



From Technologies to Solutions

# PHP 敏捷开发框架 CodeIgniter

—— 快速 Web 应用开发详解

用开源免费的、轻量级的、基于 MVC 的 CodeIgniter 框架  
提高你的 PHP 编码生产力！

David Upton 著  
CodeIgniter 中国 译

**[PACKT]**  
PUBLISHING

**David Upton** 是一家专业管理咨询公司的董事，公司总部设在伦敦，但工作在世界各地。他的客户包括一些世界上最大的公司。他对他的基于 Web 的工作越来越感兴趣了，并努力寻求最简便易行的把思想转化为健壮的专业应用的途径。他至今已为英国的两家重要的公司开发了应用。他还有一个兴趣是模拟技术，这个占用了他很多写博客的时间和思维。

#### 译序

爽快地答应 Hex 为本书作序，目的很简单，推荐读者马上在自己的 PC 上安装 CI，并且遵照《20 分钟创建一个博客》的教程体验这套框架，看看你是不是真的也很喜欢它。

我在 95 年第一次接触 Web 开发，从制作 GIF 动画，到现在管理近万台的服务器集群，各种开发语言或多或少地接触过。虽然听说 CI 至今不超过两个月，也还没有通读过其代码，却在几天的文档阅读中数次感叹他巧妙的设计。我希望在今后的产品开发中使用它，原因很简单，因为有一套简单高效，能够协助你快速开发的框架很重要。它能有效地帮助你降低开发和未来的维护成本。

开发出好的产品，和你所使用语言、操作系统没有必然联系，关键是不断地进行性能优化和功能改进，Amazon 大量使用 Perl，FaceBook 选择 PHP，Google 则偏爱 C++。谁也不能说哪种语言最好，关键是要根据你自身产品的特点，团队的禀赋做出恰当的选择。在 Web 前端开发，我个人还是推荐 PHP，当你的访问量的规模上去以后，也很容易挂接上 Memcached、DB Shard 以及分布式文件系统。PHP 的 MVC 框架有很多，CI 是其中非常有特色的一个。

和硅谷相比，国内的研发积累底蕴还不够，同行之间也不够开放，这需要时间和我们每一个人的努力去改善。希望 Hex 和他的朋友们坚持下去，做好 CI 中文社区，多听取国内使用者的反馈，多为本地用户提供实用的功能，并使用它开发出有创意的产品。

李嵩波

2008 年 6 月于北京

# 第一章 对 CodeIgniter 的介绍

大多数 PHPer 都想写出运行状态良好的应用程序，而且希望尽可能做得简单且不费事。这篇文章是有关 CodeIgniter 的（以下简称 CI），CI 是一个达成以上目标的框架。

如果你只是要达成一个最终的结果，而把中间所有的编码细节和复杂统统丢给一个框架，CI 是你最好的朋友。

CI 有很多优点：免费，轻量级，容易安装，它能使你的编程生涯变得很轻松。这一章我们会告诉你：

CI 能为你做什么？

什么是“框架”？CI 为什么能被称为框架？

“开源”商业模式。

CI 的某些不足（是的，它并不完美）。

## 1.1 CodeIgniter 能为你做什么？

如果你已经是一位 PHPer，开发过 PHP 应用，CodeIgniter 将会帮助你做得更好，更容易达成目标。CI 会减少你的代码量。你的脚本可读性也会更好，更容易升级。它会使你的网站结构更紧凑，代码更强健，如果没有很好地研究 CI 的源代码，你可能还无法察觉到它的强健。

对大多数兄弟来讲，你可能已经花了不少时间，系统地学习了 PHP、HTML 和 CSS，当然还有 MySQL 什么的，不过如果使用 CI，你只需要一些基本的 LAMP（WAMP）知识，你没有必要先成为一个专家才能使用 CI。你完全可以先借助于 CI 或别的什么框架软件，成为一个有生产力的 PHP 程序员，拿着高薪然后优雅地进一步学习 PHP 的中高级知识，直至成为一位真正的 PHP 骨灰级的人物。

下述情形，你最好不要使用 CI：

你没有一点 PHP 和 HTML 的基本知识。

你想用最少的代码，快速简便的写一个基本的内容管理系统（CMS）（可以看看 Expression Engine）。

你想写一个只有几个标准特性的简单的网站。

### 1.1.1 节省时间

CI 学习周期短，见效快。让我们试着评估一下相关的要素：

CI 如何减少代码量？

你真的可以减少很多工作量：敲击键盘的次数减少了，代码错误减少了，你只需要很少的时间调试代码。代码量减少还意味着你只需要较少的空间来存放应用程序。

举两个例子（稍后它们会被进一步分析，因此不用担心如何了解它们的工作原理！）

想象你正在写一个 MySQL 数据库查询。可能的代码如下：

```
$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]} ";
}
```

现在看看 CI 如何处理同一个问题：

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

比较字符数：前者 336，后者 112。

第二个例子，现在让我们想象你正在用 HTML 写一个数据输入窗口，你想要一个下拉输入框。下拉框中有三个选项。代码如下：

```
HTML 代码
<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>
```

CI 的写法和前例一样，因为它把相关内容放入一个数组，更容易由 PHP 进行处理：

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

$variable .= form_dropdown('url', $urlarray, '3');
```

在 HTML 中，你需要输入 154 个字符；在 CI 中，只需要 128 个字符。

### 1.1.2 使你的网站更健壮

你不需要写很多代码，是因为 CI 提供了许多标准的功能，这些经过仔细推敲的框架内的代码，对安全性和输入进行了有效的校验和考虑。初学者往往没有足够的能力全面兼顾功能和安全。（这也是中高级程序员与新手之间能力差异的一个方面）

#### 1.1.2.1 确保你的链接自动更新

设想你正在编写一个菜单页面，有许多超链接可重定向到其他页面。他们全部以传统的 HTML 格式编写：

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

后来，你决定转移网站到其他 URL。这意味你必须仔细地去找查找并修改代码中的每一处 URL，否则它们将无法正常工作。

CI 给你一个简单的函数，可以这样编写超链接：

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

CI 推荐你把你的 URL 放入一个配置文件中供你的脚本读取。CI 的 anchor 函数会自动从配置文件中提取相关 URL。因此，当你修改一个 URL 时，你只需要修改配置文件中的对应链接，然后所有超链接将自动更新。

#### 1.1.2.2 防止对数据库的攻击：对表单输入的数据进行校验和处理

数据输入可能引发许多问题。因为 HTML 和数据库的限制，数据中总包含特定的符号——举例来说，省略符号和引号——可能导致你的数据库遭到攻击，最终得到你无法预料的结果。

解决方案是在把这些数据存入数据库前对这些数据进行相关处理。这样做会浪费一些系统时间，增加一些额外编码。

CI 的表单辅助函数会自动地完成这些工作。因此，当你编写一个输入框时：

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

CI 也隐式地执行下列校验函数：

```
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;
}

```

上述函数捕获像“&”这样的特殊字符，以便在你的页面提交时不会造成混乱。你应该知道，有些字符会引起问题。

并不是所有的用户都会中规中矩的输入符合要求的信息，你也不可能知道使用浏览器输入信息的是什么人，他们在想什么，做什么。你可以使用 CI 来防止输入不符合要求的信息。当然，你大可不必知道 CI 是如何在幕后为你做到这一切的，你只需要简单地输入如下代码：

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

### 1.1.3 增强你的代码

CI 使你写代码更容易了。不像有些类库如 PEAR 等，集成比较困难，（有时候你会找不到支持 PEAR 的空间），CI 很容易集成，只要把它放入一个目录，它就能很好地工作。CI 所有代码的可读性好，也很健壮，推出前经过社区用户的认真测试，所以在你可以使用时，这些代码已经经历了很多考验。

让我们看两个例子。

#### 1.1.3.1 发送 Email 和附件很简单

发送 Email 的功能实现起来比较复杂，但是，使用 CI 将使这件事变得很简单：

```

$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.');
```

实现发送 Email 的功能中有一些不容易解决的技术问题：比如设置文本自动换行（取消设置的话则可以保持长 URL 地址不被换行或截断）或发送附件。标准的 PHP 实现起来比较复杂，CI 简化了这些工作，它的 Email 类使得发送附件很简单：

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

CI 把内部的复杂部分悄悄地完成了，举例来说，实现了列举近百种不同附件的 MIME 类型的功能。所以它知道你的相片 photo1.jpg 是一个“image/jpeg”MIME 类型。因此它在你附件的适当位置填写必要的限制符号，它处理文本的换行，让你轻松标记出不希望出现换行的文本块。

#### 1.1.3.2 压缩用户要下载的文件以加快下载速度

为了加快下载速度，常见的做法是在下载之前压缩下载文件。你可能不知道如何处理。但 CI 可以方便地让你用 4 行代码完成此功能：

```

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

运行这些代码，你会在你的 C 盘根目录下找到一个压缩文件，解压后即为原始文件。

你网站的用户并不清楚你是如何简便实现这个功能的，但他们能体会到你的网站的下载速度很快，而你只用了数分钟（而不是数小时）就实现了这个功能。

## 1.2 CodeIgniter 是什么？框架又是什么？

当发明计算机编程不久之后，便有人发现，这其中涉及到了太多的重复操作。之后，也许是 Ada Lovelace（人类历史上的首位程序员），又或许是 Alan Turing，决定将计算机程序模块化，从而使得片段程序代码可以重复使用。PHP 程序员们早已习惯了将需要重复使用的代码写在函数中，并将这些函数放在 include 文件里。

同样的，框架是为重用而发明的，存放在独立的文件中，用来简化重复操作的代码。

上面例子中连接数据库和编写 HTML 表单元素的工作都可以调用相关的 CI 函数来进行简化。

它超越了这一点。有很多种方法实现同样的功能；大多数的

框架会让你按照它实现的方法来做。他们选择了一种方式来解决这个问题，所以你也必须要遵循这种方式。如果方式得当，编程便会轻松许多，反之则会事倍功半。

好的框架设计能实现需要的功能，而且尽可能地不互相牵连。一个好框架为你做出各种功能的实现，并且给你提供一步一步的编程指导。

提到框架时，就不能不提到著名的框架：Ruby on Rails。

Rails 做得相当成功，因为它籍由最少量的编码，提供简便快速的网站开发。本质上，它是一个结构和一组工具，专为使用 Ruby 语言的用户开发，允许你快速建立 Ruby 系统原型。它不是 Ruby 语言中唯一的框架，但它一定是最有开发效率和最有名的。另一方面，如果你已经花了很大功夫学习 PHP 的话，那么从 Ruby 重新开始又要重头学起。

为 PHP 开发的框架有很多个（大约 40 个），CI 只是其中之一。其它的还包括 Zend Framework、Cake、Trax 等。下列网址可以找到一个针对十种框架的简明图表分析：

<http://www.phpit.net/article/ten-different-php-frameworks/>。

如果你访问上述网址中相关产品的官方网站，你将会注意到，每个论坛都有一个共同的热点，就是到底哪一个框架是最好的？事实似乎是每个都有它的长处，而且又都有自己的弱点。我的评估标准是：我很忙；因此框架应该节省我的时间，从中选择一个后，就坚持使用下去，因此就有了这本介绍 CI 的书。

## 1.3 关于开发者

Rick Ellis 开发了 CI，他曾经是一个摇滚音乐家，现在是一名程序员。Rick 还是 pMachine 公司的 CEO，该公司还有一个著名的内容管理系统叫做 Expression Engine。2006 年 1 月，他在他的 Blog 中写道 (<http://www.ellislab.com>)：

“我花了数星期时间搜索和安装 PHP 框架，也被它们中的许多打击了一把，令我惊讶的是，我发现大多数框架存在以下问题：文档不全或质量很差。

他们假定你水平很高，希望你能很容易地掌握使用方法。

他们是为那些有超级用户权限，或者有权修改服务器设置的人写的。

他们假定你偏爱命令行操作，事实上许多人无此爱好。

偏爱使用 PEAR 类库或其它开源类库。

模板语法过于复杂。

有的太笨重，有的又太简单。

大多数框架只能在 PHP 5 中运行，只有 5% 的使用率。

我还没有找到一个简单的 PHP 框架，健壮、易于使用、文档完整，包含建立一个完整应用需要的所有工具，并且有一个以浏览器为基础的接口，使用普通用户权限就能安装。没有别的原因，就是“市场需要”这个单一的原因促使我想开发这样一个框架产品。……”

结果是 CI 诞生了，作为一个业余时间开发的作品，Rick 慷慨地决定使它成为开源作品。在跑生意间隙，他保持经常更新 CI。他也创建了一个优秀的论坛，CI 使用者能提出问题并且分享开发心得。所有这些资源可从下列网址获得：

<http://www.codeigniter.com/>。

他能实现自己的目标吗？相信你使用后会得出自己的结论……

## 1.4 “开源”商业模式

这类软件可能会有一些让人感到困惑的地方。如果你喜欢你的软件与昂贵的支持合同和“大公司”联系起来的话，那么 CI 并不适合你。（但是，你使用 PHP 能做什么呢？PHP 的用户都知道，支持和 PHP 软件的开发，一定程度上依赖于“社区”数百或数千用户的义务劳动。）

社区支持也存在一些问题。一致性和高质量不是“有保证的”——任何人都可以发表到论坛上，有时这些发表的内容是完全错误的。（注意：如果你去看一下一些昂贵的商业软件的授权细则，也是不对产品质量做出保证的。）但是对于“开源”产品而言，则必须自行细加推敲，而不能一味的相信论坛上的表象言论。如果你对研究新事物饶有兴致，那么 CI 非常适合你！

