkannya, larik posting akan diisi, jadi Anda mengambil nilai dari fungsi validasi.

Lihat Panduan Pengguna CI untuk beberapa hal lain yang dapat Anda lakukan dengan validasi formulir. Anda dapat menggunakannya untuk:

- Secara otomatis menyiapkan data Anda, misalnya dengan memangkas atau menghapus potensi serangan skrip lintas situs
- Tulis kriteria validasi kompleks Anda sendiri, misalnya, bahwa nilai yang dimasukkan pengguna tidak boleh sudah ada di database Anda
- Tulis pesan kesalahan Anda sendiri

Kelas validasi adalah bagian yang sangat berguna dan kuat dari CI dan sepadan dengan waktu yang dibutuhkan untuk memahaminya.

5.8 RINGKASAN

Kami telah melihat cara CI menghasilkan 'tampilan', dan bagaimana CI memungkinkan Anda membuat 'tampilan mini', yang dapat Anda sarangkan di dalam tampilan lain. Ini berarti Anda dapat mengatur atas halaman judul, atau bagian dari tampilan Anda, sekali, dan kemudian menggunakannya lagi dan lagi, menjaga tampilan Anda terpisah dari konten Anda. Kami juga telah melihat bagaimana CI membantu Anda melalui tugas menulis formulir HTML, dengan seperangkat pembantu yang menyederhanakan proses dan mengurangi pengkodean yang sebenarnya.

Terakhir, kita telah melihat kelas validasi CI, yang merupakan alat yang ampuh untuk mengawasi apa yang sebenarnya coba dimasukkan oleh pengguna Anda. Tidak ada yang sempurna, tetapi ini sangat membantu menghentikan formulir pengguna yang memasukkan sampah, atau mencoba mengeksploitasi lubang keamanan di situs Anda. Itu juga terlihat jauh lebih profesional ketika situs Anda dengan sopan tetapi tegas menangkap kesalahan pengguna, daripada diam-diam menerima entri yang tidak berarti.

Dalam perjalanan, kami juga melihat proses MVC lagi, dan membuat pilihan antara penerapan prinsip-prinsip MVC yang ketat, dengan sengaja melanggar 'aturan' ini untuk membuat hidup lebih mudah. CI memiliki filosofi yang sangat fleksibel: gunakan alat jika Anda mau, tetapi—asalkan Anda memahami masalahnya—jangan ragu untuk melakukannya dengan cara lain jika itu lebih sesuai dengan prioritas Anda.

BAB 6
MENYEDERHANAKAN SESI DAN KEAMANAN

Cukup teori! Sekarang mari kita mulai menulis aplikasi kita sendiri. Dalam bab ini, kami akan menguraikan aplikasi yang akan kami buat, dan kami akan melihat salah satu pertanyaan dasar yang memengaruhi situs web apa pun, yaitu manajemen sesi dan keamanan.
Dalam bab ini, kita akan melihat:

- Bagaimana membuat halaman Anda aman
- Cara menggunakan kelas sesi CI

6.1 MULAI MENDESAIN SITUS PRAKTIS DENGAN CI

Kami telah melihat halaman selamat datang CI dan melihat bagaimana itu dibangun oleh file pengontrol dan file tampilan. Itu setara dengan 'halo dunia'. Sekali waktu, situs hobi yang ditulis oleh amatir menggunakan kode sumber terbuka dan sering dianggap lebih rendah daripada situs 'perusahaan' besar yang ditulis oleh tim pemrogram menggunakan prosedur yang rumit. Pemandangan telah berubah. Perusahaan-perusahaan besar sekarang menggunakan teknologi open-source. Misalnya, NASA dan Associated Press menggunakan MySQL, US GAO dan bahkan Yahoo menggunakan PHP untuk aplikasi tertentu. Dan saya percaya bahwa pasar untuk aplikasi 'berukuran sedang' yang fleksibel sedang berkembang, karena perusahaan besar menyadari bahwa aplikasi lama mereka tidak dapat menangani tugas baru. Terkadang lebih mudah untuk membangun program kecil yang fleksibel daripada merevisi yang lama. CI menawarkan jembatan antara kode 'buatan sendiri' dan keandalan terstruktur dari situs 'perusahaan'. Ini memegang tangan Anda dan membantu Anda memprogram lebih baik dan menghasilkan hasil yang lebih konsisten dan andal.

Untuk mendemonstrasikan fleksibilitas CI, mari buat aplikasi kita sendiri.

Untuk menyatakan kembali persyaratan saya, saya ingin membangun sesuatu untuk tujuan tertentu. Saya menjalankan beberapa situs web, beberapa di antaranya untuk saya sendiri, dan beberapa untuk klien yang sangat menuntut. Saya perlu memelihara situs-situs ini, mengujinya, dan secara umum melacaknya. Sebagian besar, ini adalah hal-hal rutin. Saya dapat mempekerjakan seseorang untuk melakukannya untuk saya, tetapi akan lebih murah untuk menulis situs web untuk mengotomatiskan sebanyak mungkin proses.
Jadi persyaratan saya adalah:

1. Untuk mengelola satu atau lebih situs web jarak jauh dengan minimal campur tangan manusia
2. Untuk menjalankan tes reguler di situs jarak jauh
3. Untuk menghasilkan laporan sesuai permintaan, memberikan perincian situs dan pengujian yang dilakukan

Saya ingin dapat mengatur situs agar berjalan di Cron, jika ISP saya mengizinkannya; jika tidak, jalankan sendiri dua kali sehari atau sekali satu jam (sesuai keinginan), dan biarkan ia melakukan pola tes yang telah diatur sebelumnya. Saya tidak ingin mengetahui detailnya, kecuali ada yang tidak beres (maka saya ingin email yang memberi tahu saya apa yang sebenarnya terjadi dan di mana tepatnya), tetapi saya ingin dapat mencetak laporan manajemen agar klien saya terkesan dengan pemeriksaan rutin dan komprehensif yang saya lakukan, dan (semoga!) kinerja situs mereka yang sempurna.

Untuk menghindari pembuatan kode yang terlalu panjang dan pengulangan, kode dalam buku ini tidak terlalu aman, jadi harap diingat jika Anda menggunakannya secara nyata. Bab ini mencakup cara dasar untuk mengamankan halaman situs Anda dari pengguna yang tidak sah, tetapi masalah keamanan PHP lainnya, yang tidak hanya terjadi pada CodeIgniter, berada di luar cakupan buku ini.

Pada tahap ini, kita akan melihat pendekatan CI terhadap hal-hal yang umum untuk sebagian besar situs web dinamis. Jadi kami akan membiarkan detail desain situs kami sampai nanti. Mari kita mulai dengan beberapa item yang sangat mendasar.

6.2 BERGERAK DI SEKITAR SITUS

Setiap situs web adalah kumpulan program yang terpisah, dan sangat penting bahwa mereka dapat berbicara satu sama lain. Seperti yang kita lihat di Bab 3, CI menghubungkan mereka dengan URL mereka.

Biasanya, URL mengambil pola:

url dasar	http://www.situssaya.com. Ini adalah alamat vanilla biasa yang digunakan semua orang untuk mengakses situs Anda. Pembaca tidak perlu mengetahui semua struktur URL lainnya karena situs membangunnya sesuai kebutuhan.
file indeks	Segmen 1: index.php Ini adalah file utama yang CI mulai dengan setiap kali situs terkena.
kelas (atau pengontrol)	Segmen 2: start Jika tidak ada pengontrol yang disetel, CI default ke pengontrol yang Anda tentukan di file konfigurasi—lihat di bawah.
metode (atau fungsi)	Segmen 3: assessme. Jika tidak ada metode yang disetel, CI default ke fungsi indeks pengontrol, jika ada. Jika tidak, Anda mendapatkan halaman '404'.
ditambah parameter apa pun	Segmen 4: fred (dan Segmen 5: 12345, Segmen 6: halo, dll.)

Jadi, untuk memanggil metode penilaian di pengontrol awal dengan parameter fred dan 12345, URL Anda akan menjadi:

http://www.mysite.com.index.php/start/assessme/fred/12345.

Kode ini mengharapkan situs Anda berisi pengontrol yang disebut start.php yang menyertakan metode penilaian, yang mengharapkan dua parameter. URL seperti ini akan memanggil fungsi apa pun di pengontrol mana pun di situs Anda. Jadi ini ideal untuk halaman berbasis menu. Untuk contoh praktis tentang cara kerjanya, mari siapkan halaman pertama yang dilihat pengguna di situs kami. Kami akan menyiapkan pengontrol yang disebut start, dan menjadikannya pengontrol default kami.

Yah, pertama-tama, CI, seperti yang keluar dari kotak, diatur untuk pergi ke sambutan controller secara default, jadi saya perlu mengubah ini. Rute default CI diadakan di /system/application/config/routes file. Saat ini berbunyi:

```
$route['default_controller'] = "welcome";
```

Jadi saya akan mengubahnya menjadi:

```
$route['default_controller'] = "start";
```

(Ingat saja, dari tabel di atas, jika pengontrol default Anda tidak memiliki metode indeks default, Anda akan mendapatkan kesalahan 404 setiap kali ada yang masuk ke URL dasar vanilla biasa Anda, yang bukan ide yang baik!)

Sekarang saya perlu menulis pengontrol awal baru saya. Ingat format dasarnya:

```
<?php
  class   Start extends Controller
  {
          function Start()
                {
                                        parent::Controller();
                 }
                function assessme($name,  $password)
                {
:                       if($name == 'fred' && $password == '12345')
                                {$this->mainpage();}
               }
               }
    ?>
```

Simpan ini di /system/application/controllers folder as start.php. (Perhatikan kasus: Mulai memiliki huruf besar dalam nama kelas dan fungsi konstruktor, tetapi tidak dalam nama file yang disimpan.)

Baris kedua memberi tahu Anda bahwa ini adalah pengontrol. Kemudian fungsi konstruktor dimulai dan memuat metode kelas pengontrol induk. Fungsi assessmentme mengharapkan dua variabel $name dan $password. CI (dari versi 1.5 dan seterusnya) secara otomatis menetapkan setiap segmen URL setelah detik sebagai parameter, jadi fred dan 12345 akan menjadi parameter $name dan $password, masing-masing. Jadi, jika saya

mengetikkan URL di halaman sebelumnya, saya akan diarahkan kembali ke fungsi mainpage(). Kami akan mengatur ini nanti di pengontrol awal. (Jika tidak, maka kode tersebut akan mati begitu saja.)

Bagi mereka yang lebih terbiasa dengan PHP prosedural daripada kelas OO, harap perhatikan bahwa fungsi di dalam kelas harus dialamatkan sebagai $this->xxxx. Jadi, jika saya memanggil fungsi mainpage() dari pengontrol awal dari fungsi lain di dalam pengontrol awal, saya harus menyebutnya $this->mainpage(). Jika tidak, CI tidak akan dapat menemukannya.

Tentu saja, tidak mungkin ada orang yang mengetikkan URL seperti:

http://www.mysite.com.index.php/start/assessme/fred/12345.

Kebanyakan, mereka hanya akan masuk dan mengharapkan situs untuk memilah semua navigasi internal. Jadi mari kita mulai itu sekarang.

http://www.mysite.com

Seringkali, hal pertama yang Anda lihat di situs adalah formulir masuk. Jadi mari kita siapkan salah satunya. Pertama, saya menambahkan fungsi baru ke pengontrol awal saya. Saya ingin situs default ke fungsi ini, jadi saya akan menyebutnya index():

```
function index()
{
$data['mytitle']   = "My site";
$data['base'] = $this->config->item('base_url');
$data['css'] = $this->config->item('css');
$data['mytitle'] = "Situs web untuk memantau situs web lain";
$data['text'] = "Silakan login di sini!";
$this->load->view('halaman masuk', $data);
}
```

Ini memanggil view, entrypage. Tampilan termasuk formulir, dan formulir memungkinkan pengguna untuk mengirimkan kata sandi dan nama pengguna. Formulir HTML harus mengarah ke halaman yang akan menangani data dalam larik $_POST. Kami telah menulis fungsi di pengontrol awal kami untuk menerima ini: ini assessmentme(). Dalam HTML lama biasa, formulir pada tampilan kami harus dimulai:

```
<form method="post" action="http:/www.mysite.com/index.php/start/assessme" />
```

Saya sudah menjelaskan sedikit tentang fungsi assessment. Tidak ada gunanya fungsi yang hanya memiliki satu kombinasi nama pengguna-kata sandi. Saya perlu beberapa cara untuk mencarinya di database. Untuk membuat struktur lebih modular, saya akan menyerahkannya ke fungsi lain, Checkme().

Jadi, seperti yang akan Anda lihat, assessme() memanggil Checkme().

Checkme()melakukan semacam tes pada kata sandi dan nama pengguna (kami belum menulisnya) dan mengembalikan 'ya' atau 'tidak' ke assessme(). Jika ya, assessme()memanggil fungsi lain, mainpage(), yang mengembalikan tampilan. Perhatikan keuntungan dari pendekatan modular. Masing-masing fungsi memiliki peran. Jika saya perlu mengubah cara sistem memeriksa kata sandi, saya hanya perlu mengubah fungsi Checkme(). Jika saya perlu mengubah halaman yang ditampilkan pada respons yang benar, maka saya pergi ke fungsi mainpage() .

Mari kita lihat struktur kode dan cara bagian berinteraksi. (Perhatikan bahwa untuk membuat contoh lebih mudah diikuti, kami tidak 'membersihkan' input dari formulir kami di sini. Tentu saja, ini membuat kode Anda terbuka untuk masalah. Kelas formulir CI secara otomatis membersihkan data yang dimasukkan.

```
/*receives the username and password from the POST array
*/function assessme()
      {
          $username =          $_POST['username'];
          $password =          $_POST['password'];
/*calls the checkme function to see if the inputs are OK*/
         if($this->checkme($username, $password)=='yes')
         {
/*if the inputs are OK, calls the mainpage function*/
                  $this->mainpage();
         }
/*if they are not OK, goes back to the index function, whichre-presents the
log-in screen */
      else{
                  $this->index();
           }
         }
/*called with a u/n and p/w, this checks them against some list. Forthe moment, there's
just one option. Returns 'yes' or 'no'*/
      function checkme($username='', $password='')
      {
          if($username == 'fred' && $password == '12345')
              {return 'yes';

          else{return 'no';}
      }
```

Pada baris 5-6, assessme() menerima output dari form dari array $_POST. Ini akan berisi sesuatu seperti:

```
[username] => fred [password] => 12345
```

Fungsi assessmentme() meneruskan kedua variabel ini ke fungsi lain, checkme(). Ini hanya menguji apakah mereka fred dan 12345, masing-masing, dan jika ya, itu mengembalikan 'ya'. Jelas, di situs nyata ini akan lebih kompleks. Anda mungkin akan melakukan pencarian basis data untuk pasangan kata sandi nama pengguna yang valid. Menjadikannya sebagai fungsi

terpisah berarti saya dapat menguji sisa kode saya sekarang, dan meningkatkan fungsi checkme() nanti, di waktu luang saya.

Jika nama pengguna dan kata sandi adalah kombinasi yang valid, fungsi assessme() memanggil fungsi lain, mainpage(), yang memungkinkan saya masuk ke situs. Jika tidak, ia akan kembali menunjukkan kepada saya fungsi index()—yaitu, formulir log-in lagi. Masalah berikutnya yang kita miliki adalah bagaimana mengelola negara. Dengan kata lain, bagaimana mengenali pengguna yang masuk ketika dia membuat permintaan halaman lain.

Keamanan Menggunakan Kelas Perpustakaan CI Lain

Jika saya ingin membangun mekanisme sesi yang akan mencegah pengguna yang tidak diinginkan mengakses file saya, berapa banyak baris kode yang dibutuhkan?

Internet bekerja dengan serangkaian permintaan. Browser Anda membuat permintaan ke server saya untuk melihat halaman tertentu. Browser saya meneruskan halaman kembali ke server Anda. Anda melihatnya, dan mungkin perlu membuat permintaan lain, jadi Anda mengklik hyperlink, yang membuat permintaan ke server saya. Dan seterusnya.

Internet adalah 'stateless'—yaitu, setiap permintaan yang dibuat oleh browser Anda ke situs web saya diperlakukan sebagai peristiwa terpisah, dan protokol HTTP, yang mendasari Internet, tidak memiliki cara langsung untuk menautkan permintaan Anda ke permintaan lain (yang Anda mungkin telah membuat). Seolah-olah Anda berada di sebuah restoran, pelayan mengambil pesanan Anda, dan membawakan Anda makanan, tetapi kemudian melupakan semua tentang Anda. Tidak apa-apa, sampai dia perlu membawakan Anda tagihan, atau untuk mengingat bahwa Anda berhak mendapatkan diskon khusus, atau sekadar ingat bahwa Anda ingin dia menelepon Anda untuk meminta taksi setelah Anda selesai makan.

Jika Anda ingin situs web Anda menghubungkan satu permintaan halaman dengan permintaan halaman lainnya, Anda harus mengelola 'status' hubungan: entah bagaimana agar situs web mengetahui bahwa beberapa permintaan berasal dari browser yang sama, dan harus diperlakukan secara khusus.

PHP menawarkan dua cara mengelola status: menggunakan cookie, atau ID sesi yang dibuat secara khusus. Fungsi sesi PHP secara otomatis memeriksa apakah situs web menerima cookie; jika tidak, ia menggunakan metode ID sesi yang diteruskan melalui URL.

Cookie adalah untaian kecil data yang diteruskan situs web ke browser apa pun yang mengakses situs. Browser secara otomatis menyimpannya. Setelah cookie ada di sana, situs web dapat memeriksanya ketika browser berikutnya mencoba mengakses situs. Jika menemukan cookie yang tepat, ia dapat menggunakan informasi di dalamnya untuk mengkonfigurasi dirinya sendiri dengan tepat. Ini mungkin berarti menutup halaman tertentu untuk pengguna yang tidak sah, atau menambahkan informasi pribadi. Dalam analogi restoran kami, pelayan akan meninggalkan tagihan Anda di atas meja, dan lain kali dia melihat Anda, itu akan mengingatkannya bahwa Anda berhak mendapatkan diskon 15%, jadi dia bisa memperhitungkannya saat menghitung tagihan Anda.

Karena beberapa orang mengatur browser mereka untuk tidak menerima cookie, PHP menawarkan pendekatan alternatif. Setiap kali browser meminta akses, situs menghasilkan string acak yang disebut 'ID sesi', dan mengembalikannya ke browser. Browser kemudian menambahkan ini ke URL ketika mereka membuat permintaan berikutnya, sehingga situs

dapat mengenali browser. (Alih-alih pelayan meninggalkan tagihan di meja Anda, Anda membuatnya membawanya bolak-balik bersamanya ke dapur.)

CI memiliki kelas sesi yang menangani banyak hal yang sama. Bahkan, ini mengurangi banyak pengkodean menjadi satu baris. Kami melihat di bab terakhir bahwa CI memiliki berbagai 'kelas perpustakaan', yang menyederhanakan sebagian besar tugas umum yang ditangani situs web. Ini adalah inti dari kerangka kerja: potongan kode yang sangat abstrak yang telah ditulis sebelumnya, yang melakukan fungsi penting untuk Anda. Yang perlu Anda ketahui adalah di mana mereka berada, bagaimana mengatasinya dan menggunakan metode mereka, dan parameter apa yang mereka harapkan. Sebagai imbalannya, Anda dapat menggunakan kode profesional tanpa harus menulisnya!

Jika Anda ingin menggunakan fungsionalitas di dalam kelas dari dalam pengontrol atau model Anda, Anda harus ingat untuk terlebih dahulu memuat kelas ke dalam pengontrol atau model. (Beberapa kelas, seperti config selalu dimuat secara otomatis, itulah sebabnya kami belum memuatnya di salah satu kode kami sejauh ini.)

Anda memuat kelas perpustakaan secara sederhana:

```
$this->load->library('newclass');
```

Biasanya, letakkan baris ini di konstruktor pengontrol atau model Anda.

Jika Anda berpikir Anda akan menggunakan kelas perpustakaan di setiap pengontrol, Anda dapat memuatnya secara otomatis seperti halnya kelas konfigurasi. Buka file /system/application/ config/autoload, dan tambahkan nama kelas yang Anda inginkan ke dalam baris:

```
$autoload['libraries'] = array();
```

Sehingga tampilannya seperti ini:

```
$autoload['libraries'] = array('newclass','oldclass');
```

Kelas perpustakaan yang akan kita gunakan pertama kali adalah kelas sesi, yang membantu Anda mempertahankan status, dan mengidentifikasi pengguna. Ini cukup sederhana untuk melakukan ini. Inilah fungsi assessmentme() kami yang diperbesar dari pengontrol awal kami dengan baris baru yang disorot:

```
function assessme()
{
    $this->load->library('session');
    $username   =       $_POST['username'];
    $password   =       $_POST['password'];

 if($this->checkme($username,  $password)=='yes')
     {
                $this->mainpage();
     }
```

```
else{$this->index();}
}
```

(Saya telah memuat perpustakaan sesi di awal fungsi sehingga Anda dapat melihatnya, tetapi biasanya, saya akan memuatnya di konstruktor pengontrol, sehingga dimuat untuk semua fungsi lain di kelas ini.) Hanya memuat kelas sesi segera memberi Anda banyak fungsi sebagai ganti satu baris kode. Ini akan secara otomatis membaca, membuat, dan sesi pembaruan. Sejujurnya, itu bukan satu baris kode. Anda harus membuat beberapa perubahan pada file konfigurasi terlebih dahulu, untuk memberi tahu kelas sesi apa yang Anda inginkan.

Periksa file system/applications/config/config.php Anda, dan Anda akan menemukan bagian seperti ini:

```
| Session Variables
|-------------------------------------------------------------------
|
| 'session_cookie_name' = the name you want for the cookie
| 'encrypt_sess_cookie' = TRUE/FALSE (boolean).  Whether to encryptthe cookie
| 'session_expiration'  = the number of SECONDS you want the sessionto last.
|  by default sessions last 7200 seconds (two hours).  Set to zero forno expiration.
|
*/
$config['sess_cookie_name']             = 'ci_session';
$config['sess_expiration']              = 7200;
$config['sess_encrypt_cookie']          = FALSE;
$config['sess_use_database']            = FALSE;
$config['sess_table_name']              = 'ci_sessions';
$config['sess_match_ip']                = FALSE;
$config['sess_match_useragent']         = FALSE;
```

Untuk saat ini, pastikan sess_use_database disetel ke FALSE.

Sekarang, setiap kali pengguna Anda terhubung, situs akan menyimpan cookie di mesin Anda, yang berisi informasi berikut:

- ID Sesi unik yang dihasilkan oleh CI (jangan dikelirukan dengan string ID sesi PHP, yang tidak dibuat dalam contoh ini). Ini adalah string acak yang dibuat oleh CI untuk sesi ini.
- Alamat IP pengguna
- Data Agen Pengguna pengguna (50 karakter pertama dari string data browser)
- Stempel waktu untuk "aktivitas terakhir"

Jika Anda menyetel sess_encrypt_cookie ke FALSE, Anda dapat membaca cookie di browser Anda dan melihat apa yang telah disimpan (sebagian dikodekan, tetapi Anda dapat membuatnya keluar) misalnya menyertakan URL:

```
ip_address%22%3Bs%3A9%3A%22127.0.0.1%22%3Bs%3A10%3A%22
```

Pengguna—dalam hal ini, 127.0.0.1). Jika Anda menyetelnya ke TRUE, cookie dienkripsi, hanya serangkaian gunk acak. Browser Anda bahkan tidak dapat membedakan bagian cookie yang terpisah, yang berarti bahwa pengguna tidak dapat mengubahnya secara bermakna tanpa membatalkannya.

Saat pengguna membuat permintaan halaman lain, situs kemudian dapat memeriksa apakah ID sesi telah disimpan di browser pengguna sebagai bagian dari cookie. Jika sudah, Anda tahu mereka adalah bagian dari sesi yang ada. Jika tidak, Anda tahu itu adalah sesi baru. Asalkan saya ingat untuk memuat kelas sesi CI pada semua pengontrol saya yang lain juga, CI juga akan melakukan pemeriksaan yang sama untuk mereka. Yang harus saya lakukan adalah memberi tahu setiap pengontrol bagaimana berperilaku jika tidak ada cookie.

Mengubah Sesi menjadi Keamanan

Ini sendiri tidak membuat sistem keamanan. Siapa pun yang mengunjungi situs memulai sesi. Kode hanya mencatat apakah mereka pengunjung baru atau bukan. Salah satu cara untuk mencegah akses tidak sah ke beberapa halaman melibatkan menambahkan sesuatu yang lain ke cookie mereka jika mereka 'masuk', sehingga saya dapat mengujinya. Kemudian, jika mereka memasukkan nama pengguna dan kata sandi yang benar satu kali, itu akan dicatat dalam cookie, dan mekanisme sesi akan menemukannya saat memeriksa cookie saat setiap permintaan baru masuk. Saya kemudian dapat mengujinya, dan jika saya menemukannya, situs akan membiarkan mereka masuk ke halaman yang dilindungi selama sisa sesi. Mereka tidak perlu terus masuk.

Menambahkan sesuatu ke cookie itu mudah. Di pengontrol assessmentme() saya, setelah saya memutuskan apakah kata sandi dan nama pengguna dapat diterima, saya menambahkan:

```
if($this->checkme($username, $password)=='yes')
{
        $newdata = array(
        'status' => 'OK',
);
$this->session->set_userdata($newdata);
$this->mainpage();
}
```

Itu mengambil konten array $newdata saya—hanya satu variabel dalam kasus ini—dan menambahkannya ke string cookie. Sekarang, setiap kali kombinasi nama pengguna kata sandi dapat diterima, assessmentme() akan menambahkan 'OK' ke cookie, dan saya dapat memulai setiap pengontrol dengan kode ini:

```
 /*remember to load the library!*/
     $this->load->library('session');
 /*look for a 'status' variable in the contents of the session cookie*/
 $status = $this->session->userdata('status');
 /*if it's not there, make the user log in*/if(!isset($status) || $status != 'OK')
            { /*function to present a login page again…*/}
/*otherwise, go ahead with the code*/
```

Di sini, Anda memiliki dasar pagar keamanan di sekitar situs Anda. Anda dapat dengan mudah membuatnya lebih rumit. Misalnya, jika beberapa pengguna Anda memiliki tingkat akses yang lebih tinggi daripada yang lain, Anda dapat menyimpan tingkat dalam variabel status daripada 'OK'—maka Anda dapat menggunakan ini dalam pengujian bersyarat untuk mengontrol akses ke fungsi.

Menyimpan data semacam ini dalam cookie tidak disukai karena pengguna dapat dengan mudah menulis ulang cookie di mesin mereka di antara kunjungan ke situs Anda. Mengingat bahwa kelas sesi CI mengenkripsinya, Anda cukup aman. Namun, alternatifnya adalah membuat basis data pengguna, dan setelah seseorang masuk, menulis 'OK' ke basis data terhadap ID sesi itu. Kemudian, untuk akses selanjutnya, Anda memeriksa ID sesi (dalam cookie) terhadap database, untuk melihat apakah memiliki 'OK' atau tingkat yang bertentangan.

Sangat mudah untuk menyimpan data sesi di database Anda. Pertama, buat tabel database. Jika Anda menggunakan MySQL, gunakan kueri SQL ini:

```
CREATE TABLE IF NOT EXISTS `ci_sessions` (
session_id varchar(40) DEFAULT '0' NOT NULL,
ip_address varchar(16) DEFAULT '0' NOT NULL,
user_agent varchar(50) NOT NULL,
last_activity int(10) unsigned DEFAULT 0 NOT NULL, status
varchar(5) DEFAULT 'no',
PRIMARY KEY (session_id)
);
```

Kemudian, ubah parameter koneksi di file system/application/config/ database.php untuk memberi tahu CI di mana database berada. Lihat Bab 4 untuk rincian lebih lanjut tentang database.

Jika semua berfungsi, Anda akan melihat data menumpuk di tabel database saat Anda menyambungkan dan memutuskan sambungan. Jika Anda memiliki sesi yang disimpan dalam tabel database, karena setiap pengguna terhubung ke situs Anda, situs akan menguji cookie. Jika ditemukan, Anda dapat membuatnya membaca id sesi, dan mencocokkannya dengan id sesi yang disimpan dalam database.

Anda sekarang memiliki mekanisme sesi yang kuat. Dan semua ini berasal dari satu baris kode! Hanya satu peringatan. Kelas sesi PHP asli dapat mengatasi pengguna yang mematikan cookie di browser mereka. (Alih-alih menyimpan cookie, ia menambahkan data sesi ke string URL.) Kelas CI tidak melakukan ini. Jika pengguna telah mematikan cookie, maka dia tidak dapat masuk ke situs Anda. Apakah ini masalah bagi Anda tergantung pada orang yang Anda harapkan untuk menggunakan situs Anda. Ini adalah salah satu peningkatan yang saya harap Rick Ellis akan segera lakukan untuk CI.

6.4 KEAMANAN

Perhatikan bahwa kelas sesi secara otomatis menyimpan informasi tentang alamat IP dan agen pengguna dari pengguna yang membuat permintaan halaman. Anda dapat menggunakan ini untuk memberikan keamanan tambahan.

Ada dua pengaturan yang dapat Anda ubah di file konfigurasi Anda untuk keamanan tambahan:

- sess_match_ip: Jika Anda menyetel ini ke true, CI akan mencoba mencocokkan alamat IP pengguna saat membaca data sesi. Ini untuk mencegah pengguna 'membajak' log-in. Namun, beberapa server (baik ISP dan perusahaan besar server) dapat mengeluarkan permintaan oleh pengguna akhir yang sama melalui alamat IP yang berbeda. Jika Anda menyetel nilai ini ke true, Anda dapat mengecualikannya secara tidak sengaja.
- sess_match_useragent: Jika Anda menyetel ini ke true, CI akan mencoba mencocokkan Agen Pengguna saat membaca data sesi. Ini berarti bahwa seseorang yang mencoba membajak suatu sesi harus memastikan bahwa pengaturan 'agen pengguna' yang dikembalikan oleh sistemnya cocok dengan pengaturan pengguna asli. Itu membuat pembajakan sedikit lebih sulit.

CI juga memiliki kelas user_agent, yang Anda muat seperti ini:

```
$this->load->library('user_agent');
```

Setelah dimuat, Anda dapat memintanya untuk mengembalikan berbagai informasi tentang agen apa pun yang menjelajahi situs Anda, misalnya, jenis browser dan sistem operasi, dan khususnya apakah itu browser, seluler, atau robot. Jika Anda ingin membuat daftar robot yang mengunjungi situs Anda, Anda dapat melakukannya seperti ini:

```
$fred = $this->agent->is_robot();if ($fred ==
TRUE)
    {$agent = $this->agent->agent_string();
/*add code here to store or analyse the name the user agent isreturning*/
}
```

Kelas bekerja dengan memuat, dan membandingkan dengan, larik agen pengguna, browser, dan robot yang terdapat dalam file konfigurasi lain:

```
system/ application/config/user_agents.
```

Jika mau, Anda dapat dengan mudah mengembangkan ini untuk memungkinkan situs Anda mengunci jenis browser tertentu atau robot tertentu. Namun, ingat bahwa penyerang mudah menulis agen pengguna robot, dan meminta mereka mengembalikan string user_agent apa pun yang Anda inginkan, sehingga mereka dapat dengan mudah menyamar sebagai browser umum. Banyak robot, termasuk yang seperti Googlebot yang tercantum dalam larik

user_agents CI, 'berperilaku baik'. Ini berarti bahwa jika Anda menyetel file robots.txt untuk mengecualikannya, mereka tidak akan masuk tanpa izin.

Tidak ada cara mudah untuk mengecualikan robot yang tidak mematuhi file ini, kecuali Anda tahu nama mereka terlebih dahulu!

Di CI, mekanisme sesi menyimpan IP dari situs yang meminta, sehingga Anda dapat menggunakan ini untuk mengoperasikan daftar hitam situs. Ambil IP dari sesi seperti ini:

```
/*remember to load the library!*/
$this->load->library('session');
/*look for an 'ip_address' variable in the contents of the sessioncookie*/
$ip = $this->session->userdata('ip_address');
```

Kemudian Anda dapat menguji variabel $ip terhadap daftar hitam.

Anda juga dapat mengembangkan mekanisme sesi CI untuk membatasi kerusakan dari permintaan berulang—seperti serangan penolakan layanan di mana robot diatur untuk membebani situs Anda dengan meminta halaman berulang kali. Atau Anda dapat menggunakan mekanisme ini untuk menangani serangan 'kamus', di mana robot diatur untuk memanggil formulir masuk Anda berulang kali, dan mencoba ratusan atau ribuan kombinasi kata sandi/nama pengguna hingga menemukan yang tepat.

Anda dapat melakukan ini karena kelas sesi CI menyimpan waktu last_activity untuk setiap sesi. Setiap kali halaman diminta, Anda dapat memeriksa berapa lama permintaan terakhir dibuat oleh pengguna ini. Meskipun satu interval waktu tidak banyak memberi tahu Anda, Anda dapat mengatur sistem untuk menyimpan lebih banyak data dan mengembangkan pola penggunaan. Serangan kamus bergantung pada mendapatkan balasan yang cepat, jika tidak maka akan memakan waktu terlalu lama. Jika Anda mendeteksi terlalu banyak permintaan secara berurutan, Anda dapat mengakhiri sesi, atau memperlambat respons.

6.5 RINGKASAN

Kami telah menguraikan aplikasi yang ingin kami bangun, dan menyerang masalah pertama yang muncul di hampir semua aplikasi: manajemen sesi dan (jika kami ingin melindungi bagian situs kami dari pengguna yang tidak sah) keamanan. Untuk melakukan ini, kita telah melihat kelas sesi CI dalam beberapa detail, dan melihat bagaimana ia membuat catatan sesi dan meninggalkan cookie di browser pengunjung. Itu kemudian dapat mencari cookie ketika permintaan berikutnya dibuat, dan Anda dapat menggunakan respons untuk mengontrol cara situs Anda merespons.

BAB 7
CODEIGNITER DAN OBJEK

Ini adalah bab geek. Ini menggambarkan cara CodeIgniter benar-benar bekerja, 'di bawah tenda'. Jika Anda baru mengenal CI, Anda mungkin ingin melewatkannya. Namun, cepat atau lambat, Anda mungkin ingin memahami mengapa sesuatu terjadi dengan cara tertentu—bukan sekadar mengetahui bahwa hal itu terjadi. Objek membingungkan saya ketika saya mulai menggunakan CodeIgniter. Saya datang ke CodeIgniter melalui PHP 4, yang merupakan bahasa prosedural, bukan bahasa Berorientasi Objek (OO). Saya dengan sepatutnya mencari objek dan metode, properti dan warisan, dan enkapsulasi, tetapi upaya awal saya untuk menulis kode CI terganggu oleh pesan kesalahan "Panggil ke fungsi anggota pada non-objek". Saya sering melihatnya sehingga saya berpikir untuk mencetaknya di kaos: memiliki nada libertarian yang misterius, anarkis, dan saya bisa melihat diri saya memakainya di pameran seni modern.

Untuk menyelamatkan dunia dari banyak kaos membosankan, bab ini membahas cara CI menggunakan objek, dan berbagai cara Anda dapat menulis dan menggunakan objek Anda sendiri. Kebetulan, saya telah menggunakan 'variabel�properti', dan 'metode�fungsi' bergantian, seperti yang sering dilakukan CI dan PHP. Anda menulis 'fungsi' di pengontrol Anda misalnya, ketika purist OO akan menyebutnya 'metode'. Anda mendefinisikan 'variabel' kelas ketika purist akan menyebutnya 'properti'.

7.1 PEMROGRAMAN BERORIENTASI OBJEK

Saya berasumsi Anda—seperti saya—memiliki pengetahuan dasar tentang OOP, tetapi mungkin telah mempelajarinya sebagai renungan untuk 'normal' PHP 4. PHP 4 bukan bahasa OO, meskipun beberapa fungsi OO telah ditempelkan di dalamnya. PHP 5 jauh lebih baik, dengan mesin yang mendasari yang ditulis dari bawah ke atas dengan OO dalam pikiran.

Tetapi Anda dapat melakukan sebagian besar dasar-dasar di PHP 4, dan CI berhasil melakukan semua yang dibutuhkan secara internal, dalam kedua bahasa tersebut.

Hal utama yang perlu diingat adalah, ketika program OO sedang berjalan, selalu ada satu objek saat ini (tetapi hanya satu). Objek dapat memanggil satu sama lain dan menyerahkan kendali satu sama lain, dalam hal ini objek saat ini berubah; tetapi hanya satu dari mereka yang dapat menjadi terkini pada satu waktu. Objek saat ini mendefinisikan 'lingkup'—dengan kata lain, variabel (properti) dan metode (fungsi) mana yang tersedia untuk program pada saat itu. Jadi penting untuk mengetahui, dan mengontrol, objek mana yang saat ini. Seperti petugas polisi dan bus London, variabel dan metode milik objek yang tidak terkini tidak ada untuk Anda saat Anda paling membutuhkannya.

PHP, sebagai campuran dari pemrograman fungsional dan OO, juga menawarkan kemungkinan bahwa tidak ada objek saat ini! Anda dapat memulai sebagai program fungsional, memanggil objek, membiarkannya mengambil alih untuk sementara waktu, dan

kemudian membiarkannya mengembalikan kontrol ke program. Untungnya, CI menangani ini untuk Anda.

Cara kerja CI 'Super-Object'

CI bekerja dengan membangun satu 'objek super': ia menjalankan seluruh program Anda sebagai satu objek besar, untuk menghilangkan masalah pelingkupan. Saat Anda memulai CI, rangkaian peristiwa yang kompleks terjadi. Jika Anda mengatur instalasi CI Anda untuk membuat log, Anda akan melihat sesuatu seperti ini:

```
1   DEBUG - 2006-10-03 08:56:39 --> Config Class: Initialized
2   DEBUG - 2006-10-03 08:56:39 --> No URI present. Default controller set.
3   DEBUG - 2006-10-03 08:56:39 --> Router Class: Initialized
4   DEBUG - 2006-10-03 08:56:39 --> Output Class: Initialized
5   DEBUG - 2006-10-03 08:56:39 --> Input Class: Initialized
6   DEBUG - 2006-10-03 08:56:39 --> Global POST and COOKIE: data sanitized
7   DEBUG - 2006-10-03 08:56:39 --> URI Class: Initialized
8   DEBUG - 2006-10-03 08:56:39 --> Language Class: Initialized
9   DEBUG - 2006-10-03 08:56:39 --> Loader Class: Initialized
10  DEBUG - 2006-10-03 08:56:39 --> Controller Class: Initialized
11  DEBUG - 2006-10-03 08:56:39 --> Helpers loaded: security
12  DEBUG - 2006-10-03 08:56:40 --> Scripts loaded: errors
13  DEBUG - 2006-10-03 08:56:40 --> Scripts loaded: boilerplate
14  DEBUG - 2006-10-03 08:56:40 --> Helpers loaded: url
15  DEBUG - 2006-10-03 08:56:40 --> Database Driver Class: Initialized
16  DEBUG - 2006-10-03 08:56:40 --> Model Class: Initialized
```

Saat startup—yaitu, setiap kali permintaan halaman diterima melalui Internet—CI menjalani prosedur yang sama. Anda dapat melacak log melalui file CI:

1. File index.php menerima permintaan halaman. URL dapat menunjukkan pengontrol mana yang diperlukan, jika tidak, CI memiliki pengontrol default (baris 2). Index.php melakukan beberapa pemeriksaan dasar dan memanggil file codeigniter.php (\codeigniter\codeigniter.php).
2. File codeigniter.php membuat instance kelas Config, Router, Input, URL, (dll.) (baris 1, dan 3 hingga 9). Ini disebut kelas 'dasar': Anda jarang berinteraksi langsung dengan mereka, tetapi mereka mendasari hampir semua hal yang dilakukan CI.
3. codeigniter.php menguji untuk melihat versi PHP mana yang menjalankannya, dan memanggil Base4 atau Base5 (/codeigniter/Base4(or 5).php). Ini membuat objek 'tunggal': objek yang memastikan bahwa kelas hanya memiliki satu instance. Masing-masing memiliki fungsi &get_instance() publik. Perhatikan &:, ini adalah penugasan dengan referensi. Jadi, jika Anda menetapkan metode &get_instance(), metode ini akan menetapkan ke satu instance kelas yang sedang berjalan. Dengan kata lain, ini mengarahkan Anda ke lubang merpati yang sama. Jadi, alih-alih menyiapkan banyak objek baru, Anda mulai untuk membangun satu 'objek super', yang berisi semua yang terkait dengan kerangka kerja.
4. Setelah pemeriksaan keamanan, codeigniter.php membuat instance controller yang diminta, atau controller default (baris 10). Kelas baru disebut $CI. Fungsi yang

ditentukan dalam URL (atau default) kemudian dipanggil, dan kehidupan seperti yang kita ketahui mulai terbangun dan terjadi. Bergantung pada apa yang Anda tulis di pengontrol Anda, CI kemudian akan menginisialisasi kelas lain yang Anda butuhkan, dan 'sertakan' skrip fungsional yang Anda minta. Jadi pada log di atas, kelas model diinisialisasi. (baris 16) Skrip 'boilerplate', di sisi lain, yang juga ditampilkan di log (baris 13), adalah skrip yang saya tulis untuk memuat potongan teks standar. Ini adalah file .php, disimpan di folder skrip, tetapi bukan kelas: hanya sekumpulan fungsi. Jika Anda menulis PHP 'murni', Anda dapat menggunakan 'include' atau 'require' untuk membawanya ke namespace: CI perlu menggunakan fungsi 'load' sendiri untuk membawanya ke objek super.

Konsep 'namespace' atau ruang lingkup sangat penting di sini. Saat Anda mendeklarasikan variabel, array, objek, dll., PHP menyimpan nama variabel dalam memorinya dan memberikan blok memori lebih lanjut untuk menampung isinya. Namun, masalah mungkin muncul jika Anda mendefinisikan dua variabel dengan nama yang sama. (Dalam situs yang kompleks, ini mudah dilakukan.) Untuk alasan ini, PHP memiliki beberapa set aturan. Sebagai contoh:

- Setiap fungsi memiliki namespace atau ruang lingkupnya sendiri, dan variabel yang didefinisikan dalam suatu fungsi biasanya 'lokal' untuknya. Di luar fungsi, ini tidak ada artinya.
- Anda dapat mendeklarasikan variabel 'global', yang disimpan dalam namespace global khusus dan tersedia di seluruh program.
- Objek memiliki ruang nama sendiri: variabel ada di dalam objek selama objek itu ada, tetapi hanya dapat direferensikan melalui objek.

Jadi $variable, $variableglobal, dan $this->variable adalah tiga hal yang berbeda. Khususnya, sebelum OO, ini dapat menyebabkan segala macam kebingungan: Anda mungkin juga banyak variabel di namespace Anda (sehingga nama yang saling bertentangan saling menimpa), atau Anda mungkin menemukan bahwa beberapa variabel tidak dapat diakses dari lingkup apa pun yang Anda masuki. CI menawarkan cara cerdas untuk memilah ini untuk Anda.

Jadi, sekarang Anda telah memulai CI, menggunakan URL www.mysite.com/index.php/welcome/, yang menentukan bahwa Anda menginginkan fungsi indeks dari pengontrol sambutan. Jika Anda ingin melihat kelas dan metode apa yang sekarang ada di namespace saat ini dan tersedia untuk Anda, coba masukkan kode 'inspeksi' ini di pengontrol sambutan:

```
$fred = get_declared_classes();
foreach($fred as $value)
{$extensions = get_class_methods($value); print "class is $value, methods are: ";
print_r($extensions);}
```

Ketika saya menjalankan ini sekarang, itu mencantumkan 270 kelas yang dideklarasikan. Sebagian besar perpustakaan lain dideklarasikan dalam instalasi PHP saya. 11 yang terakhir berasal dari CI: sepuluh adalah kelas dasar CI (config, router, dll.) dan yang terakhir adalah kelas pengontrol yang saya panggil.

Inilah 11 terakhir, dengan metode dihilangkan dari semua kecuali dua yang terakhir:

```
258: class is CI_Benchmark
259: class is CI_Hooks,
260: class is CI_Config,
261: class is CI_Router,
262: class is CI_Output,
263: class is CI_Input,
264: class is CI_URI,
265: class is CI_Language,
266: class is CI_Loader,
267: class is CI_Base,
268: class is Instance,
269: class is Controller, methods are: Array ( [0] => Controller [1]
=> _ci_initialize [2] => _ci_load_model [3] => _ci_assign_to_models
[4] => _ci_autoload [5] => _ci_assign_core [6] => _ci_init_scaffolding
[7] => _ci_init_database [8] => _ci_is_loaded [9] => _ci_scaffolding
[10] => CI_Base)
270: class is Welcome, methods are: Array ( [0] => Welcome [1] =>
index [2] => Controller [3] => _ci_initialize [4] => _ci_load_model
[5] => _ci_assign_to_models [6] => _ci_autoload [7] => _ci_assign_core
[8] => _ci_init_scaffolding [9] => _ci_init_database [10] => _ci_is_ loaded [11] =>
_ci_scaffolding [12] => CI_Base)
```

Perhatikan—dalam tanda kurung—bahwa kelas Welcome (nomor 270: controller yang saya gunakan) memiliki semua metode kelas Controller (nomor 269). Inilah sebabnya mengapa Anda selalu memulai definisi kelas pengontrol dengan memperluas kelas pengontrol—Anda memerlukan pengontrol untuk mewarisi fungsi-fungsi ini. (Dan juga, model harus selalu memperluas kelas model.) Welcome memiliki dua metode tambahan: Welcome dan index. Sejauh ini, dari 270 kelas, ini adalah dua fungsi yang saya tulis!

Perhatikan juga bahwa ada kelas Instance. Jika Anda memeriksa variabel kelas dari kelas 'Instance', Anda akan menemukan ada banyak sekali! Hanya satu variabel kelas dari kelas Instance, diambil hampir secara acak, adalah input array:

```
["input"]=> &object(CI_Input)#6 (4) { ["use_xss_clean"]=> bool(false) ["ip_address"]=>
bool(false) ["user_agent"]=> bool(false) ["allow_get_array"]=> bool(false) }
```

Ingat ketika kita memuat file input dan membuat kelas input asli? Nya
variabel kelas adalah:

```
use_xss_clean is bool(false) ip_address is bool(false) user_agent is bool(false) allow_get_array
is bool(false)
```

Seperti yang Anda lihat, sekarang semuanya telah dimasukkan ke dalam kelas 'instance'.

Semua kelas 'dasar' CI lainnya (router, output, dll.) disertakan dengan cara yang sama. Anda mungkin tidak perlu menulis kode yang mereferensikan kelas dasar ini secara langsung, tetapi CI sendiri membutuhkannya untuk membuat kode Anda berfungsi.