然而，所有明智的开发者都想把时间和精力投入到“一个人的团队”的产品中是否明智。Rick Ellis 利用业余时间编写

这个项目，并从他在 pMachine 的同事 Paul Burdick 那得到了一些帮助。它是免费的。他不对维护或开发 CI 做出任何承诺。他可能回去继续做一个摇滚音乐家。

另一方面，下载 CI 之后，您下载的版本将继续工作。您不必依赖升级和修补程序。Rick 的代码写的非常好，只有几个严重的 Bug。如果 CI 工作正常，你便没有理由不继续使用 CI。到目前为止，我只发现了两个导致我的代码出错的情况，是这个框架而不是我的代码上的 Bug。（这两个 Bug 已被修复。）

CI 的网站是社区和论坛的门户。

## 1.5 CI 不能做什么

CI 有它本身的缺点。Rick 把 CI 定义为小型“轻量级”框架。（1.5 版压缩后只有 737 KB 可以在几秒钟内下载完毕。Zend Framework 是 10 MB）CI 不能解决你所有的问题。但它能够：

- 使 PHP 编程更容易更快速。

- 帮助你架构网站或使你更容易地设计架构。

作为“轻量级”框架的一个结果是：它没有它的对手所具有的许多特征。像 Rails 因为它包含“脚手架（scaffolding）”和“代码生成器”，因此可以为你编写一些基本的脚本代码。因此，举例来说，一旦你建立了一个数据库，Rails 能自动生成简单的 CRUD 脚本（创建、读取、更新和删除）。

除此之外，Rails 还能让你编写“代码生成器”—自动地编写其他的简单脚本代码。Rails 社区中有许多这样的例子，因此你可以做很多智能化的东西。

CI 不这样做。（有基本的“脚手架（scaffolding）”功能—在 CI 中，脚手架只给开发者使用。就像在线手册描述的一样：“脚手架安全性不够……如要使用脚手架的话要确保在使用后立即关闭这个功能。在实际运行的网站上不要让脚手架处在工作状态。”说得很明确了把？）

相反地 CI 专注于使基本的东西更容易。它处理的一些事物是：

- Session 管理和 Cookie。（见第六章）

- 数据库访问和查询。（见第四章）

- 创建 HTML 相关内容，如页面和表单，并验证表单项目。（见第五章）

- 测试。（第八章）

- Internet 通信，使用 FTP 或 XMLRPC。（第九章）

很熟悉吧？这些全部是基本的功能，如果你正在创建一个动态网站，你一定会做这些工作。CI 使这些工作更容易，而且使你的代码尽可能更好地工作。

## 1.6 许可协议

如果你正在构建一个商业应用程序，那么使用的任何软件的许可协议都将是至关重要的。（如果你要筹集风险投资，那么让 VC 的律师去对其进行详述）CI 没有这方面的问题。CI 的许可协议非常宽松，许可协议文件随 CI 一起在下载回来的压缩包里。

不像我所知道的某些商业软件，CI 的许可协议一屏就可以显示出来。下面的屏幕截图就是：

## 1.7 总结

如果你已经掌握了 PHP 的基本知识，并且想“聪明”地编写动态网站脚本，CodeIgniter 框架会使你的工作更容易，它帮助你：

- 节省时间。

- 使你的网站更健壮。

- 帮助你编写更复杂的系统。

CI 使你更好地享受编程乐趣，而不是一个干苦活的体力工。

有相当多的框架并不是为 PHP 语言开发的。他们都能减少重复编码的工作，使编写复杂程序变得更容易，并且建立一个合理的系统架构。

本书并不想制造框架大战。文章中已经解释了选择 CI 理由，让它为你节省更多的时间用在学习工作和生活中吧。

本书介绍了框架的一些主要特性，并对一些框架内部的运作进行了解释。

我通过对一个真实程序代码的分析，来展示 CI 是一个可以快速简便的用于艰巨环境中的重要工具。

请享用它！



## 第二章 2 分钟：建立一个 CodeIgniter 网站

用 CI 建一个网站很容易。这一章很短，解释了用 CI 制作网站时发生了些什么，哪些文件被创建，让我们来瞧一瞧：

创建网站需要什么软件？

安装 CI 文件：一个简单的下载和解压缩操作。

CI 的基本设置：有哪些文件夹及它们是如何组织的。

CI 安装时默认的控制器和视图。

一些简单的修改来演示 CI 如何运作。

### 2.1 准备知识

CodeIgniter 有较好的版本兼容性。它工作在 PHP 4.3.2 及以上版本，或 PHP 5。由于大多数 ISP 还不支持 PHP 5，支持 PHP 4 版本是有用的。

你还需要一个数据库。CI 的在线手册说：“已支持的数据库是 MySQL、MySQLi、MS SQL、Postgre、Oracle、SQLite 和 ODBC。”

为了要开发并测试一个动态的网站，你需要一个 Web 服务器。通常，你会在本地服务器上开发并测试你的网站，也就是，这些软件运行在你自己的机器上（127.0.0.1 或 localhost），一般来讲，开发环境不会建立在远程服务器上。

如果你不熟悉如何分别建立本地开发环境，可以选择一个套装软件，像是 Xampplite，安装 Apache、PHP 和 MySQL 几乎不需要修改配置文件。Xampplite 是免费的，有简单易懂的安装指南。或者，某些版本的 Windows 附带自己的 Web 服务器。

你还需要一个称心的 PHP 编辑器。所有的编码工作都可以在文本编辑器中完成。提供语法加亮特性和自动补齐功能的编辑器会更理想一些，因为它可以帮助一般水平的程序员节约时间。

一旦你做好了这些准备工作，我担保你在 2 分钟内就可以搞定 CI 安装工作

### 2.2 安装 CodeIgniter

再次声明，CI 是完全免费的！

建立好开发环境后，访问 CodeIgniter 网站：

<http://www.codeigniter.com/> 并下载最新版的框架。1.5.3 版是最新版（译注：编写此书时的最新版是 1.5.3），是一个只有 737KB 的压缩文件，几秒钟就可以下载完成。

解压缩这个文件，把它释放到网站根目录中。如果你正在使用 Xampplite，通常在 Xampplite 文件夹里面的 htdocs 文件夹中。

CodeIgniter 的 index.php 文件应该在根目录中。这时，如果你在浏览器上打开 <http://127.0.0.1> 你也就实际打开了此文件。我们用 1-2 分钟的时间来建立一个可运行的网站！

和 CI 包含在一起是一个简单易懂的用户手册。（在 user\_guide 文件夹中）你将会经常用到它。它的内容很详细，细过这篇文章，所以，需要时，请经常查阅它。

当这些文件保存在你的机器上的时候，有两个方法来访问它们：

通过 URL，<http://127.0.0.1>

经过正常的目录路径：例如

C:/xampplite/htdocs/index.php

你应该通过浏览器访问 CI 的默认首页。真是简单！默认首页传递给你一个信息：你将看到的内容由两个文件组成：视图和控制器。

### 2.3 分析文件结构

安装 CI 文件后，我们来看一下目录结构。

你的根文件夹现在应该看起来有点像下面的图表。如果你曾经看过 Rails，这个结构将会看起来非常熟悉。

你能把这些文件夹分为三个小组：

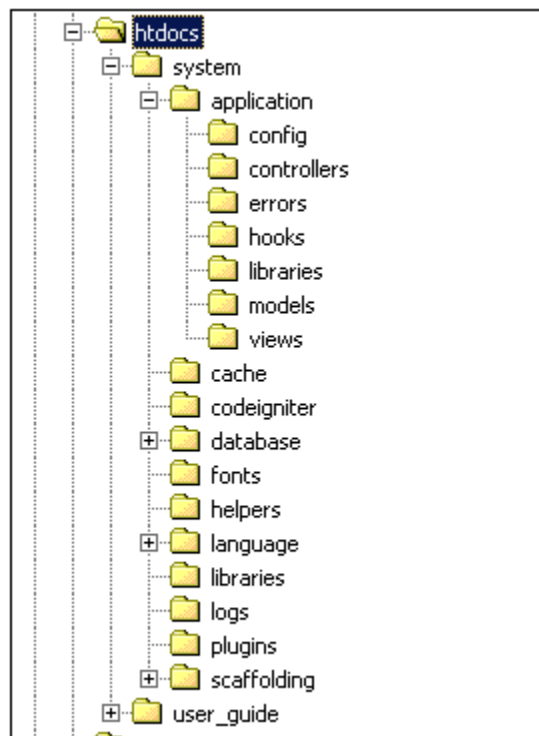
application 是你自己的项目存放文件的目录（举例来说，控制器、模型和视图：全部在 application 文件夹中）。除了你刚才见到的 welcome 视图和控制器，这些文件夹都是空的。

在 system 文件夹中的文件是 CI 本身的代码。

（system/libraries、system/codeigniter、system/drivers 等）。如果你愿意，你可以研读它们，或者改变他们——不过要等到你了解了 CI 是如何工作的，才能这么做。而且如果你改变了框架内的代码，记住，当你下载了 CodeIgniter 新版本时，备份

它们，否则，新的版本会覆盖它们。当然，你也可能不需要自己修改代码而直接使用 CI 本身的代码，Rick 写的代码应该是很不错的。

还有一些文件夹中已包含了文件，但是，可能需要增加或修改（如：language、config、errors）。这些文件夹被设置为默认的，但你可以修改它们。



### 2.4 配置文件

还记得我们要花 2 分钟建立我们的网站吗？第 2 分钟要用来做一些基本的设置。

config 文件夹包含了一些为你的网站设定基本配置的文件。打开 config/config.php 文件告诉网站应该在哪里找到它自己的结构和配置信息。文件的第一行一般是这样的：

```
/*
|-----
|
| 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/";
```

注意 CI 的注释多详尽啊！

修改引号中的数据以匹配你网站的根目录。如有疑问，请查询在线手册以得到详细指导。

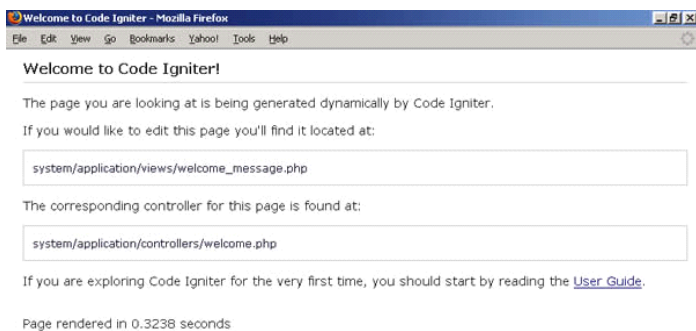
作为一项基本的原则，使用 config.php 文件储存关于你网站的信息好过散布在你项目的不同文件中。这样做有几个好处：首先，更新比较容易；其次，当你把项目从开发服务器转移到实际存放的服务器时，修改配置较容易；最后，许多 CI 函数会首先在配置文件中寻找需要的信息。

还有其他的 config 文件存放在 config 文件夹中，但是目前你可以放心地使用它们而不用修改它们的默认值。

这就是 2 分钟内运行 CI 所需要做的第二件事儿。在这一章的余下部分，我们将会上我们刚做好的网站去逛逛。

## 2.5 它能工作吗？

验证网站能否正常工作的一个简单方法就是打开你的浏览器。假定你正在本地服务器的根文件夹中运行它，在地址栏输入：  
`http://127.0.0.1`，你能看到网站的默认页面：



看到默认页面意味着你的网站正常运行了。不需要 2 分钟，对吗？

## 2.6 总结

在本章节中，我们已经见到，安装 CI 是多么容易。一旦建立好你的开发环境，你所需要做的是下载 CI 框架文件、解压并复制到一个目录而已。

随后，我们快速浏览了 CI 的目录结构。

这一章节非常短，因为 CI 容易安装，不需要太长的篇幅。其实其它章节也不长，因为 CI 的确很简单易懂，节约时间。

## 第三章 分析网站结构

既然我们已经安装了 CI，那我们就开始了解它是如何工作的吧。

读者已经知道 CI 实现了模型—视图—控制器（MVC）模式。这是管理文件和网站的方法，如果你喜欢的话你可以把他们合理的分成各个小模块，而不是把代码放在一起。

这一章，我们将会对 MVC 理论做个简短的介绍，然后再介绍 CI 的 MVC 实现方式。特别地，要了解那些文件夹是如何相互交换信息的？网站结构是怎样的？以及 CI 是如何运作的？

这一章将会介绍：

MVC 如何架构一个动态网站

CI 如何分析一个 Internet 请求，以及如何调配指定的代码来处理它

这些指定的代码如何编制

CodeIgniter 语法规则

在 CI 中，你可以找到或自己编写各种文件和类

如何使用 URL 传递参数给控制器

如何编写更好的视图并把动态内容传递给它们

如何返回信息给上网者

文件和类如何传递信息和相互调用

辅助函数和类库文件有什么用

有助于网站设计的一些特别提示

### 3.1 MVC—到底有什么用？

MVC 指的是一个动态网站的组织方法。该设计模式是 1979 年由挪威人 Trygve Reenskaug 首次提出来的，这里是一些概要：

模型是包含数据的对象，他们与数据库交互，对这些数据进行存取，使其在不同的阶段包含不同的值，不同的值代表了不同的状态，具有特定的含意。

视图显示模型的状态，他们负责显示数据给使用者。（虽然他们通常是 HTML 视图，但是，他们可能是任何形式的接口。比如 PDA 屏幕或 WAP 手机屏幕）

控制器用来改变模型的状态，他们操作模型，提供动态的数据给视图。

CI 中模型、视图和控制器文件都有自己的文件夹。文件本身是 .php 文件，通常以遵循某种命名规则的类的形式呈现。

CI 帮助你遵循 MVC 模式，使你更有效地组织代码。CI 允许你有最大的灵活性，你可以获得 MVC 结构的所有好处。

当你编程的时候，试着始终用 MVC 来思考问题。尽可能确保你的“视图”聚焦于显示内容，“控制器”纯粹地用来控制应用程序流。把应用程序逻辑保留在数据模型和数据库中。