Menyalin dengan Referensi

Anda mungkin telah memperhatikan bahwa kelas CI_Input ditetapkan oleh referensi (["input"]=> &object(CI_Input)). Ini untuk memastikan bahwa ketika variabelnya berubah, begitu juga variabel dari kelas aslinya. Karena penugasan dengan referensi bisa membingungkan, berikut penjelasan singkatnya. Kita semua akrab dengan penyalinan sederhana di PHP:

```
$one    =       1;
$two    =       $one; echo $two;
```

menghasilkan 1, karena $dua adalah salinan dari $satu. Namun, jika Anda menetapkan ulang $one:

```
$one =  1;
$two = $one;
$one = 5
echo    $two;
```

Kode ini masih menghasilkan 1, karena perubahan pada $satu setelah $dua ditetapkan tidak tercermin dalam $dua. Ini adalah penugasan satu kali dari nilai yang kebetulan ada di variabel $one pada saat itu, ke variabel baru $two, tetapi begitu selesai, kedua variabel tersebut menjalani kehidupan yang terpisah. (Dengan cara yang sama, jika saya mengubah $two, $one tidak berubah.)

Akibatnya, PHP membuat dua pigeonholes: satu disebut $one, satu disebut $two. Nilai yang terpisah hidup di masing-masing. Anda dapat, pada satu kesempatan, membuat nilai-nilai itu sama, tetapi setelah itu mereka masing-masing melakukan hal mereka sendiri.

PHP juga memungkinkan penyalinan 'dengan referensi'. Jika Anda menambahkan sederhana & ke baris 2 kode:

```
$one = 1;
$two =& $one;
$one =  5;
echo    $two;
```

Kemudian kode sekarang bergema 5: perubahan yang kita buat pada $one juga terjadi pada $two. Mengubah = menjadi =& di baris kedua berarti bahwa tugas tersebut 'berdasarkan referensi'.

Sekarang, seolah-olah hanya ada satu lubang merpati, yang memiliki dua nama ($satu dan $two). Apapun yang terjadi pada isi pigeonhole terjadi pada $satu dan $two, seolah-olah mereka hanya nama yang berbeda untuk hal yang sama.

Prinsipnya bekerja untuk objek serta variabel string sederhana. Anda dapat menyalin atau mengkloning objek menggunakan operator =, dalam hal ini Anda membuat salinan baru yang sederhana, yang kemudian menjalani kehidupan yang mandiri. Atau, Anda dapat menetapkan satu ke yang lain dengan referensi: sekarang kedua objek saling menunjuk, sehingga setiap perubahan yang dilakukan pada yang satu juga akan terjadi pada objek lainnya. Sekali lagi, anggap mereka sebagai dua nama berbeda untuk hal yang sama.

7.2 MENAMBAHKAN KODE SENDIRI KE CI 'SUPER-OBJECT'

Anda berkontribusi pada proses membangun 'objek super' saat Anda menulis kode Anda sendiri. Misalkan Anda telah menulis sebuah model yang disebut 'status', yang berisi dua variabel kelasnya sendiri, $satu dan $dua, dan sebuah konstruktor yang memberikan nilai masing-masing 1 dan 2. Mari kita periksa apa yang terjadi ketika Anda memuat model ini.

Kelas 'instance' menyertakan variabel 'load', yang merupakan salinan (berdasarkan referensi) dari objek CI_Loader. Jadi kode yang Anda tulis di controller Anda adalah:

```
$this->load->model($status)
```

Dengan kata lain, ambil variabel kelas 'beban' dari kelas super CI saat ini ('ini') dan gunakan metode 'model'. Ini sebenarnya merujuk fungsi 'model' di kelas 'loader' (/system/libraries/loader.php) dan yang mengatakan:

```
function model($model, $name = '')
{
if ($model == '')return;
        $obj =& get_instance();
        $obj->_ci_load_model($model, $name);
}
```

(Variabel $name dalam kode ini ada jika Anda ingin memuat model Anda di bawah alias. Saya tidak tahu mengapa Anda ingin melakukan ini; mungkin itu diinginkan oleh polisi di beberapa ruang nama lain.) Seperti yang Anda lihat, model dimuat dengan referensi ke dalam kelas Instance. Karena get_instance() adalah metode tunggal, Anda selalu mereferensikan instance yang sama dari kelas Instance. Jika Anda menjalankan pengontrol lagi, menggunakan kode 'inspect' kami yang dimodifikasi untuk menampilkan variabel kelas, Anda sekarang akan melihat bahwa kelas instance berisi variabel kelas baru:

```
["status"]=> object(Status)#12 (14) { ["one"]=> int(1) ["two"]=>int(2) ... (etc)
```

Dengan kata lain, 'objek super' CI sekarang menyertakan objek bernama $status yang menyertakan variabel kelas yang Anda definisikan dalam model status asli Anda, yang ditetapkan ke nilai yang kami tetapkan.

Jadi kami secara bertahap membangun satu 'super-objek' CI besar, yang memungkinkan Anda untuk menggunakan salah satu metode dan variabelnya tanpa terlalu mengkhawatirkan dari mana asalnya dan namespace apa yang mungkin mereka gunakan. Ini

adalah alasan untuk sintaks panah CI. Untuk menggunakan metode (katakanlah) sebuah model, Anda pertama-tama harus memuat model di pengontrol Anda:

```
$this->load->model('Model_name');
```

Ini membuat model menjadi variabel kelas $this->, kelas (pengontrol) saat ini. Anda kemudian memanggil fungsi variabel kelas itu dari controller, seperti ini:

```
$this->Model_name->function();
```

7.3 MASALAH DENGAN CI 'SUPER-OBJECT'

Ada satu masalah besar bagi Rick Ellis ketika dia menulis kode aslinya. PHP 4 menangani objek kurang elegan dibandingkan PHP 5, jadi dia harus memperkenalkan 'retas yang sangat jelek' (kata-katanya) ke dalam file Base4. Jelek atau tidak, peretasan itu berhasil, jadi kita tidak perlu khawatir tentang itu. Itu hanya berarti bahwa CI bekerja dengan baik pada sistem PHP 4 seperti halnya pada PHP5.

Ada dua masalah lain yang layak disebutkan di sini:

- Anda dapat menemukan diri Anda mencoba bekerja dengan objek yang tidak tersedia.
- Anda harus menyusun situs Anda dengan hati-hati, karena Anda tidak dapat memanggil metode dari satu pengontrol dari dalam pengontrol lainnya.

Mari kita lihat dua masalah ini secara bergantian. Anda ingat t-shirt yang saya sebutkan di atas: "Panggil ke fungsi anggota pada non-objek"? Pesan kesalahan yang mengganggu ini sering kali berarti Anda mencoba menggunakan fungsi dari suatu kelas (misalnya kelas model yang Anda tulis) tetapi lupa memuat kelas tersebut. Dengan kata lain, Anda menulis:

```
$this->Model_name->function();
```

tapi lupa mendahuluinya dengan:

```
$this->load->model('Model_name');
```

Atau beberapa variasi dari ini: misalnya, Anda memuat model di dalam satu fungsi kelas, yang memuat model, tetapi hanya di dalam fungsi itu, dan kemudian Anda mencoba menggunakan metodenya dari dalam fungsi lain, meskipun di kelas yang sama. Biasanya yang terbaik adalah memuat model, dll., dari fungsi konstruktor kelas: kemudian mereka tersedia untuk semua fungsi lain di kelas.

Masalahnya juga bisa lebih halus. Jika Anda menulis kelas Anda sendiri, misalnya, Anda mungkin ingin menggunakannya untuk mengakses database, atau untuk mencari sesuatu di file konfigurasi Anda—dengan kata lain, untuk memberi mereka akses ke sesuatu yang merupakan bagian dari CI 'super- obyek'. (Ada diskusi yang lebih lengkap tentang cara menambahkan kelas atau pustaka Anda sendiri di Bab 13.) Untuk meringkas, kecuali jika kelas baru Anda adalah pengontrol, model, atau tampilan, itu tidak dimasukkan ke objek super CI . Jadi Anda tidak bisa menulis sesuatu di dalam kelas baru Anda seperti ini:

```
$this->config->item('base_url);
```

Ini tidak akan berhasil, karena untuk kelas baru Anda, $this-> berarti dirinya sendiri, bukan objek super CI. Sebagai gantinya, Anda harus membangun kelas baru Anda menjadi kelas super dengan memanggil kelas Instance (terdengar familiar?) Menggunakan nama variabel lain (biasanya $obj)

```
$obj =& get_instance();
```

Sekarang Anda dapat menulis panggilan itu ke superclass CI sebagai:

```
$obj->config->item('base_url);
```

dan kali ini berhasil.

Namun, saat Anda menulis kelas baru Anda, ingatlah bahwa kelas itu masih memiliki identitasnya sendiri. Mari kita gunakan contoh garis besar singkat untuk membuatnya lebih jelas.

Anda ingin menulis kelas perpustakaan yang menyiapkan URL berdasarkan lokasi server yang meminta halaman. Jadi Anda menulis beberapa kode untuk mencari lokasi geografis dari alamat IP yang memanggil halaman Anda (menggunakan perpustakaan seperti kelas netGeo yang tersedia dari http://www.phpclasses.org/browse/package/514.html). Kemudian, dengan menggunakan fungsi sakelar, Anda memilih salah satu dari beberapa fungsi alternatif, dan Anda menyajikan halaman bahasa Inggris untuk permintaan AS atau Inggris, halaman Jerman untuk permintaan Jerman atau Austria, dan seterusnya. Sekarang, URL lengkap ke halaman spesifik negara Anda akan terdiri dari dua bagian: URL dasar situs Anda (www.mysite.com/index.php/), ditambah URL halaman individual (mypage/germanversion).

Anda perlu mendapatkan URL dasar situs dari file konfigurasi CI. Bagian kedua dari URL dihasilkan oleh pernyataan switch di konstruktor kelas baru Anda—jika klien ini di Jerman, sajikan fungsi halaman Jerman, dll. Saat ini dilakukan dalam panggilan konstruktor, Anda perlu masukkan hasilnya ke dalam variabel kelas, sehingga dapat digunakan di fungsi lain dalam kelas yang sama. Ini berarti bahwa:

- Paruh pertama URL Anda berasal dari file konfigurasi CI, yang hanya dapat direferensikan melalui superobject, yang telah Anda tautkan menggunakan $obj =& get_instance(). Dengan kata lain, Anda menyebutnya menggunakan $obj->config->item('base_url);
- Tetapi paruh kedua URL Anda dibuat di dalam konstruktor kelas baru Anda dan ditetapkan ke variabel kelas, $base. Ini tidak ada hubungannya dengan objek super CI; itu milik kelas baru Anda, dan dirujuk sebagai $this->base

Ini dapat menyebabkan penggunaan referensi $this-> dan $obj-> di baris yang sama—misalnya:

```
class my_new_class{ var
$base; My_new_class()
```

```
{
$obj =& get_instance();
// geolocation code here, returning  a value through a switch statement
//this value is assigned to $local_url
$this->base = $obj->config->item('base_url);
$this->base .= $local_url;
}
```

Membuat bingung ini adalah sumber bermanfaat lainnya dari, "Panggil ke fungsi anggota pada non-objek". Dalam contoh kami, Anda akan mendapatkan pesan kesalahan itu jika Anda mencoba menelepon juga $obj->base, atau $this->config->item().

Beralih ke masalah yang tersisa, Anda tidak dapat memanggil metode satu pengontrol dari dalam yang lain. Mengapa Anda ingin melakukan ini? Yah, itu tergantung. Jadi satu aplikasi, saya menulis serangkaian fungsi self-test di dalam setiap pengontrol. Jika saya menelepon $this->selftest() di dalam controller, ia melakukan berbagai tes yang berguna. Tapi tampaknya bertentangan dengan prinsip pemrograman kebajikan kemalasan harus berulang kali memanggil metode self-test di setiap pengontrol secara terpisah. Saya mencoba menulis satu fungsi, dalam satu pengontrol, yang akan melewati semua pengontrol, memanggil metode uji mandiri di masing-masing, menggabungkan semua hasil sementara saya menatap ke luar jendela, dan kemudian memberi saya laporan komprehensif sebagai imbalan hanya satu klik mouse. Sayangnya, tidak. Tidak bisa dilakukan.

Sebagai aturan umum, jika Anda memiliki kode yang mungkin diperlukan oleh lebih dari satu pengontrol, masukkan ke dalam model atau semacam skrip terpisah. Kemudian mereka berdua bisa menggunakannya. (Tentu saja, ini tidak membantu masalah pengujian mandiri saya, karena kode untuk menguji pengontrol harus ada di pengontrol!)

Tapi ini adalah masalah kecil. Seperti yang dikatakan Rick Ellis kepada saya:
"Saya ingin sampai pada sesuatu yang lebih sederhana, jadi saya memutuskan untuk membuat satu objek pengontrol besar yang berisi banyak instance objek lain:...ketika pengguna membuat pengontrol mereka sendiri, mereka dapat dengan mudah mengakses apa pun yang telah dipakai hanya dengan memanggilnya, tanpa khawatir tentang cakupan".

Begitulah cara kerjanya, sebagian besar waktu, efisien, dan sepenuhnya dalam Latar Belakang. Jadi saya tidak pernah mencetak t-shirt itu.

7.4 RINGKASAN

Kami telah melihat cara CI membangun satu 'super-objek' untuk memastikan bahwa semua metode dan variabel yang Anda butuhkan tersedia secara otomatis untuk Anda tanpa Anda harus mengelolanya dan mengkhawatirkan cakupannya. CI menggunakan penugasan secara ekstensif dengan referensi, membuat instance satu demi satu kelas dan menautkan semuanya bersama-sama sehingga Anda dapat mengaksesnya melalui 'kelas super'. Sebagian besar waktu, Anda tidak perlu tahu apa yang dilakukan 'kelas super', asalkan Anda menggunakan notasi 'panah' CI dengan benar. Kami juga telah melihat bagaimana Anda dapat menulis kelas Anda sendiri dan masih memiliki akses ke kerangka kerja CI. Terakhir, kami

melihat beberapa masalah yang dapat muncul, terutama jika Anda tidak terbiasa dengan program OO, dan menyarankan beberapa solusi.

BAB 8
MENGGUNAKAN CI UNTUK MENGUJI KODE

Bab ini membahas bagaimana CI dapat membantu Anda menguji kode Anda. Pengujian adalah jantung dari aplikasi kita. Kami telah membangunnya untuk menguji aplikasi jarak jauh lainnya; kami juga ingin mengujinya sendiri, saat kami mengembangkannya. CI membuat ini jauh lebih mudah. Namun, 'pengujian' dapat berarti banyak hal, jadi kami memulai dengan melihat perbedaan antara dua jenis utama, dan beberapa alasan lain yang mungkin membuat Anda ingin menjalankan pengujian.
Kemudian kita melihat kelas CI untuk membantu pengujian:

- Tes unit
- Pembandingan
- 'pembuat profil'
- Cara CI membantu Anda melibatkan database Anda dalam pengujian tanpa mengacak data langsung

8.1 MENGAPA TES, DAN UNTUK APA?

Banyak yang telah ditulis tentang pengujian. Ini telah menjadi sebuah industri. Program kompleks menggunakan pasukan penguji atau perangkat lunak pengujian. Dan konsep 'pengembangan yang digerakkan oleh pengujian' adalah Anda mendesain pengujian Anda terlebih dahulu, bahkan sebelum satu baris kode pun: kemudian tulis kode Anda untuk lulus.

Di sisi lain, banyak programmer tidak melakukan pengujian sistematis, karena tampaknya terlalu sulit, membosankan, atau memakan waktu. Mungkin kami mencoba program ini beberapa kali, dan kemudian berharap yang terbaik. CI menawarkan beberapa cara untuk membuat pengujian lebih mudah. Bahkan—jujur!—lebih menyenangkan.
Ada dua jenis tes utama:

- Tes unit: Ini mengambil pendekatan 'bottom-up'. Mereka melihat satu potongan kode Anda, mengatakan satu fungsi, memasukkan beberapa variabel, dan melihat apakah itu memberikan jawaban yang benar.
- Tes end-to-end: Ini adalah 'top-down'. Mereka fokus pada sesuatu yang seharusnya dilakukan situs, dan melihat apakah situs itu melakukannya: misalnya, mereka mencoba masuk ke situs Anda (menggunakan nama pengguna dan kata sandi yang valid) dan melihat apakah itu memungkinkan mereka. (Dan kemudian mereka mencoba masuk menggunakan kata sandi yang tidak valid…)

Seperti yang Anda lihat, ini adalah filosofi yang berbeda. Seseorang menguji potongan kode, dan tidak tahu atau peduli apa hasil akhirnya; sementara yang lain menguji hasil akhirnya, dan tidak tahu atau peduli potongan kode mana yang membawa Anda ke sana. Yang penting adalah memikirkan mengapa Anda menguji. Apa yang paling membuatmu khawatir? Apa yang paling mungkin salah dan mempermalukan Anda? Informasi seperti apa yang Anda perlukan

kembali dari pengujian Anda—hanya OK/tidak OK, atau sesuatu yang lebih kompleks? Untuk setiap aplikasi, berapa banyak waktu yang dapat Anda habiskan untuk menulis dan memelihara pengujian?

Saat kami mengembangkan situs pengujian kami, kami perlu menguji kode kami saat kami menulisnya. Tentu saja, kami mencoba mengantisipasi segala sesuatu yang mungkin dilakukan pengguna, dan setiap situasi yang mungkin muncul. Ini adalah salah satu area besar di mana pengujian unit berguna: fakta mendesain pengujian membantu Anda meningkatkan desain kode.

Setelah kode kami ada di server produksi, integritasnya sehari-hari sebagian besar berada di luar kendali kami. Paling buruk, ini menyebabkan klien menemukan pesan kesalahan atau layar kosong, dan mengharapkan Anda untuk melakukan sesuatu tentang hal itu, sering kali ketika Anda lebih suka melakukan sesuatu yang lain. Itu sebabnya kami membangun situs ini, untuk menguji situs jarak jauh lainnya.

CI dapat membantu kami memeriksa situs yang sedang berkembang untuk melihat:

- Pertama, bahwa kami telah mengantisipasi berbagai hal yang mungkin salah. Misalnya, saya mungkin melakukan kueri basis data untuk menghapus catatan dengan nomor ID yang diberikan dalam tabel tertentu. Ya, ini berhasil: Saya telah mengujinya dengan melakukannya. Tapi apa yang terjadi jika—entah bagaimana—kode memanggil tabel yang tidak ada? Atau memberikan nomor ID yang tidak valid? Atau tidak memberi sama sekali? Di sinilah unit test sangat membantu.
- Kedua, ketika saya menulis lebih banyak kode di tempat lain, apakah blok kode pertama saya masih berfungsi seperti yang saya inginkan, atau apakah saya secara tidak sengaja mengubah sesuatu yang menjadi sandaran blok pertama? Sekali lagi, pekerjaan untuk unit test. Mereka juga dapat membantu kami untuk memeriksa situs produksi secara teratur untuk melihat bahwa situs tersebut ada di sana (dan semua bagiannya—misalnya, jika database berada di server terpisah, tes 'ping' biasa tidak akan berhasil!)

CI memberi Anda banyak bantuan, posisi mana pun yang Anda ambil. Itu tidak memiliki kelas untuk menjalankan tes ujung ke ujung, tetapi Anda dapat melakukannya menggunakan kode PHP lain yang berada di luar cakupan buku ini. Tapi pertama-tama mari kita lihat bagaimana CI menampilkan kesalahan kepada Anda saat Anda mengembangkan kode.

8.2 CI'S ERROR HANDLING CLASS

CI memiliki sistem sendiri untuk mendeteksi dan melaporkan kesalahan. Di satu sisi, ini adalah tes paling sederhana dan paling umum dari semuanya: itu adalah pesan yang membantu (atau menyebalkan) yang Anda lihat ketika Anda mengembangkan kode Anda sendiri dan itu tidak berhasil. Secara default, CI menampilkan semua kesalahan di layar. Alternatifnya adalah gagal secara diam-diam; memberi Anda tidak tahu apa yang salah, jadi ini penting untuk pengembangan. Perilaku keseluruhan dikendalikan dari file index.php utama, yang dimulai:

```
/*
|---------------------------------------------------------------
| PHP ERROR REPORTING LEVEL
|---------------------------------------------------------------
|
| By default CI runs with error reporting set to ALL.  For security
| reasons you are encouraged to change this when your site goes live.
| For more info visit:  http://www.php.net/error_reporting
|
*/
error_reporting(E_ALL);
```

Ini adalah perintah PHP. Untuk mematikan pelaporan kesalahan, ganti baris terakhir dengan:

```
error_reporting(0);
```

Ini akan sesuai untuk situs produksi, di mana Anda tidak ingin detail kesalahan ditampilkan kepada pengguna.

CI memiliki tiga fungsi, show_error(), show_404(), dan log_message(), yang mengontrol bagaimana error ditampilkan pada sistem Anda. (Tidak biasanya, fungsi-fungsi ini tersedia secara global: Anda tidak perlu memuat apa pun sebelum dapat menggunakannya, langsung saja dan ketik!). Faktanya, show_error() dan show_404() biasanya terjadi secara default; yang pertama menampilkan kesalahan Anda dalam kotak kecil berformat HTML yang rapi di bagian atas layar; dan yang kedua menunjukkan halaman '404' jika Anda mencoba mengakses halaman yang tidak ada.

Fungsi ketiga, log_message(), lebih menarik. Anda mungkin ingin mengembangkan log kesalahan Anda sendiri, mungkin karena Anda tidak dapat mengakses log di server Apache ISP Anda. Pertama, Anda perlu mengatur izin untuk memastikan bahwa folder /system/logs Anda dapat ditulis. Kemudian Anda mengatur level logging di file konfigurasi:

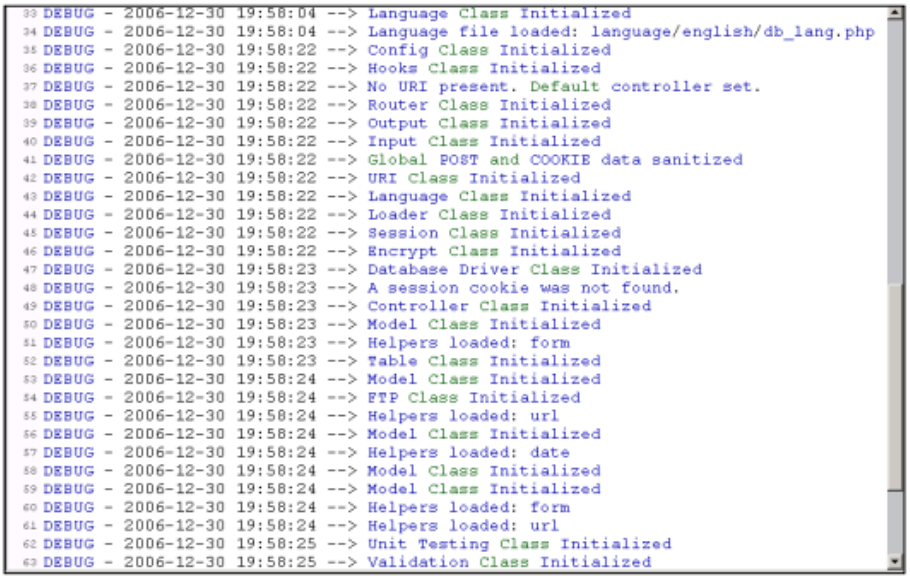
```
/*
|---------------------------------------------------------------
| Error Logging Threshold
|---------------------------------------------------------------
|
| If you have enabled error logging, you can set an error threshold to
| determine what gets logged. Threshold options are:
|
|   0 = Disables logging
|   0 = Error logging TURNED OFF
|   1 = Error Messages (including PHP errors)
|   2 = Debug Messages
|   3 = Informational Messages
|   4 = All Messages
|
| For a live site you'll usually only enable Errors (1) to be logged
| otherwise your log files will fill up very fast.
|
*/
```

```
$config['log_threshold'] = 4;
/*
```

Ini mulai masuk, secara otomatis.

Jika Anda mengubah index.php untuk mematikan tampilan layar pesan, ini tidak menghentikan pencatatan. Jadi Anda dapat melihat apa yang dilakukan sistem Anda, dan pengguna Anda tidak.

CI menghasilkan file log baru setiap hari, dan menulisnya saat Anda menginstruksikannya. Namun berhati-hatilah, file log ini dapat dengan cepat menjadi sangat besar. Berikut contoh salah satunya:



Gambar 8.1 Contoh Tampilan File Log baru tiap harinya

—dan seterusnya untuk 3000 baris lainnya, pada hari ketika program hanya memiliki penggunaan terbatas.

Dalam praktiknya, Anda mungkin ingin mengembangkan prosedur penanganan kesalahan Anda sendiri untuk menampilkan pesan default kepada pengguna saat terjadi kesalahan.

8.3 CI'S UNIT TEST CLASS

Sekarang mari kita ke pengujian yang tepat: secara proaktif melihat bit kode Anda untuk memastikan mereka bekerja dalam keadaan yang berbeda. CI membuat pengujian unit menjadi sederhana dengan kelasnya sendiri. Anda memuatnya dengan ini:

```
$this->load->library('unit_test');
```

dan kemudian, untuk setiap pengujian, Anda memutuskan tiga variabel:

- $test—tes yang sebenarnya, sebagai ekspresi PHP
- $expected_result—hasil yang Anda harapkan

- $test_name—nama tes yang Anda inginkan ditampilkan

Berikut adalah dua pengujian fungsi PHP floor() (yang membulatkan angka 'float' ke bilangan bulat terdekat di bawahnya). Perhatikan bahwa hasil yang diharapkan pertama adalah benar; yang kedua salah. (Kesalahan yang disengaja, jujur.)

```
$test = floor(1.56);
$expected_result = 1;
$test_name = 'tests php floor function';
$this->unit->run($test,  $expected_result, $test_name);
$test = floor(2.56);
$expected_result = 1;
$test_name = 'tests php floor function';
$this->unit->run($test,$expected_result,  $test_name);
```

menambahkan:

```
echo $this->unit->report();
```

menampilkan hasilnya sebagai HTML yang diformat, seperti ini

Test Name	tests php floor function
Test Datatype	Float
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\controllers\tests.php
Line Number	108

Test Name	tests php floor function
Test Datatype	Float
Expected Datatype	Integer
Result	Failed
File Name	E:\xampplite\htdocs\packt2\system\application\controllers\tests.php
Line Number	113

Gambar 8.2 Tampilan HTML untuk sintaks diatas

Jika Anda ingin sistem Anda menganalisis atau menyimpannya, gunakan:

```
echo $this->unit->result();
```

mengembalikan informasi sebagai array dua dimensi, yang dapat Anda gunakan:

```
Array (
        [0] => Array
                ([Test Name] => tests php floor function
                        [Test Datatype ] => Float
```

```
        [Expected Datatype] => Integer[Result]
        => Passed
        [File Name] => E:\myfile.php [Line Number] => 69 )
    [1] => Array
        ( [Test Name] => tests php floor function[Test Datatype ]
        => Float
        [Expected Datatype] => Integer[Result]
        => Failed
     [File Name] => E:\myfile.php[Line Number] => 73 )
```

Jadi sekarang kita memiliki cara mudah untuk mendapatkan hasilnya.

Selain membandingkan nilai (apakah floor(1.56) sama dengan 1?), kelas juga dapat menguji tipe data (is_string, is_bool, is_true, dll.—daftar lengkap ada di Panduan Pengguna online.) Anda mengganti:

```
$expected_result = 1;
```

baris dengan sesuatu seperti:

```
$expected_result = 'is_float';
```

dan tes berjalan seperti sebelumnya.

Jika Anda telah menyebarkan hal-hal ini di seluruh kode Anda, itu mungkin berjalan lambat, dan akan menampilkan semua jenis diagnostik di layar Anda. Tapi Anda bisa menghentikan ini. Cukup tambahkan baris berikut di konstruktor Anda:

```
$this->unit->active(FALSE);
```

Dan (kejutan, kejutan) jika Anda mengubah FALSE menjadi TRUE, kembali lagi. Anda bahkan dapat melakukan ini secara dinamis.

Kapan Menggunakan

Tidak ada gunanya menguji apakah fungsi PHP standar berfungsi. BENAR. Tetapi ada nilai dalam menguji fungsi Anda sendiri untuk melihat apakah mereka secara konsisten mengembalikan hasil yang diharapkan. Kekhawatiran utama selalu bahwa:

- Mereka akan berperilaku sempurna saat Anda mengujinya
- Tetapi pengguna akan segera memikirkan kombinasi keadaan yang tidak pernah Anda bayangkan, yang akan menyebabkan fungsi gagal
- Atau Anda akan menulis beberapa kode lagi, atau mengubah kode yang ada, dan fungsi Anda sendiri tidak akan berfungsi dengan baik lagi

Terkadang, kegagalan disebabkan oleh masalah pemrograman, yang dapat ditangkap oleh unit test. Anda bisa bersenang-senang memikirkan parameter yang berbeda untuk diuji.

Mari kembali ke contoh fungsi yang melakukan kueri basis data untuk menghapus catatan dengan ID tertentu dari tabel tertentu. Apa yang dilakukannya jika:

- ID adalah NULL, atau '', atau tidak disetel? (Sangat penting yang ini, karena Anda mungkin secara tidak sengaja menghapus setiap entri dalam tabel.)
- ID bukan bilangan bulat? ("x", misalnya?)
- ID adalah bilangan bulat, tetapi di luar jangkauan (Anda memiliki 1000 entri di tabel Anda, tapi ID ini 1001?)
- ID adalah angka negatif?

dan seterusnya. Cukup lucu memikirkan kondisi yang berbeda untuk diuji.

Lemparkan semuanya ke fungsi dengan unit test, dan lihat. Namun, pikirkan baik-baik tentang hasil yang Anda harapkan. Kasus pertama dan kasus kedua jelas merupakan kesalahan pemrograman. Anda harus menulis ulang program Anda untuk mencegah hal ini terjadi. Jadi jika ya, Anda ingin tes mengembalikan kegagalan.

Kami mendefinisikan hasil yang kami inginkan dari setiap pengujian, sehingga kami menguji apakah program bertindak dengan benar, bukan jika parameternya benar. Jika kami mengirimkan 'x', itu adalah parameter yang salah; tetapi jika program melempar pengecualian, itu adalah perilaku yang benar. Ini membantu untuk menulis semua tes untuk menunjukkan 'lulus' jika program melakukan apa yang kita inginkan, jadi kita hanya perlu memperhatikan pengecualian. Kasus ketiga di atas, di mana ID adalah bilangan bulat tetapi di luar jangkauan, belum tentu merupakan kesalahan pemrograman. Basis data harus dapat menyelesaikan ini dengan aman. Namun, apa yang perlu Anda lakukan tergantung pada program Anda dan tujuannya. Mungkin, sebelum Anda mengirimkan nilai integer, Anda perlu memeriksa apakah itu dalam jangkauan? Atau mungkin Anda senang membiarkannya berjalan, dalam hal ini, karena program mungkin mengembalikan pesan kesalahan basis data ke layar Anda, Anda perlu mencegat ini dan menggantinya dengan kesalahan 'maaf, tidak dapat melakukan ini saat ini'. pesan? Atau mungkin menguji operasi penghapusan untuk sukses dan bercabang?

Contoh

Mari kita buat beberapa kode untuk menguji fungsi 'hapus'. Saya telah menyiapkan fungsi 'hapus' (yang ada dalam model) sehingga mendeteksi bahwa saya sedang mengujinya, dan jika gagal mengembalikan nilai ($dbvalue)

```
if($test == 'yes')
{
$place = __FILE__._LINE__;;
$dbvalue = "exception at $place: sent state value $state to
trydelete function ";
return $dbvalue;
}
```

Jika pengujian berhasil, loop serupa mengembalikan nilai $db dari 'OK'. Kode tesnya sederhana. Pertama, kita membangun sebuah array nilai ID dan hasil yang kita harapkan dari

masing-masing. Dengan kata lain, jika kita mencoba menghapus menggunakan ID '' atau 'abc', sistem harus mengeluarkan pengecualian, mengawali baris diagnostik yang dikembalikan dengan kata 'pengecualian'. Jika 1, atau 9999, sistem harus menerimanya sebagai ID yang valid, dan awali baris yang dikembalikannya dengan 'Oke'.

Jadi kunci array adalah kondisi yang Anda uji dan nilai array adalah hasil yang Anda harapkan dari fungsi tersebut.

```
$numbers = array
(
        “       => 'exception',
        ‘NULL’  => 'exception',
        ‘x’     => ‘OK’
        ‘9999’  =>
        ‘-1’    => 'exception',
        ‘1’     => ‘OK’
);
```

Sekarang gunakan kode berikut untuk mengulang melalui array $numbers dan gunakan pengujian unit CI untuk melakukan setiap pengujian.

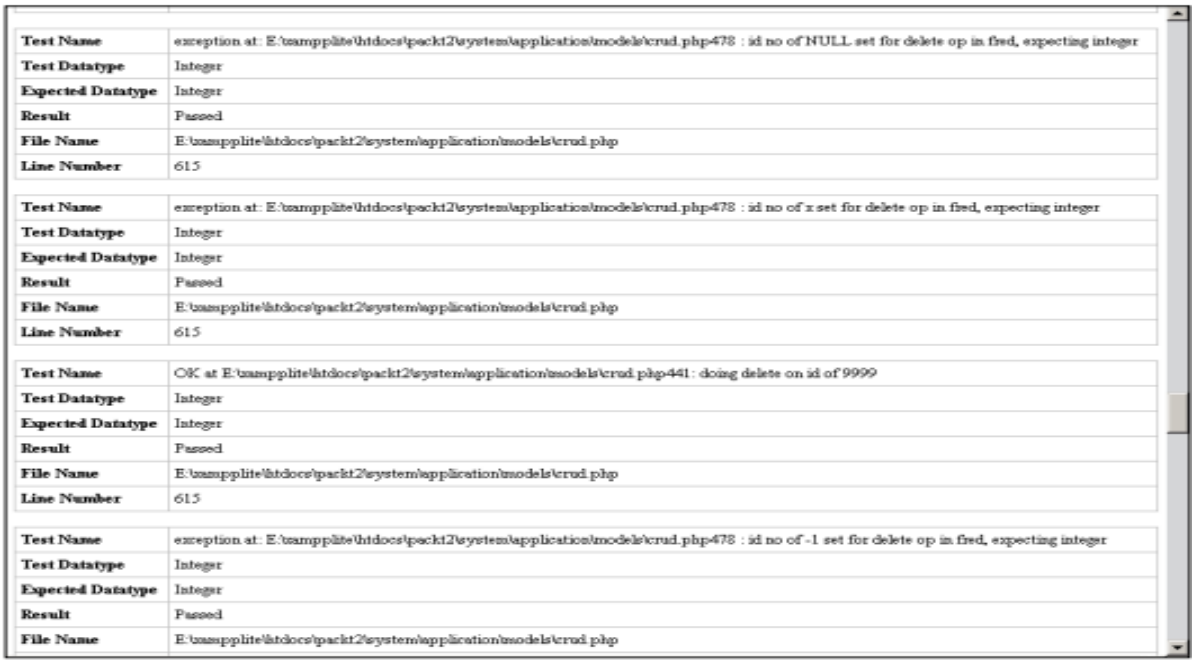
Pengujiannya adalah dengan menjalankan fungsi $this->delete(), dengan menentukan tabel yang akan Anda hapus ('fred') dan nilai ID ($testkey).

```
foreach($numbers AS $testkey => $testvalue)
{$dbvalue = $this->delete('fred', $testkey);
$result .= $this->unit->run(preg_match("/$testvalue/",
$dbvalue), 1, $dbvalue);
}
```

Ingat, pengujian unit CI memungkinkan Anda memberikan tiga parameter:

- $test: untuk setiap kunci array, kita mencoba database dengan memanggil fungsi delete dengan $testkey, kunci array—yang merupakan ID (atau kekurangan ID) yang kita kirimkan ke fungsi tersebut. Fungsi mengembalikan nilai (di sini disebut $dbvalue). $test kita adalah membandingkan nilai itu, menggunakan regex, dengan apa yang kita harapkan—yaitu $testvalue, nilai array. (Apakah itu termasuk 'OK' atau 'pengecualian'?)
- $expectedresult adalah '1', karena jika kode kita benar, kita mengharapkan regex untuk menemukan kecocokan. Kami berharap 'NULL' melempar pengecualian dan 1 menjadi OK.
- $testname: parameter ini opsional: ini adalah string yang dikembalikan oleh pengujian, yang selanjutnya menjelaskan nilai apa yang kami coba gunakan dan apa yang kami uji.

Inilah hasilnya, seperti yang ditunjukkan dalam format HTML bawaan CI: Semua mengembalikan Hasil 'lulus', sehingga kami dapat yakin dengan kode kami. (Tipe Data Uji dan Tipe Data yang Diharapkan selalu ditampilkan sebagai bilangan bulat, meskipun input kita mungkin bukan bilangan bulat, karena pengujian sebenarnya adalah perbandingan regex, yang mengembalikan 1 atau 0).



Test Name	exception at: E:\xampplite\htdocs\packt2\system\application\models\crud.php:478 : id no of NULL set for delete op in fred, expecting integer
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	615

Test Name	exception at: E:\xampplite\htdocs\packt2\system\application\models\crud.php:478 : id no of x set for delete op in fred, expecting integer
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	615

Test Name	OK at E:\xampplite\htdocs\packt2\system\application\models\crud.php:441 : doing delete on id of 9999
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	615

Test Name	exception at: E:\xampplite\htdocs\packt2\system\application\models\crud.php:478 : id no of -1 set for delete op in fred, expecting integer
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php

Gambar 8.3 Tampilan hasil HTML array pada sintaks diatas

Yang menyenangkan, dan sebenarnya cukup berguna, adalah memikirkan tes baru untuk dimasukkan ke dalam array!

Misalnya, bagaimana jika 'id' adalah angka, tetapi bukan bilangan bulat? Untuk mencoba ini dengan kode di atas, Anda akan menambahkan ke array nilai tes.

```
'3.5'    =>    'exception',
```

Namun, Anda akan terkejut (seperti saya) menemukan bahwa tes ini gagal—dengan kata lain, ini menunjukkan bahwa fungsi Anda akan menerima 3,5 sebagai bilangan bulat. Alasannya adalah PHP melakukan uji kesetaraan 'longgar'; ia menemukan nomor dan mengambil ini untuk bilangan bulat. Yang Anda butuhkan dalam kasus ini adalah mode pengujian 'ketat' yang membandingkan tipe data serta nilainya. Untuk mengatur ini, gunakan:

```
$this->unit->use_strict(TRUE);
```

8.4 CI'S BENCHMARKING CLASS

Kelas ini memungkinkan Anda untuk mengukur waktu yang dibutuhkan program Anda untuk beralih dari satu menunjuk ke yang lain. Anda memasukkan satu baris kode untuk menentukan poin pertama:

```
$this->benchmark->mark('here');
```

dan baris kode lain untuk mendefinisikan yang kedua:

```
$this->benchmark->mark('there');
```

lalu Anda memasukkan baris ketiga untuk memberi tahu Anda waktu yang telah berlalu:

```
$fred =  $this->benchmark->elapsed_time('here',  'there');
```

Anda kemudian dapat mencetak hasilnya, $fred, atau melakukan apa pun yang ingin Anda lakukan dengannya.

Tolok ukur dapat memiliki nama apa pun yang Anda suka, asalkan berbeda, dan Anda dapat memiliki pasangan sebanyak yang Anda inginkan. Anda dapat menggunakan tes ini untuk melihat apakah bagian tertentu dari kode Anda membutuhkan waktu lama yang mencurigakan. Jika salah satu halaman Anda membutuhkan waktu terlalu lama untuk dimuat, Anda dapat menyisipkan beberapa tolok ukur untuk mengidentifikasi bagian kode sebenarnya yang menyebabkan penundaan.

Namun, untuk pengujian dalam aplikasi pemantauan situs web kami, kami tidak begitu tertarik pada waktu sekali saja. Pada saat kami mendapatkan aplikasi kami di Internet, kami berharap kecepatannya dapat diterima. Perbedaan mutlak antara waktu cenderung kecil dan sebagian besar tidak berarti. Namun, jika kita melacak tolok ukur pada beberapa tes berturut-turut, kita mungkin memperhatikan bahwa itu berubah: dan ini mungkin memberi kita petunjuk untuk beberapa masalah mendasar. Kueri basis data mungkin membutuhkan waktu lebih lama; atau hosting kami mungkin kurang efektif. Jadi untuk tujuan kita, kita akan mengambil isi $fred, dan menyimpannya di database kita.

8.5 CI'S PROFILER CLASS

Kelas profiler sangat brilian. Anda meletakkan satu baris kode di suatu tempat di dalam suatu fungsi di kelas Anda. (Ini bekerja dari dalam konstruktor, jadi masuk akal untuk meletakkannya di sana.) Baris itu adalah:

```
$this->output->enable_profiler(TRUE);
```

Jika Anda berubah pikiran, hapus atau ubah ke:

```
$this->output->enable_profiler(FALSE);
```

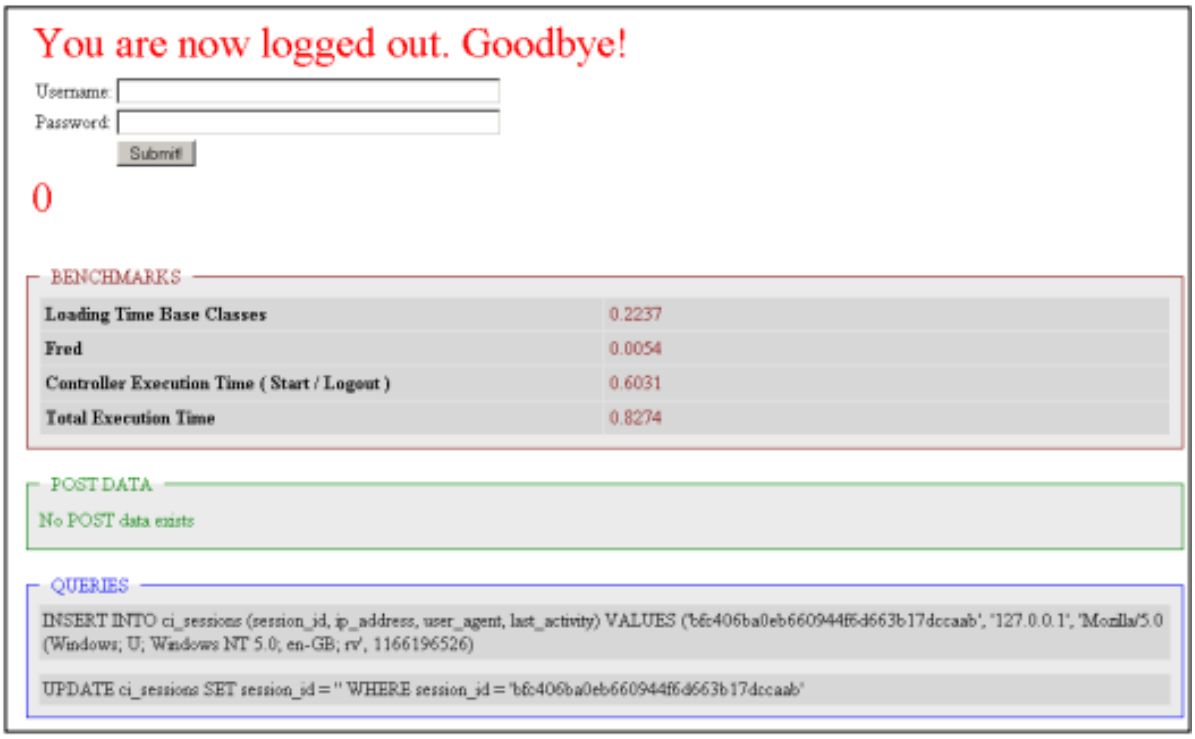
Sebagai gantinya untuk satu baris kode ini, Anda mendapatkan laporan lengkap di layar Anda, memberi Anda detail waktu yang dibutuhkan CI untuk memuat dirinya sendiri dan pengontrol Anda, tentang apa yang ada di larik POST, dan kueri basis data apa pun yang telah Anda jalankan. Pada tahap pengembangan kode, ini sangat membantu.

Jika Anda menambahkan tolok ukur Anda sendiri, itu juga akan menampilkannya. Anda harus menggunakan nama khusus untuk tolok ukur Anda—masing-masing harus diakhiri dengan _start dan _end, dan setiap pasangan harus memiliki nama yang sama:

```
$this->benchmark->mark('fred_start');
```

dan, di tempat lain,

```
$this->benchmark->mark('fred_end');
```



Gambar 8.4 contoh tampilan Profiler Class

Seperti yang Anda lihat, waktu yang berlalu antara dua tolok ukur ini kemudian ditampilkan sebagai 'fred'.

8.6 MENGUJI DENGAN BASIS DATA MOCK

Situs dinamis adalah semua tentang database. Jika Anda mengujinya dengan benar, Anda harus menguji apakah kode Anda benar-benar dapat memodifikasi database. Pengujian ujung-ke-ujung melakukan ini: misalnya, jika pengujian Anda adalah apakah Anda dapat masuk dengan kombinasi nama pengguna-kata sandi yang benar, Anda mungkin membaca database untuk melakukannya.

Tetapi menguji apakah Anda dapat memperbarui, membuat, dan menghapus entri pada basis data produksi adalah hobi yang berbahaya, karena merusak data asli Anda! Ingat bahwa CI memungkinkan Anda untuk mendeklarasikan lebih dari satu database, dan untuk bertukar di antara mereka dengan mudah?—lihat Bab 4. Dengan menggunakan ini, mudah untuk membuat database tiruan, dan kemudian menambah, mengubah, dan menghapus data di dalamnya.

Anda juga dapat menggunakan CI untuk menyiapkan dan merobohkan tabel, atau bahkan mungkin seluruh database tergantung pada host dan tingkat izin Anda. CI:

```
$this->db->query('YOUR QUERY HERE');
```

fungsi memungkinkan Anda untuk menjalankan kueri SQL apa pun, termasuk sesuatu seperti ini:

```
$this->db->query('CREATE TABLE fred(id INT, name VARCHAR(12), INDEX(id))');
```

yang akan membuat tabel baru, atau sesuatu seperti ini:

```
$this->db->query("INSERT INTO fred VALUES (1, 'smith')");
```

yang akan mengisi tabel itu dengan satu baris data.

Jadi, dengan beberapa baris kode, CI memungkinkan Anda menyiapkan kumpulan data tiruan yang sama sekali baru untuk diuji, bermain-main dengannya, menguji hasilnya, dan kemudian menghapusnya siap untuk waktu berikutnya. Anda kemudian dapat menjalankan serangkaian pengujian unit pada fungsi delete() untuk melihat apakah fungsi tersebut melakukan apa yang kita harapkan ketika memiliki parameter 'id' yang berbeda, seperti yang dijelaskan sebelumnya dalam bab ini.

Sekarang Anda melampaui pengujian unit sederhana. Jika kita menjalankan tes yang seharusnya menghapus nilai di tabel kita, maka kita perlu memeriksa apakah itu benar-benar telah dihapus. Ini mudah dilakukan dengan kode berikut, sekali lagi menggunakan kelas pengujian unit CI, dan kelas rekaman aktifnya:

```
$test = $this->db->count_all('fred');
$expected_result = 0;
$test_name = 'tests number of entries left in table after unique entry removed';
$this->unit->run($test, $expected_result, $test_name);
```

$this->db->count_all menghitung semua hasil dalam tabel, dan kita tahu bahwa sekarang seharusnya tidak ada hasil sama sekali. Anda dapat dengan mudah menggunakan kode semacam ini untuk memeriksa operasi 'masukkan', untuk melihat apakah ada satu catatan lagi di tabel sesudahnya.

Karena ini adalah data dummy, yang kami buat khusus untuk pengujian, kami tahu persis apa yang diharapkan, dan tidak masalah apa yang kami lakukan untuk itu. Ingatlah untuk menghancurkan dan membangun kembali tabel di antara tes, jika tidak, Anda mungkin mendapatkan hasil yang aneh.

8.7 KONTROL DAN WAKTU

Pengujian adalah inti dari aplikasi kami, jadi inilah sedikit penjelasan tentang cara mengontrolnya. Anda akan melihat dari spesifikasi database di akhir Bab 4 bahwa aplikasi kita memiliki tabel yang disebut tes, dan tabel yang disebut peristiwa. Setiap kali situs diminta untuk melakukan uji coba, ia melewati tabel pengujian, mencari dua bidang: frekuensi dan last_done. Jika frekuensinya, katakanlah, setiap jam, ia akan memeriksa bidang last_done untuk melihat apakah pengujian telah dilakukan dalam waktu satu jam dari waktu saat ini. Jika tidak, ia melakukan pengujian sekarang, dan memperbarui bidang last_done-nya sendiri ke waktu saat ini.

Namun, ketika pengujian selesai, program membuat entri di tabel peristiwa. Ini memberikan ID situs, dan berbagai informasi lainnya, terutama hasil pengujian. Tabel ini kemudian memberikan statistik yang dapat kita gunakan untuk membuat laporan untuk diri

kita sendiri atau untuk klien, memilah pengujian individu, atau semua pengujian di situs tertentu, dll.

Sebagai pengingat kelas benchmarking yang kita bahas sebelumnya dalam bab ini: ketika Anda menjalankan tes dari fungsi seperti yang baru saja kita diskusikan, ada baiknya untuk memasukkan benchmark sehingga Anda tahu berapa lama tes berlangsung. Menghemat waktu ke tabel acara. Ada bidang dalam tabel ini untuk timetake: ini adalah tipe data float karena kita berurusan dengan sepersekian detik. Meskipun waktu yang dibutuhkan untuk menjalankan tes tunggal mungkin tidak terlalu menarik, perubahan pola saat tes dijalankan berkali-kali pada interval mungkin menarik: perubahan tersebut dapat menunjukkan kepada Anda apakah tes semakin cepat atau lambat. Jika, misalnya, diperlukan waktu lebih lama untuk masuk ke halaman Anda, ini mungkin karena ISP Anda membebani server secara berlebihan; mungkin karena Anda meletakkan terlalu banyak kode di halaman Anda; atau mungkin situs Anda menjadi terlalu populer dan Anda membutuhkan lebih banyak bandwidth. Menguji secara teratur dan menyajikan hasil agregat dalam format yang dapat digunakan dapat memberi Anda peringatan dini yang berguna tentang masalah.

8.8 RINGKASAN

Kami telah menghabiskan banyak waktu untuk pengujian. Ini bukan subjek yang paling menarik, tetapi jika Anda menulis situs web yang penting, ini adalah cara yang bagus untuk memastikan Anda tidur nyenyak di malam hari. Kami telah melihat bagaimana CI biasanya menangani kesalahan, menampilkannya kepada Anda saat Anda menulis kode, tetapi memungkinkan Anda untuk mematikannya (atau mengalihkannya ke file log) saat situs Anda masuk ke mode produksi. Kami melihat tes unit dan alat CI untuk menangani ini. Kami juga melihat benchmarking, kelas yang memudahkan Anda untuk memantau waktu eksekusi bagian yang berbeda dari program Anda.

Alat profiler sangat bagus untuk menunjukkan banyak informasi tentang kode Anda saat Anda mengembangkannya. CI menawarkan seperangkat alat yang bagus untuk mengembangkan dan menguji kode Anda sendiri. Kami juga melihat cara pengujian dengan tabel database dummy, untuk melihat apakah operasi database benar-benar terjadi seperti yang kami harapkan. Kami kemudian mengintegrasikan beberapa kode eksternal dengan CI, untuk memungkinkan kami membangun robot web dan oleh karena itu menjalankan pengujian 'end-to-end' kami sendiri di situs jarak jauh.

BAB 9
MENGGUNAKAN CI UNTUK BERKOMUNIKASI

Kekuatan utama Internet adalah kemampuannya untuk berkomunikasi. Bab ini membahas tiga cara di mana CI membuat komunikasi lebih mudah.
Pertama, kami akan menambahkan toolkit pengujian kami dengan menggunakan kelas FTP CI untuk mengakses file jarak jauh secara langsung. Kemudian, kami akan menggunakan kelas email untuk membuat situs kami secara otomatis mengirim email kepada kami ketika kondisi tertentu terpenuhi. Terakhir, kami akan menjelajah ke wilayah Web 2.0—menggunakan XML-RPC untuk membuat 'layanan web' pribadi yang memungkinkan situs jarak jauh kami mengambil tindakan dan mengembalikan informasi pada permintaan dari situs pengujian kami.

9.1 MENGGUNAKAN KELAS FTP UNTUK MENGUJI FILE JARAK JAUH

File Transfer Protocol (FTP) adalah metode mentransfer file melalui Internet. Biasanya digunakan untuk memindahkan file ke belakang dan ke depan ke situs web Anda, menggunakan program FTP khusus. Itu adalah sesuatu yang kebanyakan dari kita hanya gunakan sesekali, ketika kita memasang situs baru.

Namun, Anda dapat mengotomatiskan seluruh proses tanpa rasa sakit dengan CI. Salah satu kegunaannya adalah untuk menguji integritas situs jarak jauh Anda: apakah file-file tersebut masih ada? Sebagai pemilik situs web, Anda selalu menghadapi kemungkinan seseorang akan mengutak-atik file di situs Anda. Mungkin saja ISP Anda atau admin server Anda, salah menghapus atau menulis sesuatu secara berlebihan. (Saya pernah mengalami hal ini pada saya, ketika ISP saya membangun kembali server mereka dan lupa memuat ulang salah satu file aplikasi saya. File yang bersangkutan tidak terlalu sering digunakan, tetapi sangat penting ketika itu. Hal ini menyebabkan kesalahan yang menarik bahwa mengambil beberapa waktu untuk melacak!)

Sebagai salah satu contoh kekuatan kelas FTP, mari kita buat program pengujian reguler, untuk periksa file di situs jarak jauh. Hanya beberapa baris kode yang kita butuhkan:

```
function getremotefiles($hostname, $username,  $password)
    {
     $this->load->library('ftp');
    $config['hostname'] = $hostname;
    $config['username'] = $username;
    $config['password'] = $password;
    $config['debug']          = TRUE;
    $this->ftp->connect($config);
    $filelist = $this->ftp->list_files('/my_directory/');
    $this->ftp->close(); return
    $list;
    }
```

Pertama, muat pustaka FTP jika Anda belum melakukannya. Kemudian, tentukan parameter konfigurasi: nama host (mis., www.mysite.com), nama pengguna, dan kata sandi untuk akses FTP Anda.

Setelah terhubung, kelas FTP CI memberi Anda beberapa opsi. Dalam hal ini, kami telah menggunakan list_files() untuk mengembalikan daftar file di folder /my_directory/. Fungsi mengembalikan array, dan Anda dapat dengan mudah memeriksanya terhadap array file yang Anda harapkan akan ditemukan di sana. Seperti sebelumnya, kami mencoba membuat daftar semua pengujian kami dalam database. Jadi kali ini kita perlu mendaftar URL FTP (atau nama host), nama pengguna dan kata sandi, dan alih-alih regex, larik file untuk diperiksa. Untuk menjaga integritas array ini, jika Anda menyimpannya di dalam database Anda, Anda perlu membuat serialisasi sebelum Anda memasukkannya, dan membatalkan serialisasi ketika Anda mengeluarkannya lagi.

Maka mudah untuk membandingkan $remotearray yang dikembalikan oleh getremotefiles() fungsi dengan $referencearray yang tidak diserialkan yang dikembalikan oleh database Anda:

```
function comparefiles($remotearray, $referencearray)
{
    $report = "<br />On site, not in reference array: ";
    $report .= print_r(array_diff($remotearray, $referencearray), TRUE);
    $report .= "<br />In reference array, not on site: ";
    $report .= print_r(array_diff($referencearray, $remotearray), TRUE); return $report;
}
```

Fungsi PHP array_diff membandingkan larik kedua dengan larik pertama, sehingga akan mencantumkan file yang ada di larik pertama, tetapi tidak di larik kedua. Jadi, jalankan fungsi dua kali, membalik urutan parameter larik: dengan cara itu Anda mendapatkan dua daftar, satu dari apa yang tidak ada di situs Anda (tetapi seharusnya ada) dan salah satu dari apa yang ada di situs Anda (tetapi tidak seharusnya). Yang pertama harus menunjukkan file apa pun yang tidak sengaja dihapus oleh ISP Anda, yang kedua file apa pun yang mungkin telah ditambahkan.

Kelas CI FTP juga memungkinkan Anda mengunggah, memindahkan, mengganti nama, dan menghapus file. Misalkan pengujian Anda di atas menunjukkan bahwa salah satu file dalam array referensi Anda (sebut saja myfile.php hilang dari situs Anda. Anda dapat menggunakan kelas FTP untuk mengunggahnya:

```
$this->ftp->upload('c:/myfile.php', '/public_html/myfile.php');
```

Dalam contoh ini, jalur lokal diberikan terlebih dahulu, dan jalur di server jauh kedua. Secara opsional, Anda dapat menentukan dalam parameter ketiga bagaimana file harus diunggah (sebagai ASCII atau biner.) Jika tidak, CI membuat keputusannya sendiri berdasarkan ekstensi file—yang biasanya benar. Jika Anda menjalankan PHP5, Anda dapat menambahkan parameter keempat untuk mengatur izin file, dengan asumsi Anda mengunggah ke server Linux.

Berhati-hatilah dengan opsi hapus. Seperti yang dikatakan panduan pengguna, "Ini akan menghapus secara rekursif semua yang ada di jalur yang disediakan, termasuk sub-folder dan semua file"". Bahkan menulis paragraf ini membuat saya gugup. Menggunakan kombinasi fungsi penghapusan dan unggah FTP, Anda dapat memperbarui file secara otomatis di situs jarak jauh Anda. Buat daftar file yang perlu Anda perbarui, dan kunjungi setiap situs secara bergantian, pertama-tama hapus setiap yang lama, lalu unggah setiap yang baru. Ada juga fungsi 'mirror' yang menarik, yang memungkinkan Anda mengatur duplikat lengkap situs web di server lain. Jika Anda menjalankan PHP5, kelas FTP juga memiliki fungsi yang memungkinkan Anda mengubah izin file. Seperti yang Anda lihat, ada banyak ruang untuk memperluas aplikasi Anda dari menguji situs web jarak jauh hingga benar-benar memelihara atau memperbaruinya. Anda dapat, misalnya, menulis kode untuk mendistribusikan pembaruan secara otomatis.

9.2 MESIN BERBICARA DENGAN MESIN LAGI—XML-RPC

Revolusi Web 2.0 sebagian besar dibangun di atas antarmuka mesin-ke-mesin, yang memungkinkan mashup dan API dan semua hal baik itu. Ini adalah dasar dari 'layanan web'. Anda dapat menawarkan antarmuka ke situs Anda yang memungkinkan orang lain menggunakannya untuk melakukan sesuatu untuk mereka. Untuk memberikan contoh sederhana, jika Anda menyiapkan 'layanan web' yang mengubah suhu dalam celcius ke Fahrenheit, klien mengirimkan permintaan dengan satu parameter (suhu yang akan dikonversi) dan server mengembalikan nilai yang dikonversi. Jadi, siapa pun dapat menambahkan fungsi konversi suhu yang tampaknya ada di situsnya sendiri, tetapi sebenarnya memanggil situs Anda.

XML-RPC memungkinkan dua mesin untuk berbicara secara langsung. Situs penerima membuat API sederhana (antarmuka pemrograman aplikasi). Siapa pun yang ingin berbicara dengannya perlu mengetahui bahwa API—metode apa yang tersedia, parameter apa yang mereka ambil, dan apa sintaksnya—untuk menanganinya. Banyak situs besar menggunakan sistem ini: Google, misalnya, memungkinkan Anda melakukan panggilan langsung ke mesin pencarinya atau ke Google Earth melalui API yang diterbitkan.

Menyiapkan API pribadi Anda sendiri relatif mudah, berkat CI. Anda memerlukan dua situs web untuk mengatur ini dan mengujinya, yang membuatnya sedikit lebih rumit daripada kebanyakan hal. Satu situs (sebut saja situs 'penerima') adalah situs yang menawarkan API, mendengarkan permintaan, dan menjawabnya. (Dalam contoh kami, ini adalah salah satu situs jarak jauh yang kami coba uji dan kelola.) Situs lain membuat permintaan menggunakan API dan mendapatkan jawabannya kembali. (Dalam contoh kami, ini adalah situs pengujian itu sendiri.)

Dalam protokol XML-RPC, kedua situs berbicara melalui XML yang sangat terstruktur. (Oleh karena itu nama XML-RPC—kependekan dari XML Remote Procedure Call.) Klien mengirimkan paket XML ke server 'situs penerima', menyatakan fungsi yang ingin digunakan dan argumen atau parameter apa pun yang akan diteruskan. Server mendekode XML dan, jika cocok dengan API, memanggil fungsi dan mengembalikan respons, juga terstruktur sebagai XML, yang didekode dan ditindaklanjuti oleh klien.

API Anda terdiri dari fungsi-fungsi yang ditawarkan situs penerima, dan instruksi tentang cara menggunakannya—misalnya, parameter apa yang mereka ambil, tipe data apa yang seharusnya, dll.

Di situs penerima, kami membuat server XML-RPC, yang membuat metode internal yang dipilih tersedia untuk situs eksternal. 'Metode internal' ini sebenarnya hanyalah fungsi normal dalam salah satu pengontrol Anda: peran server adalah menangani antarmuka antara panggilan eksternal dan fungsi internal.

Ada dua rangkaian masalah saat Anda menyiapkan proses XML-RPC:

- Membuat kedua situs saling berbicara
- Memastikan bahwa data dikirimkan dalam format yang sesuai

Keduanya sangat bergantung pada susunan multi-dimensi, yang dapat diambil oleh mesin, bahkan jika manusia perlu sedikit memikirkannya. CI membuatnya jauh lebih mudah—walaupun masih cukup sulit untuk memperbaikinya.

Menghubungkan Server XML-RPC dan Klien Satu Sama Lain

Pertama, Anda harus menyiapkan server di situs jarak jauh, dan klien di situs yang meminta. Ini dapat dilakukan dengan beberapa baris kode sederhana. Katakanlah kita melakukan server di controller yang disebut 'mycontroller' (di situs penerima) dan klien di controller yang disebut 'xmlrpc_client' (di situs yang meminta).

Dalam setiap kasus, mulailah dengan menginisialisasi kelas CI di dalam konstruktor. Ada dua; untuk klien Anda hanya perlu memuat terlebih dahulu, untuk server Anda perlu memuat mereka berdua:

```
$this->load->library('xmlrpc');
$this->load->library('xmlrpcs');
```

Sekarang, untuk servernya. Tutup fungsi konstruktor Anda, dan di dalam fungsi 'mycontroller' index(), tentukan fungsi yang Anda tawarkan ke dunia luar. Anda melakukan ini dengan membangun sub-array 'fungsi' (dalam array CI $config utama) yang memetakan nama-nama permintaan yang masuk ke fungsi aktual yang ingin Anda gunakan:

```
$config['functions']['call'] = array('function' => 'mycontroller.myfunction');
$config['functions']['call2'] = array('function' => 'mycontroller.myfunction2');
```

Dalam hal ini, ada dua panggilan fungsi bernama—'panggilan' dan 'panggilan2'. Inilah yang diminta oleh permintaan. (Ini tidak menanyakan fungsi berdasarkan nama, tetapi dengan nama panggilan. Tentu saja, Anda dapat menggunakan nama yang sama jika diinginkan.) Untuk setiap panggilan, Anda mendefinisikan sub-sub-array yang memberikan fungsi ' ' di dalam pengontrol—yaitu. 'myfunction' dan 'myfunction2' masing-masing.

Anda kemudian menyelesaikan server Anda dengan menginisialisasi dan membuat instance:

```
$this->xmlrpcs->initialize($config);
$this->xmlrpcs->serve();
```

dan sekarang siap untuk mendengarkan permintaan.

Sekarang Anda perlu membuka situs web lain—klien—dan menyiapkan klien XML-RPC untuk membuat permintaan. Ini harus menjadi pengontrol terpisah di situs klien Anda. Ini cukup singkat:

```
$server_url = 'http://www.mysite.com/index.php/mycontroller';
$this->load->library('xmlrpc');
$this->xmlrpc->set_debug(TRUE);
$this->xmlrpc->server($server_url,  80);
$this->xmlrpc->method('call');
```

Anda menentukan URL situs penerima, menentukan pengontrol yang berisi server XML-RPC yang Anda inginkan. Anda memuat kelas XML-RPC, menentukan server, dan metode yang ingin Anda gunakan—ini adalah nama panggilan yang ingin Anda buat, bukan fungsi sebenarnya yang ingin Anda gunakan. Jika fungsi yang Anda panggil membutuhkan parameter, Anda meneruskannya dengan cara ini:

```
$request = array('optimisation','sites');
```

Seperti yang Anda lihat, kami melewati dua di sini.
Kemudian, Anda memeriksa apakah respons telah diterima, dan melakukan sesuatu dengannya:

```
if ( ! $this->xmlrpc->send_request())
{
echo $this->xmlrpc->display_error();
}
else
{
print_r($this->xmlrpc->display_response());
}
```

Opsi paling sederhana adalah menampilkannya; tetapi dalam aplikasi nyata Anda lebih cenderung ingin mesin menganalisisnya, misalnya, dengan menggunakan regex, dan kemudian bertindak berdasarkan hasilnya. Misalnya, jika hasilnya berisi pesan kesalahan, Anda mungkin ingin merekam kesalahan dalam database Anda, dan mengambil tindakan untuk melaporkannya ke pengguna manusia.

Memformat Pertukaran XML-RPC

Mari kita gunakan contoh nyata, jika disederhanakan. Dalam sseeccttiionn ini,, kami akan membuat respons panggilan XML-RPC yang memungkinkan Anda memicu optimasi database dari jarak jauh. Klien yang kami tulis di atas, meminta metode yang dikenal sebagai 'panggilan' dan menyediakan dua parameter: 'pengoptimalan' dan 'situs'. Server di situs penerima memetakan permintaan 'panggilan' ini ke fungsi yang disebut 'fungsi saya'.

Mari kita lihat fungsi ini. Ini pada dasarnya adalah fungsi biasa di dalam pengontrol. Ini mencoba untuk mengoptimalkan tabel database MySQL, dan mengembalikan 'sukses' atau 'gagal' tergantung pada hasilnya.

```
function myfunction($request)
{
        $parameters = $request->output_parameters();
        $function = $parameters['0'];
        $table =   $parameters['1'];
        if ($this->db->query("OPTIMIZE  TABLE $table"))
{
        $content = 'Success';
}
        else
        {
        $content = 'failure';
        }
 $response = array(
        array(
        'function'          => array($function, 'string'),
        'table'    => array($table, 'string'),
        'result'   => array($content, 'string'),
),
        'struct');
        return $this->xmlrpc->send_response($response);
}
```

Perhatikan $request, tetapkan sebagai parameter fungsi. Ini berisi larik $request dari klien—ingat, ia memiliki dua nilai, 'optimasi' dan 'situs'. CI telah mengubah array menjadi objek, $request. Jadi Anda tidak bisa mendapatkan parameter individual dengan memperlakukannya sebagai array, sebagai gantinya Anda harus menggunakan $request->output_parameters() metode objek $request. Ini mengembalikan array, yang Anda interogasi dengan cara biasa.

Dengan menggunakan ini, kami telah memberi tahu fungsi di situs penerima tabel mana yang ingin kami optimalkan, tabel 'situs'. Kami juga telah memberi tahu apa yang harus dipanggil fungsi ('pengoptimalan'). Itu menambahkan parameter lebih lanjut yang disebut 'hasil', mendapatkan nilai, dan mengembalikan ketiganya kepada kami.

Hasil yang dikirim kembali ke situs klien terlihat seperti ini:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
    <params>
     <param>
    <value>
     <struct>
      <member>
         <name>function</name>
          <value>
            <string>optimisation</string>
```

```
                    </value>
                  </member>
                <member>
<name>table</name>
                  <value>
                    <string>sites</string>
                  </value>
                </member>
                <member>
                  <name>result</name>
                  <value>
                    <string>Success</string>
                      </value>
                  </member>
                    </struct>
                </value>
              </param>
            </params>
</methodResponse>
```

(Kecuali itu tidak menjorok: Saya melakukan itu untuk membuat struktur lebih jelas.)

Seperti yang Anda lihat, respons tiga kata sederhana kami (pengoptimalan, latihan, kesuksesan) telah dibungkus dalam lapisan tag yang bertele-tele, dengan cara yang sangat khas XML, untuk memberi tahu mesin apa yang sedang terjadi. Ada tiga pasangan tag <member></member>. Masing-masing memiliki pasangan <name></name> ('fungsi', 'tabel', 'hasil' masing-masing). Dan masing-masing memiliki pasangan <value></value>, yang menyertakan (serta tipe data) informasi aktual yang kita inginkan—yaitu. 'pengoptimalan', 'situs', 'sukses'.

Tidak peduli bahwa saya tidak menyukainya. Komputer berkembang pesat dalam hal-hal semacam ini: presisi, tidak ambigu, dan mudah dibaca oleh mesin. Bab ini adalah tentang komputer yang berbicara satu sama lain, bukan tentang antarmuka yang ramah pengguna. Sekarang, fungsi klien XML-RPC Anda di situs panggilan Anda dapat mengekstrak nilai yang diinginkan dan bertindak berdasarkan nilai tersebut. Sangat mudah untuk melakukan ini dengan regex, karena setiap jawaban jelas dibatasi oleh tanda kurung mark-up XML. Perhatikan bagaimana CI menghemat banyak waktu untuk mengutak-atik kurung sudut—Anda tidak perlu menulis apa pun tentang hal ini.

Men-debug

Segera setelah Anda mulai menguji kombinasi server klien Anda, Anda mungkin akan mendapatkan pesan ini:

> The XML data received was either invalid or not in the correct formfor XML-RPC. Turn on debugging to examine the XML data further.

Aktifkan debugging, dengan memasukkan baris:

```
$this->xmlrpc->set_debug(TRUE);
```

di klien Anda. Ini memungkinkan Anda untuk melihat dengan tepat apa yang dikirim kembali oleh kombinasi situs penerima klien Anda kepada Anda. Berhati-hatilah, di sinilah debugging menjadi cukup membuat frustrasi.

Ada beberapa tempat di mana pertukaran bisa salah:

- Situs jarak jauh tidak merespons dengan benar. (Anda mungkin harus mengaturnya sementara untuk menampilkan kesalahan untuk mengetahui mengapa tidak merespons. Ini mengganggu jika itu adalah situs yang aktif. Catch 22 tambahan adalah bahwa ia akan tampilan—yaitu kembali sebagai HTML—pesan kesalahan, yang bukan merupakan bagian dari respons XML yang diharapkan klien Anda, jadi Anda akan mendapatkan rangkaian pesan kesalahan kedua, yang disebabkan oleh rangkaian pertama…) Debugging ini mungkin melibatkan cukup banyak transfer FTP kembali dan maju, sampai Anda melakukannya dengan benar.
- Kode klien mungkin tidak bekerja dengan benar.
- Anda mendapatkan URL yang salah. (Ini harus menjadi cara CI untuk menangani pengontrol tempat server XML_RPC berada—yaitu http://www.mysite.com/index.php/mycontroller. Jika Anda meletakkan semua kode server di konstruktor pengontrol alih-alih di fungsi indeks, itu akan tetap berfungsi, tetapi Anda harus menangani fungsi yang ingin Anda panggil dengan nama—misalnya http://www.mysite.com/ index.php/mycontroller/myfunction)
- Pertukaran XML mungkin tidak sepenuhnya benar. Fungsi set_debug memungkinkan Anda untuk melihat apa yang dikirim kembali, tetapi Anda dapat menghabiskan cukup banyak waktu untuk melihat ini mencoba mencari tahu di mana kesalahannya.

Namun, setelah Anda mendapatkan semua ini dengan benar, Anda telah melakukan sesuatu yang cukup pintar. Anda telah membangun fungsi di situs jarak jauh, dan memanggilnya dari jarak jauh.

Dengan kata lain, Anda telah menyiapkan aplikasi yang dapat melakukan pemeliharaan atau operasi lain di situs jarak jauh. Jika Anda memiliki beberapa situs jarak jauh untuk dikelola, Anda dapat dengan mudah mereplikasi ini di antara mereka, memungkinkan Anda (misalnya) untuk mengoptimalkan semua tabel database Anda sekali sehari dengan satu tindakan di satu situs saja.

Masalah dengan XML-RPC?

Keamanan adalah masalah, tentu saja. Anda ingin melindungi panggilan fungsi Anda dengan kata sandi, sehingga klien harus mengirim kata sandi sebagai parameter sebelum situs penerima merespons. Ini dapat dilakukan hanya dengan mengirimkan kata sandi sebagai parameter tambahan dalam permintaan, dan meminta fungsi yang dipanggil untuk memeriksanya sebelum merespons.

Jika Anda mengekspos fungsi penting, Anda mungkin ingin semuanya terjadi di belakang lapisan SSL. Contoh kami terlihat tidak berbahaya—Anda mungkin tidak keberatan jika peretas berulang kali masuk ke situs Anda, tetapi yang dia lakukan hanyalah merapikan database Anda untuk Anda setiap waktu. Di sisi lain, itu akan menjadi dasar yang baik untuk serangan Denial of Service.

Harus dikatakan bahwa XML-RPC membuat frustrasi dan memakan waktu untuk menyiapkan dan men-debug, bahkan dengan bantuan CI yang sangat besar. Anda menulis dan men-debug dua situs sekaligus, dan format XML untuk mentransmisikan data di antara keduanya hanya bisa disebut pilih-pilih. Itu tidak membiarkan Anda lolos bahkan dengan kesalahan terkecil. Beberapa orang akan berpendapat bahwa XML-RPC adalah teknologi yang digantikan, dengan sebagian besar antarmuka atau API baru ditulis dalam bahasa yang lebih kompleks seperti SOAP (yang bahkan lebih memakan waktu untuk disiapkan).

Namun, untuk tujuan kami, XML-RPC sangat ideal. Ini memungkinkan kami untuk membuat situs web jarak jauh kami melakukan fungsi internal yang kompleks tanpa mengganggu kami dengan detailnya.

9.3 BERBICARA DENGAN MANUSIA UNTUK PERUBAHAN: KELAS EMAIL

Kami telah mengumpulkan banyak blok bangunan untuk situs pengujian web kami. Kami memiliki database pengujian, dan kami telah membangun fungsi untuk menjalankan berbagai jenis pengujian. Kami dapat mengakses situs kami dan memeriksa apakah kami melihat halaman yang benar; kita dapat memeriksa apakah semua file berada di tempat yang kita harapkan berada di server jauh. Kami dapat secara otomatis menjalankan fungsi di situs dan membuatnya mengoptimalkan dirinya sendiri. Cukup sederhana untuk menulis kode yang menggunakan alat ini untuk menjalankan serangkaian tes kapan pun kita mau, baik saat kita masuk atau dengan pengingat otomatis, seperti menyetel pekerjaan 'cron' di server Linux untuk memulai program kita berjalan pada waktu yang sesuai. interval.

Tapi itu tidak cukup untuk menjalankan tes dan hanya menyimpan hasilnya dalam database. Jika ada sesuatu yang salah, kita perlu tahu sesegera mungkin.

Di sinilah kelas email CI masuk. Ini memungkinkan kami memprogram situs kami untuk mengirimi kami email kapan pun kondisi tertentu tercapai. Anda mungkin ingin mengirim email untuk setiap pengujian yang gagal, atau Anda mungkin ingin menjalankan serangkaian pengujian, mengumpulkan hasilnya, lalu mengirim hanya satu laporan email.

Untuk menggunakan kelas email, pertama (seperti biasa) Anda harus memuatnya.

```
$this->load->library('email');
```

Kemudian kita harus mengatur beberapa variabel konfigurasi. Di sinilah kita bisa mengalami masalah, karena kelas tergantung pada server yang meng-hosting kode kita mampu (dan mau) mengirim email untuk kita. Sekali lagi, kita mungkin harus memeriksa dengan ISP. (Juga sulit untuk menguji ini di situs lokal, karena Xampplite, misalnya, mungkin tidak dapat menawarkan server email kepada Anda.)

Namun, setelah kami menyelesaikan ISP Anda, kami dapat dengan mudah mengonfigurasi kelas email.

Ada banyak pilihan, semua tercantum dalam panduan pengguna online. Yang utama adalah:

- protokol: apakah sistem Anda menggunakan email, sendmail atau SMTP untuk mengirim email?
- jalur surat: di mana program surat sistem Anda disimpan?

Anda mengaturnya seperti ini:

```
$config['protocol'] = 'sendmail';
$config['mailpath'] = '/usr/sbin/sendmail';
$this->email->initialize($config);
```

Opsi lain, yang semuanya memiliki default yang masuk akal, mencakup hal-hal seperti pembungkusan kata, rangkaian karakter, apakah Anda ingin mengirim email teks atau HTML, dan seterusnya. Menyiapkan opsi dan membuatnya berfungsi adalah satu-satunya (berpotensi) bagian yang sulit dalam menggunakan kelas ini.

Setelah Anda memuat kelas dan menginisialisasinya, menggunakannya sangat intuitif.

```
$this->email->from('david@mysite.com');
$this->email->to('someone@myownsite.com');
$this->email->bcc('fred@somewhere.com');
$this->email->subject('Test message');
$this->email->message('Hello world');
$this->email->send();
```

akan mengirimi saya email, disalin ke klien saya, melaporkan pesan apa pun yang saya inginkan. Jika Anda mengirim lebih dari satu email, mulailah setiap email baru dengan:

```
$this->email->clear()
```

hanya untuk memastikan bahwa Anda memulai dengan yang bersih setiap kali.

Anda juga dapat menggunakan kelas email untuk mengirim lampiran. Ingat bahwa file lampiran harus sudah disimpan di server yang mengirim email, dan Anda harus menentukan di mana, dalam hal file root server (memberikan alamat server, bukan alamat web).

Dapatkan alamat dan namanya seperti ini:

```
$path = $this->config->item('server_root');
$file = $path.'/my_subdirectory/myfile.htm';
```

lalu tambahkan saja baris ini:

```
$this->email->attach($file);
```

sebelum $this->email->send();.

Fungsi CI sederhana ini jauh lebih mudah daripada mencoba menulis kode PHP lengkap untuk mengirim lampiran. Ini menangani semua protokol yang terlibat, bahkan tanpa Anda harus menyadarinya.

Jika Anda menyertakan baris:

```
$result = $this->email->print_debugger();
```

dalam kode Anda, dan cetak variabel $result, Anda akan mendapatkan layar penuh informasi yang berguna, seperti ini:

```
Your message has been successfully sent using the following protocol:mail
User-Agent: Code Igniter
Date: Wed, 18 Apr 2007 13:50:41 +0100
From:
Return-Path:
Bcc: fred@somewhere.com Reply-To:
"david@mysite.com"X-Sender:
david@mysie.com
X-Mailer: Code Igniter X-
Priority: 3 (Normal)
Message-ID: <462614219c1a6@upton.cc>
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="B_ATC_462614219d14d" This is a
multi-part message in MIME format.
Your email application may not support this format.

--B_ATC_462614219d14d
Content-Type: text/plain; charset=utf-8Content-
Transfer-Encoding: 8bit
Test messagehello
world

--B_ATC_462614219d14d
Content-type: text/html; name="myfile.html"Content-
Disposition: attachment;
Content-Transfer-Encoding: base64
```

Jika terjadi kesalahan, maka informasi debug juga akan mengembalikan pesan kesalahan server. Misalnya, jika saya menyetel metode pengiriman ke SMTP tanpa menyetel host atau izin yang tepat:

```
$config['protocol'] = 'smtp';
```

itu tidak dapat mengirim pesan, dan itu memberi tahu saya:

```
You did not specify a SMTP hostname
Unable to send email using PHP SMTP. Your server might not beconfigured to send mail using
this method.
```

Namun, ingatlah bahwa 'sendmail' berpotensi menyesatkan di sini—ini mengembalikan pesan sukses jika telah meneruskan pesan di dalam server, tetapi ini tidak berarti bahwa pesan

tersebut benar-benar telah terkirim. (Jadi, jika Anda menyetel opsi 'jalur surat' yang salah, 'sendmail' mungkin melaporkan bahwa ia telah mengirim email, padahal sebenarnya tidak.) CI bergantung pada pesan yang didapatnya kembali dari aplikasi pengiriman surat, sehingga dapat tertipu. Seperti biasa dengan email, satu-satunya cara untuk memastikan mereka telah pergi adalah dengan memeriksa bahwa mereka telah tiba—tapi itu cerita lain.

Kelas email CI mencakup beberapa opsi yang berguna, semuanya dijelaskan dalam Panduan Pengguna online. Misalnya, Anda dapat mengaturnya untuk mengirim teks atau email berformat HTML—jika Anda memilih HTML, bahkan ada fungsi yang memungkinkan Anda menyetel pesan 'teks' terpisah untuk orang yang tidak menerima email HTML. Anda juga dapat mengaturnya untuk menggunakan set karakter yang berbeda, dan untuk menangani pembungkusan kata. Anda dapat mengatur ukuran batch, jika Anda ingin mengirim banyak email ke milis yang panjang, sehingga server Anda tidak kelebihan beban. (Atau ISP Anda tidak panik dan menutup Anda, mengira Anda adalah spammer.)

9.4 RINGKASAN

Kami sekarang telah menggunakan CI untuk membangun beberapa alat yang sangat canggih untuk situs web kami, yang memberikan beberapa fungsi yang signifikan.

Pertama, kami menggunakan kelas FTP CI untuk menyederhanakan dan mengotomatisasi operasi transfer file. Awalnya, kami baru saja menggunakan kelas ini untuk memeriksa apakah file yang kami harapkan ditemukan di situs kami benar-benar ada, dan catatan yang tidak diharapkan telah ditambahkan. Ini sendiri merupakan pemeriksaan yang berharga, karena banyak masalah yang dilemparkan situs web kepada Anda melibatkan perubahan file yang tidak terduga, biasanya oleh admin situs tetapi terkadang oleh peretas. Fungsi ini akan memeriksa secara teratur. Kelas FTP CI juga menawarkan kemungkinan pemeliharaan jarak jauh dan pembaruan situs.

Kemudian kami melihat pengembangan 'layanan web' pribadi kami menggunakan kelas XML-RPC CI. Ini memungkinkan kami secara otomatis memanggil fungsi di situs jarak jauh, meneruskan parameter jika perlu, dan mengembalikan hasilnya kepada kami—sama seperti jika kami telah masuk ke situs jarak jauh alih-alih ke situs pengujian kami. Kami menggunakan ini agar situs jarak jauh mengoptimalkan tabel di databasenya, dan melaporkan kembali kepada kami. Sekali lagi, kami telah melampaui rencana awal kami dengan hanya memantau situs-situs terpencil. Sekarang kami dapat menginstruksikan mereka untuk memeriksa atau mengoptimalkan diri mereka sendiri juga.

Terakhir, kami melihat kelas email CI, yang memungkinkan situs pengujian kami menghasilkan email. Kode CI sangat mudah digunakan, dan berarti situs kami dapat memberi tahu kami kapan pun dianggap ada masalah. CI mempermudah pembuatan dan pengiriman email, dan bahkan untuk mengirim lampiran.

BAB 10
BAGAIMANA CI MEMBANTU MEMBERIKAN INFORMASI DINAMIS

Kami telah banyak memikirkan untuk membangun situs web pengujian kami sekarang, dan CI telah mempermudah untuk melakukan beberapa hal yang sangat kompleks. Kami telah menyiapkan database, menggunakan FTP, membuat pengujian, dan mulai mengirim email hasil pengujian. Tetapi mudah untuk terjebak dalam hal-hal teknis dan lupa bahwa situs web sering dinilai sebagian besar berdasarkan presentasi, seberapa baik mereka memproses data, dan seberapa tepat mereka menampilkannya kepada pengguna manusia.

Berikut adalah beberapa kelas CI yang membantu mengatasi beberapa masalah yang muncul secara rutin saat Anda membuat situs web, terutama dalam hal menyampaikan informasi dinamis kepada pengguna Anda:

- Pembantu tanggal menerjemahkan format tanggal yang berbeda dan membantu Anda mengatasi zona waktu.
- Pembantu teks dan inflektor menyediakan fungsi yang berguna untuk memanipulasi dan mengonversi string.
- Kelas bahasa memudahkan untuk menulis situs web yang menampilkan informasi yang sama dalam bahasa yang berbeda, tergantung pada preferensi pengguna.
- Kelas tabel—menyimpan banyak <tr><td> yang membosankan.
- Anda dapat secara otomatis men-cache halaman dinamis dengan beban tinggi untuk respons yang lebih cepat.

Masing-masing dapat menghemat banyak waktu pengkodean, sekaligus membuat situs Anda terlihat lebih profesional (dan membuatnya lebih mudah untuk diperbarui).

10.1 THE DATE HELPER: MENGONVERSI DAN MELOKALKAN TANGGAL

Anda tahu situs web yang mengharapkan Anda memahami tanggal mesin? Format 'timestamp' asli MySQL, misalnya, sangat berguna, tetapi terlihat sangat ceroboh untuk membiarkan pengguna situs Anda melihat hal-hal seperti:

```
20070124161830
```

atau bahkan:

```
2007-01-24 16:18:30
```

Tentu saja, kebanyakan orang dapat memahami apa artinya, tetapi itu memberi situs Anda suasana yang tidak profesional dan belum selesai. CI datang untuk menyelamatkan dengan pembantu kencannya. Ini dimuat dengan:

```
$this->load->helper('date');
```

dan segera memberi Anda akses ke banyak fungsi yang berguna. Lihat Panduan Pengguna online untuk penjelasan lengkapnya.

Tanggal dapat ditentukan dengan berbagai cara. Fungsi standard_date() CI memberi Anda sepuluh cara untuk menampilkan tanggal yang sama:

```
1: atom         2006-12-31T11:34:44Q
2: cookie       Sunday, 31-Dec-06 11:34:44 UTC
3: iso          2006-12-31T11:34:44+0000
4: RFC 822      Sun, 31 Dec 06 11:34:44 +0000
5: RFC 850      Sunday, 31-Dec-06 11:12:34 UTC
6: RFC 1036     Sun, 31 Dec 06 11:34:44 +0000
7: RFC 1123     Sun, 31 Dec 2006 11:34:44 +0000
8: RFC 2822     Sun, 31 Dec 2006 11:34:44 +0000
9: RSS          Sun, 31 Dec 2006 11:34:44 +0000
10: W3C         2006-12-31T11:34:44Q
```

Yang perlu Anda lakukan adalah menentukan yang Anda inginkan. Contohnya:

```
$time = now();
echo standard_date('DATE_RFC822', $time);
```

Ada juga fungsi untuk mengkonversi antara berbagai jenis nilai tanggal-waktu. Nama mereka cukup jelas, dan sintaks yang tepat dijelaskan dalam Panduan Pengguna online. Mereka memungkinkan Anda untuk melakukan konversi yang cukup pintar dengan sangat sederhana.

Misalnya, kode ini:

```
function converttimes()
{
$this->load->helper('date');
$mysql      = '20070101120000';
$table      = '';
$table .= "<table><tr><td width='50%'>Start with MySQL
time<td>$mysql</td></tr>";
$utime      = mysql_to_unix($mysql);
$table .= "<tr><td>now convert to unix timestamp<td>$utime</td</tr>";
$htime = unix_to_human($utime);
$table .= "</td></tr><tr><td>then back to 'human' time<td>$htime</td></tr>";
$ttime = gmt_to_local($utime, 'UP25');
$table .= "</td></tr><tr><td>now convert unix stamp to local time in
Tehran<td>$ttime</td></tr>";
$ltime = unix_to_human($ttime);
$table .= "</td></tr><tr><td>and say that in human time <td>$ltime</td></tr>";
$table .= "<table>";echo
$table;
}
```

menghasilkan hasil ini:

```
Mulai dengan waktu MySQL                              20070101120000
sekarang konversikan ke cap waktu unix                1167652800
lalu kembali ke waktu 'manusia'                       2007-01-01  12:00 PM
sekarang ubah cap unix ke waktu lokal di Teheran      1167661800
dan katakan itu dalam waktu manusia                   01-01-2007  14:30
```

Ada banyak kode berguna yang tersedia untuk Anda di balik fungsi-fungsi ini, dan mereka membuat zona waktu internasional, khususnya, lebih mudah digunakan.

Pembantu tanggal juga memiliki timezone_menu(), sebuah fungsi yang menghasilkan menu drop-down zona waktu. Anda dapat menggunakan ini bersama dengan database untuk memungkinkan pengguna situs memilih zona waktu, dan kemudian menampilkan semua referensi waktu mereka dalam waktu 'lokal' mereka sendiri. Sebagai ganti tulisan:

```
echo timezone_menu();
```

Anda mendapatkan tangkapan layar berikut:

Gambar 10.1 inputan pilihan waktu yang sudah ditambahkan

Tampaknya pada awalnya CI menawarkan cara otomatis untuk menangani zona waktu juga, dalam fungsi now() pembantu tanggal. Panduan Pengguna menyarankan agar Anda mengatur 'referensi waktu master' di file konfigurasi Anda ke 'lokal' atau 'gmt', menggunakan:

```
$config['time_reference'] = 'local';
```

Lokal adalah default. Jika Anda menyetelnya ke 'gmt', kode tersebut muncul untuk mengembalikan waktu sistem (jika ada) berdasarkan fungsi PHP mktime(); jika ini tidak valid, atau Anda menyetel file konfigurasi ke 'lokal', ia mengembalikan waktu berdasarkan fungsi

time(). Namun, keduanya bergantung pada server Anda: itu harus diatur ke waktu yang akurat dan zona waktu default harus disetel. (Anda dapat memeriksanya dengan phpinfo().) Tetapi zona waktu mungkin tidak disetel, dan server Anda mungkin tidak berada di zona waktu yang sama dengan Anda: ini cukup umum di perusahaan besar, misalnya.

Jadi CI sendiri sebenarnya tidak tahu apa offset zona waktu Anda, meskipun mungkin bisa mendapatkan offset server Anda. Oleh karena itu, jika Anda menggunakan timezone_menu() untuk menangkap preferensi zona waktu pengguna, Anda tidak bisa hanya mengandalkan fungsi helper now() untuk menerjemahkan waktu GMT ke waktu lokal mereka. Anda perlu mencari preferensi mereka, dan menulis kode terpisah untuk menerjemahkan waktu kapan pun Anda ingin menampilkannya.

10.2 BEKERJA DENGAN TEKS: THE TEXT HELPER DAN INFLECTOR HELPER

Pembantu teks memiliki serangkaian fungsi yang membantu Anda memanipulasi teks dengan berbagai cara. Lihat Panduan Pengguna online untuk perincian lengkap. Saya hanya ingin menunjukkan beberapa hal berguna yang dapat Anda lakukan. Fungsi word_limiter() secara cerdas memotong string ke panjang yang Anda tetapkan. word_wrap() membungkus teks dengan panjang yang Anda tentukan. Dan word_censor() menggantikan kata-kata yang tidak ingin Anda lihat dengan padanan yang tidak berbahaya.

Ada juga fungsi untuk mengonversi ascii_to_entities() dan kembali lagi, yang dapat membantu mencegah saat-saat ketika teks dalam format seperti MS Word ditampilkan dengan aneh jika Anda menyalinnya ke halaman web. Fungsi pembantu inflektor akan mengubah kata dari tunggal ke jamak atau sebaliknya, meskipun mereka ditangkap oleh bentuk yang tidak beraturan seperti 'domba�domba' dan 'anak-anak', dan akan membuat beberapa kesalahan, misalnya mengubah 'hari' menjadi 'hari' '. Mereka juga dapat 'memperkecil', atau menggarisbawahi spasi di antara beberapa kata, lalu membaliknya kembali. Anda bisa bersenang-senang dengan ini, misalnya, kode ini:

```
function converttext()
    {
    $this->load->helper('text');
    $this->load->helper('inflector');
    $mytext = "Mr Bill Gates is a man I like. He is a very clever manand writes superb
       software";
    echo "$mytext<br />";
    $disallowed = array('like', 'clever', 'superb');
    $string = word_censor($mytext, $disallowed);echo
    "Censored, this might read: ";
    echo "$string<br />";
    $mywtext = word_limiter($mytext, 3);
    $mytext = underscore($mywtext);
    echo " His name could be written like this $mytext";
    $mytext = camelize($mywtext); echo
    "or like this $mytext";
    }
```

akan memberi Anda hasil ini:

> **Mr Bill Gates adalah pria yang saya suka. Dia adalah orang yang sangat pintar dan menulis perangkat lunak yang luar biasa Disensor, ini mungkin berbunyi: Tuan Bill Gates adalah orang yang saya ####. Dia adalah pria yang sangat ###### dan menulis ###### perangkat lunak**
> **Namanya bisa ditulis seperti ini mr_bill_gates…atau seperti ini mrBillGates…**

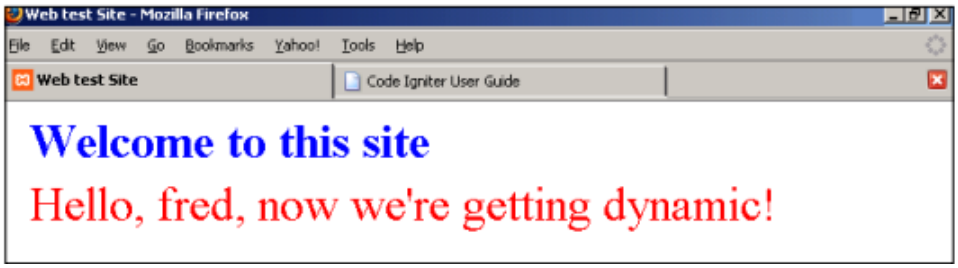
Fungsi-fungsi ini bisa sangat berguna jika Anda mengambil teks dari sumber lain dan perlu mengonversinya dengan satu atau lain cara, atau untuk menyensornya. Mereka dapat menghemat banyak waktu yang Anda habiskan untuk menulis regex.

10.3 PERGI INTERNASIONAL: KELAS BAHASA

Jika Anda menulis situs web yang dapat dilihat di lebih dari satu negara, CI dapat menyajikan halaman dalam lebih dari satu bahasa untuk Anda. Ini bekerja seperti ini:
Pertama, Anda mengidentifikasi teks yang disajikan kepada pengguna Anda yang perlu diterjemahkan. Mari kembali ke salah satu contoh pertama menampilkan data dinamis yang kita bahas dalam buku ini. Halaman selamat datang Anda mungkin dipanggil dengan kode dalam model, yang mengatakan:

```
function hello($name)
{
$data['mytitle'] = 'Welcome to this site';
$data['mytext'] = "Hello, $name, now we're getting dynamic!";
$this->load->view('testview', $data);
}
```

String yang ditetapkan ke array $data adalah pesan yang ditampilkan kepada pengguna:



Gambar 10.2 Tampilan HTML pada sintaks diatas

Tetapi Anda mungkin tahu bahwa pengguna adalah penutur bahasa Jerman—mungkin karena lokasi servernya, atau mungkin karena dia masuk, dan telah menyatakan preferensi bahasa. Alangkah baiknya jika Anda bisa menyapanya dalam bahasa Jerman. CI menyediakan metode mudah untuk melakukan ini.

Pertama, Anda perlu menyiapkan file bahasa. Jika Anda melihat di folder sistem, Anda akan melihat sudah ada folder bahasa dengan sub-folder bahasa Inggris. Ini pada gilirannya

berisi serangkaian file—misalnya, unit_test_lang.php Ini adalah file PHP yang hanya mendefinisikan array asosiatif ekspresi yang akan ditampilkan kepada pengguna:

```
<?php
$lang['ut_test_name']          = 'Test Name';
$lang['ut_test_datatype']      = 'Test Datatype ';
$lang['ut_res_datatype']       = 'Expected Datatype';
$lang['ut_result']             = 'Result';
// etc etc///
?>
```

Nilai array adalah ekspresi yang ingin Anda tampilkan, kunci array adalah singkatan apa pun yang ingin Anda gunakan untuk mengidentifikasinya sendiri. Nama file harus diakhiri dengan '_lang'.

Kita perlu mengaturnya sendiri, dalam setiap bahasa yang ingin kita tunjukkan. Mari kita panggil yang pertama welcome_lang.php dan simpan di subfolder system/language/English. Seharusnya terlihat seperti ini:

```
<?php
$lang['welcome_title']         = 'Welcome to this site';
$lang['welcome_text1']         = 'Hello ';
$lang['welcome_text2']         = ' now we're getting dynamic';
?>
```

Kunci larik bisa apa saja yang Anda suka: tetapi merupakan ide bagus untuk mengawalinya, katakan dengan 'ut' untuk larik bahasa pengujian unit, dan 'selamat datang' untuk larik yang sedang kita tulis sekarang. Mereka semua masuk ke array dasar yang sama, jadi jika Anda secara tidak sengaja memasukkan dua nilai dengan kunci yang identik, yang kedua akan menimpa yang pertama.

Fungsi asli yang mengatur halaman perlu diubah. Pertama, Anda perlu memuat file bahasa. Dalam contoh ini, saya telah memasukkannya ke dalam fungsi, tetapi biasanya, lebih masuk akal untuk melakukannya di dalam konstruktor kelas. Perhatikan bahwa meskipun nama file diakhiri dengan _lang (welcome_lang.php), Anda menghilangkan akhiran ini ketika Anda memuatnya (yaitu Anda memuat 'welcome', bukan 'welcome_lang'). Kedua, Anda menggunakan kunci array alih-alih teks yang sebenarnya—yaitu, untuk mengatakan:

```
function hello($name)
{
    $this->lang->load('welcome');
    $data['mytitle']=       $this->lang->line('welcome_title');
    $data['mytext']=        $this->lang->line('welcome_text1');
    $data['mytext'].=       $name;
    $data['mytext'].=       $this->lang->line('welcome_text2');
    $this -> load -> view('testview', $data);
}
```

Tapi ini hanya memberi kita halaman yang sama seperti sebelumnya: masih dalam bahasa Inggris. Jika kami ingin mengizinkan terjemahan ke dalam bahasa Jerman, kami memerlukan file bahasa lain. Pertama, kami membuat subfolder baru: di samping sistem/bahasa/bahasa Inggris kami membuat sistem/bahasa/jerman. Di folder baru, kami menyimpan file dengan nama yang persis sama dengan bahasa Inggris versi: 'welcome_lang.php'. (Bukan willkommen_sprach.php—maaf, hanya lelucon kecil saya.)

File ini identik dengan aslinya dalam bahasa Inggris—setidaknya di sisi kiri array. Kuncinya sama, tetapi nilai larik di sisi kanan sekarang harus dalam bahasa Jerman.