这样，如果你决定开发新的视图，你不必在任何一个控制器或模型中修改代码。如果你要更改“商业逻辑”，那么你只需要在模型中修改代码。

另一方面，你必须认识到，MVC 只是用来帮助你的一种设计方式，而不是用来约束你的。MVC 可以有不同的实现方式。CI 的论坛中包含许多如何“正确合理”的实现 MVC 的方式。（我应该在控制器部分实现数据库查询功能吗？我能直接从视图发送数据到模型吗？或者我必须通过控制器来访问？）

与其寻找理论上的正确方式，不如遵循两项有用的原则。这些在 CI 用户手册的设计和架构目标一节中有相关描述：

——松耦合：耦合是指一个系统的组件之间的相关程度。越少的组件相互依赖，那么这个系统的重用性和灵活性就越好。我们的目标是一个非常松耦合的系统。

组件专一性：专一是指组件有一个非常小的专注目标。在 CodeIgniter 里，为了达到最大的用途，每个类和它的功能都是高度自治的。

这些是 Rick Ellis 开发 CI 要实现的目标，并且它们对于你的网站也是很好的目标。实现这些目标之后，你在代码中使用这些类时就不需要担心有什么副作用了。

CI 做到了这一点，我的经验是一个站点中的“松耦合”辅助函数和类库可以很容易的应用到其他站点中，这节省了很多开发时间。

因此，如果你的控制器直接操作数据库，或你在模型中调用视图，CI 代码将会以适当的方式工作—通常没有技术上的问题—但是从 MVC 理论来看这样似乎是“不正确的”。别担心，如果喜

欢就这样做吧！

### 3.2 CI 的网站结构：控制器和视图

你的整个 CI 网站是动态的。就是，可能找不到制作“静态”网页的简单 HTML 代码。（如果需要你可以添加，但是他们将会在 CI 框架之外。）那么，你的网站到底在哪里？

当我们安装 CI 的时候，我们注意到 application 文件夹中包含名为 models、views 和 controllers 的子文件夹。每个 CI 网站都包含这三个类型的文件。

让我们看看内部细节。

再次强调我们并不处理静态网页和对应的 URL，我们将会给你看 CI 如何分析“URL”请求和如何响应它。首先，考虑一个正常的 Internet 请求。用户建立一个连接到你的网站：

www.example.com，然后经过 socket 发出一个如下的 HTTP 请求：

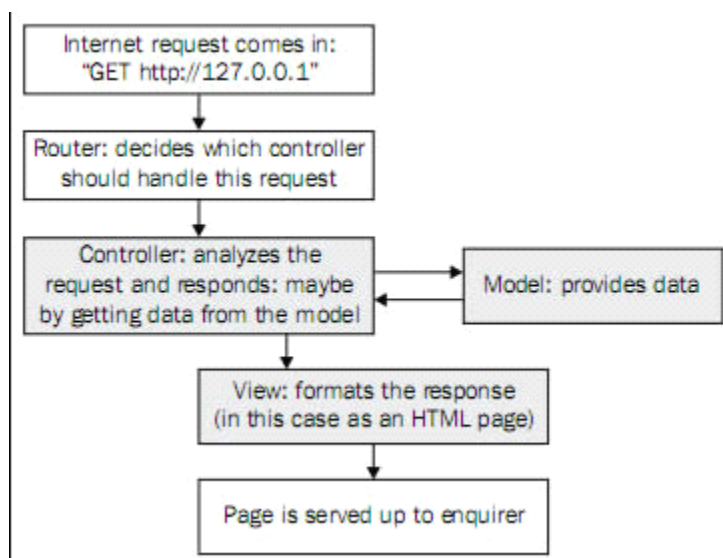
GET /folder/file.html HTTP/1.0

GET 是请求的类型，HTTP/1.0 指定 HTTP 协议的版本，中间是相对路径和文件名。但是在你的网站上，找不到简单的静态 HTML 文件。取代它的是，所有的连入请求被 index.php 文件拦截并进行处理。

如果用户正在以正确的 URL 请求你网站上的页面—通常是单击网页上的一个超链接—请求一般看起来像这样：

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

如果使用者不知道具体的 URL，CI 会设定一个默认页面（我们一会儿就告诉你怎么做。）。CI 的处理步骤是：



一个从 Internet 到你的网站根文件夹的请求被 index.php 文件拦截，作用就像一个“路由器”。换句话说，它调用一个“控制器”，然后返回一个“视图”。

“路由器”怎么知道调用哪个控制器？就像我们已经见到的，有时候请求本身包含的信息会告诉它调用哪个控制器。举例来说，如果请求说：

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

并且如果你有一个控制器叫做 welcome，这就是被调用的控制器。

#### 3.2.1 Welcome 控制器

所以，让我们看 welcome 控制器。它被存放在如下路径：system/application/controllers/welcome.php。它的内容是这样的：

```
<?php
class Welcome extends Controller
{
    function Welcome()
    {
        parent::Controller();
    }
    function index()
```



```

{
    $this->load->view('welcome_message');
}
?>

```

文件的第二行开始是一个类。每个控制器从一个 Controller 类继承。在类中包含了两个函数或方法—welcome() 和 index()。

CI 要求控制器名称首字母大写 (class Welcome)，但文件名是小写字母：

/system/application/controllers/welcome.php。

接下来的三行代码组成构造函数。注意到 CI 使用 PHP 4 构造函数命名规则，兼容于 PHP 5—CI 在 PHP 4 和 PHP 5 中都能很好的工作。构造函数在类实例化时被调用，你可以做一些初始化的工作，比如调用类库或模型，或者定义类的成员变量。

这个例子的构造函数中只有一行代码，它调用父类的构造函数：parent::Controller()。这只是显式的使用父类功能的一种方法。如果你想要详细了解 CI 中的 Controller 类，你可以查看文件 /system/libraries/controller.php。

(可以放心的是你可以随时查看 CI 的源代码，它们已经保存在你机器上的某个文件夹内。)

### 3.2.2 让视图开始工作

让我们回到连入请求那部分。路由器不仅要知道应该由哪个控制器来处理请求，而且也要知道是哪个控制器里面的哪个函数。这就是为什么请求是特定的 GET

http://127.0.0.1/welcome/index 的原因。所以，路由器在 welcome 控制器中查找一个名为 index 的函数。你要确保存在 index 函数！

来看看 index() 函数。这个函数只是用 CI 的装载函数

(this->load->view) 载入的一个视图 (“welcome\_view”)。在现阶段，它不对视图做任何操作，只是传递动态信息。稍后才会执行。

“welcome\_view” 在 CI 中有特定的含义，实际上它指向了如下文件：system/application/views/welcome\_view.php。这个视图文件中只是一些简单的 HTML 代码，但是，因为在稍后运行时，CI 会向文件里放入 PHP 编码，因此使用了 PHP 文件后缀。(如果只是使用简单的静态 HTML 则不需要修改后缀。)

下面是视图文件中的 HTML 代码 (作了精简处理)：

```

HTML 代码
<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>

```

正如你所见到的，它完全由 HTML 组成，内置 CSS 定义。在这个简单的例子中，控制器还没有传递任何变量到视图中。

### 3.2.3 默认控制器

前面提到，如果在请求中没有指明具体的控制器，CI 将会把页面重定向到一个系统默认的页面。这个默认页面可以自己设定，它存放在如下地址：/system/application/config/routes。该文件中包含下列设置：

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

如果你不在此设定默认值，不明确的 URL 请求会转到 “404 not found” 页面。

本例中，默认路由是你的 welcome 控制器。

如果没有指定函数，/index 会被默认选中。因此，如果只是要避免显示 “404” 页面，请确保你有一个 index() 函数。请注意，index 函数和构造函数并不是一码事。

你可以根据需要修改此设置，你还可以修改一个函数叫做 \_remap(\$function)，其中 \$function 是你重定向的控制器。\_remap 总是先被调用，不管 URL 是什么内容。

## 3.3 CodeIgniter 语法规则

在我们开始学习之前，让我们简单的归纳一下 CI 的语法规则。框架希望文件按一定的规则设置，否则它可能无法准确的定位和使用它们。

### 3.3.1 控制器

这是一个类 (也就是 OO 代码) 它由 URL 直接调用，例如：

“www.example.com/index.php/start/hello”。控制器使用函数名来调用函数，如：mainpage(); 不过，你不能够在一个控制器内调用其它控制器内的函数。

语法：控制器用如下格式定义：class Start extends controller (控制器名称的首字母必须大写)，并保存为 .php 文件，位于如下文件夹中：/system/application/controllers。文件名的首字母不需要大写，应该是 start.php 而不是 Start.php。还有，在构造函数中至少要有如下内容：

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

所有的其他代码一定要写在不同的函数中，例如：hello() 函数。

### 3.3.2 视图

视图是包含 HTML 代码的，带有 .php 后缀的文件。使用如下命令装载：\$this->load->view('testview', \$data)。这条命令载入并使用视图。

语法：视图用 HTML 编写，PHP 代码包含在 <?php ?> 标记中，在 view 文件夹中保存为 .php 文件。

## 3.4 CI 网站上的文件或类的类型

在 application 文件夹中有许多子文件夹。我们已经讲过了 controller、config 和 views 文件夹。

但是，什么是 libraries、models 和 scripts? 这是 CI 中似乎让人相当困惑的一个区域。(如果你用过 1.5 版以前的 CI，那么你会了解这是为什么。Rick Ellis 对早期版本感到不满意，并大幅度修改了结构。因此，为了向前兼容，一些文件夹结构必须保留。)

从技术角度看，要平等地对待这些文件夹。你为什么要把代码放在这个文件夹而不是那个文件夹是没有什么理由好讲的，这就是一种约定。

这样讲吧，假定你已经写了一段代码并命名为 display，里面包含一个函数叫做 mainpage。实现它有四种方法：模型、类库，辅助函数或插件。下列的表格列举了你如何装载和使用每一种方法。

文件	如何使用

类型	
模型	<p>是一个类（面向对象代码）</p> <p>装载方法: <code>\$this-&gt;load-&gt;model('display');</code>;</p> <p>使用方法: <code>\$this-&gt;display-&gt;mainpage();</code>;</p> <p>语法提示: 必须以如下格式开始: <code>class Display extends Model</code></p> <p>必须包含构造函数:</p> <pre>function display() {     parent::Model(); }</pre> <p>必须包含一个 <code>mainpage()</code> 函数</p> <p>概念性总结: 用户指南这样描述: “模型是设计用来表示数据库中信息的 PHP 类。”</p>
类库	<p>存放在 <code>system</code> 和 <code>application</code> 文件夹中。</p> <p>也是一个类（注意: 你自己的类库不会自动地包含在 CI 超级对象中, 因此你必须用不同的方法来调用, 详见第七章）</p> <p>装载方法: <code>\$this-&gt;load-&gt;library('display');</code>;</p> <p>使用方法: <code>\$this-&gt;display-&gt;mainpage();</code>;</p> <p>语法提示: 不需要从父类继承, 不一定需要构造函数。</p> <p>这样足够了:</p> <pre>class Display() {     function mainpage()     { //code here } }</pre> <p>概念性总结: 如果你想扩充 CI 的功能, 或者创建特别的功能可以使用类库。</p>
辅助函数	<p>可以保存在 <code>system/helpers</code> 或 <code>application/helpers</code> 文件夹中。是一段脚本（过程式代码, 不是 OO 类）</p> <p>装载方法: <code>\$this-&gt;load-&gt;helper('display');</code>;</p> <p>使用方法: 直接调用, 如: <code>mainpage();</code>;</p> <p>语法提示: 文件必须被保存为 <code>display_helper.php</code> (文件名必须加上 <code>_helper</code>)</p> <p><code>mainpage()</code> 必须作为一个函数包含在这个文件中, 整个文件就是过程式函数的集合。因此你不可以在这个文件的函数中调用 CI 的其它资源。</p> <p>概念性总结: 辅助函数是用来适度帮你实现特定目标的低级函数。</p>
插件	<p>保存在 <code>system/plugins</code> 文件夹中, 也可以保存在 <code>application/plugins</code> 文件夹中。是一段脚本（不是 OO 类）</p> <p>装载方法: <code>\$this-&gt;plugin('display');</code>;</p> <p>使用方法: 直接调用, 如: <code>mainpage();</code>;</p> <p>语法提示: 保存在如下格式的文件中: <code>display_pi.php</code> (文件名必须加上 <code>_pi</code>)</p> <p><code>mainpage()</code> 必须作为一个函数包含在这个文件中, 整个文件就是一个过程式函数的集合。因此你不可以在这个文件的函数中调用 CI 的其它资源。</p> <p>概念性总结: 用户指南如此写道: “…插件和辅助函数的最大不同就是插件通常用来提供一个单一的函数, 辅助函数是一个函数的集合…插件一般用于社区内互相分享代码。”（详见第十五章: 一个插件的例子）</p>

你可以自由地选择使用这四种方法中的任一种或几种, 当然, 如果你选择模型或类库的话, 那就必须使用 OO 编程方式。辅助函

数和插件必须使用过程编程方式。后者不能直接引用其他的 CI 类。另外, 不同文件夹的区别主要是由 CI 设定的。

你应该注意到 CI 有两组辅助函数、插件和类库, 而模型只有一组, 前三种中, 一组放在 `application` 文件夹中, 另一组放在 `system` 文件夹中, 这两组的区别主要也是 CI 设定的。