```
<?php
$lang['welcome_title']          = 'Willkommen auf dieser Web Seite';
$lang['welcome_text1']          = 'Guten tag ';
$lang['welcome_text2']          = 'jetzt sind wir dynamisch!';
?>
```

(Saya khawatir Anda harus menerjemahkan sendiri—CI tidak melakukannya untuk Anda!)
Ada satu hal yang tersisa untuk dilakukan. Saat fungsi 'halo' asli memuat berkas bahasa:

```
$this->lang->load('welcome_lang');
```

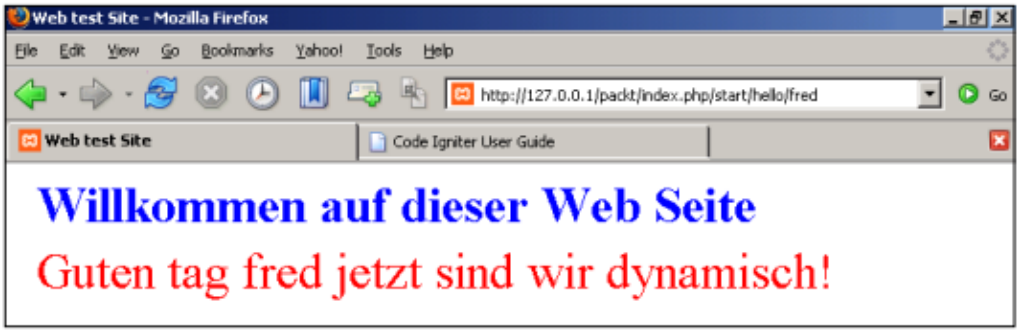
itu tidak menentukan bahasa mana, jadi defaultnya adalah bahasa Inggris. Seperti yang Anda harapkan, bahasa default ditentukan dalam file 'config':

```
$config['language']     = "english";
```

Untuk mendapatkan bahasa Jerman, ekspresi pemuatan bahasa dalam fungsi 'halo' juga harus menentukan nama folder tempat array Jerman disimpan. (Cukup logis, ini adalah 'Jerman'.)
Jadi fungsinya sekarang mengatakan:

```
function hello($name)
{
    $this->lang->load('welcome', 'german');
    $data['mytitle']=       $this->lang->line('welcome_title');
    $data['mytext']=        $this->lang->line('welcome_text1);
    $data['mytext'].=       $fred;
    $data['mytext'].=       $this->lang->line('welcome_text2);
    $this->load->view('testview', $data);
}
```

dan halaman yang dihasilkan terlihat seperti ini:



Gambar 10.3 Tampilah HTML sintaks diatas setelah diupdate

Yang perlu kita lakukan sekarang adalah memastikan bahwa fungsi kita memuat bahasa yang tepat secara dinamis.

Dengan asumsi kami telah mendeteksi preferensi bahasa pengguna, dan menyimpannya dalam variabel $user_language_pref, kita perlu sesuatu untuk memuat file bahasa secara kondisional, seperti ini:

```
if($user_language_pref == 'german')
{$this->lang->load('welcome', 'german');}
elseif($user_language_pref == 'french')
{$this->lang->load('welcome', 'french');}
// etc etc
```

Dibutuhkan tingkat disiplin diri untuk menulis kode seperti ini. Anda harus ingat untuk tidak pernah memasukkan teks aktual ke dalam kode Anda, tetapi membuat entri dalam file bahasa setiap kali. Tetapi setelah Anda selesai melakukannya, yang harus Anda lakukan adalah menyalin file bahasa dan memberikannya kepada seseorang untuk diterjemahkan ke dalam bahasa target Anda, dan situs Anda secara ajaib tersedia dalam terjemahan. Jika Anda mengubah kata-kata situs, Anda hanya perlu mengubah file bahasa. Jika Anda telah menggunakan beberapa ekspresi lebih dari sekali, Anda tidak perlu menelusuri halaman untuk mencari setiap contoh.

Jika situs Anda menggunakan teks panjang yang rumit, menerjemahkannya dengan cara ini menjadi kurang layak. Tetapi untuk teks 'boilerplate' yang tersebar di setiap situs web, kelas bahasa CI bekerja dengan baik dan membuat situs Anda terlihat jauh lebih mengesankan.

10.4 MEMBUAT TABEL HTML DENGAN CARA MUDAH: THE TABLE CLASS

Saya telah menggunakan CI selama beberapa bulan sekarang, tetapi saya terus menemukan fungsi yang membuat hidup lebih mudah. Inilah contoh yang bagus, bagi siapa saja yang menghabiskan banyak waktu untuk menulis hal-hal seperti:

```
echo "<tr><td>$value1</td><td>$value2</td></tr>";
```

Kelas tabel CI memungkinkan Anda membuat tabel HTML secara otomatis. Mari kita tampilkan detail dari beberapa tes yang telah kita jalankan. Anda memulai dengan memuat kelas, seperti biasa. Kemudian Anda dapat menentukan data tabel sebagai array, seperti ini:

```
$this->load->library('table');
$data = array(
                array('name',  'type', 'time'),
                array('test 1', 'ping', '1166627335'),
                array('test 2', 'ping', '1166627335'),
                array('test 3', 'ete', '1166702400')
                );
echo $this->table->generate($data);
```

Tetapi fungsi tersebut benar-benar muncul dengan sendirinya ketika Anda secara otomatis menghasilkan data langsung dari objek yang dikembalikan oleh kueri database. Misalnya, potongan kode pendek ini:

```
function dotable()
{
    $this->load->database();
    $this->load->library('table');
    $query = $this->db->query("SELECT name,type,time FROM events");echo
    $this->table->generate($query);
}
```

memberi Anda hasil kueri dalam tabel HTML yang diformat dengan benar. Saya tidak tahan untuk tidak menunjukkannya, meskipun format defaultnya membosankan!

name	type	time
not set	test	1166627335
not set	test	1166627882
not set	test	1166628621
not set		1166700269
ete	test	1166700394
ete	test	1166700928
ete	test	1166701193
ete	test	1166702238
ete	test	1166702361
ete	test	1166702400
ete	test	1166703348
ete	test	1166703451
ete	test	1166703835
ete	test	1166703851
ete	test	1166703928
ete	test	1166704104
ete	test	1166712129
ete	test	1166712176
ete	test	1166712209

Gambar 10.4 Array untuk Sintaks diatas

Itu sangat menghemat waktu Anda—hanya empat baris kode yang mengembalikan kueri dan membungkusnya untuk Anda dalam HTML. Bahkan, air mata kecil muncul di mata saya ketika saya memikirkan semua waktu yang saya habiskan untuk menulis:

```
<table>
<tr><td>$variable1</td><td>$variable2</td></tr> //etc.
```

Meskipun, seperti yang Anda lihat, tata letak tabel dasar CI tidak bagus, Anda dapat mengatur template Anda sendiri, menggunakan gaya CSS jika Anda mau, dan fungsinya akan mengikutinya dengan setia. Template adalah larik di dalam kelas 'tabel', jadi Anda perlu mengatur ulang setiap kali Anda memanggil kelas.

```
$tmpl = array (
    'table_open'          => '<table border="0" cellpadding="4" cellspacing="0">',
    'heading_row_start'   => '<tr>', 'heading_row_end'    => '</tr>',
    'heading_cell_start'  => '<th>', 'heading_cell_end'   => '</th>',
    'row_start'           => '<tr>',
    'row_end'             => '</tr>',
    'cell_start'          => '<td>',
    'cell_end'            => '</td>',
    'row_alt_start'       => '<tr>',
    'row_alt_end'         => '</tr>','cell_alt_start'  => '<td>',
    'cell_alt_end'        => '</td>',
    'table_close'         => '</table>'
    );
 $this->table->set_template($tmpl);
```

Ada larik templat default, yang terlihat seperti ini, di mana fungsi mendasarkan desainnya. Perhatikan bahwa ada dua set definisi baris (baris dan baris_alt), jika Anda ingin warna baris bergantian. Jika Anda mengirimkan revisi ke sebagian atau seluruh template, fungsi akan bereaksi sesuai, menghasilkan markup HTML yang berbeda.

Anda akan melihat bahwa template hanyalah sebuah array, dan Anda mengirimkan revisi dengan merevisi nilai untuk setiap kunci. Misalnya, jika Anda memiliki file CSS yang didefinisikan di suatu tempat dengan kelas yang disebut mytable, Anda dapat merujuk ke sana:

```
$tmpl = array ( 'table_open'   => '<table class="mytable">' );
```

Anda tidak perlu mengubah setiap nilai: nilai yang tidak Anda ubah tetap pada pengaturan default.

Sekarang tabel Anda secara ajaib melompat keluar dalam format yang Anda tentukan.

10.5 HALAMAN CACHING

Sekarang, kami sedang menulis beberapa kode yang cukup rumit. Server harus duduk dan memecahkan setiap halaman yang dibuat secara dinamis. Meskipun mudah bagi Anda untuk menulis fungsi seperti dotable() di atas, akibatnya server lama yang buruk harus melakukan lebih banyak pekerjaan.

Terkadang, ini dapat menyebabkan halaman Anda membutuhkan waktu lebih lama untuk dimuat daripada yang Anda inginkan. Mungkin tidak ada jalan lain untuk ini. Jika Anda

sedang menulis laporan yang akan berbeda setiap kali Anda menulisnya, maka Anda hanya perlu menunggu. Namun, Anda mungkin membuat halaman yang akan tetap sama untuk sementara waktu. Sebuah blog, misalnya, tetap sama sampai Anda memasukkan entri lain di dalamnya. Jika blog Anda mendapat seribu tampilan sehari, pada hari ketika Anda tidak menambahkan posting baru, setiap tampilan akan sama, dan itu membuang-buang waktu bagi sistem untuk membuat halaman yang sama berulang-ulang.

Cara mengatasinya adalah dengan men-cache halaman. Anda membuat halaman sekali, dan HTML yang dihasilkan disimpan dalam file 'cache' dengan stempel waktu, serta dikembalikan ke browser seseorang untuk ditampilkan di layar mereka. Kemudian, ketika penampil berikutnya meminta halaman tersebut, sistem akan memeriksa untuk melihat berapa lama halaman terakhir dibuat dan disimpan. Jika ini dalam batas waktu yang Anda tetapkan, halaman cache akan ditampilkan. Jika tidak, itu menghasilkan halaman dari awal.

Kedengarannya seperti pengkodean yang cukup rumit diperlukan di sini. Kecuali jika Anda menggunakan CI. Jika ya, Anda perlu melakukan dua hal:
Temukan file /system/cache di situs Anda. Itu harus kosong, kecuali untuk file index.html. Pastikan folder dapat ditulisi—mis. izin diatur ke 666, jika Anda menggunakan sistem Linux.
Sisipkan, di suatu tempat di fungsi pengontrol yang menghasilkan halaman HTML, baris:

```
$this->output->cache(5);
```

di mana 5 adalah jumlah menit yang Anda inginkan agar cache tetap ada sebelum halaman dibuat ulang.

Jika sekarang Anda memuat fungsinya, Anda akan melihat pemuatan halaman seperti biasa. Namun, jika Anda sekarang melihat folder /system/cache Anda, Anda akan melihat file baru di sana, dengan judul yang tidak berarti.

Buka ini (dalam editor teks) dan Anda akan melihatnya berisi kode HTML untuk halaman Anda, ditambah stempel waktu. Jika Anda meminta halaman yang sama lagi sebelum stempel waktu berumur lima menit, Anda akan mendapatkan halaman yang di-cache. Jika Anda menunggu lebih lama dan file cache kedaluwarsa, permintaan Anda berikutnya akan otomatis menghapusnya dan menggantinya dengan versi yang lebih baru.

Jika Anda berubah pikiran tentang caching halaman, hapus this->output->cache(5) baris dari pengontrol Anda, dan halaman Anda akan disajikan lagi setiap kali. (File cache terakhir akan tetap berada di folder /system/cache Anda sampai Anda menghapusnya secara manual.) Jika Anda ingin melanjutkan cache, tetapi tidak sengaja menghapus file cache kapan saja, jangan khawatir; sistem akan membuat yang baru ketika halaman itu dipanggil berikutnya.

CI membuat ini sangat cepat dan sederhana sehingga tergoda untuk men-cache setiap halaman! Ingatlah bahwa Anda tidak selalu ingin melakukan ini: yang terbaik untuk halaman dengan beban tinggi yang tidak terlalu sering berubah, tetapi mungkin tidak banyak membantu orang lain.

10.6 RINGKASAN

CI menawarkan banyak hal untuk membuat pengkodean lebih mudah dan situs web Anda lebih profesional. Bab ini hanya membahas lima di antaranya:

- Pembantu teks dan inflektor menyediakan fungsi yang berguna untuk memanipulasi dan mengonversi string.
- Pembantu tanggal memungkinkan Anda untuk mengkonversi antara format tanggal yang berbeda dan juga untuk mengatasi zona waktu.
- Kelas bahasa memudahkan penulisan situs web multibahasa, yang merespons preferensi pengguna. Sayangnya, Anda masih harus menerjemahkan!
- Kelas tabel memungkinkan Anda menampilkan tabel HTML yang dibentuk dengan benar, langsung dari kueri database jika perlu.
- Caching halaman dinamis dengan beban tinggi secara otomatis memberikan respons yang lebih cepat.

BAB 11
MENGGUNAKAN CI UNTUK MENANGANI FILE DAN GAMBAR

Bab ini membahas beberapa fungsi dan pembantu CI yang berguna. Masing-masing dari mereka adalah contoh yang baik tentang bagaimana beberapa baris kode CI memberi Anda akses tanpa batas ke berbagai aplikasi dan tindakan yang akan membutuhkan banyak pengetahuan khusus untuk membuat kode dari awal. Dalam banyak kasus, CI hanya menyediakan antarmuka ke kelas kode yang sudah di luar sana, dan yang dapat Anda unduh dari PEAR atau sumber lain. Tetapi CI memberi Anda antarmuka standar: Anda hanya memperlakukannya sebagai kode CI asli, dan kerangka kerja melakukan semua hal antarmuka untuk Anda.

Mari kita lihat lima kegiatan dalam bab ini:

- File helper memudahkan untuk menulis, dan membaca dari file.
- Pembantu unduhan memudahkan situs web Anda mengunduh file langsung ke pengguna, daripada menampilkannya sebagai HTML.
- Kelas unggah file bekerja dengan cara lain, memungkinkan pengguna untuk meletakkan file di situs Anda, dengan tindakan pencegahan keamanan bawaan untuk membatasi apa yang dapat mereka lakukan.
- Kelas manipulasi gambar memungkinkan Anda melakukan beberapa hal berguna dengan gambar, dan kita akan melihat cara mengubah ukurannya dan menandainya dengan air.
- Terakhir, kelas Zip memungkinkan Anda mengompresi file sebelum pengguna mengunduhnya.

Masing-masing contoh ini menyembunyikan banyak pengkodean yang cerdas dan memungkinkan Anda untuk menulis aplikasi praktis dengan sedikit kerumitan. Dalam banyak kasus, mereka menambahkan kode tambahan untuk membuat aktivitas lebih kuat.

Mari kita lihat satu per satu:

11.1 THE FILE HELPER

Sintaks PHP untuk membaca dan menulis file tidak mudah dipahami pada pandangan pertama. File helper CI berisi beberapa fungsi yang berguna, yang bertindak sebagai pembungkus untuk operasi penanganan file PHP sendiri. Mulailah seperti biasa dengan memuat pembantu:

```
$this->load->helper('file');
```

Kemudian hidup menjadi jauh lebih sederhana. Misalnya, untuk menulis ke file, yang perlu Anda ketahui adalah:

- Lokasi file Anda.
- Teks yang ingin Anda tulis.

- Mode di mana Anda ingin membuka file. Mode didefinisikan dalam manual PHP (lihat halaman di 'fopen'). Mereka termasuk 'r' untuk membaca, 'w' untuk menulis (menulis ke file, menimpa data yang sudah ada), dan 'a' untuk menambahkan (menulis ke file, menambahkan ke data yang ada). Dalam setiap kasus, menambahkan '+', katakan 'a+', akan terbuka file untuk operasi baca dan tulis. 'a' dan 'w', tetapi bukan 'r' atau 'r+', juga buat file, jika belum ada.

Kemudian Anda menggunakan tiga informasi ini sebagai parameter untuk write_file():

```
write_file('e:/filetest.txt', 'hello world', 'a+')
```

Ini lebih sederhana dan lebih intuitif daripada kode dua langkah PHP:

```
if ( $fp = fopen('e:/filetest.txt','r+'))
{
fwrite($fp, 'hello world');
}
```

Sekali lagi, kode CI menambahkan sedikit tambahan: secara otomatis mengunci file sebelum menulis dan membukanya setelahnya. Helper mengembalikan 'FALSE' jika operasi file tidak terjadi, sehingga Anda dapat menggunakannya untuk melaporkan keberhasilan atau kegagalan. Anda harus menentukan judul untuk file Anda, tetapi jika Anda tidak menentukan filepath, itu ditempatkan di folder root web untuk situs Anda, di mana file index.php utama Anda berada.

Tentu saja, folder apa pun tempat Anda membuat atau menulis ke file, harus memiliki izin menulis yang disetel. Ingat juga bahwa jika Anda menjalankan sistem Windows, Anda harus menggunakan garis miring — /— untuk menggambarkan jalur file Anda.

Dalam aplikasi kita, kita dapat menggabungkan helper ini dengan kelas utilitas database. Ini memungkinkan kita untuk membuat, mencadangkan, memperbaiki, dan mengoptimalkan database dan tabel, meskipun hanya pada database MySQL dan MySQLi. Campurkan dengan file helper, dan Anda membuat rutinitas pencadangan yang rapi.

```
$this->load->dbutil();
$backup =& $this->dbutil->backup();
$this->load->helper('file');
write_file('e:/mybackup.gz', $backup);
```

Kode di atas menulis versi terbaru dari database kami ke file di server. Membaca kembali file juga sama sederhananya:

```
$content = read_file('e:/filetest.txt');
```

Ada juga fungsi yang mengembalikan larik semua file dan/atau folder dalam a direktori yang diberikan:

```
$filenames = get_filenames('e:/')
```

Meskipun, jika Anda menggunakannya di direktori dengan banyak file, Anda mungkin menemukan bahwa PHP habis sebelum dapat mencantumkan semuanya. Anda dapat menggunakan ini dalam sepotong kode sederhana untuk memeriksa apakah file atau folder sebenarnya dalam folder adalah apa yang Anda harapkan. Mulailah dengan menggunakan fungsi CI untuk menemukan file yang benar-benar ada, dan array referensi file yang Anda harapkan untuk menemukan, lalu gunakan array_diff() untuk membandingkannya. Diberikan dua array, array_diff() memberi tahu Anda nilai apa yang ada di yang pertama yang tidak ada di yang lain, jadi Anda harus menggunakannya dua kali, menempatkan setiap array terlebih dahulu.

```
//list files actually found
$files_there = get_filenames('e:/rootfolder/system/application/controllers');
// list files we expected
$files_expected = array('start.php', 'index.php');
// any found that we didn't expect?
$difference = array_diff($files_there, $files_expected);echo "<br
/>Missingfiles are:";
print_r($difference);
// any expected that we didn't find?
$difference = array_diff($files_expected, $files_there);echo "<br />Extra files are:";
      print_r($difference);
```

Terakhir, tetapi terlalu mengerikan untuk dipikirkan, ada delete_files()
fungsi. Tindakan ini akan menghapus semua file dalam direktori mana pun yang Anda tentukan, sehingga:

```
delete_files('c:/mydirectory/');
```

akan menghapus semua yang ada di direktori saya. Jika Anda menambahkan parameter opsional TRUE, seperti pada:

```
delete_files('c:/mydirectory/', TRUE)?
```

itu juga akan menghapus semua subfolder di direktori itu Gunakan dengan sangat hati-hati: bayangkan saja apa:

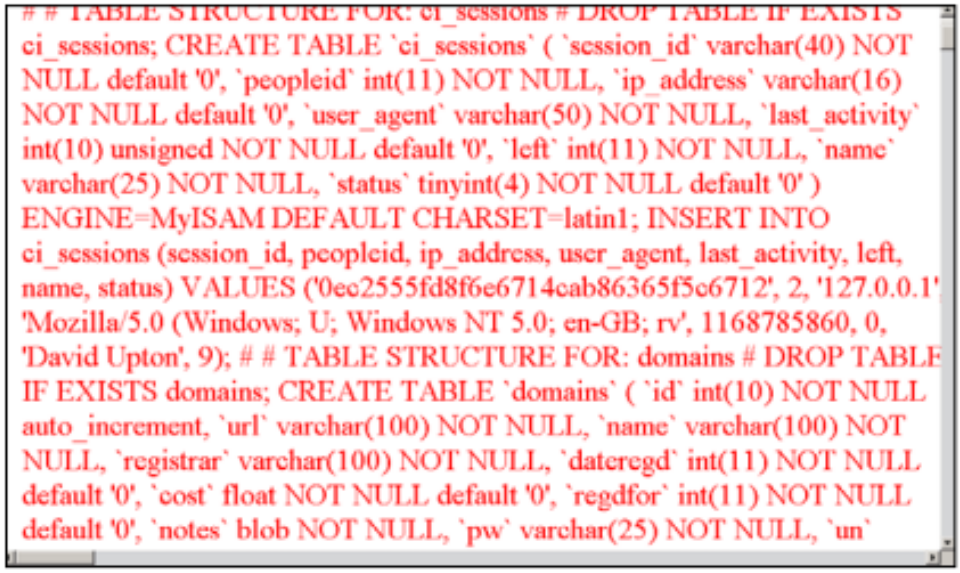
```
delete_files("c:/", TRUE)
```

mungkin bisa!

11.2 THE DOWNLOAD HELPER

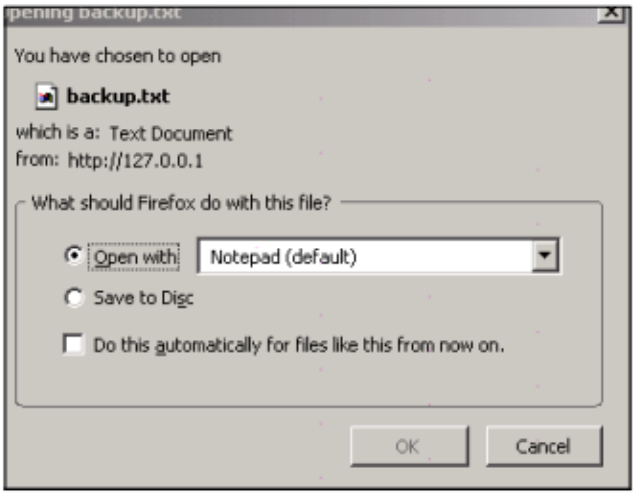
Pembantu unduhan hanya memiliki satu fungsi, tetapi ini melengkapi penolong file dengan sangat baik. Anda mungkin membuat file di situs web dan kemudian ingin menyajikannya kepada pembaca sebagai file—file teks, misalnya—daripada mengonversinya menjadi halaman web. Contoh yang baik adalah file cadangan database, seperti yang baru saja

kita buat untuk aplikasi kita. Untuk membuat ulang database jika crash, kita memerlukan file teks dalam format MySQL sendiri. Tidak banyak gunanya bagi kita untuk melihat ini di layar:



Gambar 11.1 tampilan jika database crash

Kita perlu menemukan cara untuk mengunduhnya sebagai file. Dengan kata lain, jika kita bekerja pada sistem Windows, kita ingin melihat dialog ini:



Gambar 11.2 membuka database yang sudah diunduh

Untuk membuat kode ini melalui koneksi internet, Anda harus menentukan jenis halaman yang Anda inginkan di header HTTP. Pembantu unduhan CI melakukan ini untuk Anda di latar belakang. Muat pembantu dengan:

```
$this->load->helper('download');
```

dan metode tunggalnya digunakan seperti ini:

```
force_download($name, $data);
```

di mana $name adalah nama yang Anda berikan ke file yang diunduh dan $data adalah konten file. Jika Anda ingin mengunduh file yang ada, Anda harus membacanya menjadi string terlebih dahulu:

```
$data = file_get_contents("e:/mybackup.txt");
$name = 'backup.txt';
force_download($name, $data);
```

File $data sekarang dapat digunakan secara langsung untuk membuat ulang database MySQL. Anda juga dapat menggunakan bantuan ini untuk mengunduh laporan secara langsung, daripada memaksa pengguna untuk menghapusnya dari layar.

Di balik layar, helper menangani identifikasi tipe MIME dan menyetel header HTTP. Itu bergantung pada salah satu file 'config', system/application/config/ mimes, yang juga digunakan oleh kelas Upload yang akan kita lihat selanjutnya. File konfigurasi ini menyimpan larik tipe MIME dan ekstensi HTTP yang sesuai—mis.:

```
'rtf'   => 'text/rtf',
'text' => 'text/plain',
```

yang menyelamatkan Anda harus mengingatnya!

Jika Anda secara teratur menggunakan jenis file yang tidak termasuk dalam daftar CI, Anda dapat dengan mudah menambahkannya ke file 'config'.

11.3 THE FILE UPLOAD CLASS

Terkadang, Anda ingin mengizinkan pengguna situs Anda untuk mengunggah file. Ini mungkin teks, atau gambar, atau jenis file yang lebih eksotis seperti audio MP3 atau video MPEG. Ini adalah proses yang lebih kompleks daripada unduhan file yang baru saja kita diskusikan, tetapi kelas pengunggahan file CI menangani sebagian besar pekerjaan untuk Anda. Itu juga terlihat setelah beberapa masalah keamanan. Namun, Anda harus selalu berpikir dua kali sebelum mengizinkan siapa pun mengunggah file ke situs Anda, dan Anda mungkin ingin melindungi laman unggahan untuk mencegah pengguna yang tidak berwenang melakukannya.

Pertama, Anda perlu mengalokasikan ruang untuk menyimpan file yang diunggah—folder atau direktori di server Anda. Ini harus diatur dengan izin yang benar, yang memungkinkan pengguna untuk menulis padanya. (yaitu 777 pada sistem Unix Linux). Mari kita asumsikan Anda menyebut folder ini sebagai unggahan, dan meletakkannya di folder root web Anda.

Kelas unggah file CI dimuat dengan:

```
$this->load->library('upload');
```

Maka Anda perlu melakukan tiga hal:

- Setel default
- Tulis pengontrol untuk menangani unggahan
- Berikan formulir unggah dan formulir 'sukses' kepada pengguna Anda

Mari kita bawa mereka dalam urutan ini. Pertama, atur serangkaian default. Anda melakukan ini dengan membuat $config larik.

Katakanlah Anda ingin mengatur jalur ke direktori unggahan yang baru saja Anda buat. Untuk ini, Anda perlu mengatakan:

```
$config['upload_path'] = 'uploads';
```

Baris ini dapat berada di pengontrol yang akan Anda tulis, atau Anda dapat membuat folder config/upload untuk menampungnya (ini akan menjadi system/application/config/ upload.php).

```
<?php  if (!defined('BASEPATH')) exit('No direct script accessallowed');
$config['upload_path'] = 'uploads';
?>
```

Penting untuk memahami perbedaan antara dua cara pengaturan default ini. Jika Anda mengatur default dari file config�upload, Anda tidak perlu secara khusus menginisialisasi kelas upload file. Muat saja dan itu akan menemukan default untuk dirinya sendiri.

Namun, jika Anda membiarkan default di controller, Anda perlu menentukan di mana mereka berada saat Anda memuat kelas, menggunakan parameter kedua ke fungsi pemuatan, seperti ini:

```
$this->load->library('upload', $config);
```

di mana $config adalah nama array default Anda. (Jangan mencoba mengatur beberapa default dari file config�upload dan beberapa dari controller!)

OK, jadi apa yang diributkan ini tentang default? Jadi CI menetapkan default yang masuk akal dan Anda tidak perlu mengkhawatirkannya, bukan? Ya, tetapi dalam hal ini Anda melakukannya. Ada beberapa yang penting:

- Lokasi file unggahan Anda: CI tidak membuat asumsi tentang hal ini, anda harus menceritakannya.
- Jenis file yang Anda izinkan untuk diunggah oleh pengguna Anda. Ini diatur seperti ini:

```
$config['allowed_types'] = 'gif|jpg|png';
```

di mana jenis file yang dapat diterima ditentukan dengan operator pipa (|) di antara mereka. Pengaturan ini akan memungkinkan situs Anda untuk mengunggah jenis file gambar paling populer tetapi tidak mengizinkannya untuk mengunggah file audio, misalnya.

Mengatur ini adalah langkah keamanan dasar: jika Anda hanya ingin gambar menjadi diunggah, jangan izinkan orang mengunggah file yang dapat dieksekusi atau MP3 besar. Perhatikan bahwa Anda harus menetapkan nilai sebelum jenis file dapat diunggah: pengaturan default (yaitu, tidak ada pengaturan) memungkinkan tidak ada file yang diunggah.

- Max_size: masuk akal untuk menetapkan batas ukuran maksimum file, dalam kilobyte, yang dapat diunggah. Anda tidak ingin pengguna jahat memenuhi semua ruang Anda. Pengaturan default adalah 0, yang tidak menetapkan batas.
- Timpa: jika Anda sudah memiliki file di folder unggahan Anda dengan nama yang sama dengan yang diunggah pengguna, apakah yang lama harus ditimpa dan hilang selamanya? Ini tergantung pada apa yang dilakukan situs Anda dan mengapa Anda mengizinkan unggahan. CI default ke 'FALSE', yang berarti tidak menimpa file lama dan file baru disimpan dengan nama baru. Setel default ini secara eksplisit ke 'TRUE', jika Anda ingin file baru menimpa yang lama.

  Perhatikan bahwa CI tidak secara otomatis memberi tahu pengguna bahwa ia telah mengganti nama filenya, yang mungkin membingungkan: lihat di bawah untuk cara mendapatkan laporan tentang proses tersebut.
- Anda juga dapat mengatur default untuk ukuran, lebar, dan tinggi gambar, untuk mengenkripsi file, dan untuk memangkas spasi kosong dari judulnya.

Sekarang setelah Anda memutuskan default Anda, Anda memerlukan pengontrol unggahan. Ini sederhana. Perannya adalah untuk menginisialisasi kelas unggahan, menerima unggahan dari formulir pengguna, dan kemudian memutuskan apakah itu telah berhasil. Jika ya, ini akan menampilkan laporan; jika tidak, ia kembali ke formulir unggah dengan pesan pengguna. Paling sederhana, hanya perlu menyertakan satu fungsi aktif, do_upload(), seperti ini:

```
<?php
/*constructor function to initialize controller and load the fileupload class, plus the
two other helpers it needs */
class Upload extends Controller {
      function Upload()
      {
            parent::Controller();
            $this->load->helper(array('form', 'url'));
            $this->load->library('upload');
      }
/*now the function which does all the work!*/function
do_upload()
    {
            if ( ! $this->upload->do_upload())
            {
                  $error = array('error' =>
                                          $this->upload->display_errors());

                  $this->load->view('upload_form', $error);
            }
        Else
                {
```

```
                $data = array('upload_data' =>
                $this->upload->data());
                $this->load->view('upload_success', $data);
            }
        }
    }
```

Fungsi ini membutuhkan tampilan upload_form, dan tampilan upload_success. Untuk membangun yang pertama, gunakan penolong formulir untuk membuat formulir normal, menunjuk ke fungsi do_upload dari pengontrol 'unggah' kami: tetapi alih-alih membukanya dengan:

```
echo form_open('upload/do_upload')
```

(seperti yang kita lakukan pada form yang kita buat di Bab 5), Anda membukanya dengan fungsi multipart form helper:

```
echo form_open_multipart('upload/do_upload');
```

(Ingat bahwa kita sedang menulis kode untuk menghasilkan markup HTML, jadi kita perlu menggemakannya ke layar.)

Kemudian, alih-alih fungsi form_input pembantu formulir, gunakan fungsi form_upload:

```
echo form_upload($data);
```

Dua baris kode ini menangani banyak hal yang sangat membosankan untuk Anda. Tambahkan seperti biasa tombol kirim, dan tutup formulir.

```
echo form_submit('mysubmit', 'Submit Post!');
```

Berikan variabel $view Anda ke tampilan dan muat. Pandangan Anda juga harus menggemakan variabel $error, yang dilewatkan oleh fungsi do_upload saat memuat tampilan.

```
echo $error;
```

Anda sekarang akan melihat sesuatu seperti ini:

Gambar 11.3 Tampilan dialog box untuk upload

Mengklik Browse memungkinkan pengguna untuk melihat file di komputernya sendiri (lokal), bukan server Anda. Setelah dia memilih file, mengklik unggah akan memanggil pengontrol unggah, dan file akan ditransfer ke folder unggah di server Anda.

Katakanlah saya mencoba mengunggah file teks. (Ingat bahwa jenis file kami yang diizinkan terbatas pada 'gif | jpg | png'.) Sekarang saya melihat:

Gambar 11.4 memberi ijin dan membatasi type file yang di upload

CI melaporkan kembali jenis kesalahan kepada saya: ini adalah fungsi $this->upload->display_errors() di pengontrol yang sedang bekerja, menambahkan variabel tambahan ke tampilan.

Anda juga dapat memiliki data laporan CI tentang unggahan yang berhasil kepada Anda. Seperti yang Anda lihat, pengontrol yang baru saja kami tulis memanggil tampilan yang disebut upload_success, jika unggahan berhasil. Data yang diteruskan ke tampilan ini adalah isi dari fungsi $this->upload->data. Ini memberi Anda serangkaian informasi lengkap tentang proses pengunggahan: mungkin lebih dari yang ingin Anda tampilkan.

Katakanlah saya mengunggah file bernama waltzer.jpg: laporan default terlihat seperti ini:

Gambar 11.5 tampilan file sukses di upload

Jika Anda menggunakan situs Anda untuk membuat saingan Flickr, misalnya, maka jumlah informasi ini mungkin membingungkan pengguna yang telah mengunggah foto mereka! Namun, Anda dapat dengan mudah memfilter informasi apa pun yang tidak Anda inginkan di pengontrol unggahan.

Harap dicatat, omong-omong, saya menyetel default 'timpa' ke 'FALSE' saat mengonfigurasi kelas unggah file untuk menulis contoh ini. Saya kemudian menggunakan kode untuk mengunggah file gambar waltzer.jpg—dua kali.

Tangkapan layar sebelumnya adalah laporan CI tentang unggahan kedua yang berhasil. Anda akan melihat bahwa file telah berganti nama menjadi waltzer1.jpg. Jika saya melihat di direktori unggahan saya, saya dapat melihat file waltzer.jpg asli dan file waltzer1.jpg yang baru. Bergantung pada aplikasi Anda, Anda mungkin ingin membandingkan nilai untuk raw_name dan orig_ name dan memberi tahu pengguna bahwa nama file telah diubah.

CI tidak membandingkan kedua file tersebut, hanya namanya saja. Jika Anda mengizinkan beberapa pengguna untuk mengunggah file, sangat mungkin bahwa dua dari mereka secara tidak sengaja akan menggunakan nama file yang sama untuk file yang berbeda, dan Anda mungkin tidak ingin kehilangan yang pertama. Di sisi lain, jika Anda menggunakan situs untuk mengunggah laporan yang selalu memiliki nama yang sama, Anda mungkin lebih suka hanya menyimpan versi terbaru di situs—dalam hal ini, penimpaan adalah cara sederhana untuk menghemat ruang.

Berikut gambar itu, omong-omong. Kami akan melakukan lebih banyak hal dengannya di bagian berikutnya.

Gambar 11.6 Gambar yang sudah diunggah

11.4 CI'S IMAGE CLASS

Jika Anda mengizinkan pengguna untuk mengunggah gambar ke situs Anda, Anda juga perlu melihat kelas Manipulasi Gambar CI. Ini bekerja dengan tiga perpustakaan gambar paling populer untuk PHP: GD GD2, NetPBM, dan ImageMagick. (Gunakan phpinfo() untuk mengetahui mana yang didukung server Anda.) Penandaan air gambar hanya berfungsi dengan GD GD2, meskipun:

Kelas manipulasi gambar memungkinkan Anda untuk melakukan empat fungsi dasar dengan gambar:

- Ubah ukuran: Anda mungkin ingin memasukkannya ke dalam ukuran standar di layar Anda; atau anda mungkin ingin memotongnya menjadi gambar 'thumbnail'.
- Pangkas
- Putar

- Watermark (hanya tersedia dengan GD◆GD2): Ini sering digunakan untuk menempatkan pemberitahuan hak cipta pada sebuah gambar, sehingga orang tidak bisa begitu saja mengunduhnya dari situs Anda dan menyebarkannya sebagai karya mereka sendiri.

Mungkin yang paling berguna dari fungsi ini adalah mengubah ukuran, jadi kita akan melihatnya dengan sedikit detail. Memotong dan memutar kurang berguna karena Anda tidak dapat melakukannya secara bermakna kecuali Anda dapat melihat gambar di layar. Untuk melakukan ini, Anda memerlukan semacam antarmuka pengguna yang memungkinkan pengguna menentukan apa yang ingin dia lakukan dan mengontrol cara CI mengimplementasikan fungsi-fungsi ini, dan Anda harus membuatnya sendiri!

Katakanlah Anda telah mengupload gambar waltzer.jpg Anda, menggunakan kelas upload file yang baru saja kita bahas, ke folder /uploads Anda. (Izin untuk folder ini harus diatur ke 777 untuk Anda unggah—dan juga bagi Anda untuk memanipulasi gambar, karena CI perlu menulis hasil manipulasi kembali ke folder.) Pertama, muat perpustakaan:

```
$this->load->library('image_lib');
```

Kemudian, Anda perlu mengatur beberapa detail konfigurasi. (Seperti kelas unggah file, Anda dapat melakukannya dalam kode Anda, atau dalam file system/application/config/ image_lib.php yang terpisah.)

Ada beberapa preferensi yang dapat Anda atur dan tercantum dalam Panduan Pengguna online. Mungkin yang paling penting adalah:

- Pustaka gambar mana yang Anda gunakan. Standarnya adalah GD2, jadi jika Anda tidak menggunakannya dalam instalasi PHP Anda, Anda perlu menentukan yang sedang Anda jalankan, mis., `$config['image_library'] = 'ImageMagick'` (Anda akan juga harus menyediakan jalur ke perpustakaan ImageMagick menggunakan: `$config['library_path'] = '/mypath';`.)
- Gambar yang ingin Anda manipulasi. Ini harus menjadi jalan (relatif terhadap Anda folder root situs) dan nama file.
- Ukuran gambar yang Anda inginkan setelah diproses—di mana 'x' adalah angka piksel, lebarnya diatur oleh `$config['width'] = x;`,dan tingginya dengan `$config['height'] = x;`.

Ini cukup untuk mengubah ukuran gambar Anda, menimpa file gambar lama dengan gambar yang diubah ukurannya. Kodenya terlihat seperti ini:

```
function do_image($image_name)
{
    $this->load->library('image_lib');
    $config['image_library'] = 'GD';
    $config['source_image'] = "$image_name";
    $config['width'] = 75;
    $config['height'] = 50;
    $this->image_lib->initialize($config); if(!$this-
    >image_lib->resize())
    {echo "failed";} else{echo
    'success!';}
}
```

Perpustakaan dapat melakukan beberapa hal pintar lainnya juga. Jika Anda tidak ingin menimpa gambar asli, tentukan nama dan jalur file baru untuk versi baru, dengan menambahkan:

```
$config['new_image'] = 'newfolder/newname.png';
```

Atau, jika Anda ingin membuat thumbnail gambar, cukup tambahkan:

```
$config['create_thumb'] = TRUE;
```

alih-alih. Ini memiliki efek penamaan ulang file yang diubah ukurannya dengan akhiran default _thumb, sehingga waltzer.jpg menjadi waltzer_thumb.jpg. (Anda juga dapat mengubah sufiks default—lihat Panduan Pengguna.) Jadi, Anda memiliki dua file dalam folder yang sama: file asli dan thumbnail.

Perhatikan bahwa pengaturan gambar mini tidak melakukan hal lain—Anda masih harus mengatur ukuran yang Anda inginkan.

Berikut gambarnya, diperkecil menjadi 75 kali 50 piksel:

Gambar 11.6 Gambar upload yang sudah diperkecil ukurannya

Fungsi tambahan dari kelas gambar memungkinkan Anda untuk menandai gambar dengan air. Jadi, jika Anda telah menempatkan foto brilian Anda sendiri di situs web Anda, Anda juga dapat menambahkan pemberitahuan hak cipta ke dalamnya.

Sekali lagi ada banyak opsi, dijelaskan sepenuhnya di Panduan Pengguna, tetapi kode dasarnya sederhana. Inisialisasi kelas, beri tahu gambar mana yang ingin Anda tandai air dan apa yang ingin Anda tandai airnya, dan panggil fungsi tanda air.