那些在 `system` 文件夹中的代码是 CI 的核心代码, 所有的项目都会使用。如果你升级到一个新版 CI, 这些文件会被修改。

那些在 `application` 文件夹中的代码只能用于一个应用程序。如果你升级到一个新版 CI, `application` 文件夹不会被覆盖。

当你装载一个辅助函数、插件或类库时, CI 会在两个文件夹中查找, 比如你要装载一个类叫做 `display`, CI 会先查找 `system/application/libraries` 文件夹。如果在这个文件夹中不存在, CI 会寻找 `system/libraries` 文件夹。

这意味着, 可以通过把同名的文件放入 `application` 文件夹来取代 CI 核心的类库、辅助函数和插件。不要轻易尝试这样做。因为这种高度的灵活性需要你足够多的使用 CI 的经验。如果你想扩充 CI 基本类和脚本的功能, 请参考第十三章。

### 3.5 这些文件夹的含义?

在对系统各文件夹进行了一番介绍后, 我们通过下表来总结一下各文件夹的作用。

application	<table> <tr> <td>config</td><td>配置文件: 包含网站的基本配置信息</td></tr> <tr> <td>controllers</td><td>控制器</td></tr> <tr> <td>errors</td><td>包含出错信息页, 你不必修改这个文件夹</td></tr> <tr> <td>hooks</td><td>首次安装时空, 用来存放你创建的“钩子”。钩子是用来装载其它文件的控制方法。</td></tr> <tr> <td>libraries</td><td>代码库, 针对本项目的专用代码</td></tr> <tr> <td>models</td><td>代码库, 也是针对本项目的专用代码</td></tr> <tr> <td>views</td><td>要显示信息的模板</td></tr> </table>	config	配置文件: 包含网站的基本配置信息	controllers	控制器	errors	包含出错信息页, 你不必修改这个文件夹	hooks	首次安装时空, 用来存放你创建的“钩子”。钩子是用来装载其它文件的控制方法。	libraries	代码库, 针对本项目的专用代码	models	代码库, 也是针对本项目的专用代码	views	要显示信息的模板
config	配置文件: 包含网站的基本配置信息														
controllers	控制器														
errors	包含出错信息页, 你不必修改这个文件夹														
hooks	首次安装时空, 用来存放你创建的“钩子”。钩子是用来装载其它文件的控制方法。														
libraries	代码库, 针对本项目的专用代码														
models	代码库, 也是针对本项目的专用代码														
views	要显示信息的模板														
cache	第一次安装时空, 如果你打开缓存设置（详见第十章）, 该文件夹存放缓存数据														
codeigniter	基本系统文件														
database	CI 的数据库类库文件														
fonts	没有在用户指南中介绍, 存放水印图像使用的字体														
helpers	系统级“辅助函数”														
language	你可以存放你本国语言的键名列表—详见第十一章														
libraries	系统级类库														
logs	如果你需要系统记录错误, 那么日志文件默认保存在这个文件夹中														
plugins	更多的系统级代码块														

scaffolding	系统级类库，实现简单的“脚手架”功能
-------------	--------------------

### 3.6 设计一个较好的视图

在现阶段，你可能要问我们，为什么这么努力的去为一个简单的 HTML 页面服务？为什么不把所有的东西放在一个文件里？对于一个简单的网站来说，这个观点是对的，但是现在谁还听得到简单的网站？CI 中最酷的一点就是它能始终保持内部结构的一致，架构清晰，易于维护。

一开始，我们需要三个步骤：

编写一个视图页面

编写一个样式表

更新我们的 config 文件以指定样式表在哪里

在这三点做完之后，我们需要更新控制器接受从 URL 传来的参数，把变量传给视图。

首先，让我们重新设计视图并把它保存到如下路径：

system/application/views/testview.php

#### HTML 代码

```
<html>
<head>
<!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>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>
```

它基本上仍然是 HTML，但是注意已高亮的 PHP 代码。

你将会注意到有一个 PHP 代码片断存放在样式表中。让我们在站点根文件夹下保存一个简单的样式表为 mystyles.css。它很简单：

#### CSS 代码

```
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;
}
```

这给了我们两种可选风格，而且你将会在视图中用到它们。

首先，让我们在 config 文件加入：

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

这只是告诉网站我们刚才编写的 CSS 文件的文件名和位置。

但是注意样式表的实际位置是 \$base/\$css——从哪里获取变量 \$base 和 \$css 的内容呢？你还可以联想到，变量 \$mytitle 和 \$mytext 的内容呢？答案是我们需要一个新的控制器！

### 3.7 设计一个较好的控制器

现在，我们需要一个新的控制器。我们将其命名为 Start 并保存在：/system/application/controllers/start.php

该控制器必须做几件事：

调用视图

将基本 URL 信息和刚编写的 css 文件的名称传递给视图

把另一些数据传递给视图：它正在期待标题 (\$mytitle) 和一些本文 (\$mytext)

最后，我们想要控制器接受来自使用者的一个参数（例如通过 URL 请求）

换句话说，我们必须传递变量到视图中。因此让我们从 Start 控制器开始。这是一个 OO 类：

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

在这里请注意，我们已经声明了 \$base (网站的根地址) 和 \$css (css 文件名) 为变量或类的属性。如果我们在每个类中要编写超过一个函数，这样做会很方便。你也可以把它们定义为一个函数内的变量，如果你硬要这么做也可以。

构造函数现在可以通过查找 config 文件来定义我们刚声明的变量了。要实现这个功能，我们可以做如下定义：

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

当作在：

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

CI 会从 config 文件中读取相关的设置内容。

使用这一机制，不管我们编写多少个控制器和函数，如果我们的网站访问量很大，我们需要搬迁服务器的话，那么这些参数也只需修改一次。

#### 3.7.1 把参数传给一个函数

现在，在 Start 控制器类里面，让我们定义将会实际工作的函数。

```
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);
}
```

这个函数期待一个参数，\$name，（但你可以设置一个默认值——myfunction(\$myvariable=0)），通常会把一个字符串赋给 \$mytext 变量。好吧，我们现在要问一个问题，\$name 变量来自哪里？

在这个例子中，它需要来自 URL 请求的第三个参数。因此，它由 HTTP 请求提供：

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

换句话说，当你输入 URL：

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

注意：这个例子中的代码“干净地”传递了变量 fred，没有用任何方式检查它。你需要在编程时对它进行检查。我们将会在第七章学习到如何检查输入。通常，参数必须在检查后确保没有问题再传递给控制器。如果不检查的话，一个不怀好意的用户可以很容易地侵入系统，他可以发送这样一个 URL 如：

http://www.mysite.com/index.php/start/hello/my\_malicious\_variable。所以，你应该检查接收的变量，确保它们符合要求再进行处理。

URL 的最后一段作为一个参数传给函数。事实上，你能增加更多的参数，但不能超过你所使用的浏览器的设置。



让我们总结一下 CI 处理 URL 的具体细节：

URL 段	用途
http://www.mysite.com	定位你网站的基本 URL
/index.php	定位 CI 路由器并读取 URL 的其它部分，分析后定们到相关页面。
/start	CI 要调用的控制器的名称（如果没有设置控制器名称，CI 将调用你在 config 文件中设置的默认控制器）
/hello	CI 要调用的函数名称，位于所调用的控制器内。（如果不存在该函数，默认调用 index 函数，除非你使用 _remap）
/fred	CI 把这个作为传递给函数的变量
如果还有其他 URL 段，例如/bert	CI 把这个作为传递给函数的第二个变量
更多变量	CI 把更多的参数作为变量传递给函数

### 3.7.2 传递数据到视图

让我们回去再看一下 hello 函数：

```
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);
}
```

注意 hello() 函数如何先设置一个名为 \$data 的数组，并把一些对象的属性及文本读入数组。

然后它通过名称装载视图，并把刚生成的 \$data 数组作为第二个参数。

在幕后，CI 很好地利用了另外一个 PHP 函数：extract()，这个函数的作用是把数组中的元素取出放入变量表，其中每个键值对中的键名即为变量名，对应该键名的值为变量的值。因此我们刚才定义的 \$data 数组在视图中转换成一个单一的变量：\$text（等于“Hello, \$name, now we're getting dynamic”），\$css（等于来自 config 文件的设置值），等等。

换句话说，当它被建立的时候，\$data 数组看起来像这样：

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

但是当它被传递给视图的过程中，它被解开，并且下列变量在视图对象中生成，与 \$data 的每个键/值相对应：

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

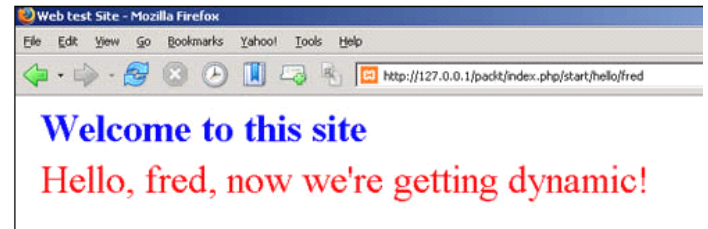
虽然你只传送一个变量到视图中，但是，你能把许多数据装进那个变量中。\$data 数组的每个值还可以是数组，这被称为多维

数组，因此，用一个数组变量可以把大量的变量传递给视图，这是一个很好的编程技巧。

现在访问

http://127.0.0.1/packt/index.php/index/start/fred（注意：这个 URL 跟以前的不同—它要求在 index 控制器中寻找 start 函数，并把参数“fred”传递给该函数），你可以看到如何使用 MVC 架构编写动态网站。（当然，到目前为止至少是 VC，因为我们还没有介绍 M。）

你将看到下列内容：



你可以看到参数 fred 是 URL 的最后一部分，并被传递给函数，再经由函数传给了视图。

请记住你的视图一定要与你的控制器相对应。如果视图中并没有一个变量安排显示的位置，它将不被显示。如果视图正在期待一个变量，而这个变量并没有在控制器中声明并赋值，你可能收到一个错误信息。（当然，如果变量被声明，则不会有错误信息，但它不会正常显示。）

### 3.8 CI 中的类彼此之间如何操控

当你编写你的控制器、模型等内容时，你将会需要在他们之间进行互操作并传递数据。让我们看看如何实现这些功能。

#### 3.8.1 调用视图

我们已经见到控制器如何调用视图并传递数据给它：

首先它创建数组 (\$data) 传给视图；然后它装载并调用视图：

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

#### 3.8.2 直接调用函数

如果你想要使用来自类库、模型、插件或辅助函数的代码，你必须首先装载他们，然后按上面表格里的方法调用它们。因此，如果“display”是一个模型，并且我想使用它的 mainpage 函数，我的控制器可能这样调用：

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

如果函数需要参数，我们可以通过如下方式传递：

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

#### 3.8.3 与控制器互动

你可以在任何控制器内调用类库、模型、插件或辅助函数，并且模型和类库也能彼此调用，同插件和辅助函数一样。

然而，你不能从一个控制器调用另外一个控制器，或从一个模型或类库调用一个控制器。只有两个方法可以让一个模型或者类库与一个控制器关联：

第一，它可以返回数据。如果控制器作如下赋值：

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

函数中返回一个值，那么控制器将得到赋给 \$fred 的值。

第二，你的模型或类库可以创建（并传递给视图）一个 URL。控制器调用相关视图与使用者产生交互。

你不能把超链接直接传递给一个模型或类库。用户总是与控制器交互，从不直接面对其它对象—但是，你能在控制器里写一个调用函数。换句话说，你的视图可能包含一个超链接指向一个控制器函数：

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

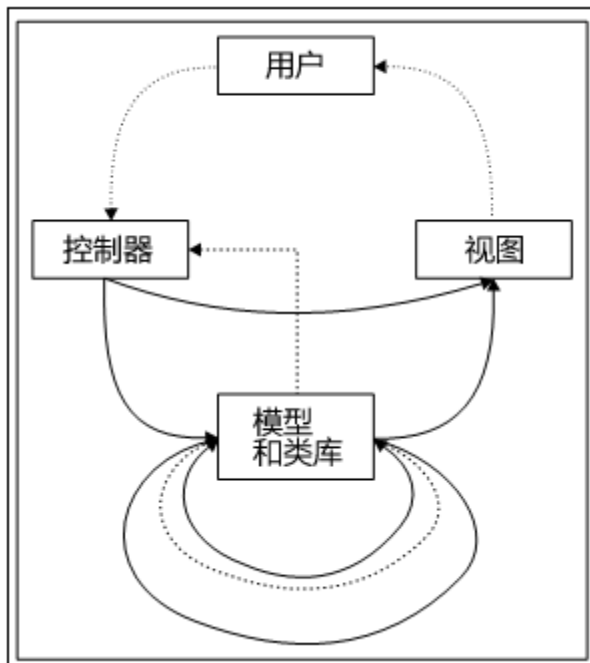
通过 callmodel 函数，调用一个模型中的函数：

```
function callmodel()
{
    $this->load->model('mymodel');
```



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

### 3.8.4 这就像一个鸡蛋杯



这个图显示了组件间信息流的走向。  
实线表示直接函数调用，如：

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

这些信息流可以从控制器到视图，也可以从控制器到类库或模型。（模型也能调用视图，但理论上这样做不合适。）相反方向就不能调用，如：视图不能调用控制器。然而，类库和模型能互相调用。

虚线表示通过返回值传递信息。模型和类库可以在内部互相传递数据，也可以把值返回给控制器。视图不能返回任何值。

虚线表示通过用户传递信息或控制—换句话说，视图会在屏幕上显示一些内容，并可能让用户去单击一个超链接（调用一个控制器）。

这个图很像一个“鸡蛋杯”纯属是一种巧合。它只是看起来像而已。

### 3.8.5 一个 CI 辅助函数的例子：URL 辅助函数

下面介绍一个如何使用辅助函数的例子，使用辅助函数能使你的代码更简洁，更有针对性。CI 的 URL 辅助函数包含一组帮你操作 URL 的函数。你可以像这样装载它：

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

然后，你可以用它查询并返回设置在 config.php 文件中的 site 和/或 base URL：

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

你也可以用它创建超链接。我们在上一节看到过在 start 控制器中访问 hello 函数，并把参数 fred 传递给该函数，这些都是通过一个 URL 实现的：

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

如果你想要你自己的代码创建一个这样的 URL，你可以使用 URL 辅助函数实现。语法是：

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

这将创建一个相同 URL 的超链接，并显示：“Say hello to Fred”。换句话说，它等同于：

HTML 代码

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

请记住，使用 CI 辅助函数有两个好处。第一，输入字符较少。（49 个字符对 82 个字符，均包括空格。如果包括装载 URL 辅助函数的语句—另外的 27 个字符，每个控制器只需装载一次—那么，它还是 76 个字符而非 82 个字符。）

第二，URL 辅助函数自动在 config 文件中查找网站 URL（和 index 文件名）。这意味着如果你改变你的网站 URL，你只需要改变 config 文件一次，你不需要查遍代码修改超链接。

URL 辅助函数还有其他有用的功能。比如：它能创建“mailto”超链接。

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

和下面的 HTML 代码等效：

HTML 代码

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

如果你担心机器人在你的网站上搜集 Email，并用它们发送垃圾邮件的话，那么用 safe\_mailto 代替 mailto。屏幕显示的内容相同，并且工作也正常。

但是如果检查你的 HTML 代码，现在变成了复杂的 JavaScript 代码，机器人也许什么也抓不到（或者无法很容易的抓到）。

JavaScript 代码

```
<script type="text/javascript">
//
var l=new Array();
l[0]='&gt;';l[1]='a';l[2]='/' ;l[3]='&lt;';l[4]='|01';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]='&gt;';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]='|97';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]='&lt;';
for (var i = l.length-1; i &gt;= 0; i=i-1) {
if (l[i].substring(0, 1) == '|')
document.write("&amp;#"+unescape(l[i].substring(1))+";");
else document.write(unescape(l[i]));
}
//]]&gt;
&lt;/script&gt;</code></pre>
</div>
<div data-bbox="517 773 960 830" data-label="Text">
<p>你，和你的用户，不需要看到这些代码。它只是用来迷惑机器人的，以确保你的 Email 地址的安全。而你只需要增加四个字母和一个下划线：你用 safe_mailto 代替 mailto，CI 为你做其它所有的一切。</p>
</div>
<div data-bbox="547 829 930 844" data-label="Text">
<p>在 URL 辅助函数中有许多有用的函数。请查阅用户指南。</p>
</div>
<div data-bbox="517 843 956 871" data-label="Text">
<p>总结一下 URL 辅助函数，它符合我们在这一章开始时讨论的话题：</p>
</div>
<div data-bbox="517 871 960 898" data-label="Text">
<p>高度的“组件专一性”。URL 辅助函数只做它要做的事情，简化对应的编码工作。</p>
</div>
<div data-bbox="517 898 960 940" data-label="Text">
<p>松耦合—URL 辅助函数有一个简单的接口，不依赖调用它的代码。你可以在任何 CI 项目中使用它。例如，大多数项目都需要很多的超链接。你可以使用这个辅助函数一遍一遍地创建它们。</p>
</div>
<div data-bbox="482 967 522 980" data-label="Page-Footer">
<p>- 11 -</p>
</div>
```

如果你查看 URL 辅助函数代码的话（在 `system/application/helpers/url_helper.php`），那么你会发现它是过程式的代码—只是一组函数而不是 OO 类。它不装载任何其他 CI 类或辅助函数。（它本身不是对象，不能这样做。）

### 3.8.6 一个简单的类库例子：创建一个菜单

现在让我们看一些在 CI 框架中实际使用的代码。

举例来说，这里是一个创建三个菜单项的类库文件：

```
1 <?php
2 class Menu{
3     function show_menu()
4     {
5         $obj =& get_instance();
6         $obj->load->helper('url');
7         $menu = anchor("start/hello/fred", "Say hello to Fred
8 |");
9         $menu .= anchor("start/hello/bert", "Say hello to Bert
10 |");
11         $menu .= anchor("start/another_function", "Do something
12 else |");
13     return $menu;
14 }
15 }
```

（这时，不要为不寻常的语法担心—在第 6 行中是“`$obj->`”而不是“`$this->`”。这在第七章会详细介绍。）

注意：这些代码是 OO 代码，函数 `show_menu()` 包含在一个简单的类“Menu”中。它可以访问其他的 CI 类和辅助函数：在这个例子中，它使用 URL 辅助函数，我们刚刚介绍过它。

首先，它装载 URL 辅助函数，然后创建一个字符串（`$menu`），包含链接到三个控制器和函数的 HTML 代码。然后它返回 `$menu` 字符串。

你可以在一个控制器里这样调用它：

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

然后控制器可以使用 `$menu` 变量调用视图：

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

这个类生成一个菜单，对特定网站而言的菜单。正因为这样，我把它保存在 `system/application/libraries`，而不是 `/system/libraries`。它不像 URL 辅助函数那样松耦合，可以在任意的网站上使用。

它具有高度的专一性：它创建一个菜单，我可以在任何控制器中调用它，显示一个标准菜单。

## 3.9 总结

MVC 框架已被广泛地用于构建复杂的动态网站。CI 使用它帮助你编写高度可复用的代码。

当你编写你自己的代码时，要始终保证代码的“松耦合”和“组件专一性”。不要担心你的项目是否严格遵守了 MVC 的理论。关键是理解文件的类型和它们如何彼此交互。然后，你再确定是否使用类库、模型、辅助函数或是插件编写代码。

我们已经了解了 CI 的文件结构，而且知道我们可以随时随地的查看 CI 的源代码，不过我们通常并不需要这样做。

`config` 文件中包含网站的基本信息，可以使我们方便地修改信息，有利于我们快速迁移服务器，减少出错的机会。

我们已经见到了控制器的基本结构，而且用构造函数从 `config` 文件中读取数据并把它存入对象的属性中。我们还动态地把数据从控制器传递给视图。到目前为止，CI 中的一些重要知识已经学习过了。当我们继续学习时，我们会很清楚这些知识是非常重要的。

同时，我们也学习了 CI 组件间传递数据的方式，了解这些对你的编程是很重要的。

最后，我们分析了两个例子，一个是 URL 辅助函数的例子，还有一个是创建“menu”类库的例子。

## 第四章 使用 CI 简化数据库开发

你学习 CI 是因为你想要使编程更容易和更有生产力。这一章讲述 CI 的 Active Record 类。如果 CI 只提供一个 Active Record 类,它还是物超所值的。当然,CI 是免费的,只不过我要强调 Active Record (以下简称 AR) 类的价值是非常高的,它是你提高生产力的主要工具。

AR 使你以最小的代价获得最大的回报。它简单,易于使用和维护。

这一章描述 CI 如何连接到一个数据库,你如何使用 AR 操纵数据库。你将会见到:

AR 类与“经典” PHP/MySQL 接口的比较

如何编写“读”查询并显示结果

如何编写创建、更新和删除的查询

CI 保留让你用传统的方法编写数据库查询,但是我不详细介绍这部分内容。它的知识完全被在线手册覆盖。使用 AR 类后,你可能不会再用传统的方式来数据库查询了。

### 4.1 配置设置

你或许已经注意到在这本书的大多数的章节会谈到 system/application/config 文件夹和里面的 config 文件。这些文件对控制 CI 按要求工作相当必要。而且你可以让大部分的配置参数等于系统的默认值。数据库 config 文件在正常使用数据库之前需要进行设置。

基本上,你仅仅必须告诉它你的数据库在哪里、它是什么类型。文件的默认值为:

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

其他的选项可以保留为默认值。必选项是:

hostname: 你的数据库的位置,举例来说, 'localhost' 或 IP 地址

username 和 password: 使用者名称和密码必须有充分的权限,允许你的网站存取数据库中的数据。

database: 你的数据库的名字,举例来说, 'websites'

dbdriver: 你正在使用的数据库的类型 - CI 可受的有选项有 MySQL、MySQLi、PostgreSQL、ODBC 和 MS SQL

以我的经验来看,最困难的事情之一就是把新的 CI 网站连接到数据库。你可能需要查询你的 ISP。有时他们的数据库运行在与他们的 web server IP 地址不同的地方。如果你正在使用 MySQL,他们可能提供 phpMyAdmin,通常告诉你 hostname-这可能是 'localhost' 或者它可能是一个 IP 地址。

你可能注意到 config 文件的内容实际上是一个多维数组。在 \$db 数组里包含一个叫做 default 的数组,你所做的设置就是往里增加键/值对,例如 hostname = 127.0.0.1。你还可以增加其他的数据库设置,通过改变 \$active\_group 的设置可以容易地更改数据库。

这为网站连接到数个数据库提供了可能性-举例来说,一个测试数据库和一个产品数据库。你可以很容易地在他们之间切换。或者你可以在两个数据库之间交换数据。

### 4.2 为我们的网站设计数据库

我想表达的是 CI 能用来开发正式的网站。我现在正在维护客户的一些网站,而且我想要监控他们,用我设计的方法测试它们,用数据库保存我想要的数据库,而且可以得到这些网站的分析报告。因此让我们试着创建它。先让我们确定一些目标。它们是:用最少的人工干预管理一个或更多的远程网站

对远程网站进行定期的测试

生成符合要求的分析报告,提供网站的细节和测试结果

因此,第一件事情是我们将会需要一个网站的数据库。建立一个名为 websites 的 MySQL 数据库,你也可以使用别的数据库产品。

现在,我们需要增加一些表来保存各种数据。让我们为网站增加一张表,字段有 URL,他们的名字和密码/用户名,和他们的

类型。我们也将为每个网站建立一个 ID 字段。而且在 MySQL 数据库中,至少需要为实体生成一个唯一标识符,可以使用自动增量类型来达到这一目的。

每个网站必须有一个不同的主机,我们需要另一表来保存主机信息。一般有一个域名与主机相关联,所以我们需要一个域名表来保存有关域名的信息,还需要一个人员表来记录这些人的姓名,密码,邮件地址,备用邮件地址,手机号码,曾至宠物的名字,可能还有其它的一些什么。

因此我们的网站表需要包括这样一些字段: domain ID, host ID, 两个 people ID, 一个存放网站站长的 ID 一个存放网站管理人的 ID (管理人为网站提供技术支持,保证网站正常运行。)

你能见到,这是一个完整的关系型数据库,让我们来建立它! (本章的附录中有该数据库的详细资料,如果你想创建这个数据库,请执行该 MySQL 查询。)

我们现在要用一个更简单容易的方法来实现这一切。所以,让我们看看 CI 框架为我们提供了什么功能,我们要重点介绍 AR 类。

### 4.3 Active Record

AR 是一个“设计模式”。另一方面又是一个高度抽象的东西,就象 MVC。它本身不是代码,只是一个模式。对于代码,有一些不同的解释。它的核心是把你的数据库和 PHP 对象建立一个对应关系。每次当你执行一个 QUERY 语句。每一张数据库里的表是一个类,每一行是一个对象。所有你需要做的只是创建它,修改它或者删除它。例如,“方法”,是从类继承而来。Ruby on Rails 是使用 AR 模式的,CI 也是,尽管这两种框架实现 AR 的方式有一点不同。

理论的东西够多了-但是这是什么意思呢?好吧,用代码简单和清楚地描述一下吧。

#### 4.3.1 使用 Active Record 类的优点

AR 节省你的时间,自动运作,使 SQL 语句方便易懂。

##### 4.3.1.1 节省时间

当你在 PHP 编程时,每写一个数据库查询的时候,你每次一定要与数据库建立连接。对 CI 来说,第一次连接数据库时,你在每个控制器或模型的构造函数里放入这样一行语句:

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

一旦你这样做了,你不需要重复连接,在那个控制器或模型就可以做任意多次的查询。

你已经在 config 文件中设置了关于数据库的参数,就象我们这一章开始时看到的一样。再一次,这使更新你的网站比较容易,如果你想要改变数据库名字、密码或位置的话。

##### 4.3.1.2 自动机制

你一旦已经连接到数据库,CI 的 AR 生成隐含的代码。举例来说,如果你进行下列的插入操作:

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

你正在插入的数据已经被在幕后转换成这样一行代码:

```
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;
}
```

换句话说，CI 框架使你的代码变得更强健。现在，让我们看看它是如何工作的。

第一，连接数据库非常简单。在传统的 PHP 编程时，你可能是这样做的：

```
$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]} ";
}
```

换句话说，你必须重复地输入 host、username 和 password，建立一个连接，然后选择一个连接的数据库。你必须每次都这样做，然后再开始一个查询。CI 以一条命令替换连接工作：

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

在每个控制器、模型或者其它类的构造函数里放入这条命令。之后，你就可以直接开始查询了。连接数据被保存在你的数据库 config 文件中，CI 每次都会去那里查询它。

因此，在每个 CI 函数中，你可以直接进行数据库查询。上面的 query 在 CI 中被转换成：

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

很简单，不是吗？

这一个章的余下部分给出不同 query 的用法。

#### 4.3.2 “读取”查询

常用的查询是取回来自数据库的数据。等同于 select 的查询是：

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

这是一个“select \*”查询，目标是 site 表。换句话说，它取回所有的行。如果你偏爱分开来写，你能这样做：

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

如果你想要得到特定的列，而不是全部列，这样做：

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

你可能要对结果排序，你可以在 get 语句前插入：

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

desc 是降序的意思。你也能选择 asc(升序) 或 rand(random 随机)。

你也可能想要限制返回的行数，比如你想要最初五个结果。你可以在 get 语句前插入：

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

当然，在大多数的查询，你不可能在表中返回所有记录。数据库的具有按给定条件过滤返回结果的能力。这通常使用 where 子句来实现，CI 这样表达：

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

这条语句会查找客户号是 1 的客户相连的所有的网站。但是这对我们并不是很有帮助，我们并不会记住人员表的 ID 号。对人来说，用姓名是比较合理的办法，因此我们需要连接到 people 表：