```
function wm_image()
{
    $this->load->library('image_lib');
    $config['source_image'] = 'uploads/waltzer.jpg';
    $config['wm_text'] = 'Copyright 2007 - David Upton';
    $config['wm_type'] = 'text';
    $this->image_lib->initialize($config); if(!$this-
    >image_lib->watermark())
{echo 'failure to watermark';}else {echo 'success';}
}
```

(Opsi wm_type yang disetel ke teks memungkinkan Anda memberi tanda air dengan teks. Jika tidak, atur opsi ini ke overlay, dan berikan gambar, yang akan ditumpangkan pada gambar asli Anda.)

Seperti inilah gambarannya sekarang.

Gambar 11.7 memberikan Watermark pada gambar

Kode saya yang sebenarnya sedikit lebih kompleks daripada contoh yang ditunjukkan di atas, sehingga saya dapat mengontrol ukuran dan posisi tanda air saya agar lebih mudah terlihat di halaman ini. Kode default yang ditunjukkan di atas akan memadai untuk sebagian besar tujuan, tetapi tanda air yang keluar terlalu kecil untuk terlihat jelas pada halaman yang dicetak. Lihat Panduan Pengguna online CI untuk informasi lebih lanjut tentang bekerja dengan font eksternal.

11.5 KOMPRESI FILE MUDAH DENGAN CI ZIP CLASS

Jika Anda memindahkan file besar seperti gambar, Anda mungkin perlu mengompresnya. CI berisi perpustakaan yang berguna untuk melakukan ini. Seperti biasa, Anda mulai dengan menginisialisasi kelas Zip. Setelah Anda selesai melakukannya, Anda harus memberi tahu CI file mana yang ingin Anda zip, dan membuat arsip untuk memasukkannya. Kemudian Anda menggunakan fungsi read_file untuk membacanya dan zip, dan fungsi download untuk mengunduhnya desktop Anda.

```
function zip_image()
{
    $this->load->library('zip');
    $this->zip->archive('my_backup.zip');
    $path = 'uploads/waltzer1.jpg';
    $this->zip->read_file($path);
    $this->zip->download('my_backup.zip');
}
```

Kelas penyandian CI Zip lebih kompleks dari ini dan memungkinkan Anda beberapa opsi. Seperti biasa, semuanya diatur dalam Panduan Pengguna online. Tetapi ini akan memberi Anda gambaran tentang betapa mudahnya CI membuat zip unduhan dari situs Anda, meminimalkan bandwidth yang mereka konsumsi, dan menghemat waktu bagi pengguna Anda.

11.6 RINGKASAN

Bab ini mengumpulkan beberapa pembantu CI dan kelas dengan tema serupa. Mereka membantu Anda untuk:

- Tulis dan baca file di sistem Anda, dengan pengkodean minimal, sementara CI mengunci dan membuka kunci file di latar belakang.
- Unduh file daripada menampilkannya sebagai HTML di layar, dengan CI
- menyediakan header HTTP dan mengkhawatirkan jenis MIME untuk Anda.
- Unggah file ke situs Anda, memungkinkan Anda menentukan batasan keamanan seperti ukuran dan jenis file yang Anda izinkan.
- Memanipulasi gambar dengan mudah, untuk mengubah ukuran atau menandainya dengan air.
- Kompres file Anda sebelum Anda mengunduhnya ke pengguna Anda.

Kelas ini sangat mudah digunakan, dan hanya ketika Anda melihat kode yang mendasarinya di perpustakaan manipulasi gambar (dalam sistem file/libraries/Image_lib.php) Anda menyadari betapa rumitnya pengkodean CI telah menyelamatkan Anda!

BAB 12
VERSI PRODUKSI, PEMBARUAN, DAN KEPUTUSAN BESAR

Hari besar telah tiba. Situs pengembangan Anda berjalan cukup baik di pengembangan lokal Anda sehingga Anda dapat mentransfernya ke situs produksi yang dihosting di server web jarak jauh. Seharusnya mudah untuk melakukan ini. Salin semua file, termasuk seluruh folder sistem, perbarui pengaturan konfigurasi, salin dan tautkan ke database, dan pergilah. Terkadang, itu sangat mudah.

Tetapi ketika tidak, selalu malam sebelum Anda memberikan presentasi yang sangat penting kepada pemodal ventura atau prospek utama. Jadi, jika ini terjadi pada Anda, bab ini mencakup:

- Apa yang harus dicari di file konfigurasi Anda
- Beberapa alat diagnostik untuk digunakan jika Anda buntu
- Beberapa perbedaan potensial antara server yang mungkin membuat Anda tersandung
- Beberapa catatan tentang keamanan, sekarang Anda berada di luar sana di dunia besar

Selanjutnya, bab ini mencakup pemutakhiran, dan melihat beberapa cara di mana CI telah berubah pada tahun itu tersedia. Seberapa stabil? Keputusan apa yang harus Anda buat jika Anda melakukan situs penting untuk itu? Dan apa yang harus Anda lakukan, setelah situs Anda aktif dan berjalan, jika Rick Ellis mengeluarkan versi CI yang lebih baru?

Terakhir, kami membahas secara singkat membuat perubahan Anda sendiri pada inti CI. Semuanya ada di sana; itu open-source itu mungkin. Apakah itu masuk akal atau tidak adalah masalah lain.

12.1 KONEKSI: PERIKSA FILE KONFIGURASI

Sistem biasanya gagal pada antarmuka. Untuk itulah file konfigurasi Anda ada: untuk memberi Anda tempat untuk meletakkan semua antarmuka itu. Jika Anda belum melakukannya, Anda telah melewatkan salah satu kekuatan utama CI.

Masalah antarmuka utama kemungkinan adalah:

URL

CI bekerja dengan mencari file. Pengguna Anda terhubung ke index.php, dan kemudian seluruh proses pemuatan dimulai. Setidaknya, itu harus dilakukan. Pastikan Anda telah mengatur alamat web dan alamat server dengan benar di file konfigurasi Anda. Alamat web adalah folder root web Anda; Anda mungkin harus menanyakan alamat server ISP Anda, meskipun biasanya alamat tersebut jelas dari program 'pengelola file' mereka sendiri.

Saya mengalami masalah tertentu saat mencoba menjalankan sub-domain. Banyak host mengizinkannya, tetapi memetakan domain ke folder dengan cara yang tidak Anda harapkan.

Database

Menemukan dan menghubungkan ke database Anda sering menjadi masalah besar. Lihat file konfigurasi Anda dan file config/database Anda. Anda perlu memastikan bahwa Anda

memiliki situs dan alamat server yang benar, dan nama database, alamat, nama pengguna, dan kata sandi yang benar. Hati-hati dengan awalan—terkadang ini ditambahkan secara otomatis. (Situs Anda disebut 'fred'. Basis data Anda, menurut Anda, disebut 'mydata' dan nama pengguna Anda adalah 'mylogin'. Tetapi server mengira mereka disebut 'fred_mydata', 'fred_mylogin', dll.)

Terkadang, membantu untuk membuat pengguna baru di database Anda, bahkan jika Anda sudah memilikinya, dan mengatur login untuk nama dan kata sandi pengguna ini. Saya tidak tahu mengapa ini berhasil, tetapi itu berhasil.

Dalam file konfigurasi, Anda dapat mengatur CI untuk menerima berbagai jenis protokol URI, yang menentukan bagaimana server menangani string URI. Standarnya adalah:

```
$config['uri_protocol'] = "auto";
```

tetapi ada empat opsi lain yang dapat Anda coba jika ini tidak berhasil. Jika opsi yang benar tidak disetel, Anda mungkin menemukan situs Anda berfungsi sebagian, tetapi (misalnya) formulir tidak memanggil halaman target.

File konfigurasi lainnya

File config/routes menetapkan jalur default yang diikuti aplikasi, jika pengguna tidak menentukan metode pengontrol melalui URL (yaitu jika mereka hanya masuk ke www.mysite.com). Ini biasanya harus diatur pada default:

```
$route['default_controller'] = 'index';
```

Jika Anda mengganti nama folder sistem, ingatlah bahwa Anda juga perlu mengubah index.php di direktori root situs. Pengaturan default adalah:

```
$system_folder = "system";
```

12.2 PERHATIKAN PHP 4/5 DAN PERBEDAAN SISTEM OPERASI

CI harus dapat hidup dengan versi PHP apa pun termasuk dan yang lebih baru dari 4.3. Namun, ini tidak berarti bahwa kode PHP apa pun yang Anda tulis juga akan kompatibel—jadi jika Anda menulisnya di Xampplite menggunakan PHP 5, dan pindah ke ISP dengan server PHP 4, perhatikan masalah karena perbedaan bahasa.

PHP (dari versi apa pun) dapat diatur dengan cara yang berbeda. Layak untuk dijalankan phpinfo() di situs lokal dan jarak jauh Anda, dan memeriksa perbedaannya. Sensitivitas huruf besar-kecil berbeda antara server Microsoft dan Linux. Jadi, jika Anda telah mengembangkan situs di PC, lalu mengunggahnya ke server berbasis Linux, harap server melaporkan bahwa ia tidak dapat menemukan beberapa model atau pustaka yang ingin Anda muat. Jika sudah dicek dan sudah diupload, pastikan kapitalisasinya sudah benar. Karena definisi kelas dan konstruktor di CI memiliki untuk memulai dengan huruf kapital, mudah untuk memulai nama file dengan huruf kapital juga. Jika Anda memuat model, katakanlah, di dalam pengontrol, dan beri nama dengan huruf besar ($this->load->Mymodel), maka Windows dan Linux mungkin memiliki pandangan yang berbeda tentang panggilan ke $this->mymodel.

Sebagai contoh ekstrem dari perbedaan server, saya pernah mulai menulis pengontrol, memutuskan untuk menjadikannya model, dan menyimpannya di folder model tanpa menyadari bahwa saya telah meninggalkan beberapa baris pertama sebagai:

```
class Myclass extends Controller {

    function Myclass()
    {
                parent::Controller();
```

alih-alih mengubahnya menjadi:

```
class Myclass extends Model {
    function Myclass()
    {
        parent::Model();
```

Berjalan secara lokal di Xampplite, ini tidak menimbulkan pengecualian. Ditransfer ke server Linux jarak jauh, itu segera gagal (dan Anda dapat membayangkan berapa lama waktu yang dibutuhkan untuk melacak ...).

Beberapa fungsi PHP juga tampak berperilaku berbeda pada sistem operasi yang berbeda: misalnya, include_once() tidak peka huruf besar/kecil pada Windows tetapi tidak pada sistem lain. Itu bukan masalah CI secara khusus.

Selain itu, database Anda mungkin versi yang berbeda—banyak ISP yang sangat tradisional, misalnya menjalankan MySQL 3.23! Ini tampaknya menyebabkan beberapa ketidakcocokan, yang berarti bahwa mengunggah database dengan kueri SQL kurang mudah dari yang seharusnya. (Misalnya, mungkin tidak menerima komentar pada tabel db.)

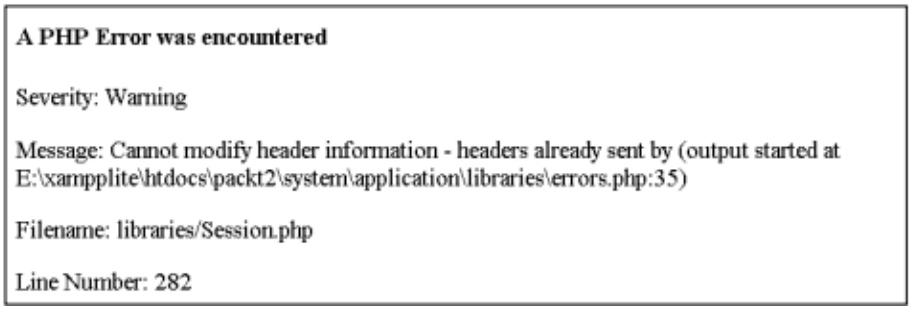
Linux memiliki sistem izin file yang berbeda dengan Windows. Pastikan Anda memiliki izin yang benar pada file dan folder Anda. Izin file CI tertentu harus diatur dengan benar sebelum sistem dapat bekerja.

Alat Diagnostik

Baris pertama dari file index.php adalah:

```
error_reporting(E_ALL);
```

yang menampilkan kesalahan PHP di layar Anda, seperti ini:



Gambar 12.1 tampilan kesalahan di PHP

Jelas, laporan kesalahan seperti ini terlihat buruk, dan mungkin memberi terlalu banyak informasi kepada peretas, jadi untuk versi produksi Anda mengubahnya menjadi:

```
error_reporting(0);
```

Tapi kemudian, masalah apa pun mungkin hanya menghasilkan layar kosong, tanpa bantuan informasi diagnostik. Anda mungkin harus mengaktifkan kembali pelaporan kesalahan, hingga situs Anda dapat berjalan. Salah satu kompromi adalah mengaturnya ke tingkat menengah, seperti:

```
error_reporting(E_ERROR);
```

Ini akan mencegah 'peringatan' tetapi akan tetap memberi Anda informasi tentang masalah serius. 'Peringatan' biasanya merupakan kondisi yang tidak menghentikan program untuk dijalankan, tetapi mungkin menunjukkan masalah mendasar lainnya yang belum Anda pertimbangkan.

Kelas CI Profiler—lihat Bab 8—juga sangat berguna: ini menunjukkan kepada Anda pertanyaan apa yang Anda lakukan, dan apa yang ada di array POST.

Berbagai alat lain yang menurut saya berguna ketika semuanya tidak berfungsi:

1. Atur CI untuk mencetak file log. (Selesai dari file konfigurasi—lihat Bab 8—Anda perlu untuk mengatur:

   ```
   $config['log_threshold'] = 4;
   ```

   4 menunjukkan semua pesan, termasuk hanya pemberitahuan dan peringatan; kadang-kadang ini adalah petunjuk untuk masalah mendasar. Kemudian lihat log (dicetak di /system/logs, diarsipkan berdasarkan tanggal.) Ini akan memberi tahu Anda bagian mana dari sistem CI yang telah dipanggil, sehingga Anda setidaknya dapat melihat di mana proses berhenti. (Setel nilai kembali ke 0 untuk mencegah pencatatan lebih lanjut. Ingatlah untuk melakukan ini, dan untuk menghapus file log setelah Anda selesai: sungguh menakjubkan betapa banyak ruang yang mereka gunakan.)
2. Jika Anda dapat mengaksesnya, cetak server PHP dan variabel sesi: print_r($_SERVER) dan:

   ```
   print_r($_SESSION)
   ```

   Dan gunakan untuk memeriksa apakah nilai document_root dan script_filename sesuai dengan yang Anda harapkan. Jika tidak, Anda mungkin perlu menyesuaikan nilai file konfigurasi untuk base_url dan server. Anda juga dapat melihat apakah ada set nilai [HTTP_COOKIE], yang akan menunjukkan kepada Anda apakah kelas sesi Anda diaktifkan dan berfungsi.
3. Periksa apa yang telah dimuat CI ke superobject-nya dengan menggunakan metode PHP:

   ```
   get_declared_classes();
   ```

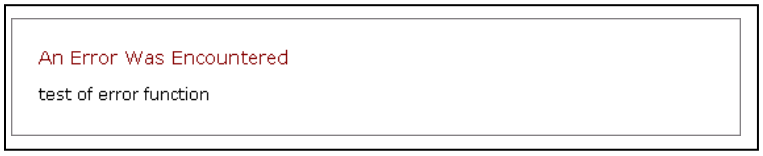
   dan:

   ```
   get_class_methods();
   ```

4. Fungsi show_error() CI sendiri hanyalah sarana untuk memformat laporan kesalahan yang Anda buat. Jadi sertakan baris berikut dalam kode Anda, katakan di beberapa cabang yang tidak boleh dijangkau:

```
show_error('test of error function');
```

akan menghasilkan layar Anda menunjukkan:



Gambar 12.2 memberikan kalimat saat terjadi kesalahan

Saya tidak menemukan itu sangat berguna. Yang saya inginkan adalah sistem yang akan memberi saya laporan kesalahan yang lengkap dan bermanfaat, kapan dan di mana saya menginginkannya, tetapi tidak akan menunjukkannya ketika saya tidak ingin mereka muncul. Saya menulis fungsi saya sendiri, yang berbunyi:

```
function reportme($file, $line, $message)
{
    $obj =& get_instance();
    if(isset($_POST))
            {$bert = print_r($_POST,  TRUE);} else
    {$bert = 'no post array';} if(isset($_SESSION))
          {$sid = print_r($_SESSION,  TRUE);}
    else{$sid = 'no session array';}
     $time = Gmdate("H:i j-M-Y");
/*full report*/
     $errorstring =  "$time - $file - $line: $message: POST array:
$bert SESSION array: $sid\n";
/*short report*/
    $shortstring =  "$file - $line: $message";
/*set $setting to 'test' if you want to write to the screen*/
    $setting = 'test'; if($setting ==
    'test')
            {echo $errorstring;}
/*set $action to 'log' if you want to log errors*/
    $action = 'log'; if($action ==
    'log')
            {
            $filename = $obj->config->item('errorfile');
            $fp = fopen("$filename", "a+")or die("cant open file");fwrite($fp, $errorstring);
            fclose($fp);
            }
}
```

Ini tinggal di perpustakaan yang disebut kesalahan. Saya harus ingat untuk memuat perpustakaan, dan kemudian, setiap kali saya memiliki bagian kode yang tidak saya yakini, saya menyertakan fungsinya:

```
$this->errors->reportme(__FILE__,__LINE__,'if the code has reached here it is because...');
```

Saya kemudian dapat mengatur fungsi reportme() untuk memberi saya laporan di layar, atau di file log saya.

Ada beberapa keuntungan dari metode sederhana seperti ini. Pertama, saya dapat mengubah fungsi reportme() dengan mudah, membuatnya menulis kesalahan ke file, atau tidak melakukan apa-apa sama sekali: jadi saya bisa membuat semua laporan saya hilang dari layar sekaligus, atau kembali lagi, dengan mengubah satu baris kode.

Kedua, katakanlah saya mengharapkan variabel memiliki nilai tertentu. (Nomor ID menjadi bilangan bulat, katakanlah.) Saya membuat pesan ini selengkap dan selengkap mungkin. Saya mencoba mengatakan apa yang saya harapkan untuk ditemukan (bilangan bulat), serta memasukkan nilai yang sebenarnya saya dapatkan. Panggilan fungsi juga menggunakan FILE PHP dan 'konstanta ajaib' LINE untuk memberi tahu saya di mana tepatnya itu terjadi.

Jadi, jika potongan kode ini menjadi masalah ketika saya mentransfernya ke server lain, mungkin beberapa saat setelah saya menulisnya, saya dapat segera menemukan kodenya, dan teks membantu saya mengingat mengapa itu menjadi masalah. Enam bulan setelah Anda menulis kode, Anda tidak bisa langsung mengambilnya, terutama jika sudah larut malam dan klien menelepon untuk meminta penjelasan! Semakin membantu teks kesalahan, semakin mudah untuk merespons dengan bijaksana.

Ketiga, jika integritas situs sangat penting, saya dapat mengatur fungsi untuk mengirim email kepada saya dengan laporan kesalahan. Itu mungkin menghasilkan kotak surat yang sangat penuh selama fase pengembangan, tetapi setelah situs stabil dan digunakan, mungkin sangat berguna untuk memiliki peringatan segera jika situs mengalami masalah. Anda akan tahu tentang mereka sebelum pengguna Anda memberi tahu Anda.

12.3 MENGATASI PERUBAHAN DALAM VERSI CI BARU

Antara 28 Februari 2006 dan 30 Oktober 2006, CI beralih dari versi beta pertama ke versi 1.5. Itu tingkat perkembangan yang cukup mengesankan.

Selama waktu itu Rick Ellis membuat beberapa perubahan yang cukup radikal, terutama pada struktur situs. Untuk sebagian besar, dia telah berhati-hati untuk membuat mereka kompatibel ke belakang — tetapi tidak semuanya. Jika Anda baru mengenal CI dan telah mengunduh versi terbaru, Anda dapat melewati bagian ini. Tetapi jika Anda menulis program menggunakan versi sebelumnya, Anda mungkin perlu memeriksa perubahan ini. Anda mungkin juga perlu memeriksa apakah Anda menggunakan pustaka CI atau plug-in yang ditulis oleh orang lain.

Rick telah bergulat dengan dua masalah utama:

Cara Memuat Model, dan Cara Memanggilnya

Pada awalnya, tidak ada model, hanya folder untuk skrip dan perpustakaan. Tidak ada ketentuan untuk menginisialisasi mereka secara otomatis sebagai bagian dari 'super-objek' CI. Akibatnya, Anda memiliki sistem MVC tanpa file 'model', yang tampaknya membingungkan.

Selain itu, ada dua folder perpustakaan: /system/application/libraries menyimpan file apa pun yang Anda tulis sendiri, sementara /system/libraries menyimpan file operasi sistem itu sendiri. Ini

mungkin membingungkan beberapa orang: keduanya sangat berbeda! Anda harus menambah atau mengubah yang pertama; Anda mungkin tidak perlu mengubah yang terakhir. (Dan jika Anda melakukannya, Anda menghadapi risiko ketidakcocokan yang serius jika Anda meningkatkan ke versi CI yang lebih baru: lihat di bawah.)

Dengan versi 1.3 datang kelas 'model' baru. Panduan Pengguna mendefinisikan model sebagai, "kelas PHP yang dirancang untuk bekerja dengan informasi dalam database Anda". Saat pertama kali diperkenalkan, model CI terhubung secara otomatis ke database. Namun, sejak Versi 1.3.3, Anda harus memuat database secara khusus dari dalam model atau pengontrol yang memanggilnya.

Atau, saat Anda memanggil model dari pengontrol, Anda dapat melakukannya dalam format ini:

```
$this->load->model('Mymodel', '', TRUE);
```

dan kemudian 'TRUE' memuat model dengan koneksi database default yang dibuat, seperti yang didefinisikan dalam file konfigurasi Anda. (Parameter kedua, dibiarkan kosong di sini, adalah alias opsional untuk model.)

CI mungkin masih akan berfungsi jika Anda memasukkan fungsionalitas 'model' (dalam arti MVC) ke dalam 'perpustakaan' atau 'skrip' (usang), seperti yang harus Anda lakukan pada hari-hari awal ketika tidak ada 'model' folder: tetapi Anda harus mengakses sumber daya CI secara berbeda—lihat bagian selanjutnya!

Cara Menginisialisasi 'library' Classes Sendiri

Awalnya, Anda tidak dapat menjadikan kelas Anda sendiri sebagai bagian dari 'objek super' CI. Ini adalah masalah, karena itu berarti bahwa kode perpustakaan Anda tidak dapat, misalnya, mengakses database melalui Rekaman Aktif, atau menggunakan perpustakaan CI lainnya, dan itu menjadi sangat membatasi.

Versi 1.2 menambahkan fungsi get_instance() yang memungkinkan Anda mengakses 'objek super'. (Lihat Bab 7.) Anda dapat memasukkannya ke dalam 'perpustakaan' atau 'skrip' Anda dan kemudian menggunakan sumber daya CI. (Kecuali file baru Anda adalah skrip fungsional daripada kelas OO, tentu saja. Namun, file skrip mungkin paling baik digunakan untuk sederhana fungsi tingkat rendah.)

Versi 1.4 memperkenalkan sistem baru. Anda harus membuat dua file untuk setiap kelas 'perpustakaan'. Yang pertama adalah kelas itu sendiri, misalnya Newclass.php, disimpan di folder application/libraries, dan yang kedua, disimpan di folder application/init, memiliki disebut init_newclass.php dan berisi beberapa baris kode standar yang menginisialisasinya sebagai bagian dari 'objek super'. Namun, Anda masih harus menggunakan fungsi get_instance() untuk mengakses sumber daya CI.

Di versi 1.5, folder init tidak digunakan lagi, dan inisialisasi terjadi secara otomatis. Anda sekarang hanya membutuhkan satu file untuk setiap kelas 'perpustakaan'. Folder skrip lama juga tidak digunakan lagi. 'Deprecate' dalam konteks ini, biasanya berarti bahwa hal yang bersangkutan sampai dikenali dan harus tetap berfungsi, tetapi pengembang tidak memberikan jaminan bahwa ia akan melakukannya di semua versi yang akan datang. Dengan

kata lain, jangan panik jika Anda masih memiliki skrip di folder sistem/aplikasi/skrip—tetapi jangan menulis lagi.

Jika Anda berencana untuk menggunakan perpustakaan atau plug-in yang ditulis oleh komunitas CI, harap periksa terlebih dahulu apakah mereka sudah diperbarui dengan versi CI terbaru. Masih ada beberapa yang ditulis untuk 1.4.1 dan memiliki file 'init' yang terpisah. Memperbarui mereka tidak sulit, tetapi perlu hati-hati untuk melakukannya dengan benar.

Jadi Haruskah Memperbarui Jika Versi CI Baru Keluar?

Versi baru CI keluar dari waktu ke waktu. Mereka datang dengan instruksi komprehensif untuk memperbarui. Biasanya, ini melibatkan penyalinan sekumpulan file baru ke folder sistem Anda. Kadang-kadang, Anda perlu mengubah file konfigurasi, atau file index.php Anda, juga, tetapi tidak satu pun dari ini adalah perubahan besar dan tidak satupun dari mereka adalah ilmu roket. Karena struktur folder menyimpan file aplikasi Anda di tempatnya sendiri, biasanya mudah untuk memperbarui sistem tanpa menyentuh aplikasi.

Tapi, katakanlah Anda telah menulis aplikasi pembunuh Anda di versi 1.5. Ini diunggah ke sistem produksi Anda dan berfungsi dengan baik. Kemudian, Rick Ellis mengeluarkan CI versi 1.6 (atau 2.8 atau apa pun…). Ini memiliki fitur baru yang menarik, dan beberapa perbaikan bug. Apakah Anda meningkatkannya?

Saya akan mengatakan, 'Ya', jika itu adalah peningkatan kecil, katakanlah antara 1.5.2 dan 1.5.3. Tetapi jika itu adalah perubahan versi utama, dan sistem Anda yang ada berfungsi, biarkan saja. Anda dapat membedakan sebagian dari penomoran, tetapi juga dari 'perubahan log' yang diterbitkan dengan setiap peningkatan ketika keluar. Jenis perubahan yang telah dibuat di CI selama setahun terakhir terbagi dalam tiga kategori:

Perbaikan bug: Ada beberapa yang mengejutkan—CI adalah kode yang sangat baik, dan sebagian besar kelas dasar telah diuji dengan baik oleh ratusan bahkan ribuan pengguna. Fitur baru:. Ini muncul secara teratur, tetapi jika Anda berhasil membangun aplikasi Anda tanpa mereka, apakah mereka akan sangat membantu sekarang?

Perubahan halus: Seperti yang telah saya jelaskan, CI telah melalui proses evolusi internal, dan mungkin akan terus demikian. Seperti yang Anda lihat dari tabel berikut, beberapa di antaranya mungkin kompatibel ke belakang, atau mungkin memerlukan penulisan ulang kode Anda yang cukup besar.

Tabel 12.1 Perubahan antar versi CI:

Versi	Ubah Log
1.2	Menambahkan fungsi global bernama get_instance() yang memungkinkan objek CodeIgniter utama dapat diakses di seluruh kelas Anda sendiri.
1.3	Menambahkan dukungan untuk Model.
1.3	Menambahkan kemampuan untuk meneruskan parameter inisialisasi Anda sendiri ke pustaka inti kustom Anda saat menggunakan $this->load->library().

1.3	Menambahkan kelas yang lebih baik dan spasi nama fungsi untuk menghindari tabrakan dengan kelas yang dikembangkan pengguna. Semua kelas CodeIgniter sekarang diawali dengan CI_ dan semua metode pengontrol diawali dengan _ci untuk menghindari tabrakan pengontrol.
1.3.3	Model tidak terhubung secara otomatis ke database pada versi ini.
1.4	Menambahkan kemampuan untuk mengganti kelas sistem inti dengan kelas Anda sendiri.
1.4	Memperbarui fungsi pemuat Model untuk memungkinkan banyak pemuatan dari model yang sama.
1.4.1	Plugin, helper, dan kelas bahasa yang diperbarui untuk memungkinkan folder aplikasi Anda berisi plugin, helper, dan folder bahasanya sendiri. Sebelumnya, mereka selalu diperlakukan sebagai global untuk seluruh instalasi Anda. Jika folder aplikasi Anda berisi salah satu dari sumber daya ini, mereka akan digunakan sebagai pengganti sumber daya global.
1.4.1	Tidak digunakan lagi folder aplikasi/skrip. Ini akan terus berfungsi untuk pengguna lama, tetapi Anda disarankan untuk membuat pustaka atau model Anda sendiri. Ini awalnya ditambahkan sebelum CI memiliki perpustakaan atau model pengguna, tetapi tidak diperlukan lagi.
1.5	Menambahkan kemampuan untuk memperluas perpustakaan dan memperluas kelas inti, selain dapat menggantikannya.
1.5	Folder init yang tidak digunakan lagi. Inisialisasi terjadi secara otomatis sekarang.

Jangan salah paham. Semua ini adalah perubahan yang masuk akal dan semuanya adalah perbaikan. Jika Anda memulai proyek baru, mulailah dengan versi CI terbaru. Tetapi jika Anda menulis kode menggunakan versi 1.3, katakanlah, Anda akan menemukan bahwa folder skrip Anda tidak digunakan lagi dan model Anda tidak lagi terhubung secara otomatis ke database. Secara pribadi, saya akan membiarkan kode itu berjalan pada CI versi 1.3, daripada mencoba memutakhirkannya. Hidup ini terlalu singkat.

12.4 CARA MENAMBAHKAN KELAS DASAR CI

Pengguna normal tidak perlu mengubah kelas CI dasar. Ini adalah kerangka kerja yang cukup bagus, ia melakukan banyak hal, dan bagaimanapun juga, inti dari kerangka kerja adalah untuk mempermudah, bukan? Namun, jika Anda harus ….

CI adalah open source, dan Anda dapat melihat semua kode segera setelah Anda mengunduhnya. Ini termasuk pustaka dasar yang membuat CI berfungsi (disimpan di sistem/perpustakaan) serta yang Anda tulis di sistem/aplikasi/perpustakaan.) Jadi selalu mungkin untuk mengubah CI sesuka Anda.

Mengubah file perpustakaan sistem memiliki dua masalah, namun:

- Tidak ada jaminan bahwa kode baru Anda akan kompatibel dengan CI lainnya, atau dengan versi yang diperbarui. Hal ini dapat menyebabkan kesalahan halus atau aneh yang tidak mudah dilacak.
- Jika nanti Anda memperbarui versi CI, folder sistem kemungkinan besar akan berubah. File perpustakaan yang Anda ubah mungkin akan ditulis ulang dan diperbarui, jadi Anda harus melalui perubahan Anda dan mentransfernya ke versi yang diperbarui.

Namun, sejak versi 1.5, sekarang ada dua 'penyelesaian' yang masuk akal untuk mengotak-atik kelas perpustakaan CI (kecuali untuk kelas 'database' dan 'pengontrol' yang mendasarinya, yang Anda sentuh dengan risiko Anda sendiri.)

- Pertama, Anda dapat membuat file dengan nama yang sama dengan salah satu kelas dasar sistem di folder /system/application Anda. Sistem kemudian menggunakan yang ini, dalam preferensi untuk yang standar di folder / sistem. Ini memerlukan konvensi penamaan yang tepat—lihat Panduan Pengguna online. Itu juga mengharuskan Anda untuk menyalin semua fungsionalitas di kelas yang ada serta tambahan Anda sendiri atau perubahan.
- Kedua, dan lebih mudahnya, Anda dapat membuat kelas baru yang memperluas kelas sistem. (Jadi mungkin lebih baik disebut sebagai 'sub-kelas'.) Sekali lagi, ada konvensi penamaan—lihat Panduan Pengguna online. Memperluas kelas sistem yang mendasari berarti bahwa sub-kelas baru Anda mewarisi semua sumber daya dari kelas CI yang mendasarinya, tetapi menambahkan beberapa metode tambahan Anda sendiri. Ini berarti bahwa, jika Anda memperbarui versi CI Anda, kelas CI yang mendasarinya akan diganti, tetapi sub-kelas baru Anda (yang harus Anda masukkan ke dalam folder sistem/aplikasi) tidak akan tersentuh.

Namun, tak satu pun dari metode ini akan menjamin bahwa kode Anda (atau tetap) kompatibel dengan CI lainnya.

Melihat melalui forum online CI, ada berbagai saran untuk memperluas kelas Validasi, Pengujian Unit, dan Sesi. Unit Testing, misalnya, hanya memiliki dua fungsi dan jumlah perbandingan yang terbatas. Mungkin Anda ingin fungsi menampilkan kesalahan dengan warna merah, sehingga menonjol saat hasil tes dikembalikan?

Jika Anda ingin menggunakan beberapa fungsi pengujian lainnya secara ekstensif, akan lebih mudah untuk menambahkannya melalui sub-kelas, memperluas Pengujian Unit, daripada menuliskannya di pengontrol setiap kali Anda memanggil Pengujian Unit.

Jika Anda ingin melakukan ini, Anda akan memulai sub-kelas baru Anda dengan cara ini:

```
class MY_Unit_test extends CI_Unit-test
{
        function My_Unit_test()
        {
                parent::CI_Unit_test();
        }
        function newfunction()
        {
                //new code here!
```

```
        }
    }
```

Perhatikan tiga hal di sini:

- Nama kelas pengujian unit yang mendasarinya adalah is CI_Unit_test, meskipun nama file kode kelasnya adalah system/libraries.unit_test.
- Jika Anda perlu menggunakan konstruktor di sub-kelas Anda, pastikan Anda memperluas konstruktor induk terlebih dahulu, seperti di sini.
- Nama sub-kelas baru Anda harus diawali dengan MY_, dan disimpan sebagai application/libraries/MY_unit_test.php. (Tidak seperti kelas utama, di mana awalan CI_ adalah bagian dari nama kelas tetapi bukan dari nama file, di sini awalan MY_ adalah bagian dari keduanya.)

Setelah Anda membuat sub-kelas, Anda memuatnya seperti ini:

```
$this->load->library('unit_test');
```

Dengan kata lain, persis sama seperti sebelum Anda menulis sub-kelas; dan Anda juga memanggil fungsi dengan cara yang sama, kecuali bahwa kali ini Anda tidak hanya dapat memanggil fungsi unit test yang ada, tetapi juga fungsi baru yang Anda tulis sendiri:

```
$this->unit_test->newfunction();
```

Saat Anda memperbarui instalasi CI Anda berikutnya, perpustakaan pengujian unit di folder sistem akan ditimpa, tetapi yang ada di folder aplikasi tidak, jadi kode Anda akan tetap ada di sana. Tentu saja, Anda harus memeriksa apakah pustaka sistem yang diperbarui masih kompatibel dengan kode Anda sendiri.

12.5 RINGKASAN

Dalam bab ini, kita telah melihat beberapa hal yang bisa salah ketika Anda mencoba mentransfer sistem Anda dari server lokal ke server jarak jauh. Ini mungkin melibatkan:

- Versi PHP atau MySQL yang berbeda
- Sistem operasi yang berbeda

Secara khusus, kami telah melihat sensitivitas huruf besar-kecil, perbedaan PHP, dan masalah MySQL. Kami juga telah melihat alat diagnostik.

Kemudian kami melihat pembaruan CI. Ini semua merupakan peningkatan besar, tetapi saran saya adalah, jika Anda memiliki sistem yang bekerja pada versi CI saat ini dan yang baru keluar, pikirkan baik-baik sebelum Anda meningkatkan.

Terakhir, kami melihat pro dan kontra dari penambahan ke kelas dasar CI. Sebagian besar pengguna tidak perlu melakukan ini, tetapi jika Anda mau, saya sangat menyarankan bahwa cara terbaik untuk melakukannya adalah dengan membuat sub-kelas kelas perpustakaan yang ada.

BAB 13
CRUD INSTAN ATAU MENYATUKAN SEMUANYA

Bagian paling penting—dan paling membosankan—dari menulis situs dinamis apa pun adalah CRUD. Anda memiliki satu atau lebih tabel database; Anda harus dapat membuat, Membaca, Memperbarui, dan Menghapus entri pada masing-masing entri tersebut. Nanti, Anda akan melakukan hal-hal cerdas dengan data, tetapi sampai ada beberapa cara yang mudah digunakan untuk meletakkannya di sana dan memeliharanya, situs Anda tidak layak.

Tapi ini melibatkan penulisan fungsi CRUD dan ini, meskipun secara konseptual cukup mudah, cukup rumit dan memakan waktu. Jadi untuk situs kami, saya telah menulis model CRUD umum, menggunakan kelas dan pembantu CI untuk membuatnya lebih mudah. Dalam bab ini, Anda akan melihat bagaimana model ini bekerja dan bagaimana mengintegrasikannya ke dalam aplikasi kita.

Model CRUD melakukan lebih dari sekedar CRUD. Ini memvalidasi data yang dimasukkan pengguna, dan juga memeriksanya dalam beberapa cara—misalnya, untuk melihat bahwa Anda tidak melakukan operasi 'hapus' tanpa membatasinya pada baris tabel tertentu, atau bahwa Anda tidak secara tidak sengaja mengulangi 'buat' operasi dengan kembali ke formulir entri dan memuatnya kembali di browser Anda.

Terakhir, model CRUD berisi kerangka pengujian sendiri, sehingga Anda dapat melakukan pengujian pengembangan saat Anda membangun atau mengadaptasi kode Anda. Ada kerangka kerja CRUD lain di luar sana, yang mungkin lebih komprehensif dan mungkin kodenya lebih baik (lihat Bab 15). Namun, yang satu ini berhasil. Dan ini adalah cara yang baik untuk menyimpulkan dan menggunakan banyak pelajaran yang telah kita pelajari di bab-bab sebelumnya.

Bab ini menetapkan kode untuk model, dengan beberapa komentar:

- Pertama-tama, kita melihat filosofi desain.
- Kemudian kita melihat pengontrol standar untuk digunakan dengan model.
- Kemudian kita melihat bagaimana tabel database harus terstruktur.
- Kemudian kita akan melihat model itu sendiri: pertama, array yang menyimpan informasi tentang database, dan kemudian pada fungsi yang terpisah.
- Terakhir, kita akan melihat fungsi self-test.

13.1 CRUD MODEL: FILOSOFI DESAIN

Ide di balik model CRUD ini adalah bahwa ia dapat dipanggil oleh pengontrol apa pun untuk tabel apa pun. Data tentang tabel itu, dan bagaimana Anda ingin formulir pembaruannya ditampilkan disimpan sekali, dalam sebuah larik. Segala sesuatu yang lain adalah standar: pengontrol hanya mengidentifikasi dirinya sendiri (dan dengan demikian tabel tempat kerjanya) dan jika perlu, memberikan nomor ID untuk catatan. Jadi yang harus Anda lakukan adalah menulis beberapa pengontrol sederhana, dan semua pekerjaan mengatur formulir dan membuat koneksi database selesai untuk Anda.

Ingat bahwa pengguna tidak dapat berbicara dengan model secara langsung, jadi dia harus melalui pengontrol setiap saat. Anda dapat meletakkan semua kode di pengontrol, tetapi kemudian Anda harus menyalin semuanya untuk setiap pengontrol baru. Dengan cara ini, hanya ada satu set kode CRUD dalam model, jadi hanya satu set yang diperbarui dan dipelihara. Harganya adalah Anda harus terus meneruskan informasi bolak-balik antara pengontrol dan model, yang membuat kode sedikit lebih sulit untuk diikuti.

Demi kesederhanaan, saya telah menggunakan dua fungsi eksternal yang tidak didefinisikan dalam kode ini:

- failure(), yang melaporkan kesalahan; Namun, saya ingin ini dilakukan.
- Model yang disebut tampilan—ini membuat menu dan mengatur URL dasar dan seterusnya. Jadi semua fungsi CRUD membangun setumpuk data, memasukkannya ke dalam variabel $data, dan cukup panggil:
  $this->display->mainpage($data);

Saya juga ingin model CRUD dapat menguji dirinya sendiri, jadi ini termasuk rangkaian pengujian mandiri. Memanggil ini selama proses desain memungkinkan saya untuk memeriksa apakah model akan berperilaku seperti yang saya inginkan, dalam kondisi apa pun yang dapat saya pikirkan. (Mengejutkan bagaimana menulis rangkaian pengujian membuat Anda menyadari hal-hal baru yang bisa salah—tetapi sekarang lebih baik daripada saat klien Anda menggunakan situs ini.)

Harap diingat bahwa setiap model seperti ini adalah kompromi. Semakin Anda memintanya, semakin ia meminta Anda. Misalnya, model ini tidak akan berfungsi kecuali tabel database Anda ditata dengan cara tertentu. Ini akan menyusun formulir dengan cara yang cukup canggih, tetapi tidak fleksibel tanpa batas. Itu tidak termasuk JavaScript untuk kontrol yang lebih baik dari pengalaman pengguna. Itu tidak dapat menangani pengecualian untuk aturannya sendiri. Di sisi lain, asalkan Anda hanya ingin melakukan serangkaian hal standar (yang cukup umum), itu membuat hidup jauh lebih mudah.

13.2 FORMAT STANDAR CONTROLLER

Pertama, untuk setiap tabel database, Anda memerlukan pengontrol standar. Beginilah cara pengguna berinteraksi dengan tabel Anda—misalnya, untuk menambahkan situs baru, mengubah detail situs yang sudah ada, dll. Untuk menambahkan orang baru, pengguna akan berinteraksi dengan tabel orang, jadi kita memerlukan pengontrol yang berbeda : tetapi hampir sama dengan pengontrol situs.

Ini adalah pengontrol untuk tabel situs kami:

```
<?php
class Sites extends Controller {
/*the filename, class name, constructor function names and this variable are the only
thing you need to change: to the name of thetable/controller (First letter in upper
case for the Class name andconstructor function, lower case for the file and
variable.lower case!)*/
    var $controller         = 'sites';
/*constructor function*/function
    Sites()
```

```
        {
                parent::Controller();
                $this->load->model('crud');
        }
    /*function to update an entry (if an ID is sent) or to insert a newone. Also includes
    validation, courtesy of CI */
        function insert($id)
        {
                $this->crud->insert($this->controller, $id);
        }
    /*interim function to pass post data from an update or insert throughto Crud model,
    which can't receive it directly*/
        function interim()
        {
                $this->crud->insert2($this->controller, $_POST);
        }
    /*function to delete an entry, needs table name and id. If calleddirectly, needs
    parameters passed to function; if not, from Postarray*/
        function delete($idno=0, $state='no')
        {
                if(isset($_POST['id'])&& $_POST['id'] > 0)
                        {$idno = $_POST['id'];}
                if(isset($_POST['submit']))
    {$state = $_POST['submit'];}

                $this->crud->delete($this->controller, $idno, $state);
        }
    /*function to show all entries for a table*/function showall()
        {
                $this->crud->showall($this->controller, $message);
        }

    /*function to show all data in a table, but doesn't allow anyalterations*/
        function read()
        {
                $this->crud->read($this->controller);
        }
    /*function to set off the test suite on the 'crud' model. This function need only appear
    in one controller, as these tests are madeon a temporary test table so that your real
    data is not affected*/
        function test()
        {
        $this->crud->test();
        }
    }
?>
```

Seperti yang Anda lihat, ini cukup ramping dan sepenuhnya digeneralisasi. Jika Anda ingin menjadikannya pengontrol orang alih-alih pengontrol situs—dengan kata lain, untuk memungkinkan Anda membuat, membaca, memperbarui, atau menghapus entri di tabel orang, yang perlu Anda lakukan hanyalah:

- Ubah nama kelas dari Sites menjadi People (huruf awal huruf besar!).

- Ubah variabel $controller dari sites menjadi people (huruf kecil).
- Ubah nama fungsi konstruktor dari Sites menjadi People (huruf awal huruf besar).
- Simpan pengontrol baru sebagai:
  system/application/controllers/people.php.

Nama pengontrol harus sama persis dengan nama tabel database yang terkait—jadi untuk tabel orang, harus orang. Nama harus memiliki huruf besar pertama di baris definisi kelas dan fungsi konstruktor, tetapi tidak di tempat lain.

13.3 TABEL DATABASE

Ada tiga aturan sederhana untuk tabel database Anda:

1. Bidang ID utama di setiap tabel harus selalu disebut 'id' dan harus berupa bidang yang bertambah otomatis. (Ini adalah tipe kolom MySQL standar. Secara otomatis membuat nomor unik baru setiap kali Anda membuat entri baru.)
2. Harus ada field yang disebut 'nama' di setiap tabel, jika Anda ingin menggunakannya sebagai
3. dasar untuk kotak drop-down dinamis dalam bentuk apa pun.
4. Anda juga memerlukan bidang 'kirim' untuk menahan status, dan hal-hal seperti itu.

Jika tidak, Anda dapat memiliki bidang apa pun yang Anda suka, dan beri nama apa pun yang disukai oleh sistem basis data Anda. Segala sesuatu yang lain ditangani oleh model CRUD, untuk pasangan tabel pengontrol yang dirancang sepanjang garis ini.

Jantung Model: Array

Babak penyisihan selesai. Mari kita mulai dengan model CRUD.

Pertama, Anda perlu mendefinisikan model CRUD dan fungsi konstruktor. Barang standar sekarang:

```
<?php
class Crud extends Model {
/*create the array to pass to the views*/ var $data =
    array();
    var $form = array(); var
    $controller;

    function Crud()
    {
        // Call the Model constructor
        parent::Model();
        $this->load->helper('form');
        $this->load->helper('url');
        $this->load->library('errors');
        $this->load->library('validation');
        $this->load->database();
        $this->load->model('display');
```

Save it as system/application/models/crud.php.

Kemudian datanglah bagian yang membosankan, tetapi Anda hanya perlu melakukannya sekali. Anda perlu menulis array multi-dimensi. (Saya mulai belajar PHP dari sebuah buku—lebih baik tanpa nama—yang mengatakan 'array multi-dimensi tidak terlalu sering ditemukan, jadi kita tidak akan membahasnya lebih detail di sini'. Sepertinya saya telah menggunakan mereka sejak itu.)

Dimensi pertama dari array kita adalah daftar tabel. (situs, orang, dll.)

Dimensi kedua adalah daftar bidang dalam setiap tabel. Untuk tabel situs, ini adalah id, nama, url, dll.)

Dimensi ketiga menjelaskan setiap bidang dan menyediakan satu set parameter yang mengontrol bagaimana hal itu akan ditangani oleh formulir insert�update. Ini adalah:

- Teks yang Anda ingin pengguna lihat pada formulir sisipan: bagaimana bidang ini dideskripsikan ke manusia, bukan nama sebenarnya. (Jadi yang pertama adalah Nomor ID situs ini bukan hanya id.) Ini untuk membantu Anda membuat formulir Anda ramah pengguna.
- Jenis objek formulir yang ingin Anda gunakan untuk menampilkan bidang ini pada formulir sisipkan/perbarui Anda: ini mungkin kotak input, atau area teks, atau kotak drop-down. (Model CRUD ini mencakup beberapa tetapi tidak semua opsi.)
- Aturan validasi CI apa pun yang ingin Anda terapkan saat pengguna mengisi formulir ini. Ini dapat dibiarkan kosong.
- Jika Anda ingin menampilkan bidang ini sebagai kotak drop-down dinamis, nama tabel yang akan digambar. Lihat di bawah untuk penjelasannya. Ini juga bisa dikosongkan.

Kita telah mendeklarasikan array sebagai variabel kelas $form, jadi setelah itu kita harus merujuknya sebagai $this->form). Itu didefinisikan di dalam konstruktor, yaitu, segera mengikuti dari kode sebelumnya.