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

用 sites 表中的每个 people ID 去查询这个 ID 在 people 表中的信息。

注意 SQL 的约定，如果一个列名在二张表中是重复的，你需要在列名前加上表名和一个“.”号。因此 sites.peopleid 在位置

桌子中意味着 peopleid 所在的表是 sites。在进行 SQL 多表查询时，最好把列名进行唯一性的标识，这样可以避免产生歧义，也可以让你自己明了。

你可以增加更多的 where 子句的操作符。举例来说，增加否定操作符：

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

或比较操作符：

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

或组合语句（“WHERE...AND...”）：

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

或使用 \$this->db->orwhere()；来表示（“WHERE...OR”）：

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

现在让我们建立一个完整的查询：

```
$this->db->select('url','name','clientid','people.surnam
e 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();
```

这应该给我们前五个（用姓名排序）网站，这些网站属于 3 号客户，而且还显示客户的姓和他或她的身份证数字！

使用 AR 的潜在好处是已经进行了自动的转义，因此，你不必关心转义的问题。这适用于这样的函数象 \$this->db->where()，以及在下一个段中被描述的数据插入和修改语句。（安全警告：这不同于阻止交叉脚本攻击-对付这个你需要 CI 的 xss\_clean() 函数。它也不相同于验证你的数据-对付这个你需要 CI 的验证类，见第 5 章。）

#### 4.3.3 显示查询结果

在 CI 显示查询结果相当简单。假定我们定义了上述的查询语句，最后一句是：

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

然后，如果有多个结果，他们被保存在 \$row 对象中，你可以用一个 foreach 循环：

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

或如果我们只想要一个结果，它可以作为一个对象被返回，或在这里当做一个 \$row 数组：

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

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

我比较喜欢对象语法胜过数组-更简洁！

如果你遵守 MVC 模式，你将会在模型中保存你的查询和数据库交互，然后通过视图显示数据。

#### 4.3.4 “创建”和“更新”查询

AR 有三个函数帮助你在数据库内生成新的实体，它们是 \$this->db->insert(), \$this->db->update(), \$this->db->set()。

“create”和“update”的不同之处是“create”是向表中插入一条全新的记录，而“update”是修改表中已经存在的记录。因此对“update”，你必须首先定位需要修改的记录。

CI 用数组来保存数据，或是使用 \$this->db->set(); 你可以



任选一种。

因此，如果要在 websites 数据库中加入一行。首先，确保在我们的控制器中的构造函数加入：

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

我们想要增加一个新的网站，包含有一个网址，一个名字，一个类型和一个客户 ID。如果用数组的方式，这可能是：

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

把这些信息增加到 sites 表，我们使用：

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

或者，我们也可以使用 `$this->db->set()` 设置每一个值：

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

如果我们正在更新一笔现有的记录，我们也可以创建一个数组或使用 `$this->db->set()`，但是现在有两个不同。

第一，我们必须定位我们想要更新的记录；其次，我们需要使用 `$this->db->set()`，如果我想要在 sites 中更新一笔记录（针对 'id' 列的值 1 的那行记录），使用数组的方式是这样处理的：

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

你也可以使用 `$this->db->set()` 方式，就象前面做过的那样。

CI 提供几个函数检查数据库是否成功执行了相关操作。最有用的：

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

在执行 insert 或 update 后应该返回 '1' - 但是如果我正在进行 update 一批记录的话，可能返回更大的一个整数。

你已经注意到当我 insert 一笔新的记录时，我没有设定 ID 这一列。这是因为 ID 这列被设定为自动插入类型。但是当我 update 一笔现有的记录时候，我必须引用 ID 属性，否则数据库不知道该改变哪一笔记录。

如果我正在 insert 一笔新的记录，在实际产生它之前，我们并不知道 ID 具体的值。如果我需要引用新的记录的 ID，使用下列语句：

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

（这一行代码必须跟在 insert 语句之后，否则可能得到错误的结果。）

还有一点需要知道，CI 的 AR 操作，包括

`$this->db->insert_id()` 和 `$this->db->update_id()` 会自己转义。

从 1.5 版，CI 也包含了对事务的支持，即指定的一批 SQL 操作要么全成功，要么全失败，换句话说，要么提交，要么回滚。这在复式记帐应用和许多商业应用中是很重要的。举例来说，说你正在卖电影票。你需要接受付款，同时分配座位。如果你的系统收费成功但分配座位失败，这个客户肯定是要光火的。

CI 现在也让事务处理变得很简单，即要么“提交”，要么回滚。你可以参考用户手册以得到更多关于事务的信息。

#### 4.3.5 “删除”查询

删除操作也许是最简单的。你只要定位好需要删除记录，比如我们要删除 ID 为 2 的记录：

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

要小心使用删除操作，因为如果你不注意 where 中的条件，可能会误操作，甚至最坏的结果是删除整张表。

#### 4.3.6 Active Record 和传统 SQL 编程的结合

CI 不要求你只能使用 AR，你也能用 CI 直接发送 SQL 查询。比如：假定你已在构造函数里已连接了数据库，你可以在需要的

地方直接使用类似的 SQL 查询：

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

我个人觉得 AR 更容易使用。借助数组为 AR 准备数据更直观，更容易理解，虽然可能需要更多的代码。AR 还能自动转义，对不熟悉 SQL 语法的程序员会更有吸引力。

然而，有那么少数几种情况，可能你需要使用原始的 SQL 语句。比如你想使用复杂的 join，或者又比如你需要使用多个“where”条件。如果你想找出 client 为 3 的网站，并且只是这两个指定的 type，那么你需要给 SQL 语句加上括号以便查询能被正确执行。

在这些情况下，你可以随时把 SQL 作为一个字符串，并放入一个变量中，然后在 CI 的 `$this->db->where()` 函数中使用这个变量：

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

如果没有括号就是含糊不清的。你的意思是：

SQL 代码

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

或

SQL 代码

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

嗯，是的，当然，这是显而易见的，但机器通常的猜测是错误的。顺便说一句，小心 `$condition` 的语法。实际的 SQL 查询是：

SQL 代码

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

双引号是用来定义变量的：

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

这很容易让你的单引号和双引号产生混淆。

我上面描述的一些 CI 表达式，比如

`$this->db->affected_rows()`，并不是 Active Record 模型的一部分。但他们可以很容易的混合使用。

需要指出的是，如果你试着混合使用 AR 和直接的 SQL 查询，你可能会麻烦。（我还没有尝试过。如果你手中有大把的时间，你可以对其进行测试，但坦率地说，我认为这表明一个不太好的生活方式。如果你有这闲工夫，不如尝试去数火车厢，至少还能让你呼吸点新鲜空气。而我本人，则是太忙了以至于不能不用 CI！）

#### 4.4 总结

我们已经介绍了 CI 的 Active Record 类，而且见到，它是多么容易使用：

连接到一个或更多的数据库

编写标准 SQL 的“读取”、“更新”、“创建”和“删除”查询

正确使用数据库所需要执行的其他功能

CI 的 AR 功能概念清晰又容易使用，而且使代码非常易读。它自动化数据库连接，并把连接数据保存至一个 config 文件。

它几乎可以做“传统”SQL 所能做的任何事情——比我在这里讲的要多的多。更多内容请参考在线《用户指南》。

#### 4.5 附录：使用 MySQL 查询语句创建“websites”数据库

MYSQL 代码

```
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=utf8;

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,
  `datereg` 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=utf8 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=utf8 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=utf8;

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,
  `nslurl` VARCHAR(36) NOT NULL,
  `nslip` 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=utf8 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=utf8 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=utf8 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=utf8 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=utf8;
```

## 第五章 简化 HTML 页面和表单的设计

本章介绍了另一种节约你的时间，并使你的代码更具安全性和逻辑性的方法。

首先，我们将会介绍创建视图的各种不同方法-与你的控制器和模型协同并用来显示结果的页面。

然后，你将会学到如何很快地创建 HTML 表单，与实现内建的保护；而且你也将看到该如何校验你的表单。

我假定这本书的读者熟悉 HTML 和 CSS。下列的例子非常简单，因此，我们能把重点放在 CI 代码上。而且我已经假定我们已经写一个 CSS 文件并已把它保存在网站的某个目录中。

### 5.1 编写视图

视图是用户能看到你的网站的所有。他们使用一个统一的接口，而且可以根据需要进行修改。MVC 的好处之一是你分开了表示层和逻辑层，一切都显得很干净。

到现在为止，我们已经完成了那非常简单的“welcome”页面，（记得第 3 章吗？）现在让我们看看该如何使它变得更精细。

视图实际上是一组包含有你的内容的 HTML 结构。结构中有各种元素，如颜色，字体，文字布局等；不过视图不关心这些，它要做的只是取来内容，显示出来。

创建视图，首先你需要创建一个 HTML 网页的骨架，并保存为.php 后缀。让我们称它为 basic\_view.php。保存在 application/views 目录中。（CI 的 loader 会在这个目录寻找视图文件。）

HTML 代码

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

然后当你想要从一个控制器装载它时，使用在某个函数中调用\$this->load->view()：

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

注意，如果这是一个 model 或者一个 helper，你将会首先装载它，然后根据需要使用它。通过视图，调用它只需要一行代码。

当然，那是一个空的视图。为了要使它有用，我们需要内容。因此我们要增加名称和一些文本。首先我们在控制器中定义他们：

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

注意我们并没有把它们定义为单独的变量，而是作为数组 \$data 的元素。对于第一个元素，键名是 'mytitle'，值是 "A website monitoring tool"。

然后，我们调用装载函数：

```
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);
}
```

我们把 \$data 数组作为\$this->load->view() 的第二个参数，在视图名称之后。视图接收到 \$data 数组后，使用 PHP 函数 extract() 把数组中的每个元素转换成内存变量，数组的键名即为变量名，值为变量内所包含的值。这些变量的值能直接被视图引用：

HTML 代码

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

虽然你只能传送一个变量到视图，但是通过建立数组，你能把大量变量整洁地传入视图。它似乎复杂，但是实际上是一种紧凑和优秀的信息传输方式。

### 5.2 PHP 语法的长格式和短格式

在我们继续之前，先了解一下 PHP 标记的两种不同格式。

常用的方式是：

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

然而，如果你不喜欢这种，CI 也支持一个较短的版本：

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

<?php ?>被<? ?>取代，而且 echo 由“=”代替，你也能用短格式来使用 if, for, foreach, 和 while 循环。完整的介绍请参考在线用户手册。

我个人偏好标准格式，因为我已经习惯使用它。如果你使用短格式，注意有些服务器不能正确地解释短格式。如果你仍然想要使用短标签，可以打开 config 文件，改变下列设置：

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

如设置为 TRUE，CI 在把它们送到服务器之前，将把短格式改成标准格式。不过这样做会对调试造成困难。因此，建议使用标准格式。

CI 也有一个‘模板语法分析器’类，允许你把变量放入 HTML 代码而不需要任何的 PHP 代码。本文不涉及这个内容。如果在与可能被弄糊涂的 HTML 设计者合作，它相当有用，细节请参见用户手册。

### 5.3 嵌套视图

为了最大程度地重用代码，我们可以提取 HTML 页面的公共部分，例如，header 和 footer，然后在显示实际视图时把它们组合起来。

让我们创建一个视图的 header 部分，这是一个符合 W3C 标准的 header、包含 HTML 声明和 meta 数据。

首先，我们列出 header 部分的代码：

HTML 代码

```
<!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; ?>" />
```

把这保存为 views/header\_view。下面介绍它包含的变量：

\$mywebtitle，是标题（meta 标签；这将不在屏幕上出现，但是搜索引擎将会读取，每个页面的 meta 有可能有变化，因此，我把它设为一个变量。）

\$myrobots，这个变量用来告诉机器人当前页面不需要被编入索引。

\$base 和 \$css，描述基本网址的字符串。\$css 包含 css 文件的路径信息，这些信息会从 CI 的 config 文件读取（也可以用 CI config 变量 site\_url 代替。）

现在我们需要知道：

我们如何调用嵌套视图？

我们如何获取变量的值？

有两个方法可选择。第一，在主调用视图中读入其它视图，因此我们的主视图，也就是 basic\_view，应该加上一行：

HTML 代码

```
<html><head>
```



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

变量可以在控制器中加上两行:

```
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);
}
```

在这里新的变量\$myrobots, \$css, \$base, \$mywebtitle 被创建为数组\$data 的新元素, 当 header\_view 被 basic\_view 调用时, CI 使用 extract() 解开他们, 在视图中显示出来 (在两个视图中不要出现同名的变量, 否则会引起冲突)。

第二个方法将把视图加入控制器里面, 给它分配一个变量:

```
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);
}
```

从严格的 MVC 设计原则来看, 这样做似乎更正确。

实际上有三个参数可传给 load->view 函数。

第一个, header\_view, 是要装载的视图的名字。这是必选。

第二个, 是可选项, 是装入视图的数据。

第三个是布尔值。如果你不指定它, 默认是 FALSE, 将视图送到浏览器。然而, 在嵌套视图这种情况下, 你需要将 header\_view 送到主视图 basic\_view 中, 因此需要将第三项参数设置为 TRUE。

现在我们已经建立了与 stylesheet 的关联, 我们能够用定义好的 css 中的类来更新视图中的显示部分:

```
HTML 代码
<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>
```

请注意 CI 的 MVC 系统能让你分离显示的内容。视图只为内容提供结构, 结构的风格则由 css stylesheet 控制。

视图不关心什么 \$mytext 的内容是什么, 它只是按照正确的格式在正确的位置上显示它。定义 \$mytext 的控制器甚至不知道 (也不关心) 它产生的数据如何被显示。

因此, 如果我们需要改变我们网页的外观, 或在一个不同的系统 (如 WAP) 上显示他们, 那么我们只需要改变视图和 CSS stylesheet。我们不需要修改控制器。

而且如果我们想要改变在网页上的信息, 我们不需要去改动视图, 而是只需要改变控制器里的变量值。

记得“松耦合”原则吗? 这里再一次体会到了这个原则, 这使设计、升级和维持你的网站比较容易。

## 5.4 网站架构的现实问题

请稍等片刻, 我们在 header\_view 中动态生成了 CSS 样式表的地址:

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

这意味着控制器必须生成变量的值, 这些值与数据如何被显示有关, 但是我们在上面说过控制器不应该知道或者关心它们具体的值是什么。这样不就符合了我们刚才提及的所谓‘松耦合’原则? 动态地产生这些数据需要这样一些操作: 首先, 控制器必须在 config 文件中读取它们, 然后控制器必须在 \$data 数组中装入它们而且传送它们到视图, 然后视图必须解开成为内存变量 \$base 和 \$css, 真正使用这两个变量的是 HTML 协议。

似乎这样做太绕圈子了, 为什么不直接在视图中静止地插入数据?