```
$this->form =array
('sites' => array
(
'id'            => array('ID  number of this site','readonly', 'numeric'),
'name'          => array('Name  of site', 'textarea','alpha_numeric'),
'url'           => array('Qualified  URL,eg http://www.example.com', 'input', ''),
'un'            => array('username  to log in to site','input', 'numeric|xss_clean'),
'pw'            => array('password for site', 'input','xss_clean'),
'client1'       => array('Main  client','dropdown', '', 'people' ),
'client2'       => array('Second client', 'dropdown','', 'people'),
'admin1'        => array('First admin', 'dropdown','', 'people'),
'admin2'        => array(  'Second Admin', 'dropdown','', 'people'),
'domainid'      => array('Domain name', 'dropdown','numeric', 'domains'),
'hostid'        => array( 'Host', 'dropdown','numeric', 'hosts'),
'submit'        => array(  'Enter details', 'submit', 'mumeric')
),
'domains' => array
(
```

```
'id'                    => array('ID  number of this domain','hidden', 'numeric'),

//etc etc etc!!
```

Anda dapat melihat bahwa, di dalam array $form, ada sub-array untuk setiap tabel (di sini, situs dan domain, meskipun saya baru saja memulai yang terakhir, karena alasan ruang) yang masing-masing berisi sub-sub-array mereka sendiri, satu untuk setiap bidang ('id', 'nama', dll). Masing-masing sub-sub-array ini pada gilirannya adalah array yang berisi tiga atau empat nilai yang kami jelaskan di atas.
Mungkin sulit untuk mendapatkan sintaks array yang benar, tetapi secara konseptual sederhana.

Untuk set lengkap tabel dalam aplikasi kita, array ini membutuhkan sekitar 120 baris untuk ditentukan. Tetapi Anda hanya perlu melakukannya sekali! Ini adalah jantung dari model Anda. Akhiri fungsi konstruktor dengan tanda kurung tutup '}', dan lanjutkan ke fungsi lain dalam model CRUD.

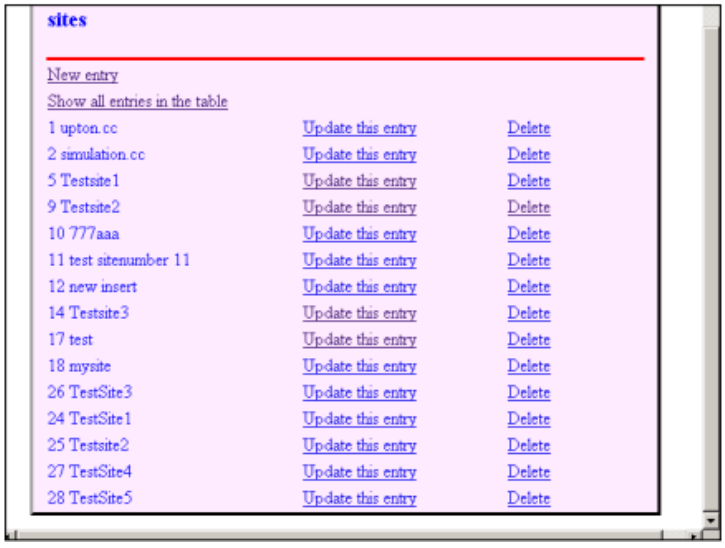
Jika Anda perlu mengubah tabel database Anda (misalnya menambahkan bidang baru) atau Anda ingin mengubah aturan validasi, maka Anda hanya perlu mengubah nilai dalam larik ini. Segala sesuatu yang lain akan berubah secara otomatis: misalnya, lain kali Anda mencoba menambahkan entri baru, Anda akan melihat formulir entri yang mencerminkan perubahan tersebut.

13.4 FUNGSI DEMI FUNGSI: CRUD MODEL

Berbagai fungsi yang membentuk model CRUD adalah sebagai berikut:

Tampilkan

Ini adalah fungsi yang paling sering dikunjungi pengguna. Ini bertindak sebagai titik masuk untuk semua operasi lainnya—membuat entri baru, memperbarui, atau menghapus entri. Ini menunjukkan kepada Anda apa yang sudah ada di tabel. Dengan beberapa data uji di tabel situs, terlihat seperti ini:



Gambar 13.1 Tampilan model Fungsi

Seperti yang Anda lihat, dari layar ini Anda dapat memperbarui entri untuk sebuah situs, atau menghapusnya. Anda juga dapat membuat entri baru, atau membaca semua data dalam tabel.

Omong-omong, harap diingat bahwa model ini tidak menyertakan ketentuan keamanan apa pun. Di situs nyata, Anda mungkin ingin menyempurnakan opsi pengguna — mis., izinkan mereka memperbarui tetapi tidak menghapus. Dan Anda ingin memastikan bahwa peretas tidak dapat mengakses fungsi model CRUD dengan mengetikkan URL seperti www.example.com/index.php/sites/delete/18. Struktur berbasis URL CI membuatnya relatif mudah untuk menyimpulkan bagaimana sistem mengakses perintah ini, jadi Anda mungkin ingin memastikan bahwa pengguna harus masuk ke situs sebelum model CRUD diaktifkan sama sekali.

Benar, kembali ke mekanisme CRUD. Ingatlah bahwa manusia tidak dapat memanggil model secara langsung. Dalam setiap kasus, opsi (untuk menghapus, memperbarui, dll.) dilakukan melalui panggilan ke pengontrol. Hanya untuk melompat kembali, pengontrol situs yang memanggil showall function melakukannya dengan baris kode ini:

```
$this->crud->showall($this->controller);
```

dengan kata lain, itu menggantikan situs untuk $this->controller, dan meneruskan nilai itu sebagai parameter ke fungsi CRUD, untuk memberi tahu pengontrol mana yang berfungsi.

Sekarang mari kita lihat fungsi showall. Telah melewati parameter pertamanya, situs. Kami akan meninggalkan $message sampai nanti. Berkonsentrasi pada garis yang disorot.

```
/*this function lists all the entries in a database table on one page. Note that every db
table must have an 'id' field and a 'name'field to display!
This page is a jumping-off point for the other functions - ie tocreate, read, update
or delete an entry.
When you've done any of these, you are returned to this page. It has a'message' parameter, so
you can return with a message - either successor failure.*/

function showall($controller='', $message = '', $test ='no')
 {
                $result = '';
                $mysess = $this->session->userdata('session_id');
                $mystat = $this->session->userdata('status')
                ;if(!$this->db->table_exists($controller))
                        {
                          $place = __FILE__._LINE__;
                          $outcome = "exception:$place:looking  for table
                          $controller:  it doesn't exist'";
 /*test block: what if there is no controller by that name?*/if($test =='yes')
                                {
                                  return $outcome;
                                }
                             else{
                                   $this->failure($outcome, 'sites');
                                    }
                              }
```

```
                /*end test block*/
                    $this->db->select('id, name');
                    $query = $this->db->get($controller); if
                    ($query->num_rows() > 0)
                            {
                            $result .= "<table class='table'>";

                            $result .= "<tr><td colspan='3'><h3>$controller</h3></td></tr>";
                            $result .= "<tr><td colspan='3' class='message'>
                            $message</td></tr>";
                            $result .= "<tr><td colspan='3'>";
                            $result .= anchor("$controller/insert/0", 'New entry');
                            $result .= "</td></tr>";
                            $result .= "<tr><td colspan='3'>";
                            $result .= anchor("$controller/read", 'Show all entries in the table');
                            $result .= "</td></tr>"; foreach ($query->result() as $row)

                                    {
                                    $result .= "<tr><td>";
                                    $result .= $row->id;
                                    $result .= " ";
                                    $result .= $row->name;
                                    $result .= "</td><td>";
                                    $result .= anchor("$controller/insert/ $row->id",'Update this
                                    entry');
                                    $result .= "</td><td>";
                                    $result .= anchor("$controller/delete/$row->id",'Delete');

                        $result .= "</td></tr>";
                        }
                        $result .= "</table>";
                        $data['text'] = $result;
                        $this->display->mainpage($data, $this->status);
                        }

                        {$place = __FILE__ . __LINE__ ;
                        $outcome = "exception: $place:
                        no results from table $controller";

                        /*test block: were there results from this table/ controller?*/ if($test == 'yes')
                        {$place = __FILE__ . __LINE__ ; return $outcome;
                        }
                        /*end test block*/

                else{
                $message = "No data in the $controller table";
/*note: this specific exception must return to another controller which you know does contain data.....
otherwise, it causes an infinite loop! */
                $this->failure($message, 'sites');
                }
        }
}
```

Ini membuat tabel, menampilkan beberapa data (id dan nama) tentang setiap entri. Setiap baris entri juga memberi Anda opsi untuk memperbarui atau menghapus entri: ini dicapai dengan menggunakan fungsi jangkar CI untuk membuat hyperlink ke fungsi yang sesuai di pengontrol yang sesuai.

Ada juga satu baris yang menawarkan Anda kesempatan untuk membuat situs baru, sekali lagi dengan menawarkan hyperlink ke fungsi insert controller. (Catatan: Saya telah memanggil fungsi insert baik untuk membuat entri baru dan memperbarui yang lama. Ini karena model mengasumsikan bahwa jika insert dipanggil dengan nomor ID, itu untuk memperbarui entri yang sesuai. Jika dipanggil tanpa ID nomor, itu membuat entri baru.)

Banyak kode yang diambil dengan penanganan pengecualian: bagaimana jika tabel tidak ada, bagaimana jika kueri tidak mengembalikan informasi? Pengecualian diteruskan ke fungsi kegagalan. Ada juga dua blok tes untuk memungkinkan saya menjalankan tes mandiri. Selain itu, ada baris yang memungkinkan Anda membaca (tetapi tidak mengubah) semua data dalam tabel. Mari kita lihat fungsi read terlebih dahulu, karena ini yang paling sederhana.

Membaca Data

Saya telah menggunakan Tabel HTML CI (lihat Bab 10) dan kelas Rekaman Aktif (lihat Bab 4) untuk menunjukkan betapa sederhananya fungsi ini. Saya ingin halaman berformat sederhana yang menampilkan semua data dalam database dalam tabel HTML. Itu tidak mengizinkan perubahan apa pun: ini secara harfiah adalah halaman 'baca'. Pertama, harus ada fungsi di pengontrol untuk memanggil model dan memberi tahu model tabel pengontrol mana yang akan ditampilkan. Itulah fungsi read() di controller standar.

Ini memanggil fungsi berikut dalam model CRUD:

```
/*queries the table to show all data, and formats it as an HTML table.*/
function read($controller)
{
        $this->load->library('table');
        $tmpl = array (
                'table_open'     => '<table border="1" cellpadding="4" cellspacing="0"
                                    width="100%">',
                'row_alt_start'    => '<tr bgcolor="grey">',
            );
        $this->table->set_template($tmpl);
        $this->load->database();
        $this->load->library('table');
        $query = $this->db->get($controller);
        $result =   $this->table->generate($query);
        $data['text'] = $result;
        $this->display->mainpage($data);
}
```

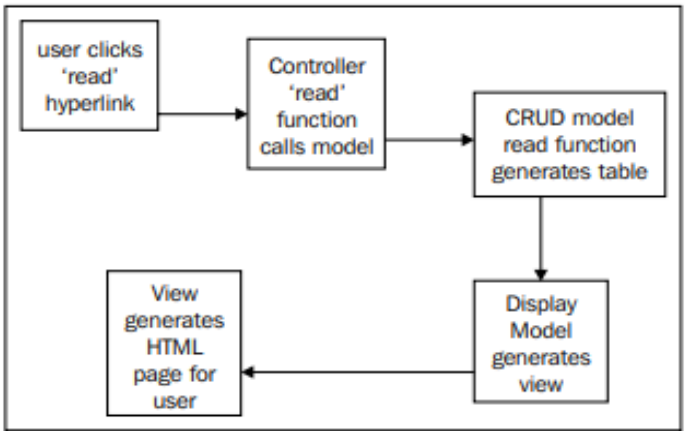
Dua baris yang disorot melakukan semua pekerjaan kueri database dan memformat hasilnya. Saya telah menggunakan fungsi halaman utama di kelas tampilan untuk menyediakan pemformatan halaman: fungsi baca di sini hanya membuat data dan menyerahkannya sebagai bagian dari array.

Hasilnya adalah halaman penuh data dari file tes:

26	TestSite3	http://www.example3.com	alfred	albert	1	1	1	1	1	1	/files
24	TestSite1	http://www.example1.com	fred	bert	1	1	1	1	1	1	htdocs
25	Testsite2	http://www.example2.com	bert	fred	1	1	1	1	1	1	public_html
27	TestSite4	http://www.example4.com	alphonse	aloysius	1	1	1	1	1	1	

Gambar 13.2 Pengujian file sintaks diatas

Mari kita ingatkan diri kita sendiri bagaimana kontrol melewati antara controller, model CRUD, dan bagian lain dari program.



Gambar 13.3 Alur kontrol model CRUD

Hapus dan Coba Hapus

Menghapus adalah operasi yang paling permanen! Untuk alasan ini, fungsi hapus kami memeriksa untuk memastikan dua hal:

1. Bahwa variabel status 'ya' telah disetel di bidang 'kirim': jika tidak, ia akan meneruskan permintaan ke fungsi trydelete. Itu menanyakan pengguna apakah dia benar-benar ingin melakukan penghapusan. Jika dia mengkonfirmasi, fungsi trydelete menetapkan variabel status 'ya' dan mengirimkan permintaan kembali ke fungsi hapus, yang sekarang menerima instruksi hapus.
2. Sebelum melakukan kueri penghapusan, ia memeriksa apakah nomor ID telah ditetapkan (jika tidak, semua entri mungkin akan dihapus). Kemudian, ia menggunakan Rekaman Aktif CI untuk melakukan penghapusan dan untuk memeriksa apakah satu baris memang telah dihapus dari tabel. Jika satu baris dihapus, maka ia kembali ke fungsi showall. Anda akan melihat bahwa itu mengembalikan dua parameter—nama pengontrol, dan pesan yang melaporkan bahwa penghapusan telah berhasil dilakukan. (Ini adalah parameter kedua yang ditampilkan. Jika disetel, parameter ini akan muncul di kotak merah di bagian atas tabel, memberi tahu pengguna apa yang sedang terjadi.)

Pertama, inilah fungsi hapus. Anda akan melihat kode ini juga rumit oleh banyak baris 'test block'. Abaikan ini untuk saat ini: cukup ikuti kode yang disorot..

```
/*DELETE FUNCTION:  given table name and id number, deletes an entry*/ function delete($controller, $idno, $state='no', $test='no')
{
/*first check that the 'yes' flag is set. If not, go through the trydelete function to give them a chance to change their minds*/
if(!isset($state) || $state != 'yes')
{
/*test block: are 'yes' flags recognised?*/
if($test == 'yes')
{
$place =  __FILE__ . __LINE__ ;
$outcome = "exception at $place: sent state value $state to trydelete function ";
return $outcome;
}
else
/*end test block*/

else{

{$this->trydelete($controller, $idno, 'no');}
}

/*'yes' flag is set, so now make sure there is an id number*/ if(isset($idno) && $idno > 0 && is_int($idno))
/*test block: with this id no, am I going to do a delete?*/
{
if($test == 'yes')
{
$place =  __FILE__ . __LINE__ ;
$outcome = "OK at $place:
doing delete on id of $idno ";
return $outcome;
}
else{
/*end test block*/
/*if there is an id number, do the delete*/
$this->db->where('id', $idno);
$this->db->delete($controller);
$changes = $this->db->affected_rows();
}
if($changes != 1)
{
/*test block: did I actually do a delete? */
$place =  __FILE__ . __LINE__ ;
$outcome = "exception at $place: cdnt do delete op on $controller with id no of $idno";
if($test == 'yes')
{return $outcome;}
else
/*end test block*/
/*if there was no update, report it*/
```

```
{$this->failure($outcome);}
}
else{
/*test block: I did do a delete*/
if($test == 'yes')
{return 'OK';}
else{
/*end test block: report the delete*/
$this->showall($controller,
"Entry no. $idno deleted.");}
}
}
else
/*test block: report id number wasn't acceptable'*/
{
$place =  __FILE__ . __LINE__ ;
$outcome = "exception at: $place : id no of $idno set for delete op in $controller, expecting integer";
if($test == 'yes')
{return $outcome;}
else
/*endtest block: if I failed, report me*/
{$this->failure($outcome);}
}
}
}
```

Saya berjanji untuk menjelaskan parameter $message yang kami gunakan saat kami memanggil showall. Anda dapat melihatnya di sini: jika fungsi ini berhasil, ia kembali ke halaman showall, dengan memanggilnya dengan pesan yang sesuai:

```
$this->showall($controller, "Entry no. $idno deleted.");}
```

Penting tidak hanya bahwa tindakan telah dilakukan, tetapi pengguna tahu bahwa itu telah dilakukan.

Sekarang, kembali ke pencegahan penghapusan yang tidak disengaja. Jika fungsi delete tidak dipanggil dengan parameter state=yes, ia akan merutekan ulang permintaan ke fungsi trydelete—'kesempatan kedua'. Sebenarnya, hanya fungsi trydelete yang akan disetel parameter ini menjadi ya, sehingga formulir hapus akan selalu menampilkan opsi apakah Anda yakin kepada pengguna.

Mari kita lihat fungsi trydelete. Itu menciptakan bentuk sederhana, yang terlihat seperti ini:

Gambar 13.4 tampilan fungsi trydelete

Mengklik ya akan memanggil kembali fungsi hapus. (Perhatikan lagi bahwa formulir tidak dapat kembali langsung ke mentah/hapus, karena formulir tidak dapat menunjuk ke model. Itu harus menunjuk ke situs/menghapus fungsi di controller, yang hanya meneruskan semuanya langsung ke mentah /hapus fungsi dalam model lagi.)

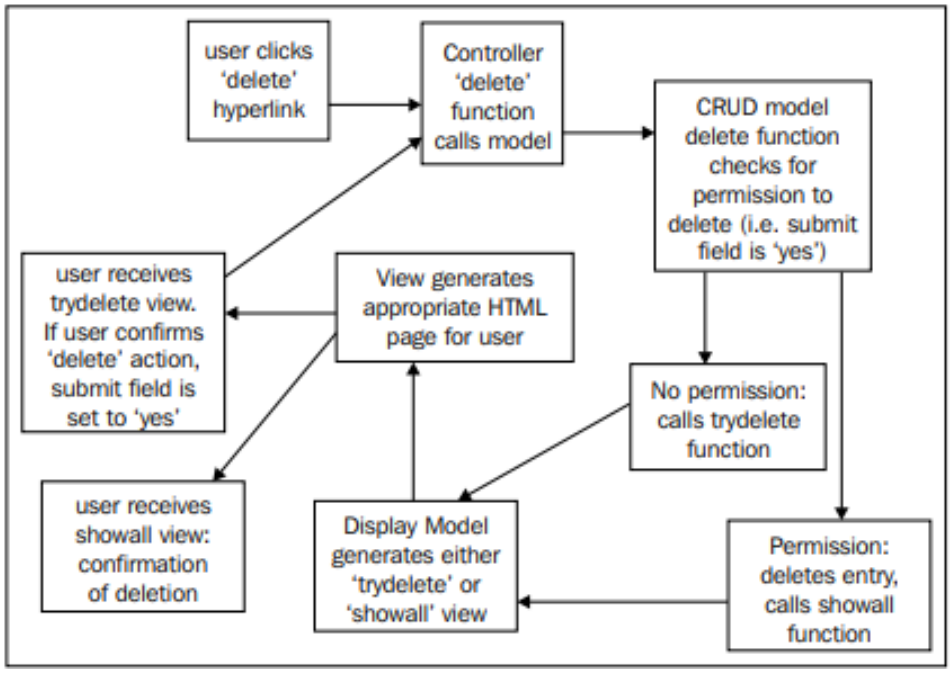
Perubahan halusnya adalah, jika pengguna mengonfirmasi penghapusan, formulir trydelete menambahkan (sebagai bidang tersembunyi) parameter submit=yes, yang masuk ke larik posting, dan dikembalikan ke fungsi hapus pengontrol. Fungsi delete controller membaca parameter submit=yes dari larik post, dan melakukan panggilan ke fungsi crud/delete, yang kali ini menyertakan state=yes sebagai parameter, sehingga fungsi delete berpindah ke langkah berikutnya.

Jika pengguna tidak ingin melakukan penghapusan, dia mengklik hyperlink yang dibuat oleh fungsi jangkar CI, dan diteruskan kembali ke fungsi showall, yang kemungkinan besar dari mana dia berasal.

Inilah kode yang melakukan semua ini:

```
/*TRYDELETE FUNCION: interrupts deletes with an 'are you sure?screen'*/
    function trydelete($controller, $idno, $submit = 'no')
            {
            if($state == 'yes')
            {$this->delete($controller, $idno, 'yes');}
            else{
                    $result .= "<table><tr><td>Are you sure you want todelete this
                    entry?</td></tr>";
                    $result .= form_open("$controller/delete");
                    $result .= form_hidden('id', $idno);
                    $result .= "<tr><td>";
                    $result .= form_submit('submit', 'yes');
                    $result .= "</td></tr>";
                    $result .= form_close();
                    $result .= "</table>";
                    $result .= anchor("$controller/showall","No, don't delete");
                    $data['text'] = $result;
                    $this->display->mainpage($data);
                    }
}
```

Hanya untuk kejelasan, berikut adalah diagram tentang bagaimana kontrol lewat selama operasi penghapusan.



Gambar 13.5 diagram kontrol penghapusan data sederhana

Seperti yang Anda lihat, ini cukup kompleks, lebih dari contoh kita sebelumnya. Model melakukan semua pekerjaan, tetapi pengguna hanya dapat berbicara dengan pengontrol, jadi jika Anda perlu kembali dan menyajikan kembali pertanyaan kepada pengguna, Anda perlu melibatkan pengontrol lagi.

Namun, setelah Anda menyelesaikannya, itu bekerja dengan baik dan sangat logis. CI memaksakan kerangka kerja ini pada Anda, tetapi dalam jangka panjang itu adalah keuntungan. Kode Anda konsisten, dan modular. Perhatikan bagaimana model tampilan yang sama dan tampilan yang sama dipanggil setiap kali: apa yang mereka tunjukkan kepada pengguna bergantung pada fungsi model CRUD yang memanggilnya.

Sisipkan

Ini adalah fungsi yang paling kompleks, karena menghasilkan formulir untuk diisi pengguna. (Antarmuka dengan manusia selalu menjadi hal yang paling sulit…)

Daripada menulis dua fungsi terpisah, satu untuk menyisipkan dan satu untuk memperbarui, dan harus membuat formulir dua kali, saya telah menulis satu fungsi yang berfungsi untuk keduanya. Jika Anda memberikan nomor ID yang valid, itu memperbarui catatan yang sesuai; jika tidak, itu memasukkan entri baru. Untuk membuatnya lebih mudah diikuti, saya belum menyertakan blok uji yang kita lihat di fungsi hapus.

Di sinilah kita menggunakan array yang kita definisikan di awal bab ini. Fungsi mengatur formulir, menggunakan pembantu formulir CI, dan berdasarkan jenis elemen formulir yang kami tentukan dalam array (dropdown, textarea, dll.). Inti dari fungsi adalah pernyataan switch, yang menyelesaikan ini.

Kode menggunakan kelas validasi CI untuk membantu kita memeriksa data yang masuk: ingat kita menetapkan aturan validasi dalam larik awal kita.

```
/*the most complex function. This creates an HTML form, based on the
description of the fields in the form array. This is sent to our
display model, which sets up a view and shows it to the user.
The view then sends a POST array back to the controller. The form
can't call this model directly, so it has to call the controller,
which refers it back to the model.
Note the function parameters:
1. The controller parameter is whichever controller/ table has called
the model - eg the 'sites' controller, or the 'domains' controller.
The controller has the same name as the table it manipulates.
2. The optional id parameter is the id of an individual entry in that
table.
3. The optional 'test' parameter is so you can set the form up to make
usable responses to self-test functions.
*/
    function insert($controller='', $id=0, $test='no')
    {
          $myform = '';
          $myid = 0;
             $currentvalue = array();
   /*test if the table exists*/
             if(!$this->db->table_exists($controller))
                    {
                      $place =  __FILE__.__LINE__;
                      $outcome = "exception: $place:looking for table
   $controller: it doesn't exist'";
                     if($test =='yes')
                          {
                           return $outcome;
                          }
                    else{
                     $this->failure($outcome, $controller);
                           }
                     }
              else
             {
                                 if($test =='yes')
                                   {
                                   return 'OK';
                                   }
             }
   /*end test block*/
   /*next check if there is an id number. If there is, we need to get the
   values to populate the table fields*/
             if(isset($id) && $id > 0)
                   {$myid = $id;
                   $this->db->where('id', $id);
                   $query = $this->db->get($controller);
                   if ($query->num_rows() > 0)
                          {
                           $row = $query->row();
   //--------------work out the values we want!
                          foreach($row as $key =>$value)
   /*
   first of all work out what value you want to show as the existing
   value in each line of the form. In priority order these are:
   1. the last value the user entered, from the post array
   2. the value from the database
   3. nothing, if neither of these is set.
   if we got here, the id does exist and is returning values, so get the
   existing values into a value array. Or, if there is something in the
   validation array, use that instead*/
                          {
                          $_POST[$key] = $this->validation->$key;
```

```
                        if(isset($_POST[$key]))
                                {$currentvalue[$key] = $_POST[$key];}
                        else
                                {$currentvalue[$key] = $value;}
                        }
/*test block: there was an id number, so has the program gone for an
update? if this is not a test, of course, just do the update*/
                        if($test == 'yes')
                                {
                                $place =  __FILE__.__LINE__;
                                $outcome = "exception: $place: id of $id
returned results from $controller table so have gone for update";
                                return $outcome;
                                }

/*end test block*/
                        $myform .= "<tr><td colspan='2'>Update
existing entry number $id</td></tr>";
                        }
/*now catch situation where this query isn't returning results. We
could only have got here with an integer set as our ID number, so
this probably means we are trying to delete an entry that doesn't
exist.*/
                else{
                                $place =  __FILE__.__LINE__;
                                $outcome = "exception: $place: despite
id of $id cant get any results from $controller table";
                        if($test == 'yes')
/*test block: there was and ID but there were no results*/
                                {
                                return $outcome;
                                }
/*end test block*/
                         else
                                {$this->failure($outcome, $controller);}
                         }
                }
/*there was no ID number, so this is a new entry*/
          else{
/*If the user has filled in values, and has returned here because some
of them didn't validate, we still need to repopulate the form with
what he entered, so he only has to alter the one that didn't validate.
Get these from the post array*/
                if(isset($_POST))
                        {
                        foreach($_POST as $key => $value)
```

```
                                {
                                 if(isset($_POST[$key]))
                                   {$currentvalue[$key] = $_POST[$key];}
                                 }

                        }
                $myform .= "<tr><td colspan='2'>New entry</td></tr>";

/*test block: there was no ID, so this is a new entry*/
                        if($test == 'yes')
                                {
                                $place =  __FILE__.__LINE__;
                                $outcome = "exception: $place: id of $id
treated as no id, so going for new entry";
                                return $outcome;
                                }
/*end test block*/
                }

/*the table exists, whether this is an update or new entry, so start
to build the form*/
          $myform       .=      "<table class='table'>";
          $myform .= form_open("$controller/interim");
          $myform .= '<p>This entry could not be made because...</P>';
          $myform .= $this->validation->error_string;


/*the rest of this function is common to inserts or update.
Look up in the form array which form field type you want to display,
and then build up the html for each different type, as well as
inserting the values you want it to echo.*/


                foreach($this->form[$controller] as $key => $value)
                {
/*This switch statement develops several types of HTML form field
based on information in the form array.
It doesn't yet cover checkboxes or radio or password fields. It adds
a 'readonly' type, which is a field that only displays a value and
doesn't let the user modify it*/

                $fieldtype = $value[1];
                $val_string = $this->validation->$key;
                switch($value[1])
                        {
/*a simple input line*/
                        case 'input':
                        $data = array(
                'name'          => $key,
```

```
                'id'          => $key,
                'value'       => $currentvalue[$key],
                'maxlength'   => '100',
                'size'        => '50',
                'style'       => 'width:50%',
        );
                $myform .= "<tr><td>$value[0]</td><td>";
                $myform .= form_input($data);
                $myform .= "</td></tr>";
                if($test == 'second')
                        {
                        return 'input';
                        }
                break;

               case 'textarea':
/*a text area field.*/
                $data = array(
                'name'        => $key,
                'id'          => $key,
                'value'       => $currentvalue[$key],
                'rows'         => '6',
                'cols'        => '70',
                'style'       => 'width:50%',
        );
                $myform .= "<tr><td valign=
                                'top'>$value[0]</td><td>";
                $myform .= form_textarea($data);
                $myform .= "</td></tr>";
                break;
                case 'dropdown':
/*a drop-down box. Values are dynamically generated from whichever
table was specified in the forms array. This table must have an id
field (which is now entered in the form) and a name field (which is
displayed in the drop-down box).*/
                $dropbox = array();
                if(isset($value[3]))
                        {
                        $temptable = $value[3];
                        $this->db->select('id, name');
                        $query = $this->db->get($temptable);
                        if ($query->num_rows() > 0)
                              {
                             foreach ($query->result() as $row)
                              {
                               $dropbox[$row->id] = $row->name;
                              }
                              }
                        }
```

```
                        $myform .= "<tr><td valign=
                                                'top'>$value[0]</td><td>";
                        $myform .= form_dropdown($key, $dropbox,
$currentvalue[$key]);
                        $myform .= "</td></tr>";
                        break;

                        case 'submit':
/*a submit field*/
                        $myform .= "<tr><td>$value[0]</td><td>";
                        $time  =  time();
                        $data = array(
                  'name'         => 'submit',
                  'id'           => 'submit',
                );
                        $myform .= form_submit($data);
                        $myform .= "</td></tr>";

                        break;

                        case 'hidden':
/*generates a hidden field*/
                        $myform .= form_hidden($key,
$currentvalue[$key]);
                        break;

                        case 'readonly':
/*generates a field the user can see, but not alter.*/

                        $myform .= "<tr><td>$value[0]</td><td>$current
value[$key]";

                        $myform .= form_hidden($key,
$currentvalue[$key]);
                        $myform .= "</td></tr>";


                        break;

                        case 'timestamp':
/*generates a timestamp the first time it's set*/
//                      $myform .= "<tr><td>$value[0]</td><td>now()";
                        $timenow = time();

    if($currentvalue[$key]==''||$currentvalue[$key]==0)
                                {$time = $timenow;}
                        else{$time = $currentvalue[$key];}

                        $myform .= form_hidden($key, $time);
                        $myform .= "</td></tr>";


                        break;

                        case 'updatestamp':
/*generates a timestamp each time it's altered or viewed*/
//                      $myform .= "<tr><td>$value[0]</td><td>now()";
```

```
                        $timenow = time();
                        $myform .= form_hidden($key, $timenow);
                        $myform .= "</td></tr>";
                        break;

                        default:
                        $place =  __FILE__.__LINE__;
                        $outcome = "exception: $place:
                                    switch can't handle $fieldtype";
/*test block: what if the switch doesn't recognise the form type?'*/
                         if($test == 'second')
                               {
                               return $outcome;
                              }
/*test block ends*/
                        else {

                        $this->failure($outcome, $controller);
                               }
                        }
/*end the foreach loop which generates the form*/
          }
          $myform .= form_hidden('submit',$time);
          $myform .= form_close();
          $myform .= "</table>";
/*Finally we've built our form and populated it! Now, stuff the form
in an array variable and send it to the model which builds up the rest
of the view.*/
          $data['text'] = $myform;
          $this->display->mainpage($data);
}
```

Beberapa hal untuk dijelaskan di sini. Semua tipe bidang formulir adalah standar, kecuali untuk hanya-baca—yang merupakan bidang formulir tersembunyi yang memungkinkan Anda melihat, tetapi tidak mengubah, apa yang tertulis di dalamnya. Ini tidak aman, tentu saja: pengguna yang cerdas dapat dengan mudah meretas nilainya. Itu hanya dirancang untuk menyederhanakan pilihan yang dihadapi pengguna.

Anda akan melihat formulir menunjuk ke fungsi yang disebut interim, pada pengontrol mana pun yang memanggilnya. Sekali lagi, itu karena Anda tidak dapat menangani model secara langsung melalui URL-nya. Jadi, jika diatur oleh pengontrol 'sites', formulir menunjuk ke 'sites�interim' dan nilai yang dimasukkan oleh pengguna, atau dari data yang ada, dikemas dalam array $_POST dan dikirim ke sana. Seperti yang akan Anda ingat dari awal bab ini, fungsi itu hanya memanggil fungsi mentah insert2, meneruskan array $_POST ke sana sebagai parameter.

insert2

Insert2 menerima larik $_POST sebagai parameter dan memeriksa apakah ia memiliki set bidang 'id'. Jika ya, itu memperbarui entri itu. Jika tidak, itu membuat entri baru. Agar kelas validasi CI, yang memerlukan larik $_POST, dapat berfungsi, fungsi kami mengganti nama larik yang diterimanya sebagai parameter sebagai $_POST.

```
   function insert2($controller, $newpost, $test = 'no')
          {
          $myform = '';
/*test the incoming parameters*/
          if(!$this->db->table_exists($controller))
          {
//test here!
          }

          $this->load->library('validation');
/*handle the validation. Note that the validation class works from
the post array, whereas this function only has a $newpost array: same
data, but different name. So we re-create the $_POST array.
*/
          $_POST = $newpost;
/*now build up the validation rules from the entries in our master
array*/
          $errorform = '';
          $newtemparray = $this->form[$controller];
          foreach($newtemparray as $key => $value)
                {$rules[$key]= $value[2];}
          $this->validation->set_rules($rules);
/*and the name fields*/
          foreach($newtemparray as $key => $value)
                {$fields[$key]= $value[0];}
          $this->validation->set_fields($fields);

    $this->validation->set_fields($fields);
/*now do the validation run*/
          if ($this->validation->run() == FALSE)
                {
/*if the validation run fails, re-present the entry form by calling
the 'insert' function*/
                $id = $_POST['id'];
                $this->insert($controller, $id, 'no', $_POST);
```

```
                }
            else
            {
/*The validation check was OK so we carry on. Check if there is an id
number*/
            if(isset($_POST['id']) && $_POST['id'] > 0)
                {
/*if yes: this is an update, so you don't want the id number in the
post array because it will confuse the autoincrement id field in the
database. Remove it, but save it in $tempid to use in the 'where'
condition of the update query, then do the update*/
                $tempid = $_POST['id'];
                unset($_POST['id']);
                $this->db->where('id', $tempid);
                $this->db->update($controller, $_POST);
                if($this->db->affected_rows()== 1)
                    {$this->showall($controller, "Entry number
$tempid updated.");}
                else{$this->failure("Failed to update $controller for
id no $tempid", __FILE__,__LINE__);}

/*if no id number, we assume this is a new entry: no need to unset the
post array id as it isn't there! the database will create its own id
number. Do the new entry*/
                $this->db->insert($controller, $_POST);
                if($this->db->affected_rows()== 1)
                    {$this->showall($controller,
                                    "New entry added.");}
                else{$this->failure("Failed to make new entry in
$controller ", __FILE__,__LINE__);}
                }
                }
            }
```

Dan itu saja. Beberapa ratus baris kode, yang memungkinkan Anda melakukan CRUD di meja mana pun.

Suite Test

Ingat 'blok uji' itu di fungsi hapus? Tujuannya hanyalah untuk mendeteksi apakah fungsi sedang dijalankan 'nyata' atau untuk pengujian, dan, jika yang terakhir, untuk memastikan bahwa ia mengembalikan nilai yang dapat kita uji dengan mudah.

Ini semua karena, pada akhir model CRUD, kami memiliki rangkaian 'pengujian sendiri'. Ini dipanggil oleh fungsi tes di pengontrol apa pun (tidak masalah yang mana) dan melakukan tes CRUD umum menggunakan tabel dummy.

Pertama di kelas CRUD ada fungsi master 'test', yang hanya ada untuk memanggil yang lain.

```
/*now a suite of self-test functions.*/
/*first function just calls all the others and supplies any formattingyou want. Also it builds/
destroys temporary data table before/ aftertests on the database.*/
    function test()
            {
            $return = "<h3>Test results</h3>";
```

```
        $this->extendarray();
        $return  .= $this->testarray();
        $this->reducearray();
        $return  .= $this->testarray();
        $this->testbuild();
        $return  .= $this->testdelete();
        $this->testdestroy();
        $return  .= $this->testinsert();
        $return  .= $this->testinsert2();
        $return  .= $this->testshowall();
        $data['text'] = $return;
        $this->display->mainpage($data);
    }
```

Ini hanya merakit tes apa pun yang Anda inginkan, dan menjalankannya.
Namun, daripada membahas semua fungsi ini, mari kita tunjukkan satu saja: fungsi pengujian yang disebut testdelete().

Namun, pertama-tama, kita membutuhkan dua fungsi: satu untuk membangun, dan satu lagi untuk menghancurkan, tabel pengujian dummy khusus kita, 'fred'. Fungsi pertama menghancurkan tabel 'fred' yang ada, membangun yang lain, dan menempatkan sederet data uji di dalamnya:

```
/*this function builds a new temporary table. 'fred', in your databaseso you can test the
CRUD functions on it without losing real data*/
    function testbuild()
    {
       $this->db->query("DROP TABLE IF EXISTS fred");
       $this->db->query("CREATE TABLE IF NOT EXISTS fred (id INT(11) default
       NULL, name varchar(12)  default NULL)");
       $this->db->query("INSERT INTO fred VALUES (1, 'bloggs')");
    }
```

Bergantung pada pengujian yang ingin Anda jalankan, Anda dapat membuatnya lebih rumit—misalnya, mengisi lebih banyak bidang, atau memiliki lebih banyak baris data.
Yang kedua menghancurkan meja sehingga kita bisa mulai dari awal. Seharusnya tidak ada nilai yang tersisa di dalamnya setelah kita melakukan tes hapus, tetapi jika itu gagal, atau jika kita menulis tes lain, mari kita pastikan:

```
/*this function destroys the temporary table, to avoid any confusionlater on*/
     function testdestroy()
     {
     $this->db->query("DROP TABLE IF EXISTS fred");
     }
```

Sekarang kita dapat mulai menguji fungsi hapus:

```
    function testdelete()
    {
    $result = '<p>Deletion test</p>';
```

Tes pertama yang mungkin kita lakukan adalah memastikan bahwa fungsi 'delete' memotong panggilan hapus tanpa parameter $state dari yes, dan mengirimkannya ke fungsi trydelete untuk menanyakan 'apakah Anda yakin?'

Ingatlah bahwa kita ingin pengujian mengembalikan 'OK' jika program menangani setiap kemungkinan dengan benar—bukan jika kemungkinan itu sendiri 'benar' atau 'salah'. Jadi jika parameter 'status' mengatakan 'haggis', yang jelas-jelas 'salah', pengujian harus mengatakan 'OK' selama program memperlakukannya sebagai 'tidak ya'! Idealnya, kami hanya ingin daftar singkat yang menyoroti kegagalan: jika pengujian berhasil, kami tidak perlu mengetahui detailnya.

Pertama-tama kita menyiapkan sebuah array, di mana setiap kunci adalah ekspresi yang mungkin kita gunakan dalam pengujian, dan nilai yang sesuai adalah hasil yang kita harapkan:

```
$states = array(
                        'no'    =>      'exception', '1'  =>
                        'exception', 'haggis'=> 'exception',
                        'yyyes'  =>  'exception', 'yes'    =>
                        'OK'
                         );
foreach($states AS $testkey => $testvalue)
        {$test = $this->delete('fred', 1, $testkey, 'yes');
 /*if you got the value you want, preg_match returns 1*/
        $result .=  $this->unit->run(preg_match("/$testvalue/",
 $test), 1, $test);
}
```

Dengan asumsi kode kami berfungsi dengan baik, itu akan mengembalikan:

Test Name	exception at E:\xampplite\htdocs\packt2\system\application\models\crud.php657: sent state value no to trydelete function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	376

Test Name	exception at E:\xampplite\htdocs\packt2\system\application\models\crud.php657: sent state value 1 to trydelete function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	376

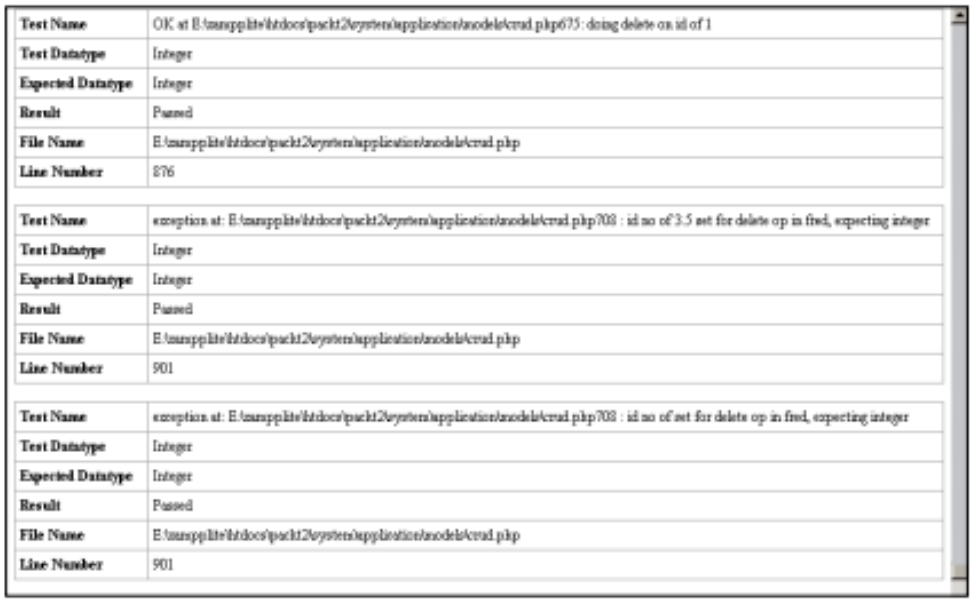
Test Name	exception at E:\xampplite\htdocs\packt2\system\application\models\crud.php657: sent state value yyyes to trydelete function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	376

Gambar 11.6 Tampilan array file yang dihapus

Pengujian kami berikutnya adalah untuk melihat bagaimana, dengan status yang benar, fungsi hapus bereaksi terhadap serangkaian nilai ID—termasuk non-bilangan bulat, nilai negatif, dll. Berhati-hatilah dengan perincian pengujian. Misalnya 9999 adalah nomor ID yang valid karena merupakan bilangan bulat dan lebih besar dari 0, tetapi itu tidak akan mengarah ke operasi penghapusan karena kami hanya memiliki satu catatan, dengan ID 1! Anda harus jelas tahap proses mana yang Anda uji.