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

用变量方式做这件事情的好处是: 如果你迁移网站, 或移动你的 CSS 文件, 你只需要在 config 文件中改变设置, 而且每个控制器和视图将会立刻反映变化。而如果把 CSS 位置硬编码到每个视图的后果是一旦需要变化, 你必须到每个视图中去修改它们, 明白这样做的好处了吧?

没有一个正确的答案。这取决于你的优先级。关键是要合理运用 MVC 设计原则, 让它作为一种工具, 而不是束缚。为此, CI 给了你很大自由。

第三种方法, 也是我使用的方法, 创建一个特殊的“Display”模型。这个模型用于构建标准的页面。把页面头部和 CSS 文件引用等内容放进模型中, 并且模型会接受一个存放某些信息的参数。我将在本章其它章节讲解这个模型。

## 5.5 CI 的表单辅助函数: 输入数据

让我们把目光转向你如何使用你的 HTML 页。任何动态的网站最重要部份之一是和用户互动, 而且这通常意味着使用 HTML 表单。编写表单是重复和无聊的。

CI 的表单辅助函数是非常有用的代码片断。它有一个稍稍不同的定义, 使表单创建起来比较容易。让我们建立一个表单, 这个表单允许我们在浏览器中录入数据。在 websites 数据库的 sites 表中, 我们想要录入网站的名字、类型和网址, 和更新的日期。

你能用简单的 HTML 代码建立表单, 或者你能在一个控制器内建立它, 把它赋给一个变量, 然后调用视图, 而且传送该变量到视图。我正在按第二种方式做。

第一, 我们必须装载表单辅助函数到我们需要使用它的控制器内。然后, 我们把下列的代码放入控制器的构造函数:

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

然后, 我们必须开始编写表单。

现在, 为了生成表单的输入项, 我们不用这样写:

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

CI 允许你这样做:

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

(记得 'name' 是输入项的名称, 'value' 是你想输入的内容。在这里可以设定 value 的初始值, 或你能动态地从表单中获取。)

嗯, 你可能会说, 48 个字符变成 33 个字符, 没少几个字符, 特别是我必须先装载这个辅助函数 (另外的 28 个字符)。何必呢? 理由是:

### 5.5.1 使用表单辅助函数的好处之一: 清楚

使用 CI 表单辅助函数的第一个好处是你的代码绝对的清楚。如果你想要一个比较精细的输入框, 如果用 HTML 是这样的:

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

(记住, type 是输入框的类型: text、hidden 等等。

name 是将在\$\_POST 数组中取得的变量名称。id 是在网页上定位这个输入框的标识符, 如果你使用 JavaScript 的话。

value 是输入框里显示的值, 它一开始是一个默认值, 用户也可以在输入一个新的值。

maxlength 和 size 是明显的; style 一组 HTML 格式或者在 css stylesheet 中定义。)

CI 用一个数组代替上述的 HTML 代码:

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

它看上去蛮长的, 实际上并不比 HTML 代码长, 而且, 它非常清楚, 容易理解和维护。而且是动态的。

隐藏的表单输入框非常简单。如果我们想要自动地记录我们的数据库被更新的日期。我们把日期放入一个\$date 变量, 然后:

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

如果你想要一个'文本'输入框, 给你的使用者提供一个可以输入超过一行的地方, 可以使用 CI 的 form\_textarea() 函数, 下面的代码使用默认的长度, 在网页上显示一个文件输入框:

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

CI 的表单辅助函数在你编写下拉框, 多选框和单选框时特别有用, 如果我们要改变我们的 URL 输入框为一个下拉框, 允许用户从下拉列表中选一个 URL。首先, 把下拉列表的选项存入一个数组, 然后调用 form\_dropdown() 函数:

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

被传给表单中 url 下拉框的第一个参数是输入框的名字; 第二个是包含下拉列表的数组, 第三个默认选项。换句话说, 如果使用者接受默认值, 你的\$\_POST 数组将会包含值 'url=1', 但是你的用户将会见到选项 'www.this.com'。

如果使用 HTML 代码编写:

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

CI 实现的代码实际上比较短, 很容易学会。

如果你在一个数据库表('urls')中储存你的可能选择的网址的目录, 那么生成一个动态下拉框很容易。首先把数据从表中读出放到一个数组中:

```
$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;
    }
}
```

然后重复我们以前用过的 CI form\_dropdown() 功能:

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

只有\$urlarray 的内容会发生变化; 代码总是一样的。

如果你正在更新一个表中的记录而不是插入, 你不想为你的用户显示默认值。你想要为那一个记录显示已经存在的值。你应该已经知道你想要的修改的记录的 id 值, 因此, 你需要先读取数据库中'site' 表中相关记录。确定把查询结果赋给一个变量, 使用第二个变量取出第一个变量的中的相关记录, 再调用 CI 的 form\_dropdown 函数, 把第二个变量和对应的列名作为参数传入:

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

然后你的 CI 下拉框函数会从中读取相关信息:

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

本书没有太多的篇幅讨论所有的表单辅助函数。它还能编写单选框, 隐藏文件框, 多选框和一些其它的输入框, 完整的资料请参考 CI 用户手册。

## 5.5.2 使用表单辅助函数的好处之二: 自动化

使用表单辅助函数的第二个好处是可以自动化实现一些功能, 不然的话, 你只能自己编写相关的脚本了。

首先, 它拦截 HTML 的一些字符, 比如用户输入的引号, 并且转义它们以免破坏表单。

其次, 它自动链接。当你打开一个表单时, 你必须声明目标页, 它将会接受表单的数据并且处理它。(在 CI 中, 这是一个控制器里面的一个功能而不是一个静态页。比如它指向控制器的更新函数。)因此, 如果你用纯 HTML 代码, 你将会这样写:

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

如果你用 CI 打开你的表单, 你只需要这样做:

```
form_open('websites/update');
```

CI 自动地在你的 config 文件中取出基本 URL 并定位到对应的控制器函数。再次强调, 如果你迁移你的网站, 你只需要修改 config 文件, 而不是去一个一个地修改代码文件。

顺便提一下, CI 假定你的表单将会总是以 POST 的方式提交数据而不是 GET 方式。CI 普遍使用 URL 本身, 因此, 不要搞错。

## 5.6 我的 "Display" 模型

作为示范 (稍微简化了一下), 这里是我的 Display 模型:

```
<?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);
}
}
??

```

我能用下面的代码在任何的控制器中调用这个主页：

```

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

```

并且我也知道我的视图正在被动态地装配，完全符合我的需要。

## 5.7 CI 的验证类：方便地检验数据

在你编写 HTML 表单时一个重要的工作是检查输入。我们都知道我们应该这样做，但是…直到现在为止，我们已经编写过一种简单的表单，将会信任地接受任何用户输入的任何数据。你应该

意识到可能有一些用户是不怀好意的，而且所有的其余都是不负责任的。（别直接告诉他们。）如果他们有可能犯一个简单的错误，他们就会犯。确保你始终检查用户输入的数据，并使它们符合你的要求。

你能在客户端用 javascript 做到这一点，但是这样做作用有限，使用者能容易地绕过它。而在服务器端的校验需要一个额外的信息来回，这点额外的开销是值得的。

编写校验代码也相当复杂，但是，你一定猜到了。CI 提供了一个验证类可以使这项工作变得非常容易。

让我们改变我们自己的表单处理过程来实现校验。你需要在表单里作一些调整，还要在它指向的函数里作一些调整。

如果你的表单由 `form_open('sites/update')` 开始，你需要修改的函数是 `'sites'` 控制器里的 `'update'` 函数。如果你没有使用 CI 的表单辅助函数，HTML 等价代码是：

### HTML 代码

```

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

```

你需要做三件事情：

设置校验

设置控制器

设置表单

### 5.7.1 设置验证规则

在你的表单指定的那个函数中装载验证类并声明你的校验规则：

```

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

```

`'url'` 和 `'name'` 输入框一定要有输入内容。CI 提供了各种操作，确保一些操作一定要进行，用户手册全面地解释了这些内容。他们的含义非常明了：`min_length[6]` 显然意味着输入的信息长度一定要大于等于六个字符。`numeric` 意味着只能输入数字，等等。你还能组合规则，用 `"|"` 把它们连接起来：

```

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

```

意味着不能为空，字母，长度至少 12 个字符。你甚至能编写你自己的规则。

### 5.7.2 设置控制器

仍然在相同的函数中，创建一个 `'if/else'` 语句：

```

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

```

你进行确认测试，而且如果输入内容不能通过测试的话，就再返回到输入页面。（如果你在一个控制器内的一个函数中生成你的视图，则使用 `$this->myfunction` 代替

`$this->load->view('myform')`。

如果校验成功，就生成 `view("success")`，告诉用户输入的信息已被接受，然后给出一个链接让他进到下一步。

### 5.7.3 设置表单

录入信息的表单也要做相应的调整。每次校验没有通过的话，你不但要让系统返回到录入界面，而且必须说明哪一项出错，以及为什么出错。因此你必须在表单的某处给出一个附加信息：

```

$this->validation->error_string;

```

这行代码显示适当的信息，避免用户在那里犯嘀咕。

你也需要自动地填写用户已正确输入的那些内容，否则，用户必须再次录入上一次他们已经正确录入的信息。

首先，你需要在控制器里增加更多的代码。而且是立刻加在校验规则之后，加入一个数组来存放给用户的提示信息。数组的键名是你表单中的输入框名，值是给出的错误提示信息：



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

然后，增加一行代码：

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

现在你已经在控制器里声明了一个存有信息的数组，你只需要在表单内加入显示它们的代码。对于 HTML 代码，这会是：

HTML 代码

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

或者，如果你正在使用 CI 的表单辅助函数：

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

如果使用这个表单插入一个新的记录到数据库的表中，上面的代码已经够用了。如果你正在使用表单更新一个已经输入过的记录，当表单第一次显示时，应该在输入框中显示数据库表中的实际信息，这个时候，它的值应该是从数据库里读回来的（记得前面调用 `$siterow->url` 的例子吗？）

如果你在更新一个现有的记录时，上一次的录入内容由于一个输入框内容没有录入而无法通过校验，在重新回到表单之前，你需要在通过校验的输入框中填写用户刚录入的信息，而在校验出错的输入框里再次放入从数据库表中读入的信息，否则，你就需要再次录入已经校验通过的信息了。

还好，这可以通过一个简单的“if/else”语句来实现：

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

第一次表单显示的时候，在 `$_POST` 数组中将会没有内容；因此你从数据库的相关表中读取信息。但当你提交一次以后，`$_POST` 数组中有数据存在，所以你选择 `validation` 函数中返回的值。

查阅 CI 用户手册，了解表单校验的其它内容，你还可以做到：自动地准备你的数据，举例来说，通过它消除可能产生的跨站脚本攻击

编写你自己的复杂校验标准，举例来说，用户录入的值不能已经存在于数据库中

编写你自己的错误信息

CI 的验证类非常有用而又功能强大，值得花时间好好研读并掌握。

## 5.8 总结

我们已经学习了 CI 中生成视图的方法，以及它如何让你创建“迷你-视图”，你能把视图嵌套到其它视图中去。这意谓着你能建立共用的 HTML 头部和 HTML 尾部，实现视图的重用。

我们也已经见到 CI 如何帮助你编写 HTML 录入表单，通过表单辅助函数简化 HTML 表单的编写工作。

最后，我们学习了 CI 的验证类，这是检查用户录入信息的有用工具。没有什么是完美的，但是这个工具的确能阻击你的用户录入垃圾，或企图进行攻击。它也使你的网站看起来更加专业，能够有效地捕捉用户造成的各种输入错误，而不是一味地接受无意义的输入。

在整个学习过程中，我们也再次玩味了 MVC 的原则，而且有时稍稍地做一些变通会让生活变得更容易。CI 有一种非常有柔性的哲学：如果要有效率地解决问题，就要学会灵活地使用工具。



第六章 简化使用 Session 和安全

理论说得够多了！现在让我们开始写我们自己的应用。在这一章里，我们将会大致描述一下我们要建立的一个应用系统，而且我们分析一些会影响到网站系统的基本问题也就是 Session 管理和安全。