```
        /*given $state set to 'yes', test another array of values for the idnumber. Start by
        building a test table*/
            $this->testbuild();
        /*then another array of values to test, and the results you expect..*/
$numbers = array(
                                '9999'    =>      'OK',
                                '-1'      =>      'exception',
                                'NULL'    =>      'exception',
                                '0'       =>      'exception',
                                '3.5'     =>      'exception',
                                ''        =>      'exception',
                                '1'       =>      'OK'
                                );
        /*now do the tests*/
              foreach($numbers AS $testkey => $testvalue)
                        {$test = $this->delete('fred', $testkey, 'yes', 'yes');
                      $result           .=      $this->unit->run(preg_match("/
        $testvalue/", $test), 1, $test);
                      }
        /*destroy the test table, just in case*/
              $this->testdestroy();
        /*return the results of this test*/return $result;
}
```

Semua baik-baik saja, itu akan mengembalikan sesuatu seperti ini:



Test Name	OK at E:\xampplite\htdocs\packt2\system\application\models\crud.php675: doing delete on id of 1
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	876

Test Name	exception at: E:\xampplite\htdocs\packt2\system\application\models\crud.php708 : id no of 3.5 set for delete op in fred, expecting integer
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	901

Test Name	exception at: E:\xampplite\htdocs\packt2\system\application\models\crud.php708 : id no of set for delete op in fred, expecting integer
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampplite\htdocs\packt2\system\application\models\crud.php
Line Number	901

Gambar11.7

Anda dapat menambahkan tes sebanyak yang Anda inginkan.

Pengujian membantu proses pengembangan. Saat Anda memikirkan nilai yang berbeda untuk dimasukkan ke dalam array pengujian Anda, Anda harus mempertimbangkan apakah kode Anda sebenarnya akan menanganinya dengan baik. Ini juga dapat membantu nanti, jika Anda mengubah kode dan secara tidak sengaja membuat kesalahan; dan itu akan meyakinkan Anda, setelah kode masuk ke situs produksi, untuk menjalankan tes sesekali.

13.6 RINGKASAN

Ini adalah bab yang panjang tetapi telah menarik banyak hal bersama-sama. Kami telah melihat:

- Bagaimana menggeneralisasi operasi CRUD sehingga Anda dapat melakukannya dengan dua kelas: satu untuk controller, dan satu untuk model CRUD. Yang pertama perlu diulang untuk setiap tabel, yang terakhir tetap sama.
- Sebagai hasilnya, kami dapat membangun berbagai pemeriksaan dan pengamanan, serta pengujian, sehingga kami dapat yakin bahwa operasi CRUD kami selesai.

Menggunakan CI memungkinkan kita untuk menulis semua ini dalam beberapa ratus baris kode (relatif) sederhana, yang dapat kita gunakan kembali di hampir semua situs yang kita buat, asalkan kita mematuhi beberapa aturan penamaan dan tata letak sederhana. Bagi saya, itulah kerangka kerja.

BAB 14
PUTUSAN ATAS CI

Buku ini dimulai dengan beberapa contoh spesifik tentang bagaimana CodeIgniter dapat menghemat waktu dan usaha Anda ketika Anda menulis situs web menggunakan PHP. Kami telah melalui beberapa dari banyak hal yang dapat dilakukan CI, dengan menggunakan sebagai dasar beberapa bagian situs web yang akan melakukan pengujian rutin terhadap situs web lain. Saya harap contoh-contoh ini telah menunjukkan bagaimana CI membuat pengkodean lebih mudah di tingkat makro.

Dalam bab ini, saya ingin mundur sedikit dan melihat dampak keseluruhan dari penggunaan framework CodeIgniter. Apakah menulis aplikasi lengkap lebih mudah? Bisakah itu menghasilkan hasil yang profesional?

Saat Anda menulis buku seperti ini, penting untuk membaginya menjadi beberapa bagian dan fokus pada satu trik baru pada satu waktu. Itu berarti terkadang sulit untuk melihat bagaimana semua bit cocok bersama. Saya harap kode 'Instant CRUD' di bab terakhir dapat digunakan untuk menyatukan bit-bit kode yang berbeda, menggabungkan Rekaman Aktif dan pengujian unit dan formulir. Dalam bab ini, saya ingin menunjukkan bagaimana semua bagian dapat disatukan untuk menghasilkan proyek yang telah selesai. Dengan kata lain, apakah proyek pengujian situs web kami berfungsi?

Melihat dari atas ke bawah pada beberapa kode spesimen dari situs akan membantu kita menyusun neraca:

- Dimana CI membantu
    1. Mengatur situs
    2. Membuat kode lebih sederhana
    3. Melakukan sesuatu untuk Anda (validasi data, dll.)
- Dimana CI tidak banyak membantu.

14.1 BEBERAPA KODE: MODEL 'DO_TEST'

Berikut ini adalah tampilan rinci pada bagian dari satu model di situs kami. Model ini bekerja dari tabel 'tests' dan tabel 'events'. Tujuannya adalah untuk mengontrol fungsi paling sentral dari situs kami, yaitu pengujian yang dilakukannya di situs jarak jauh. Model ini:

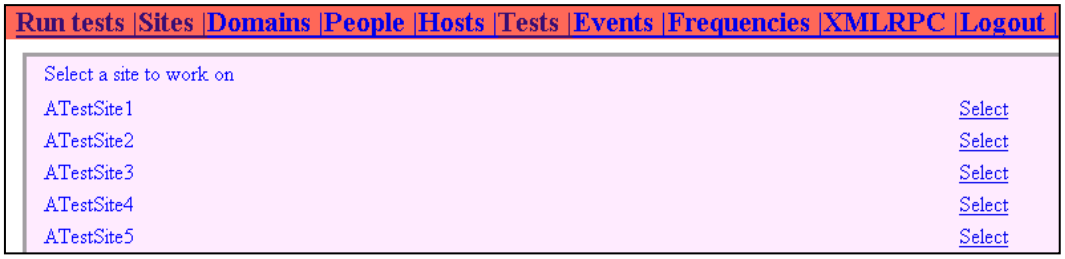
- Menghubungkan tabel database yang mencantumkan situs dan pengujian
- Memperbarui tabel lain, yang mencantumkan peristiwa (yaitu, setiap kali pengujian dilakukan)
- Membantu antarmuka dengan pengguna, memungkinkan dia untuk memilih pola pengujian atau untuk mendapatkan informasi dalam berbagai format

Karena ini adalah model, itu perlu dipanggil oleh pengontrol, dan mengembalikan hasilnya ke tampilan. Jika tampilan berisi hyperlink, ini pada gilirannya memanggil fungsi pengontrol, yang bertindak sebagai 'ujung depan' untuk fungsi model lain.

Inilah salah satu fungsi utama model do_test, untuk menghasilkan tabel informasi langsung dari kueri basis data. Idenya adalah untuk membuat daftar situs yang tersedia dan memilih satu untuk diuji. Jika Anda belum memilihnya, fungsi ini akan menghasilkan daftar situs dan meminta Anda untuk memilih. Kodenya terlihat seperti ini:

```
/*this function prepares a report on existing tests and allows you tochoose which to do.
  First, it selects a site and reports on that*/function report($site=0,$message='')
    {
/*have you chosen a site yet?*/
    $siteid = $this->uri->segment(3, 0);if(!$siteid >0)
                {
                $text = "<table class='table'>";
                $text .="<tr><td colspan = '2'>Select a site to workon</td></tr><tr>";
                $this->db->select('name, id');
                $query = $this->db->get('sites');if ($query->num_rows() > 0)
                          {
                          foreach ($query->result() as $row)
                                    {
                                        $text .= "<tr><td>";
                                        $text .= $row->name;
                                        $text .= "</td><td>";
/*note the next line uses the CI anchor function to generatehyperlinks*/
                                $text .= anchor("tests/report/$row->id","Select");
                                $text .= "</td></tr>";
                                          }
                                }
                $response['mytext'] = $text;
                $response['message']= $message;
                $this->display->mainpage($response);
                }
```

Hasilnya, ketika dipanggil melalui controller, terlihat seperti ini:



Gambar 14.1 hasil ketika dipanggil melalui controller

Perhatikan ada hyperlink untuk setiap opsi, sehingga Anda dapat memilih situs itu. Saya ingin menggunakan kelas 'Tabel' HTML untuk melakukan ini secara otomatis, tetapi saya tidak dapat melihat cara mudah untuk memasukkan hyperlink di baris hasil jika saya melakukannya. Tetapi:

- Kelas rekaman aktif CI membuat kueri basis data lebih mudah untuk ditulis
- Fungsi jangkar CI (bagian dari pembantu URL) memudahkan penulisan hyperlink

Dan, tentu saja, ketika kami datang untuk memindahkan situs kami dari server pengembangan saya ke web, kami tahu bahwa kedua fungsi CI ini akan memastikan bahwa informasi koneksi database, dan detail konfigurasi situs seperti URL, diubah secara otomatis.

CI membantu saya mengatur kode juga. Variabel $siteid diteruskan sebagai bagian dari segmen URI saat model dipanggil dari menu. Perhatikan bagaimana halaman web yang sebenarnya dibuat melalui fungsi halaman utama dari tampilan model. Yang harus dilakukan model do_test adalah mengambil informasi spesifik yang kita butuhkan dari database, dan mengirimkannya. Model tampilan memastikan bahwa itu diformat sesuai dengan file .css kami, dan diletakkan di halaman dengan bilah menu, dll.

Saya tidak cukup tepat di sini, tentu saja, karena pemformatan tabel dalam model dotest. Ada terlalu banyak kurung sudut di sana: idealnya, mereka harus berada dalam file tampilan. Namun, ini akan mengharuskan kami untuk merancang file 'tampilan' khusus untuk bagian data ini. Tampaknya lebih sederhana untuk melakukannya dengan cara ini. Setidaknya informasi desain disimpan di file .css.

Sekarang kita beralih ke bagian selanjutnya dari fungsi ini. Jika Anda telah memilih situs, tampilan sekarang memungkinkan kami melihat pengujian yang telah dilakukan untuk situs tersebut.

Kodenya terlihat seperti ini—loop else mengikuti:

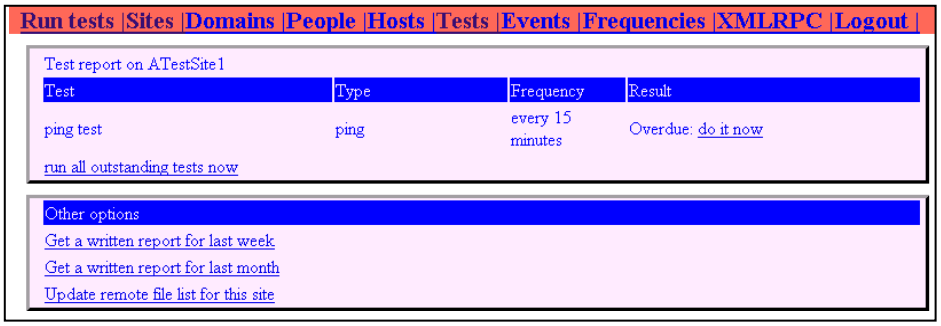
```
if(!$siteid > 0)
```

Dengan kata lain, inilah yang terjadi jika $siteid yang valid diteruskan.

```
        else
        {
 /*ok, you've chosen a site. let's go to town*/
     $this->db->select('sites.name AS sitename, sites.url AS siteurl,tests.name AS
 testname, tests.lastdone AS lastdone, tests.id AS testid, frequency, sites.id AS
 siteid, tests.type AS type');
     $this->db->join('sites', 'sites.id = tests.siteid');
     $this->db->orderby('frequency desc, sitename asc');
     $this->db->where('sites.id', $siteid);
     $query = $this->db->get('tests');if ($query-
     >num_rows() > 0)
     {
     $xrow                 = $query->row();
     $report = "<table class='table'><tr ><td colspan='4'>Test report on
     $xrow->sitename</td></tr>";
     $report .= "<tr class='header'><td width ='25%'>Test</td><tdwidth
 ='15%'>Type</td><td width ='10%'>Frequency</td><td width
 ='40%'>Result</td></tr>";
                  foreach ($query->result()  as $row)
                  {
                  $report .= "<tr><td width ='25%'>";
                  $report .= $row->testname;
                  $report .= "</td><td width ='15%'>";
                  $report .= $row->type;
```

```
                $report .= "</td><td width ='10%'>";
            //          $report .= $row->lastdone;
                         $this->db->select('name');
                $this->db->where('id', $row->frequency);
                $fquery = $this->db->get('frequencies');if ($fquery-
                >num_rows() > 0)
                        {
                         $frow = $fquery->row();
                        $sid = $frow->name;
                        }
                $report .= $sid;
                $report .= "</td><td width ='40%'>";
                $alf =   $this->deadline($row->testid);if($alf==FALSE)
                                {
                        $report .= "Overdue: ";
                        $report .= anchor("tests/runtest/$row->testid/human",'do it now');}
                else{$report .= "Last done: $alf";}
                $report .= "</td></tr>";
                }
        $report .= "<tr><td colspan='4'>";
        $report .= anchor("tests/runalltests/$row->siteid", 'run alloutstanding tests now');
        $report .= "</table>";
        }
        else
        $report = "no tests for this site yet.";}
        $report .="<table class='table'><tr class='header'><td>Otheroptions</td></tr>";
        $report .= "<tr><td>";
        $report .= anchor("tests/getwrittenreport/$row->siteid/604800", 'Get a written report for last
    week');
        $report .= anchor("tests/getwrittenreport/$row->siteid/2592000", 'Get a written report for
    last month');
        $report .= "</tr><tr><td>";
        $report .= anchor("tests/getbaseremotefiles/$row->siteid", 'Updateremote file list for this
    site');
        $report .= "</td></tr><table>";
        $response['mytext'] = $report;
        $this->display->mainpage($response);
          }
}
```

Hasil dari kode ini terlihat seperti ini:

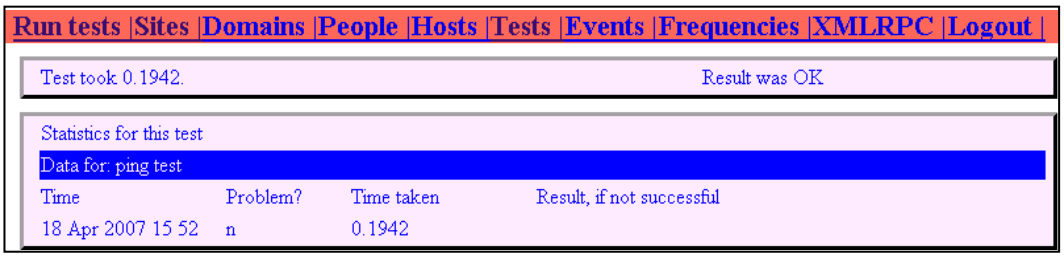


Gambar 14.2 hasil dari tipe data

Anda dapat melihat bahwa ia telah berkonsultasi dengan database untuk mengetahui tes mana yang harus diterapkan ke situs pilihan kami. Hanya ada satu tes, 'ping' sederhana yang seharusnya dilakukan setiap 15 menit.

Itu belum dilakukan dalam 15 menit terakhir, jadi itu ditampilkan sebagai terlambat dan hyperlink memungkinkan saya untuk melakukannya sekarang jika saya mau. Jika saya mengklik hyperlink, saya memanggil fungsi runtest pengontrol tes, menyediakannya dengan nomor ID tes yang saya inginkan.

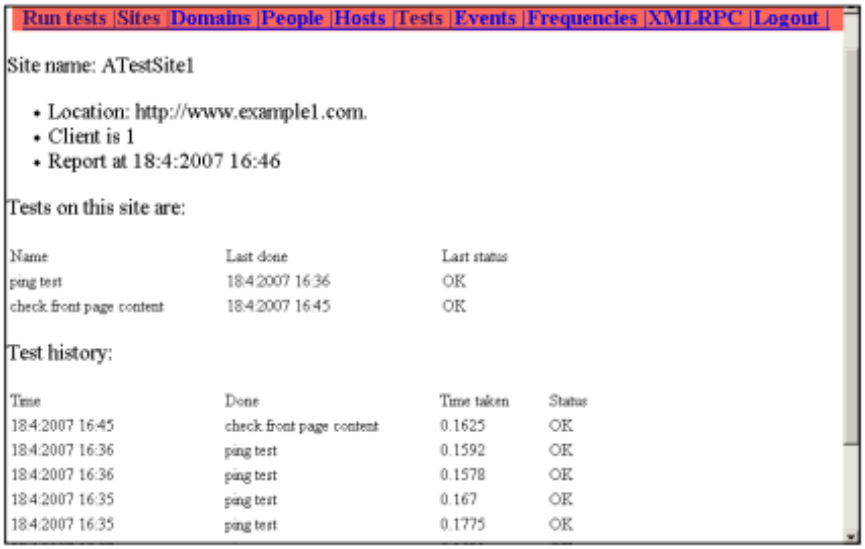
Hasilnya terlihat seperti ini:



Gambar 14.3 tampilan test

Sistem memberi tahu saya ketika tes telah dilakukan, bahwa tidak ada masalah dengan itu, dan berapa lama waktu yang dibutuhkan.

Jika sekarang saya kembali ke fungsi laporan, Anda akan ingat bahwa ada opsi untuk menghasilkan laporan historis pada pengujian yang dilakukan di situs yang saya pilih selama seminggu terakhir atau sebulan terakhir. Saya hanya mengatur situs ini sebagai contoh, sehingga tidak memiliki riwayat pengujian yang sebenarnya: tetapi ini adalah tampilannya jika saya melakukan serangkaian pengujian identik secara berurutan. Saya juga telah menambahkan tes lain, dan laporannya juga menyertakannya.



Gambar 14.4 pengujian laporan dan test

Seperti yang Anda lihat, kami membuat laporan yang dapat kami tunjukkan kepada klien, untuk menunjukkan bahwa situs itu hidup dan merespons saat diuji dengan cara yang berbeda pada waktu yang berbeda. Waktu yang dibutuhkan untuk setiap tes untuk merespons tidak terlalu menarik pada satu waktu, tetapi dapat membantu kita untuk melihat pola dalam jangka waktu yang lebih lama.

Fungsi yang benar-benar menjalankan tes dibangun di sekitar pernyataan switch. Dibutuhkan dua parameter:

- Nomor ID tes, dari mana kita mencari informasi dasar yang kita perlukan untuk menjalankannya: URL yang akan digunakan, pasangan nama-nilai, dan teks apa pun yang kita harapkan akan ditemukan jika tes berhasil.
- Jenis pengguna. (Jika pengguna adalah 'manusia', program mengembalikan lebih banyak informasi dalam format yang lebih ramah pengguna—dengan kata lain, jika Anda ingin menampilkan hasilnya di layar, panggil dengan parameter pengguna yang disetel ke manusia. Jika Anda ingin untuk menangani hasil secara terprogram, setel parameter ini ke yang lain.)

Dalam kutipan kode ini, kami telah mendefinisikan beberapa jenis pengujian. Dua contoh adalah:

- tes 'ping' yang hanya memanggil URL. Jika mereka mendapatkan hasil, mereka memeriksanya baik untuk frasa yang diharapkan (disebut 'regex' dalam database) atau untuk istilah HTML umum jika tidak ada kumpulan frasa 'regex'.
- pengujian 'ete' yang menggunakan beberapa kode yang kami kembangkan untuk membuat pengujian 'end-to-end' dari halaman yang dilindungi, masuk dan mencari frase yang diharapkan. Kode ini tidak dijelaskan dalam buku ini karena tidak termasuk dalam fungsi CI.

Setiap tes mengembalikan variabel $result dan $timetaken; ini dimasukkan ke dalam tabel 'peristiwa' database sebagai catatan pengujian, bersama dengan informasi lain dari database. Inilah kodenya, yang sangat bergantung pada CI

Model Active Record untuk membaca dan menulis ke database, dan pada kelas benchmark untuk mendapatkan waktu yang diambil oleh setiap tes.

```
/*function to run an individual test*/
   function runtest($testid, $user='human')
   {
/*first, look up the test details */
   $this->db->where('id', $testid);
   $query = $this->db->get('tests');
   if ($query->num_rows() > 0)
          {
   foreach ($query->result() as $row)
                {
                 $type = $row->type;
/*then work out which type it is and forward it accordingly*/
```

```
                switch ($type){
                case 'ping':
                $this->benchmark->mark('code_start');
                $result =$this->pingtest($testid);
                $this->benchmark->mark('code_end');
                $timetaken = $this->benchmark->elapsed_time('code_
start', 'code_end');
              break;

                case 'ete' :
                $this->benchmark->mark('code_start');
                $result = $this->httppost($testid);
                $this->benchmark->mark('code_end');
                $timetaken = $this->benchmark->elapsed_time('code_
start', 'code_end');
                break;

                default:
                $result = 'noid';
                }
/*work out which site the test belongs to*/
                $this->db->select('tests.siteid AS id');
                $this->db->where('id', $testid);
                $query = $this->db->get('tests');
                if ($query->num_rows() > 0)
                      {$srow = $query->row();
                      $mysiteid = $srow->id;
                      }
                else{$mysiteid = 0;}
/*build the rest of the result set and enter it into the database*/
                $time = now();
                if($result == 'OK')
                      {$isalert = 'n';}
                else{$isalert = 'y';}
                $this->db->set('name', $type);
                $this->db->set('type', 'test');
                $this->db->set('timetaken', $timetaken);
                if($result != '')
                {$this->db->set('result', $result);}
                $this->db->set('testid', $testid);
                $this->db->set('userid', 0);
                $this->db->set('siteid', $mysiteid);
                $this->db->set('time', $time);
```

```
                $this->db->set('isalert', $isalert);
                $this->db->insert('events');

              $mydata = array(
              'lastdone'      => $time,
              'notes'         => $result,
              'isalert'       => $isalert
          );
                $this->db->where('id', $testid);
                $this->db->update('tests', $mydata);

/*only return this info to screen if user is human. Otherwise, no need
to do anything more; you've updated the database.*/
                if($user == 'human')
                      {$mytext = "<table class='table'><tr>";
                      $mytext .= "<td>Test took $timetaken.</td>";
                      $mytext .= "<td>Result was ".$result.
                                                  '</td</tr></table>';
                      $mytext .= $this->testhistory($testid);
                      $response['mytext'] = $mytext;
                      $this->display->mainpage($response);}
                else{return $response;}
                       }
                }
        }
```

Bagaimana kami benar-benar mencetak laporan pengujian adalah contoh menarik dari tindakan CI. Seperti biasa, ada berbagai pilihan. Anda dapat mencetak laporan Anda dengan cara yang sulit, menggunakan HTML dalam kode Anda seperti yang kami lakukan sebelumnya, seperti ini:

```
/*do database query here!*/
/*now format the results the hard way……*/

   $report .= "<p>Test history:</p>";
   $report .= "<table width='100%'><tr><td width ='20%'>Time</
td><td width ='20%'>Name</td><td width ='10%'>Time taken</td><td
width='45%'>Status</td></tr>";if ($query->num_rows() > 0)
         {
         foreach ($query->result() as $row)
               {
                $report .= "<tr><td width='20%'>";
                $report .= gmDate("j:n:Y H:i", $row->time);
                $report .= "</td><td width='20%'>";
                $report .= $row->name;
                $report .= "</td><td width=10%>";
                $report .= $row->timetaken;
                $report .= "</td><td width = 45%>";
                if($row->isalert == 'n')
                      {$report .= "OK";}
                else
                      {$report .= "problem: $row->result";}

                $report .= "</td></tr>";
               }
          $report .= "</table>";
```

Di sisi lain, Anda dapat menggunakan pustaka Tabel HTML untuk mempermudah hidup:

```
/*do db query here */
/*format the results using the CI HTML table library*/
   $report .= "Test history:";
/*redefine our CI table layout if we want to, using our css file*/
   $tmpl = array ('table_open'       => '<table border="1",
class="table">',);
   $this->table->set_template($tmpl);
   if ($query->num_rows() > 0)
          {
          $this->table->set_heading('Time of test', 'Name ,'Time
taken','Result');
          $report .= $this->table->generate($query3);
          }
          $this->table->clear();
```

Jelas, lebih pendek dan lebih sederhana. Hasilnya tampak persis sama, hanya bergantung pada format HTML. Namun, mari kita lihat dua masalah yang diangkat oleh kode ini.

Ketika Anda melihat lebih dekat, ada kompromi yang datang dengan kenyamanan. Pada versi pertama, yang lebih panjang, dimungkinkan untuk memformat tanggal:

```
$report .= gmDate("j:n:Y H:i", $row->time);
```

Ini memberi saya tanggal 'manusia' daripada tanggal Unix yang sebenarnya saya simpan di database saya. Dengan kata lain, tabel mengatakan '24 Apr 2007 09 04' daripada '1177405479'.

Namun, tidak mudah untuk melakukan ini dengan fungsi Tabel HTML CI. (Anda mungkin dapat melakukan ini dalam kueri basis data itu sendiri di beberapa sistem basis data; tetapi fungsi tanggal MySQL hanya beroperasi pada tanggal dan waktu yang disimpan dalam format tanggal unik MySQL, dan kami memilih untuk menggunakan format Unix sebagai gantinya.) Kami memiliki hal yang sama masalah sebelumnya ketika kita ingin menghasilkan hyperlink. Anda tidak bisa melakukannya dengan cara ini.

Ada pertanyaan besar lainnya tentang ini. Di mana format harus pergi? Semua kode yang ditulis sebelumnya akan muncul di controller atau model. (Seperti yang saya tulis, itu ada dalam model, yang dipanggil oleh pengontrol ketika pengguna manusia mengklik hyperlink untuk menghasilkan laporan. Alasan untuk memasukkannya ke dalam model adalah agar saya dapat memanggil kode dari beberapa pengontrol jika saya perlu.)

Puritan MVC akan mengatakan bahwa Anda tidak boleh memasukkan pemformatan ke dalam model. Itu harus dalam tampilan. Dan, memang, mereka ada benarnya. Saya mungkin ingin menulis ulang kode untuk menghasilkan laporan teks, dan menggunakan pembantu unduhan CI untuk memaksa situs mengunduhnya dalam format teks daripada menampilkannya di layar. (Lihat Bab 11 untuk lebih lanjut tentang download helper.) Karena itu, saya harus menulis ulang semua kode untuk menghasilkan sesuatu yang diformat dengan jeda teks daripada pasangan <tr></tr>—atau, lebih buruk lagi, untuk menghasilkan format dalam .rtf, atau format teks kaya yang serupa.

Pilihan lainnya adalah menggunakan sintaks PHP alternatif atau kelas Template Parser dan meletakkan variabel, atau placeholder, sebenarnya dalam tampilan itu sendiri. (Lihat Bab 5 untuk liputan singkat tentang opsi ini.) Kemudian, semua yang Anda berikan ke tampilan adalah data aktual, tidak diformat. Meskipun ini dapat memuaskan para puritan MVC, saya merasa ini menambahkan lapisan kompleksitas tambahan pada kode. Tapi ini adalah pandangan pribadi, dan banyak yang tidak setuju.

Poin utamanya adalah CI menawarkan berbagai opsi: terserah Anda untuk memilih yang paling Anda sukai. Tidak ada cara yang 'benar' atau 'salah': hanya ada beberapa cara yang bekerja lebih baik daripada yang lain, dan beberapa cara yang lebih sesuai dengan gaya pribadi Anda daripada yang lain.

14.2 NERACA

Mari kita lihat kembali apa yang telah kita bahas dalam buku ini. Apakah CI membantu?

Dimana CI Helped: Struktur

Bahkan dari melihat bagian dari hanya satu model di situs kami, jelas bahwa setiap aplikasi yang bermanfaat dengan cepat menjadi kompleks. CI, dengan menyarankan dan sampai batas tertentu menegakkan struktur MVC, membantu mengatur beberapa kompleksitas itu. Masih mungkin untuk melupakan di mana Anda meletakkan sepotong kode, atau bahkan menulis fungsi serupa dua kali di pengontrol atau model yang berbeda: tetapi kemungkinan besar kode Anda akan berada dalam potongan logis.

Mekanisme URL CI membantu Anda dengan cepat menautkan dari satu file kode ke file kode lainnya. Struktur 'objek super' CI memastikan bahwa potongan kode dapat saling memanggil dan menyampaikan informasi, tanpa konflik namespace. Kami cenderung tidak memiliki kebingungan antara variabel dengan nama yang sama, karena masing-masing terbatas pada areanya sendiri. Pada saat yang sama, Anda dapat dengan mudah mengakses semua sumber daya CI di situs Anda, dari mana pun Anda berada dalam kode Anda. File 'config' CI mendorong Anda untuk membangun kumpulan informasi terpusat tentang situs Anda. Semua manfaat ini berarti situs Anda akan lebih mudah dikembangkan, lebih mudah dirawat, dan lebih mudah dipahami oleh programmer lain.

Dimana CI Helped: Kesederhanaan

Ada banyak tempat di mana CI membantu menyederhanakan kode Anda. Mungkin contoh terbaik datang dari kelas Rekaman Aktif, tetapi ada banyak contoh lainnya dalam buku ini. CI sangat baik dalam mengambil potongan kode yang kompleks, menyembunyikannya dalam fungsi beberapa kelas perpustakaan, dan memungkinkan Anda untuk menggunakannya dengan panggilan fungsi sederhana.

Di mana CI Helped: Fungsionalitas Ekstra

Banyak fungsi CI membawa manfaat tambahan saat Anda menggunakannya. Fungsi seperti kelas URL dan Rekaman Aktif secara otomatis merujuk ke pengaturan 'konfigurasi' Anda, sehingga Anda tidak perlu terus mengulang informasi, dan agar setiap perubahan yang Anda buat di satu tempat secara otomatis diterapkan di seluruh situs.

Ada banyak contoh kecil—cara mudah Anda dapat menyembunyikan alamat email Anda dari robot, misalnya (lihat Bab 3), atau kelas Rekaman Aktif juga menyiapkan data Anda.

Faktanya, salah satu kurva pembelajaran utama (setidaknya bagi saya) dengan CI adalah mencari tahu pintasan apa yang tersedia, dan mengingat untuk menggunakannya alih-alih menulis semuanya dengan cara PHP tangan panjang yang melelahkan seperti yang biasa saya lakukan. Jika buku ini menarik perhatian Anda pada beberapa jalan pintas ini, buku ini akan memiliki tujuan yang berguna untuk itu saja.

14.3 MASALAH DENGAN CI

CI tidak sempurna. Ini benar-benar berarti bahwa ini adalah tindakan penyeimbang: ringan dan mudah digunakan melawan kerumitan dan kelengkapan. Seperti yang pernah dikatakan seseorang, 'ringan' berarti, "mencakup semua yang saya inginkan dan tidak ada yang tidak saya inginkan".

Kelengkapan

CI berisi fungsi untuk hampir setiap aplikasi biasa yang dapat Anda pikirkan. Ada beberapa pengecualian untuk mata pelajaran yang dicakup oleh kelas CI utama, dan banyak di antaranya dicakup oleh perpustakaan tambahan yang disumbangkan oleh pengguna CI (lihat bab berikutnya) atau tersedia melalui repositori PEAR.

Kelalaian yang paling jelas meliputi:

- Kelas AJAX
- Kelas untuk menulis robot web
- Kelas untuk membangun situs yang aman, menangani login, halaman yang dilindungi, dan sejenisnya, serta pemeliharaan sesi dasar
- Kelas 'scaffolding' yang ditingkatkan, menghasilkan sesuatu yang dapat digunakan di situs publik daripada murni untuk pengembangan

CI mungkin juga mengambil daun dari buku Rails, dan membangun kelas pembuat kode — yang akan memungkinkan pengembang untuk membangun kelas yang dapat disesuaikan sendiri.

Kemudahan penggunaan

CI menuntut beberapa upaya untuk belajar, seperti yang Anda harapkan. Tapi, dengan asumsi bahwa Anda sudah mengetahui beberapa PHP, itu cukup mudah. Faktanya, 'kurva pembelajaran' utama yang saya temukan adalah, ketika Anda mengetahui cara standar PHP dalam melakukan sesuatu, Anda cenderung melakukannya dengan cara itu. Baru kemudian, ketika Anda mencari di Panduan Pengguna CI untuk sesuatu yang lain, apakah Anda menyadari bahwa Anda bisa melakukan hal pertama jauh lebih cepat dan sederhana dengan kelas atau pembantu CI.

Berbicara secara pribadi, saya menemukan dua kelas CI cukup sulit untuk dipahami dan diikuti. Ini adalah kelas XMLRPC, dan kelas Validasi. Ini sebagian besar karena keduanya memerlukan antarmuka antara halaman yang berbeda, atau situs yang berbeda, untuk bekerja, dan terkadang sulit untuk menyiapkannya dengan benar. (Lihat Bab 9 dan 8, masing-masing)

Saya juga merasa sulit pada awalnya untuk mengikuti cara CI menggunakan 'super-objeknya'—lihat Bab 7. Melakukannya dengan benar mungkin merupakan kurva belajar CI yang paling curam, dan kesalahan dapat menyebabkan beberapa hasil yang aneh dan membuat frustrasi sampai Anda memahaminya.

Selebihnya: mudah. Jika ragu, Anda selalu dapat menjelajahi kode sumbernya.

14.4 RINGKASAN

Dalam bab ini, kita telah melihat beberapa contoh pengkodean, menyatukan banyak fungsi , yang telah kita bahas sedikit demi sedikit di bab sebelumnya.

Kami juga telah melihat cara CI membantu:

- Untuk mengatur situs Anda
- Untuk membuat pengkodean lebih sederhana
- Untuk menambahkan fungsionalitas

Saya harap buku ini telah meyakinkan Anda bahwa, jika Anda ingin membuat kode situs web dinamis di PHP, CI adalah cara yang sangat cerdas untuk melakukannya.

Ini juga merupakan penghargaan untuk gerakan open-source, dan kepada orang-orang murah hati yang mendukungnya, bahwa koleksi kode yang begitu kaya tersedia, secara bebas dan universal.

Mengambil tema kemurahan hati ini lebih jauh, bab terakhir dalam buku ini membahas beberapa cara lain untuk mengembangkan pengkodean CI Anda—komunitas pengguna CI yang dapat (dan biasanya akan) memberikan bantuan, dukungan, dan sumber daya kode tambahan.

BAB 15
SUMBER DAYA DAN EKSTENSI

Yah, kami telah melihat CI dengan cukup teliti, dan saya harap Anda terkesan. Kami juga telah mengembangkan beberapa kode kami sendiri dalam prosesnya. Saya yakin ketika Anda melihat beberapa kode saya, Anda mulai berpikir "Saya bisa menulisnya dengan lebih baik...". Setiap orang memiliki gayanya sendiri, dan CI memberi Anda banyak kebebasan.

Komunitas CI penuh dengan orang-orang yang menulis kode yang bagus, dan untungnya banyak dari mereka yang bersedia menyediakannya secara gratis untuk kita semua. Jadi ada banyak kode di luar sana yang dapat menghemat banyak pekerjaan Anda. Jika, untuk mengambil satu contoh, Anda ingin membuat grafik dinamis dari data yang diambil dari database Anda, Anda dapat duduk dan menulis kode sendiri, tetapi sebenarnya, setidaknya tiga orang telah menangani masalah ini, dan semuanya telah membuat kode mereka tersedia untuk Anda.

Bab terakhir ini membahas beberapa sumber daya yang dapat Anda gunakan, untuk membuat pengkodean Anda lebih cepat dan lebih mudah. CI memiliki komunitas pengguna yang berkembang dan aktif dan sumber daya yang tersedia berubah setiap saat, jadi saya belum mencoba membuat daftar lengkap, hanya memberi Anda gambaran tentang apa yang ada dan di mana mencarinya.

Ada catatan peringatan juga. Ada begitu banyak kode di luar sana yang dapat dengan mudah membingungkan. Orang-orang menulis proyek kesayangan mereka, beberapa brilian, beberapa cukup bagus. Banyak dari kita lebih baik dalam menulis kode daripada menulis penjelasan atau komentar. Akibatnya, cukup sulit untuk mengetahui apa yang dilakukan setiap perpustakaan atau plug-in, dan apakah itu yang terbaik untuk Anda.

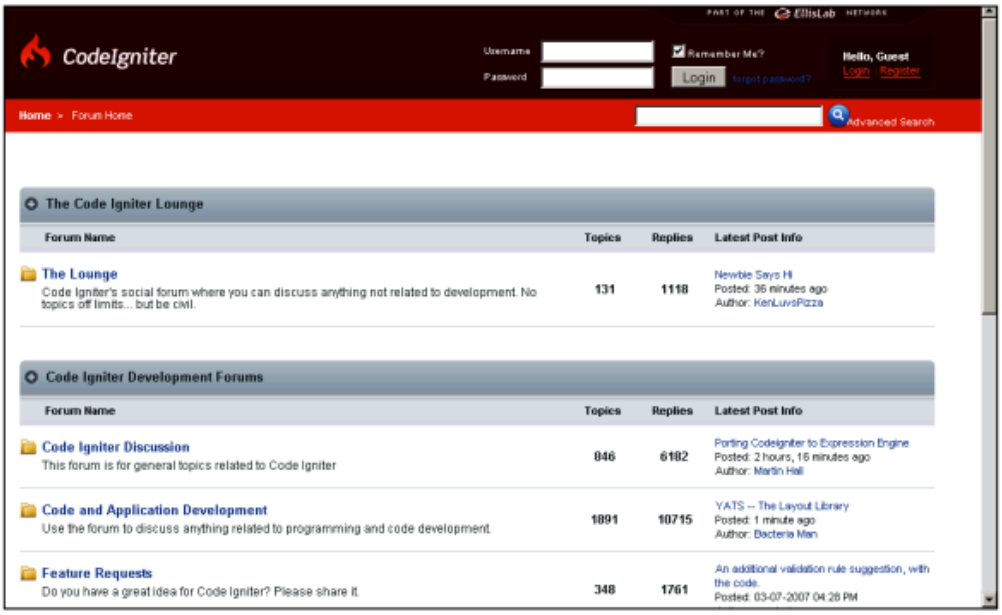
Jadi, mari kita habiskan bab terakhir buku ini dengan melihat bantuan yang tersedia.

- Pertama, mari kita lihat sumber kodenya.
- Kemudian, mari kita lihat beberapa subjek dan bandingkan kode yang tersedia.
- Terakhir, mari kita lihat sumber bantuan yang lebih umum: di PHP, MySQL, dan Apache.

15.1 CI'S USER FORUMS

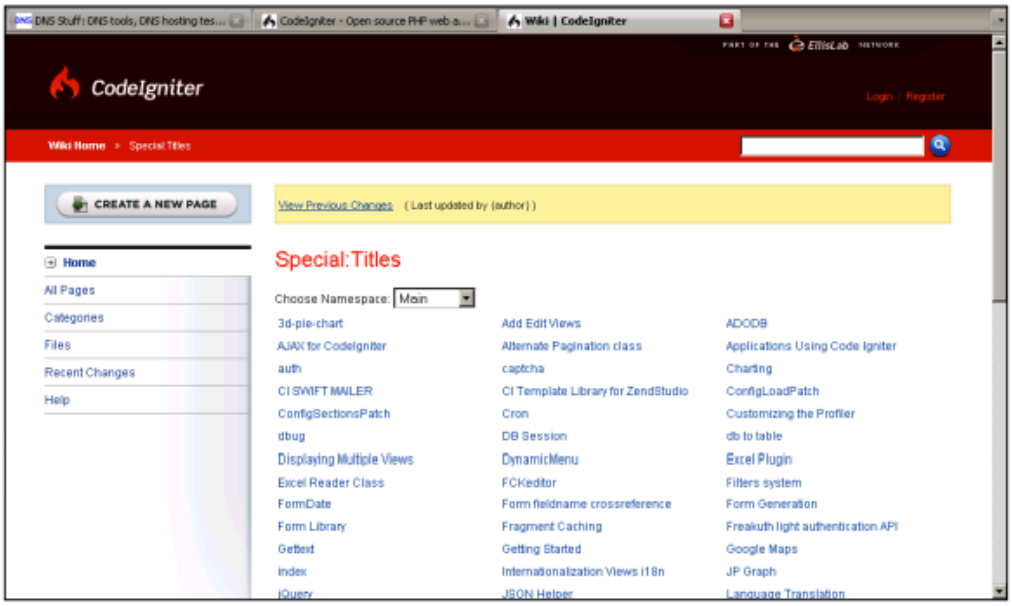
CI memiliki dua sumber daya utama:

- Forum pengguna, di http://www.codeigniter.com/forums/ menawarkan diskusi berkelanjutan yang hidup dan cukup baik tentang sebagian besar masalah CI. Komentar dan saran yang dibuat tidak selalu membantu (atau akurat), tetapi ada sejumlah 'anggota senior' yang biasanya sangat masuk akal. Ini juga merupakan forum yang baik; orang mengajukan pertanyaan 'baru' yang sangat jelas, tetapi mendapatkan jawaban yang sabar dan membantu. Kadang-kadang, Rick Ellis sendiri ikut campur, tetapi dia tidak mencoba menangani setiap masalah sendiri.



Gambar 15.1 tampilan Forum Web CI

- Wiki, di http://www.codeigniter.com/wiki/ .Ini dimaksudkan sebagai "repositori untuk tips, trik, peretasan, pengaya, dan peningkatan". Ini berisi banyak kode yang berguna, meskipun cakupannya tidak sistematis.



Gambar 15.2 Tampilan Forum repository CI

Menggunakan forum atau wiki itu mudah: Anda cukup membuat keanggotaan untuk diri Anda sendiri (gratis) lalu masuk dan lakukan pekerjaan Anda. Jika Anda serius menggunakan CI, ada baiknya mengatur pembaca RSS Anda untuk berlangganan umpan 'perubahan terbaru' di Wiki.
Ingat, bagaimanapun, bahwa:

- Tidak semua penulis plug-in memiliki kompetensi teknis seperti Rick Ellis. Produk mereka mungkin memiliki bug atau masalah.
- Beberapa plugin lama yang ditulis sebelum CI versi 1.5 keluar mungkin perlu diubah, karena cara perpustakaan diinisialisasi telah diubah (lihat Bab 12). Ini seharusnya tidak terlalu sulit untuk dilakukan, tetapi itu berarti file perpustakaan ini tidak akan langsung berfungsi.

15.2 VIDEO TUTORIAL

Jika Anda ingin benar-benar dibicarakan melalui aplikasi CI pertama Anda, ada tiga tutorial video luar biasa di situs CI.

- Pengantar CI.
- Buat blog dalam 20 menit. Derek Jones membuat halaman blog dasar, menunjukkan kepada Anda cara mengatur situs, membuat kueri basis data, dan menyajikan hasilnya dalam tampilan.
- Tautan ke video eksternal oleh Derek Allard (lihat http://video.derekallard.com/), yang menjelaskan, antara lain, bagaimana menggunakan library Scriptaculous untuk mengintegrasikan efek AJAX dan JavaScript. Menggunakan tampilan di bawah, ini menunjukkan kepada Anda cara membuat dropdown entri teks pelengkapan otomatis, menggunakan Ajax untuk memperbaruinya.

Gambar 15.3 Tautan ke video eksternal oleh Derek Allard

Plug-in dan Pustaka yang Tersedia

Maksud dan harapan Rick Ellis adalah bahwa pengguna CI akan menyumbangkan 'plug-in' atau perpustakaan untuk membantu pengguna CI lainnya. Framework ini baru tersedia sekitar satu tahun, tetapi sudah ada banyak kode menarik yang tersedia.

Jumlah plug-in dan perpustakaan terus bertambah, dan yang sudah ada sedang diubah. Jadi, bagian selanjutnya bukanlah penjelasan sistematis tentang apa yang ada di sana: hanya beberapa catatan tentang beberapa hal yang mungkin berguna bagi Anda. Saya minta maaf karena saya harus melewatkan banyak hal bagus: silakan lihat sendiri wiki.

AJAX/JavaScript

Wiki berisi dua paket AJAX: satu menggunakan XAJAX, dan yang lainnya menggunakan library prototype.js�scriptaculous.js.

Tabel 15.1 deskripsi AJAX dan XAJAX

Nama	**Ajax for CI 1.5.1**
URL	http://www.codeigniter.com/wiki/AJAX_for_CodeIgniter/
Menggunakan perpustakaan prototype.js dan scriptaculous.js	
Unduhan termasuk file .js serta .php dan Panduan Pengguna lengkap. (Ini tidak mudah dimengerti jika Anda belum memiliki pemahaman yang baik tentang AJAX dan DOM, dan itu mungkin memiliki beberapa contoh yang lebih panjang.) Mudah untuk menginstal: letakkan file .php di folder application/libraries Anda dan file .js di direktori root Anda. Baru dirilis, jadi sangat sedikit komentar di Forum CI.	
Pengarang	siric
Nama	**XAJAX**
URL	http://www.codeigniter.com/wiki/XAJAX/
CI 'front end' untuk perpustakaan XAJAX. Termasuk file 'include' JavaScript-nya sendiri, xajax.js	
Pengarang	Greg McLellan—based on the xajax php library (see http://www.xajaxproject.org/)

Otentikasi

Pengguna Wiki juga bergulat dengan keamanan: ketiga paket ini melihat otentikasi pengguna Anda dan menghindari kemungkinan jebakan menyimpan data sesi dalam cookie.

Tabel 15.2 3 otentikasi pengguna CI

Nama	**FreakAuth_light**
URL	http://www.4webby.com/freakauth/
Ini termasuk perpustakaan untuk dilakukan • pengguna masuk�keluar • pendaftaran pengguna • ingat kata Sandi • ganti kata sandi • penguncian area yang dipesan situs web aplikasi administrasi backend ke: • mengelola pengguna	

• mengelola administrator Ini memungkinkan Anda untuk mengatur empat tingkat akses (dari superadmin hingga tamu) dan kemudian mengatur metode 'periksa' di pengontrol. Ini dapat diatur baik dalam konstruktor pengontrol atau dalam fungsi individual. Jika pengguna memanggil pengontrol (atau fungsi individu), kode akan memeriksa apakah dia login, konsultasi	
Ada diskusi ekstensif tentang kode ini yang terjadi di forum CI pada saat penulisan. Beberapa kesalahan telah diidentifikasi, tetapi kode tersebut sekarang dalam rilis ketiga dan sepertinya masalah sedang diselesaikan.	
Pengarang	danfreak
Nama	**Auth**
URL	http://www.codeigniter.com/wiki/auth
Paket ini menawarkan fungsionalitas login�logout, registrasi, dengan aktivasi, dan bahkan reset password yang terlupa. Pengaturannya cukup rumit: Anda harus menyiapkan tabel database, menyertakan beberapa pustaka inti dan pembantu baru, dan juga melakukan beberapa konfigurasi.	
Bekerja dengan CI1.5.	
Pengarang	Anonymous
Nama	**DB Session**
URL	http://www.codeigniter.com/wiki/DB_Session/ http://dready.jexiste.fr/dotclear/index. php?2006/09/13/19-reworked-session-handler-for-code- igniter
Mengubah kelas sesi CI (lihat Bab 6) yang menyimpan data sesi dalam cookie. (Yang dapat dienkripsi, tentu saja.) Kelas ini hanya menyimpan pengidentifikasi sesi: Anda menambahkan tabel tambahan ke database Anda, dan itu mencari semua informasi sesi lainnya di sana.	
Bekerja dengan CI1.5.	
Pengarang	dready

15.3 SITUS EKSTERNAL

Ada beberapa 'power user' dari CI yang menyumbangkan kode mereka sendiri. Salah satu contoh yang baik adalah Glossopteris, sebuah situs yang dijalankan oleh perusahaan desain web AS. Ini menyediakan beberapa perpustakaan mereka sendiri, misalnya (di http://www.glossopteris.com/journal/post/table-relationships-in-ci) perpustakaan CRUD lain, yang mereka klaim "akan memungkinkan untuk tabel yang kompleks inter- hubungan yang akan ditugaskan dan tindakan CRUD sederhana yang harus diselesaikan." Ini mengikuti preseden Rails dengan cukup dekat: Anda dapat mendefinisikan hubungan antara tabel seperti tautan 'memiliki satu' dan 'memiliki banyak'. Kode tersedia, tetapi dapat dilakukan dengan lebih banyak komentar atau panduan pengguna.

Pengembangan lainnya adalah CI_Forge (http://www.ciforge.com/), yang dimaksudkan sebagai, "Tempat untuk proyek yang dirancang untuk meningkatkan atau memperluas kerangka kerja PHP ringan CodeIgniter." Ini menyediakan hosting Subversion dan Trac, wiki, pelacak masalah bug, dan dukungan perubahan log. Ini adalah aplikasi baru, tetapi (per Juli 2007) sudah menampung 20 proyek.

Perbandingan: Pustaka Charting Mana yang Digunakan?

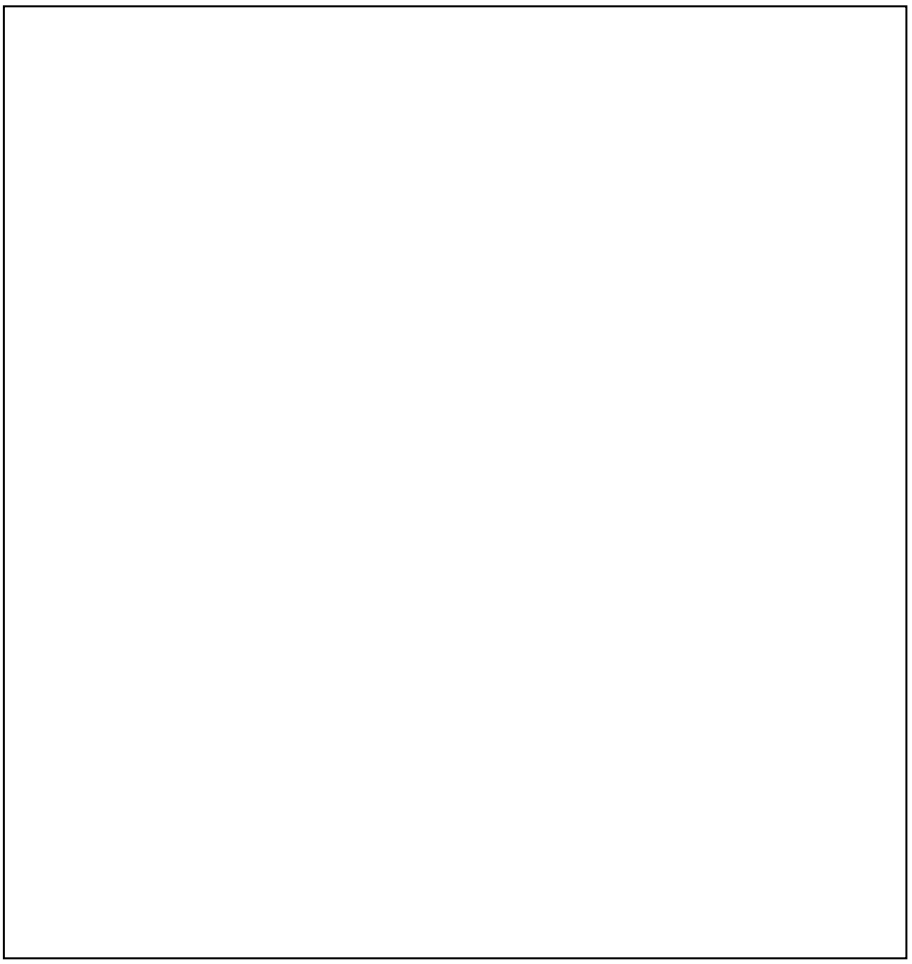
Itu cukup banyak pilihan. Terkadang, ada terlalu banyak pilihan.

Untuk mendemonstrasikannya, mari kita lihat tiga opsi untuk melakukan hal yang sama, dan lihat perbedaannya. Membuat grafik data yang dinamis bukanlah hal yang mudah untuk dikodekan sendiri. Tapi itu membuat situs Anda terlihat bagus.

Mari kita lihat tiga add-on yang tersedia untuk CI yang melakukan hal ini, dan coba bandingkan kekuatan dan kelemahannya, serta lihat hasil yang mereka hasilkan.

Tabel 15.3 Add-on yang tersedia untuk CI

Nama	**3d-pie-chart**
URL	http://codeigniter.com/wiki/3d-pie-chart/
Menghasilkan diagram lingkaran dari dua larik data (label dan nilai) dan menyimpannya di situs Anda. Tampak hebat, tapi hanya ini yang dilakukannya.	
Mudah diatur: letakkan file piechart.zip di folder aplikasi/perpustakaan Anda, dan tulis pengontrol berdasarkan contoh. Memerlukan font, dan Anda perlu mengubah tampilan untuk menampilkan hasilnya. Bekerja dengan CI versi 1.5	
Pengarang	Craig
Nama	**Panaci**
URL	http://bleakview.orgfree.com/ or http://codeigniter.com/wiki/Charting/

Secara dinamis menghasilkan bagan dan grafik, termasuk bagan batang, garis, area, langkah, dan impuls (tetapi bukan bagan pai). Entri wiki menyatakan: "Harap dicatat, ini BUKAN perpustakaan kelas komersial seperti jpgraph atau chartdirector, tetapi cukup memadai untuk plot dasar". Contoh kode, dan plot spesimen, di bawah ini, menunjukkan seperti apa dan bagaimana menggunakannya. 	
Bekerja dengan CI versi 1.5. Seperti 3d-pie-chart, Anda menyalin file ke folder application/libraries Anda, dan memanggilnya dari controller Anda, menyediakan parameter dasar dan array data. Diskusi singkat di Forum CI, tidak ada bug utama yang ditemukan pada saat penulisan.	
Pengarang	Oscar Bajner
Nama	**JP Graph**
URL	http://codeigniter.com/wiki/JP_Graph/

Ini tidak sepenuhnya plug-in: ini adalah kode yang memungkinkan Anda untuk antarmuka antara CI dan perpustakaan JP Graph eksternal. Anda perlu mengunduh pustaka JP Graph, membuat serangkaian plug-in untuk setiap jenis grafik yang ingin Anda gunakan, dan kemudian memanggil plug-in dari pengontrol saat Anda membutuhkannya. Karena contoh-contoh ini membentuk situs webnya, http://www.aditus.nu/jpgraph/features. phpshow, JP Graph menawarkan rentang grafik yang jauh lebih luas, dan terlihat bagus.
Ada dua kelemahan dengan JP Graph. Seperti yang dikatakan entri wiki: "Perlu diingat bahwa JpGraph memiliki basis kode yang sangat besar, jadi pastikan untuk menyertakan hanya pustaka khusus yang Anda perlukan untuk setiap bagan." Kedua, JP Graph gratis untuk penggunaan pribadi, tetapi tidak untuk penggunaan komersial.
Pengarang | Aditus Consulting

Tiga opsi: dua yang pertama relatif sederhana, yang kedua lebih kompleks. Tergantung pada apa yang Anda butuhkan (dan jika Anda siap untuk membayar).

CRUD: Perbatasan Terakhir

Anda perlu menulis halaman CRUD di hampir setiap aplikasi. Tampaknya sederhana, dan logis, untuk mengotomatisasi proses pembuatan halaman-halaman itu! Mereka standar yang menggiurkan—namun mereka memiliki sejumlah besar kemungkinan variasi yang menipu. Tidak mungkin untuk menulis satu tanpa mulai memaksakan aturan dan asumsi Anda

sendiri pada pengguna. Juga, selalu ada trade-off antara mencakup lebih banyak opsi yang memungkinkan di satu sisi, dan kesederhanaan penggunaan di sisi lain. Semakin banyak pengecualian dan kemungkinan yang Anda coba cakup, semakin kompleks kode Anda dan semakin besar unduhannya. Jadi, beberapa orang telah mencoba menyederhanakan operasi CRUD dasar.

Kami mencoba mengembangkan aplikasi CRUD kami sendiri di Bab 13. Ini adalah model yang cukup sederhana yang memotong banyak sudut dan hanya memungkinkan Anda untuk menggunakan subset dari objek formulir HTML yang tersedia; tetapi berhasil memasukkan fungsi validasi CI. Kami telah menyebutkan, dalam bab ini, perpustakaan Glossopteris.

Pendekatan menarik lainnya adalah 'CodeCrafter', yang terdaftar di CI wiki dan tersedia dari Datacraft Software Consulting di Afrika Selatan, di: http://www.datacraft.co.za/index.php?contents=codecrafter/codecraft.

Ini mengklaim bahwa, "CodeCrafter akan membantu Anda menghasilkan seluruh aplikasi CodeIgniter Anda hanya dalam hitungan detik." Muncul dengan manual online 26 halaman, yang menunjukkan kepada Anda bagaimana menggunakan antarmuka untuk menghasilkan kode CI. Ini adalah metode yang berbeda dengan sebagian besar penawaran lain yang diulas di sini: metode ini membuat kode CI untuk Anda, menggunakan antarmuka grafis, daripada menyediakan pustaka atau kode untuk Anda tambal.

SuperModel (lihat http://codeigniter.com/wiki/SuperModel/) mengklaim: "Perpustakaan SuperModel adalah perluasan model untuk mengotomatisasi sebagian besar tugas pembuatan dan validasi bentuk biasa. Anggap saja sebagai perancah pada steroid."

Komentar penulis menjelaskan rasa frustrasi menulis kode semacam ini—dan juga risiko bagi pengguna. Dia berkata: "Harap diperhatikan bahwa perpustakaan ini sedang dalam proses. Saat ini saya membuat banyak perubahan, termasuk perubahan API yang akan merusak aplikasi. Saat saya menulis ini (30 Mei 2006) saya sedang mengerjakan implementasi satu<>banyak dan banyak<>banyak yang bergabung……Tidak mungkin menulis sesuatu seperti ini, tetapi tetap fleksibel seperti CodeIgniter Sayangnya, perpustakaan ini memaksa Anda untuk melakukan beberapa hal-hal dengan cara tertentu. Saya telah mencoba untuk menjadi sefleksibel mungkin, tetapi pada saat yang sama, harus ada garis yang ditarik antara menjadi fleksibel, dan benar-benar kembung. Itulah mengapa ini adalah perpustakaan pihak ketiga eksternal—Anda bebas mengimplementasikan model dengan caranya Anda inginkan, atau gunakan perpustakaan pihak ketiga serupa lainnya yang melakukan hal serupa."

15.4 SUMBER DAYA UNTUK PROGRAM LAIN, MIS. XAMPLITE, MYSQL, PHP

Ada banyak sumber daya yang berguna untuk PHP. Mari kita sentuh beberapa di antaranya secara singkat.

- PHP sendiri dapat diunduh gratis dari www.php.net, yang juga menyertakan manual lengkap.
- Editor PHP murah dapat dibeli dari MP Software di http://www.mpsoftware.dk/.

Ada banyak buku bagus tentang PHP, termasuk Pemrograman PHP dengan PEAR, oleh Carsten Lucke, Aaron Wormus, Stoyan Stefanov dan Stephan Schmidt, diterbitkan oleh Packt.

Untuk menjalankan server web lokal di mesin Anda sendiri, coba lihat http://www.apachefriends.org/en/index.html—situs yang menawarkan unduhan gratis paket XAMPP. Ini menginstal server web Apache dengan MySQL, PHP, dan Perl. Jika paket XAMPP terlalu lengkap untuk Anda, cobalah Minixampp dari situs yang sama, di mana kode yang digunakan dalam buku ini ditulis.

MySQL juga memiliki halaman webnya sendiri—http://www.mysql.com/—meskipun jika Anda ingin mengunduh versi terbaru secara gratis, kunjungi http://dev.mysql.com/. (Ingatlah meskipun banyak ISP tidak menggunakan versi terbaru. Meskipun MySQL sudah mencapai versi 5, sebagian besar ISP masih menggunakan Versi 4. Ini mencegah Anda menggunakan beberapa fitur baru yang lebih menarik, seperti prosedur tersimpan.) Lihat , Membuat Database MySQL Anda: Tips dan Teknik Desain Praktis, oleh Marc Delisle, diterbitkan oleh Packt.

Meskipun MySQL hadir dengan alatnya sendiri, alat yang paling populer (dan paling umum) adalah PHPMyAdmin. (Lihat Menguasai phpMyAdmin 2.8 untuk Manajemen MySQL yang Efektif, juga oleh Marc Delisle, diterbitkan oleh Packt.)

15.5 RINGKASAN

Dalam Bab ini, kita telah melihat beberapa sumber daya yang tersedia untuk Anda ketika Anda mulai membuat kode dengan CI. Ada banyak kode siap pakai yang tersedia. Anda harus melihat sebelum menggunakan: jangan hanya mengambil plug-in atau perpustakaan pertama yang tampaknya melakukan apa yang Anda inginkan dan mulai menggunakannya. Anda perlu mempelajari setiap penawaran untuk melihat apa yang sebenarnya dilakukannya, dan itu juga membantu untuk mempelajari kode dan memastikan Anda memahaminya. Namun, jika Anda siap untuk melakukan ini, Anda dapat menemukan perpustakaan di berbagai tingkat cakupan dan kompleksitas yang akan mengambil banyak tugas yang seharusnya melibatkan banyak pengkodean tangan.

Secara khusus, kami melihat perpustakaan untuk

- AJAX dan JavaScript
- Autentikasi
- Bagan
- CRUD

Terakhir, kami melihat beberapa sumber daya yang tersedia untuk PHP dan MySQL dan untuk menjalankan server web lokal.

BAB 16
PENGUJIAN BROWSER DENGAN CODECEPTION

Pada bab sebelumnya, kita telah menggunakan PHPUnit, tetapi dalam bab ini kita akan mengeksplorasi beberapa alat lain. Kami akan menginstal Codeception dan Selenium Server, mempelajari cara mengonfigurasinya, dan cara menulis tes browser.

16.1 MENGINSTAL DAN MENGONFIGURASI CODECEPTION

Apa itu Codeception?

Codeception adalah kerangka pengujian PHP yang menyediakan tiga tingkat pengujian: Tes Penerimaan, Tes Fungsional, dan Tes Unit. Kami menggunakan Tes Penerimaan sebagai sinonim untuk Tes Browser dalam bab ini. Setiap situs web dapat dicakup dengan tes penerimaan yang disediakan oleh Codeception. Untuk menggunakan tes fungsional, kami memerlukan modul untuk kerangka kerja kami, tetapi kami tidak memiliki modul untuk CodeIgniter. Untuk alasan ini, kami hanya akan menggunakan Codeception untuk tes penerimaan dalam bab ini.

Pada bab sebelumnya, kita telah menggunakan PHPUnit dengan ci-phpunit-test yang menyediakan integrasi framework dengan CodeIgniter. Karena kami tidak memiliki integrasi kerangka kerja untuk Codeception, kami akan menggunakan Codeception sendiri, yang seharusnya menyederhanakan proses pembelajaran untuk menggunakannya untuk tes penerimaan.

Menginstal Codeception

Unduh versi terbaru (dalam buku ini kami menggunakan v2.1.4) dari http://codeception.com/install dan letakkan codecept.phar yang diunduh di direktori root proyek PHP Anda.

```
CodeIgniter/
└── codecept.phar
```

Untuk pengguna Komposer

Codeception juga dapat diinstal melalui Composer sebagai berikut:

```
$ cd CodeIgniter/
$ composer require  codeception/codeception --dev
```

Setelah diinstal melalui Composer, gunakan vendor/bin/codecept sebagai ganti php codecept.phar dalam instruksi selanjutnya di bab ini.

Apa itu Selenium Server?

Selenium adalah proyek yang mengotomatiskan browser yang terdiri dari tiga bagian: WebDriver, Server, dan klien. Selenium WebDriver adalah driver untuk memanipulasi

browser. Selenium Server adalah aplikasi server Java yang berkomunikasi dengan klien Selenium (dalam kasus kami, facebook/webdriver[1] di Codeception) dan Selenium WebDriver. Singkatnya, Codeception bertindak sebagai klien Selenium kami, dan menggunakan Server Selenium untuk mengontrol browser yang digunakan dalam pengujian kami. Selenium Server menggunakan WebDriver untuk mengontrol browser yang ditentukan saat kami menjalankan Codeception.

Memasang Server Selenium

Unduh versi terbaru (dalam buku ini kami menggunakan v2.48.2) dari http://docs.seleniumhq.org/download/ dan tempatkan selenium-server-standalone-x.x.x.jar yang diunduh di direktori root proyek PHP Anda.

```
CodeIgniter/
└── selenium-server-standalone-2.48.2.jar
```

Menginisialisasi Codeception

Untuk memulai, inisialisasi Codeception dengan menjalankan perintah berikut:

```
$ php codecept.phar bootstrap
```

Perintah ini membuat beberapa file dan direktori tes. Anda akan melihat output yang mirip dengan ini:

```
 Initializing Codeception in .../CodeIgniter

File codeception.yml created       <- global configuration
tests/unit created                 <- unit tests
tests/unit.suite.yml written       <- unit tests suite configuration
tests/functional created           <- functional tests
tests/functional.suite.yml written <- functional tests suite configuration
tests/acceptance created           <- acceptance tests
tests/acceptance.suite.yml written <- acceptance tests suite configuration
tests/_output was added to .gitignore
 ---
tests/_bootstrap.php written <- global bootstrap file

 Building initial Tester classes
 Building Actor classes for suites: acceptance, functional, unit
  -> AcceptanceTesterActions.php generated successfully. 0 methods added
 \AcceptanceTester includes modules: PhpBrowser, \Helper\Acceptance
 AcceptanceTester.php created.
  -> FunctionalTesterActions.php generated successfully. 0 methods added
 \FunctionalTester includes modules: \Helper\Functional
 FunctionalTester.php created.
  -> UnitTesterActions.php generated successfully. 0 methods added
 \UnitTester includes modules: Asserts, \Helper\Unit
 UnitTester.php created.

 Bootstrap is done. Check out .../CodeIgniter/tests directory
```

Dalam buku ini, kami hanya menggunakan Codeception untuk pengujian penerimaan, jadi semua file kasus uji ditempatkan di direktori tes/penerimaan.

```
CodeIgniter/
├── codeception.yml  ... Codeception's global configuration file.
└── tests/
    ├── acceptance/                ... Your test files will be placed here.
    ├── acceptance.suite.yml ... Configuration file for acceptance tests.
    └── ...
```

16.2 MENGONFIGURASI TES PENERIMAAN

Menguji dengan Firefox

Kami menjalankan tes penerimaan dengan browser web kami. Pertama, kita akan menggunakan Firefox, karena Selenium Standalone Server memiliki driver Firefox bawaan. Atur browser dan URL situs kami di acceptance.suite.yml.

```
--- a/CodeIgniter/tests/acceptance.suite.yml
+++ b/CodeIgniter/tests/acceptance.suite.yml
@@ -7,6 +7,7 @@
 class_name: AcceptanceTester
 modules:
     enabled:
-        - PhpBrowser:
-            url: http://localhost/myapp
+        - WebDriver:
+
+            url: http://127.0.0.1:8000/
+            browser: 'firefox'
         - \Helper\Acceptance
```

Daftar di atas dalam format yang disebut unified diff. Ini menunjukkan perbedaan antara dua file. Singkatnya, hapus garis merah yang dimulai dengan - dari file asli, dan tambahkan garis hijau yang dimulai dengan + untuk mendapatkan file baru. Lihat Konvensi yang Digunakan dalam Buku Ini untuk perincian tambahan.

16.3 TES MENULIS

Konvensi untuk Tes Penerimaan Codeception. Berikut adalah beberapa konvensi dasar untuk tes penerimaan Codeception.

1. File uji memiliki akhiran Cept.php.
2. Kode tes dimulai dengan "$I = new AcceptanceTester($scenario);".
3. Metode `$I->wantTo()` mendefinisikan skenario Anda (nama pengujian).

Kami meletakkan file di direktori `tests/acceptance`.

Cept adalah format berbasis skenario untuk file kasus uji di Codeception. Meskipun kami tidak menggunakannya dalam buku ini, Codeception juga memiliki format Cest, yang menggabungkan pendekatan pengujian berbasis skenario dengan desain OOP. Jika Anda ingin mengelompokkan beberapa skenario pengujian menjadi satu, Anda harus

mempertimbangkan untuk menggunakan format Cest. Lihat dokumentasi Kelas Cest[2] untuk detailnya.

Menulis Tes Pertama

Sekarang setelah kita mengetahui konvensi dasar, kita dapat mulai menulis tes penerimaan pertama kita.

tests/acceptance/WelcomeCept.php

```
1  <?php
2  $I = new AcceptanceTester($scenario);
3  $I->wantTo('ensure that frontpage works');
4  $I->amOnPage('/');
5  $I->see('Home');
```

Dengan membaca kode di atas, Anda mungkin memiliki ide bagus tentang apa yang dilakukan tes ini.

Kita harus selalu menulis baris "$I = new AcceptanceTester($scenario);" di bagian atas kode tes. Metode $I->wantTo() mendefinisikan skenario kita (nama pengujian). Sisa pengujian mengakses /, dan memeriksa apakah halaman yang diterima dalam respons berisi string Beranda.

16.4 MENJALANKAN TES

Menjalankan Server Selenium

Untuk menjalankan tes, kita perlu memulai server Selenium untuk mengontrol browser web. Jalankan server dengan perintah berikut:

```
$ java -jar selenium-server-standalone-2.48.2.jar
```

Menjalankan Server Web

Kami juga membutuhkan server web. Buka terminal lain, dan jalankan server web bawaan PHP:

```
$ CI_ENV=testing php -S 127.0.0.1:8000 index.php
```

Kami menambahkan CI_ENV=testing untuk menjalankannya di bawah lingkungan pengujian. Karena kami ingin mengatur lingkungan CodeIgniter yang berjalan di server ke nilai pengujian. Ini adalah hal yang sama yang Anda setel CI_ENV di file konfigurasi .htaccess atau Apache Anda.

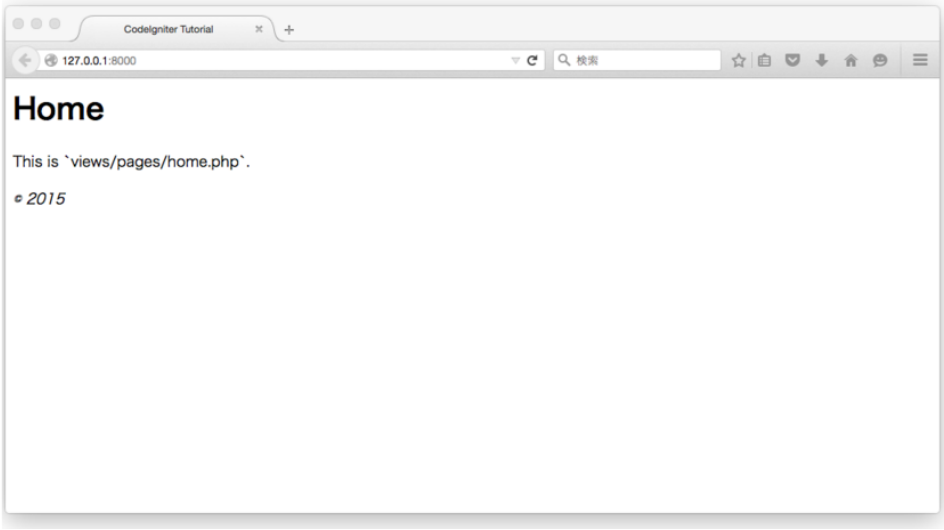
Pengujian PHPUnit kami berjalan di lingkungan pengujian, karena ci-phpunit-test mengubah lingkungan secara otomatis. Namun dalam hal ini, proses PHP pada web server berbeda dengan proses menjalankan Codeception. Jadi kami tidak memiliki cara untuk mengubah lingkungan CodeIgniter di server dari kode pengujian kami.

Menjalankan Codeception

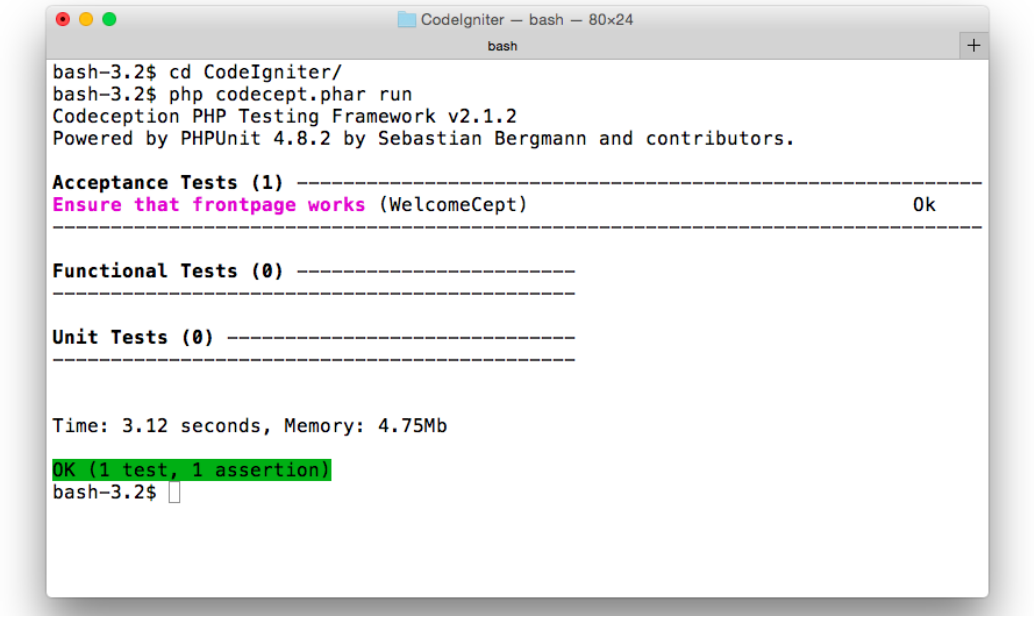
Karena kita akan menggunakan Firefox[3], itu perlu diinstal juga. Buka terminal lain. Jika Anda menggunakan perintah run codecept, Codeception menjalankan semua tes:

```
$ php codecept.phar run
```

Anda akan melihat Firefox memulai dan mengakses http://127.0.0.1:8000.



Gambar 16.1 Tampilan Firefox



Gambar 16.2 Jalankan perintah codecept run

Anda akan melihat OK hijau di terminal Anda. Tes pertama kami lulus, tetapi kami juga melihat garis di bawah ini:

```
Functional Tests (0) ----------
----------

Unit Tests (0) ----------
----------
```

Kami tidak memiliki tes fungsional atau tes unit dengan Codeception. Karena kami menggunakan PHPUnit untuk pengujian tersebut, kami tidak membutuhkannya di Codeception. Kami hanya dapat menjalankan tes penerimaan menggunakan perintah berikut:

```
$ php codecept.phar run acceptance
```

```
Codeception PHP Testing Framework v2.1.4
Powered by PHPUnit 4.8.10 by Sebastian Bergmann and contributors.

Acceptance Tests (1) ------------------------------------------------------------
Ensure that frontpage works (WelcomeCept)                                    Ok
----------------------------------------------------------------------------------


Time: 3.75 seconds, Memory: 4.75Mb

OK (1 test, 1 assertion)
```

16.5 PENGUJIAN BROWSER: PRO DAN KONTRA

Ada pro dan kontra dalam pengujian browser. Kelebihan:

- dapat menguji seluruh sistem (ini adalah pengujian sistem)
- dapat menguji permintaan JavaScript dan Ajax
- kurang terpengaruh oleh perubahan kode sumber
- dapat dijalankan di situs web mana pun
- dapat ditunjukkan kepada klien dan manajer Anda Kontra:
- mereka sangat lambat (membutuhkan server web, browser web, dan database yang berjalan)
- terpengaruh oleh perubahan tampilan
- pemeriksaan yang lebih sedikit dapat menyebabkan hasil positif palsu
- masalah rendering dan JavaScript dapat menyebabkan hasil yang tidak terduga

Tes browser menguji seluruh sistem, jadi kami membutuhkan database. Itu berarti kita juga membutuhkan perlengkapan database. Codeception mencakup fungsionalitas untuk memulihkan file dump SQL ke database saat menjalankan tes penerimaan. Namun, karena kami telah menggunakan migrasi database dan seeder untuk pengujian kami, kami tidak memiliki file dump SQL.

Kita dapat membuat file dump SQL, tetapi migrasi dan seeder memiliki banyak manfaat, seperti yang dibahas dalam Bab 5. Karena kita sudah memiliki migrasi, seeder, dan kemampuan untuk menjalankannya, kita harus menggunakannya di sini, demikian juga.

Kami menggunakan pengontrol Dbfixture, yang kami uji di Bab 8, dan menjalankan pengontrol sebelum menjalankan tes penerimaan. Jika kita melakukannya dengan cara ini, kita tidak perlu memelihara file dump SQL, dan kita memanfaatkan infrastruktur pengujian yang ada dengan baik.

application/controllers/Dbfixture.php

```php
<?php

class Dbfixture extends CI_Controller
{
    public function __construct()
    {
        parent::__construct();

        // Only accessible via CLI
        if (is_cli() === false) {
            exit();
        }
    }

    public function migrate()
    {
        $this->load->library('migration');
        if ($this->migration->current() === false) {
            echo $this->migration->error_string() . PHP_EOL;
        }
    }

    public function seed()
    {
        $this->load->library('seeder');
        foreach (glob(APPPATH.'database/seeds/*Seeder.php') as $file) {
            $seeder = basename($file, '.php');
            $this->seeder->call($seeder);
        }
    }

    public function all()
    {
        $this->migrate();
        $this->seed();
    }
}
```

Kontroler ini memeriksa untuk memastikannya dijalankan dari CLI, dan metode all() menjalankan migrasi, dan mengeksekusi semua file seeder yang memiliki akhiran Seeder.php di direktori application/database/seeder.

16.6 KASUS UJI UNTUK PENGONTROL BERITA

Kami menulis kasus uji untuk pengontrol Berita di Bab 7, Kasus Uji untuk Pengontrol Berita. Sekarang, kita akan menulis tes yang sama dengan Codeception.

tests/acceptance/NewsCept.php

```php
<?php
$I = new AcceptanceTester($scenario);
$I->wantTo('ensure that News works');
```

Perlengkapan Basis Data

Sebelum menjalankan tes, kita harus menyiapkan database kita. Kami telah membuat Dbfixture controller untuk membantu kami melakukan ini. Bagaimana kami menjalankan

pengontrol dari pengujian Codeception kami? Kita bisa menjalankannya dengan fungsi system()[4] di PHP.

tests/acceptance/NewsCept.php

```
<?php

system("CI_ENV=testing php index.php dbfixture all");

$I = new AcceptanceTester($scenario);
$I->wantTo('ensure that News works');
```

Untuk pengguna Komposer

Jika Anda menginstal CodeIgniter melalui Composer, jalur index.php berbeda.

```
system("CI_ENV=testing php public/index.php dbfixture all");
```

Menguji Konten Halaman

Kita dapat menulis tes dengan cara yang sama seperti yang kita lakukan di WelcomeCept. Setel URI dengan $I->amOnPage() metode dan periksa hasilnya dengan metode $I->see() .

tests/acceptance/NewsCept.php

```
$I->amGoingTo('access the archive page');
$I->amOnPage('/news');
$I->see('News archive');
$I->see('News test');
$I->see('News text', '.main');
$I->seeInTitle('CodeIgniter Tutorial');
```

Komentar

Metode $I->amGoingTo() memungkinkan Anda mendeskripsikan tindakan yang akan Anda lakukan. Ini hanyalah metode untuk menambahkan komentar ke pengujian Anda. Anda juga dapat menggunakan metode $I->expect() dan $I->expectTo() untuk menambahkan komentar yang menjelaskan hasil yang diharapkan.

```
$I->amGoingTo('access the archive page');
```

Pernyataan

Metode $I->see() adalah metode pernyataan. Ini memungkinkan kita untuk memeriksa apakah string argumen ada di halaman.

```
$I->see('News archive');
$I->see('News test');
```

Kita dapat menggunakan argumen kedua untuk menentukan kelas CSS yang kita harapkan akan diterapkan ke string.

```
$I->see('News text', '.main');
```

Kami juga dapat memeriksa judul halaman.

```
$I->seeInTitle('CodeIgniter Tutorial');
```

Untuk mempelajari lebih lanjut tentang metode asersi, lihat bagian asersi dokumentasi[5].

Formulir Pengujian

Codeception juga memungkinkan kita untuk mengisi dan mengirimkan formulir. Kami menggunakan metode $I->fillField() untuk mengisi kolom di formulir, kemudian menggunakan metode $I->click() untuk mengklik tombol kirim.

tests/acceptance/NewsCept.php

```
$I->amGoingTo('post a valid item');
$I->fillField('title', 'CodeIgniter is easy to write tests');
$I->fillField('text', 'You can write tests for controllers very easily!');
$I->click('Create news item');
$I->see('Successfully created');
```

Jika Anda ingin tahu lebih banyak tentang manipulasi formulir, lihat bagian Formulir dokumentasi[6].

BeritaCept

Berikut ini adalah daftar kode lengkap untuk NewsCept.php.

tests/acceptance/NewsCept.php

```
<?php

system("CI_ENV=testing php index.php dbfixture all");

$I = new AcceptanceTester($scenario);
$I->wantTo('ensure that News works');

$I->amGoingTo('access the archive page');
$I->amOnPage('/news');
$I->see('News archive');
$I->see('News test');
$I->see('News text', '.main');
$I->seeInTitle('CodeIgniter Tutorial');

$I->amGoingTo('access an item which does not exist');
$I->amOnPage('/news/not-existing-slug');
$I->see('404 Page Not Found');
$I->dontSeeInTitle('CodeIgniter Tutorial');

$I->amGoingTo('access an existing item');
$I->amOnPage('/news/news-test');
$I->see('News test');
$I->seeInTitle('CodeIgniter Tutorial');

$I->amGoingTo('access the create page');
$I->amOnPage('/news/create');
$I->see('Create a news item');

$I->amGoingTo('post a valid item');
$I->fillField('title', 'CodeIgniter is easy to write tests');
$I->fillField('text', 'You can write tests for controllers very easily!');
$I->click('Create news item');
$I->see('Successfully created');

$I->amGoingTo('access the archive page');
$I->amOnPage('/news');
$I->see('News test');
$I->see('CodeIgniter is easy to write tests');

$I->amGoingTo('post an invalid item');
$I->amOnPage('/news/create');
$I->click('Create news item');
$I->see('The Title field is required.');
$I->see('The Text field is required.');
```

Jalankan dan konfirmasikan bahwa tes lulus. Jangan lupa untuk menjalankan server Selenium dan server web sebelum menjalankan tes.

```
$ php codecept.phar run acceptance tests/acceptance/NewsCept.php
```

Anda akan melihat OK hijau seperti berikut:

```
Codeception PHP Testing Framework v2.1.4
Powered by PHPUnit 4.8.10 by Sebastian Bergmann and contributors.

Acceptance Tests (1) ------------------------------------------------------------
Ensure that news works (NewsCept)                                          Ok
---------------------------------------------------------------------------------


Time: 5.86 seconds, Memory: 5.25Mb

OK (1 test, 14 assertions)
```

Jika Anda ingin tahu lebih banyak tentang fungsionalitas tes penerimaan Codeception, lihat dokumentasi Acceptance Testing[7].

16.7 PENGUJIAN DENGAN GOOGLE CHROME

Untuk menjalankan pengujian dengan Google Chrome, kita perlu menginstal Google Chrome[8] dan ChromeDriver untuk Selenium.

Memasang ChromeDriver

Anda dapat menemukan drivernya di http://www.seleniumhq.org/download/
atau https://github.com/SeleniumHQ/Selenium/wiki/ChromeDriver.
Anda mungkin harus mengikuti beberapa tautan untuk mendapatkan driver terbaru. Dalam buku ini, kami menggunakan v2.20[9]. Unduh driver untuk platform Anda dan unzip. Pindahkan driver ke direktori root proyek PHP Anda.

```
CodeIgniter/
└── chromedriver
```

Untuk pengguna Windows

Nama file driver untuk Windows adalah chromedriver.exe.

Mengonfigurasi Tes Penerimaan

Untuk menggunakan driver Chrome dalam pengujian Anda, tentukan lingkungan yang berbeda di dalam env root di accceptance.suite.yml. Beri nama lingkungan (firefox, chrome, dll.), lalu definisikan ulang parameter konfigurasi yang telah ditetapkan sebelumnya.

```
--- a/code/CodeIgniter/tests/acceptance.suite.yml
+++ b/code/CodeIgniter/tests/acceptance.suite.yml
@@ -11,3 +11,11 @@ modules:
             url: http://127.0.0.1:8000/
             browser: 'firefox'
         - \Helper\Acceptance
+env:
+    chrome:
+        modules:
+            config:
+                WebDriver:
+                    browser: 'chrome'
+    firefox:
+        # nothing changed
```

Daftar di atas dalam format yang disebut unified diff. Ini menunjukkan perbedaan antara dua file. Singkatnya, tambahkan garis hijau yang dimulai dengan + ke file asli. Lihat Konvensi yang Digunakan dalam Buku Ini untuk rinciannya.

Menjalankan Server Selenium

Saat Anda menjalankan server Selenium, tambahkan jalur file driver dengan opsi berikut:

```
$ java -jar selenium-server-standalone-2.48.2.jar  \-Dwebdriver.chrome.driver=./chromedriver
```

Karena kesulitan teknis dalam sistem penerbitan buku ini, saya mengubah gaya perintah Bash yang panjang dengan memecahnya menjadi lebih dari satu baris.

Untuk pengguna Windows

Nama file driver untuk Windows adalah chromedriver.exe.

Sekarang, kami siap untuk mulai menjalankan pengujian dengan Chrome.

Menjalankan Tes

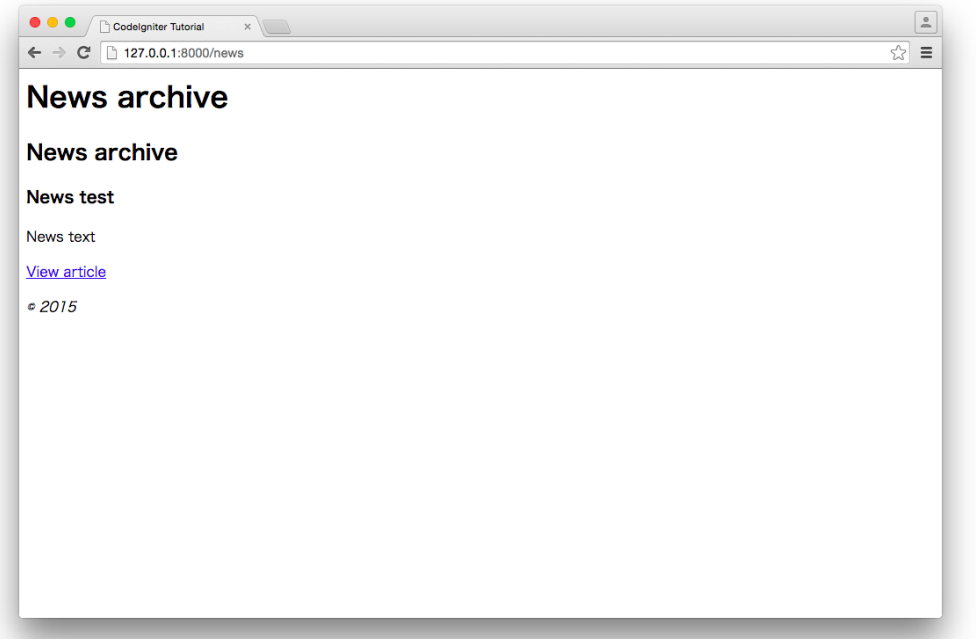
Jalankan server web bawaan PHP sebelum menjalankan Codeception:

```
$ CI_ENV=testing  php -S 127.0.0.1:8000 index.php
```

Jalankan perintah codecept dengan opsi --env:

```
$ php codecept.phar run acceptance --env chrome
```

Anda akan melihat Google Chrome memulai dan menjalankan tes.

f 

Gambar 16.3 Tampilan pada Chrome

```
Codeception PHP Testing Framework v2.1.4
Powered by PHPUnit 4.8.10 by Sebastian Bergmann and contributors.

Acceptance-chrome Tests (2) ---------------------------------------------------
Ensure that news works (NewsCept)                                          Ok
Ensure that frontpage works (WelcomeCept)                                  Ok
-------------------------------------------------------------------------------


Time: 6.4 seconds, Memory: 5.00Mb

OK (2 tests, 15 assertions)
```

Anda juga dapat menjalankan tes di beberapa browser:

```
$ php codecept.phar run acceptance --env firefox --env chrome
```

```
Codeception PHP Testing Framework v2.1.4
Powered by PHPUnit 4.8.10 by Sebastian Bergmann and contributors.

Acceptance-firefox Tests (2) ----------------------------------------------------
Ensure that news works (NewsCept)                                          Ok
Ensure that frontpage works (WelcomeCept)                                  Ok
---------------------------------------------------------------------------------

Acceptance-chrome Tests (2) -----------------------------------------------------
Ensure that news works (NewsCept)                                          Ok
Ensure that frontpage works (WelcomeCept)                                  Ok
---------------------------------------------------------------------------------


Time: 11.74 seconds, Memory: 5.25Mb

OK (4 tests, 30 assertions)
```

Output di atas menunjukkan bahwa kami telah menjalankan semua tes dengan dua browser, dan hasilnya berwarna hijau.

LAMPIRAN

MYSQL QUERY UNTUK MENGATUR BASIS DATA 'SITUS WEB'

```
DROP TABLE IF EXISTS `ci_sessions`;
CREATE TABLE IF NOT EXISTS `ci_sessions` (
        `session_id` varchar(40) NOT NULL default '0',
        `peopleid` int(11) NOT NULL,
        `ip_address` varchar(16) NOT NULL default '0',
        `user_agent` varchar(50) NOT NULL,
        `last_activity` int(10) unsigned NOT NULL default '0',
        `left` int(11) NOT NULL,
        `name` varchar(25) NOT NULL,
        `status` tinyint(4) NOT NULL default '0' )
ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `domains`;
CREATE TABLE IF NOT EXISTS `domains` (
        `id` int(10) NOT NULL auto_increment,
        `url` varchar(100) NOT NULL,
        `name` varchar(100) NOT NULL,
        `registrar` varchar(100) NOT NULL,
        `dateregd` int(11) NOT NULL default '0',
        `cost` float NOT NULL default '0',
        `regdfor` int(11) NOT NULL default '0',
        `notes` blob NOT NULL,
        `pw` varchar(25) NOT NULL,
        `un` varchar(25) NOT NULL,
        `lastupdate` int(11) NOT NULL default '0',
        `submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=10 ;

DROP TABLE IF EXISTS `events`;
CREATE TABLE IF NOT EXISTS `events` (
        `id` int(10) NOT NULL auto_increment,
        `name` varchar(50) NOT NULL default 'not set',
        `type` enum('test','alert','report') NOT NULL,
        `testid` int(10) NOT NULL,
        `siteid` int(10) NOT NULL,
        `userid` int(10) NOT NULL,
        `reported` int(11) NOT NULL,
        `result` blob NOT NULL,
        `time` int(11) NOT NULL,
        `timetaken` float NOT NULL,
        `isalert` varchar(2) NOT NULL,
        `emailid` int(11) NOT NULL,
        `submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=69 ;

DROP TABLE IF EXISTS `frequencies`;
CREATE TABLE IF NOT EXISTS `frequencies` (
        `id` int(10) NOT NULL,
```

```
	`name` varchar(16) NOT NULL,
	`submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `hosts`;
CREATE TABLE IF NOT EXISTS `hosts` (
	`id` int(11) NOT NULL auto_increment,
	`cost` float NOT NULL,
	`name` varchar(100) NOT NULL,
	`hosturl` varchar(100) NOT NULL,
	`un` varchar(50) NOT NULL,
	`pw` varchar(50) NOT NULL,
	`ns1url` varchar(36) NOT NULL,
	`ns1ip` varchar(36) NOT NULL,
	`ns2url` varchar(36) NOT NULL,
	`ns2ip` varchar(36) NOT NULL,
	`ftpurl` varchar(100) NOT NULL,
	`ftpserverip` varchar(36) NOT NULL,
	`ftpun` varchar(50) NOT NULL,
	`ftppw` varchar(50) NOT NULL,
	`cpurl` varchar(36) NOT NULL,
	`cpun` varchar(36) NOT NULL,
	`cppw` varchar(36) NOT NULL,
	`pop3server` varchar(36) NOT NULL,
	`servicetel` varchar(50) NOT NULL,
	`servicetel2` varchar(50) NOT NULL,
	`serviceemail` varchar(100) NOT NULL,
	`webroot` varchar(48) NOT NULL,
	`absoluteroot` varchar(48) NOT NULL,
	`cgiroot` varchar(48) NOT NULL,
	`booked` int(11) NOT NULL,
	`duration` int(11) NOT NULL,
	`lastupdate` int(11) NOT NULL default '0',
	`submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;

DROP TABLE IF EXISTS `people`;
CREATE TABLE IF NOT EXISTS `people` (
	`id` int(11) NOT NULL auto_increment,
	`uname` varchar(25) NOT NULL,
	`pw` varchar(25) NOT NULL,
	`status` smallint(3) NOT NULL default '1',
	`name` varchar(50) NOT NULL,
	`firstname` varchar(50) NOT NULL,
	`surname` varchar(50) NOT NULL,
	`email` varchar(120) NOT NULL,
	`lastupdate` int(11) NOT NULL default '0',
	`submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

DROP TABLE IF EXISTS `sites`;
CREATE TABLE IF NOT EXISTS `sites` (
	`id` int(10) NOT NULL auto_increment,
```

```
        `name` varchar(100) NOT NULL,
        `url` varchar(100) NOT NULL,
        `un` varchar(50) NOT NULL,
        `pw` varchar(50) NOT NULL,
        `client1` int(10) NOT NULL default '0',
        `client2` int(10) NOT NULL default '0',
        `admin1` int(10) NOT NULL default '0',
        `admin2` int(10) NOT NULL default '0',
        `domainid` int(10) NOT NULL default '0',
        `hostid` int(10) NOT NULL default '0',
        `webroot` varchar(50) NOT NULL,
        `files` text NOT NULL,
        `filesdate` int(11) NOT NULL default '0',
        `lastupdate` int(11) NOT NULL default '0',
        `submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=15 ;

DROP TABLE IF EXISTS `tests`;
CREATE TABLE IF NOT EXISTS `tests` (
        `id` int(11) NOT NULL auto_increment,
        `siteid` int(11) NOT NULL default '0',
        `name` varchar(250) NOT NULL,
        `type` varchar(25) NOT NULL,
        `url` varchar(120) NOT NULL,
        `regex` varchar(250) NOT NULL,
        `p1` varchar(250) NOT NULL,
        `p2` varchar(250) NOT NULL,
        `p3` varchar(250) NOT NULL,
        `p4` varchar(250) NOT NULL,
        `p5` varchar(250) NOT NULL,
        `p6` varchar(250) NOT NULL,
        `frequency` int(10) NOT NULL default '0',
        `lastdone` int(10) NOT NULL default '0',
        `isalert` varchar(2) NOT NULL,
        `setup` int(10) NOT NULL default '0',
        `lastupdate` int(10) NOT NULL default '0',
        `notes` varchar(250) NOT NULL,
        `submit` varchar(25) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=11 ;

DROP TABLE IF EXISTS `types`;
CREATE TABLE IF NOT EXISTS `types` (
        `id` varchar(7) NOT NULL,
        `name` varchar(50) NOT NULL, PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

DAFTAR PUSTAKA

Ardhana, Y. K. (2013). Pemrograman PHP: Codeigniter Black Box. Jakarta: Jasakom.

Basuki, A. (2010). Membangun Web Berbasis PHP Dengan Framework Codeigniter. Yogyakarta: Lokomedia.

Binarto, S. (2012). Tips & Trik Membuat Program Penjualan Menggunakan Visual Basic 6.0 . Jakarta Selatan: Mediakita.

Hakim, Lukmanul. (2010). Membangun Web Berbasis PHP dengan Framework Codeigniter. Yogyakarta : Lokomedia.

Hapsari, S. (2010). Pembuatan Website Pada Google Original Movie Rental Pacitan . Journal Speed, 48-54.

Pratama, Antonius Nugraha Widhi. (2010). Codeigniter: Cara Mudah Membangun Aplikasi PHP. Jakarta Selatan: Mediakita.

Prihatna, H. (2005). Kiat Praktis Menjadi Webmaster Profesional. Jakarta: PT. Elex Media Komputindo.

Riyanto. (2011). Membuat Sendiri Aplikasi E-commerce dengan PHP dan MySQL Menggunakan Codeigniter dan JQuery. Yogyakarta: Andi.

Shalahuddin, M. &. (2011). Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek). Bandung: Modula.

Supaartagor, C. (2011). PHP Framework For database Management on MVC Pattern. Thailand: Ubon ratchathani University.

Supardi, I. Y., & Hermawan, A. (2018). Semua Bisa Menjadi Programmer CodeIgniter Basic. Elex Media Komputindo.

PHP FRAMEWORK

Oleh :
Iman Saufik Suasana, M.Kom

BIODATA PENULIS

Iman Saufik Suasana, M.Kom Lahir di Semarang tanggal 9 September 1976. Alumni S1 Sistem Komputer dari Sekolah Tinggi Elektronika dan Komputer serta Magister Teknik Informatika Universitas Dian Nuswantoro. Saat ini menjadi Dosen dan Ketua Program Studi S1 Sistem Komputer Universitas Sains dan Teknologi Komputer.

Mengampu beberapa matakuliah seperti Pengantar Teknologi Informasi, Logika dan Algoritma Pemrograman, Komunikasi Data dan Jaringan, Pemrograman Web, Perancangan Sistem Informasi, Internet Marketing dan lain-lain. Buku yang pernah di hasilkan di antaranya Pengantar Teknologi Informasi dan Pemrograman JavaScript.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-48-2 (PDF)
9 786238 120482