- 在这一章，我们将会看到：
- 如何使你的网页安全
- 如何使用 CI 的 Session 类

6.1 开始用 CI 设计一个实际的网站

我们已经看过了 CI 的 welcome 页面，并且了解了它是怎样通过控制器和视图文件创建的。实际上这就是 CI 的“hello world”。

从前，业余人士写的发烧友网站都使用开源系统，而且总是被一些所谓的大公司看成是低档的玩艺儿。不过现在，套用一句老话，时代不同了。很多大公司开始使用开放源代码技术。举例来说，NASA（美国太空总署）和美联社使用 MySQL、US GAO 和 Yahoo 正在使用 PHP 的应用。而且我相信规模适中，具有高度灵活性的动态网站正在迅速增长。当大公司了解到那些他们一直使用的商业技术无法满足新的需求后，他们引进规模适中、具有高度灵活性的系统代替原来的系统。

CI 无疑是设计开发规模适度、具有高度灵活性的网站的利器。它手把手地教你规划网站，并做出一致而可靠的应用。为了示范 CI 的灵活性，让我们开始创建一个应用。重新说一下我的需求，我要让这个网站为我解决问题。我正在维护着一些网站，有些是我自己的，有些是我的客户的。我需要对这些网站进行日常维护，测试它们，确保它们运转良好。大部分的工作内容是重复性的，我可以雇人来做这件事，但是写一个网站将会是比较合理的方案，因为它能自动化的完成工作，不管白天还是黑夜。

- 因此我的需求是：
- 不需要人工干预，管理一个或更多的远程网站
- 定期检测远程网站
- 生成报表，提供网站的细节和测试结果
- 如果 ISP 允许的话，我想设定网站能运行 Cron（注：Linux 下的计划任务软件），如果不允许，我会自己一天两次或者 1 小时 1 次地运行，让它运行一个预设的测试方案。
- 我不想知道细节，除非什么东西出现故障（然后我想要让它发送一封 Email 告诉我出了什么事，是哪个环节出的事），然后我希望能打印出管理报告，让我的客户意识到我一直在做定期的检查，并正常地运行他们的网站。

注意：为了避免使代码太长和过于重复，因此在这本书里的代码并不足够安全，因此请记住不要在实际应用环境中使用这些代码。这一章涉及了针对未授权用户的相关的安全知识。但是，确保网站安全的一般概念等其他 PHP 安全问题，本书没有涉及。

在现阶段，我们可以看到 CI 的实现方式与一般动态网站的方式相似，因此让我们先放下设计网站这个主题，从一些非常基本的概念谈起。

6.2 关于网站

任何的网站是一个各类应用的集合，而且它们彼此之间应该能够交互。我们在第 3 章看到，CI 使用 URL 进行链接。典型的 URL 如下表：

基本 URL	http://www.mysite.com 这是最基本的 URL，用户用它定位你的网站。用户不需要知道 URL 的其余部分，网站会在需要时自动加上其它部分
index 文件	第 1 段：index.php 这是 CI 的网站主文件
类（或称为控制器）	第 2 段：start 如果没有设置控制器，CI 会从 config 文件中寻找默认控制器
方法（或	第 3 段：assessme

称为函数）	如果没有设定函数，CI 会使用本控制器中的默认函数，找不到就重定向到 404 页
附加的任何的参数	第 4 段：fred（第 5 段：12345，第 6 段：Hello，等等。）

因此，在 start 控制器中调用 assessme 函数，并传入 fred、12345 两个参数，你的网址将会是：  
http://www.mysite.com/index.php/start/assessme/fred/12345

系统会查找 start.php，其中有一个 Start 类，它的内部有一个 assessme 方法，需要传入两个参数，现在传给这两个参数的值分别是 fred 和 12345。一个这样的 URL 可以调用你网站上任何一个控制器中的任何功能。

为了体会它如何工作，让我们建立我们网站的第一个页面。我们将会建立叫做 Start 的控制器，而且把它设定为默认控制器。好吧，首先，因为安装 CI 时，它指定 welcome 为默认控制器，因此我需要改变这个设置。CI 默认的路由存放在下列路径：  
/system/application/config/routes。当前设置为：

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

我们把它改成：  
\$route['default\_controller'] = "start";

（记住，如果你的默认控制器中没有有一个 index 方法，如果使用基本 URL，系统会显示 404 错误）  
现在我需要编写我的 start 控制器。记住基本的格式：

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

把这个文件保存为  
/system/application/controllers/start.php。  
第二行告诉你这是一个控制器。然后构造函数装载父控制器构造方法。assessme 函数需要两个变量 \$name 和 \$password。CI（从 1.5 版开始）自动分配 URL 的相关部分作为参数，因此 fred 和 12345 将会变成那\$name 和\$password。

因此，如果我们键入上面的网址，如果用户存在，密码无误，会被重定向到 mainpage() 函数。我们会在 start 控制器中增加这个函数。（如果用户检测不能通过，系统运行结束。）

对于那些过去一直使用过程式编程而不是 OO 编程的人来说，请注意一个类中的方法必须被表示成\$this->xxxx。因此，如果我调用 start 控制器里的 mainpage() 函数，我必须这样表示 \$this->mainpage()。否则，CI 会找不到它。

当然，一般用户不会这样定位一个网站：  
http://www.mysite.com/index.php/start/assessme/fred/12345

他们仅仅输入：  
http://www.mysite.com  
而且希望网站给出导航条。因此让我们从这里开始。通常，你在网站上看到的第一个页面是登录页面。因此让我们来做一个。首先，我把一个新的函数加入 start 控制器。我想要网站默认调用这个函数，因此，我将它命名为 index()

```
function index()
{
    $data['mytitle'] = "My site";
```

```

    $data['base'] =
$this->config->item('base_url');
    $data['css'] = $this->config->item('css');
    $data['mytitle'] = "A website to monitor other
websites";
    $data['text'] = "Please log in here!";

    $this->load->view('entrypage', $data);
}

```

它调用视图：entrypage。视图中包含一个表格，这个表格允许用户输入用户名和密码。HTML 表格需要指定一个处理\$\_POST 的函数，我们已经把它放在 start 控制器中，就是 assessme()。用 HTML 代码表示，在我们的视图上的表格应该是：

HTML 代码

```

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

```

我已经稍稍解释了 assessme 函数，它接受一个用户/密码组合，我需要一个数据库中查询它。为了要使代码更结构化，我们要引入一个函数叫做 checkme()。

所以，你会见到 assessme() 调用 checkme()，checkme() 在数据库里检查用户是否存在，密码是否有效，然后把‘是’或‘不’返回给 assessme()。如果返回是，assessme() 调用另外一个函数，mainpage()，返回一个视图。

注意代码模块的好处。每个函数扮演一个角色。如果我需要改变系统检查密码的方式，我只需要改变 checkme() 函数。如果我需要修改页面，我只要修改 mainpage() 函数。

让我们看看代码结构以及各部分如何交互。注意，为了要使例子简洁，我们不校验用户的输入。当然，这不会造成问题，CI 的表格类自动为输入数据“消毒”。

```

/*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,
which re-presents the log-in screen */
    else{
        $this->index();
    }
}
/*called with a u/n and p/w, this checks them against some
list. For the moment, there's just one option. Returns 'yes'
or 'no' */

function checkme($username='', $password='') {
    if ($username == 'fred' && $password == '12345') {
        return 'yes';
    } else {
        return('no';
    }
}

```

在第 5-6 行上，assessme() 接收来自表格的\$\_POST 数组，包含以下一些数据：

```
[username] => fred
```

```
[password] => 12345
```

assessme() 函数把这两个变量传递给另一个函数

checkme()。该函数检测具有密码 12345 的用户 fred 是否是合法的用户，如果他们是，它返回‘yes’。显然地，在一个真正的网站上，这会更复杂。你或许会为有效的用户名/密码做数据库查询。把它定义成一个单独的函数意味着我可以测试我的代码的其它部分，并在稍后改进 checkme() 函数。

如果使用者名称和密码是一个有效的组合，assessme() 函数调用另一个函数 mainpage()，让用户进入网站。否则，它返回到 index() 函数—即，再次显示登录窗口。

下一个问题是如何处理状态。换句话说，当进入到另一页面时如何识别这是一个已登录的用户。

### 6.3 安全/Session: 使用另一个 CI 类库

如果我想要建立一个会话机制确保未授权用户无法存取我的文件，需要多少代码？

英特网通过一系列请求工作。你的浏览器给我的服务器发请求，要求某一特定页。我的浏览器把该页处理后返回你的服务器。服务器处理后向浏览器发出另一页面，你再按下一个超链接，对我的服务器提出新的页面请求。如此而已。

英特网是‘没有状态的’——就是，你的浏览器对我的服务器做的每个请求都被当做一个单独事件，HTTP 协议没有办法把你的请求和任何其他请求联系起来。

如果你想要你的网站页面与页面之间建立联系，你必须管理“状态”，要服务器记得哪些请求来自你的浏览器，需要做一些特别的处理。

PHP 提供管理状态的两种办法：使用 cookie 或使用 session 会话。PHP 的 session 自动地检查你的浏览器是否接受 cookie。如果不接受，它会使用会话 ID 的方法，这个 ID 直接通过 URL 进行传递。

Cookie 是小块的数据由服务器发送到你的浏览器。浏览器自动地保存它。一旦保存了 cookie，当浏览器尝试访问网站的时候，网站就会检查是否存在 cookie。如果它找到正确的 cookie，它能读取它里面的数据适当地配置它本身。可能关闭特定页面，或显示你的个人数据。

因为一些人设定他们的浏览器不接受 cookie，PHP 提供其它变通的方法。每次当浏览器请求访问时，网站产生‘会话 ID’，把它发送到浏览器端，当浏览器发送下个请求时把这个 ID 加入 URL，所以网站能够识别浏览器。

CI 有一个 Session 类来处理会话相关工作。事实上，它大大减少了编码量。我们在上一章学习到 CI 有各种各样的 library，大大简化了 PHP 的编程。这些就是框架的核心：集成大量代码，为你提供各种强大的功能。你要做的只是使用它们，而不必关心它们是如何运作的。作为结果，你的应用使用了最专门的代码，而你却不需自己去编写它们！

如果你想要在你的控制器内或模型中使用某个类内的函数，你一定记得首先要装载这个类，（有些类，比如 config 总是自动地被装载，这是为什么我们不需要在代码中装载它的原因。）

你可以这样很简单地装载类库：

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

通常，把这一行放入你的控制器或模型的构造函数中。

如果你认为你会在每个控制器中使用某个 library，你能象 config 类一样自动地装载它。找到 /system/application/config/autoload 文件，增加你想自动装载的类名：

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

因此它看起来像这样：

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

在这里，我们首先要用到 session 类，这帮助你维持状态，而且可以识别用户。做到这一点相当简单。下面列出增强的 assessme() 函数：

```

function assessme() {

    $this->load->library('session');

```

```

$username = $_POST['username'];
$password = $_POST['password'];

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

```

(你可以看到，在函数一开始就装载 session library，但是通常，我们将会控制器的构造函数中装载它，因此，在这一个类中的其它函数都可以使用这个类。)

当这个类装入后，立即通过它本身强大的功能为你提供对会话的读取，创建，修改等功能。

好吧，坦白地讲，不仅仅就一行代码那么简单。你必须首先修改 config 文件，告诉 Session 类你想要它做的。

检查你的 system/application/config/config.php 文件，你会找到像这样的一个部分：

```

/*-----
| Session Variables
|-----
|
| 'session_cookie_name' = the name you want for the cookie
| 'encrypt_sess_cookie' = TRUE/FALSE (boolean). Whether to
| encrypt the cookie
| 'session_expiration' = the number of SECONDS you want the
| session to last.
| by default sessions last 7200 seconds (two hours). Set
| to zero for no 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;

```

现在，确定 [sess\_use\_database] 被设定成 FALSE。

保存设置，现在，每次当你的客户登入网站，网站会在你的机器上保存一个 cookie，包含下列的数据：

CI 生成的一个唯一的 SESSION ID，这是 CI 为这个会话生成的任意组合的字符串。

用户的 IP 地址

用户的 User-Agent 数据（浏览器数据的最初 50 个字符）

最大有效时间

如果你设定 sess\_encrypt\_cookie 为 FALSE，你能在你的浏览器上看到 cookie 的内容：（它部分被编码了，但是你能辨认出来）：

ip\_address%22%3Bs%39%3%22127.0.0.1%22%3Bs%310%3A22

包括使用者的 URL，本例为 127.0.0.1。如果你设定 sess\_encrypt\_cookie 为 TRUE，cookie 被加密，只是一行无意义的代码。浏览器不能识别它，当然使用者也无法修改它。

当使用者发出另一个页面请求时，网站会检查会话 ID 是否已经以 cookie 的形式保存在用户的机器里，如果有，这会作为一个已经存在的会话的一部分，如果没有，就会建立一个新的会话。记得在所有的其它控制器上加载 Session 类。CI 将作同样的检查。我必须做的只是告诉每个控制器当没有 cookie 时该如何运作。

### 6.3.1 使 Session 更安全

这本来不会造成安全问题。任何访问网站的用户都会开始一个会话。代码只是判断他们是否是一个新用户。避免未认证的用户访问某些页面的一个方法是如果他们已经登录，就在 cookie 中加入一些信息，因此我们可以对此进行验证。一旦用户输入正确的用户名和密码一次，就会在 cookie 中被记录，当发出新的请求

时，会话机制会检查 cookie，如果是已登录的用户，就返回用户请求的页面，如是不是，就返回登录画面。

把信息加入 cookie 很容易。在我的 assessme() 控制器中，如果密码和使用者名称验证通过，加上如下代码：

```

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

```

这样会把 \$newdata 数组中内容-上例中只有一个变量-加入到 cookie 中。现在，每当密码/用户名组合是可接受的，assessme() 将会把 'OK' 加入 cookie，而且我们可以把这些代码加到每个控制器中：

```

/*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*/

```

这样，你在你的网站周围构建起安全的地基。你可以很容易地使它变得更精细。举例来说，如果你的部分用户有更高级的权限，你可以在 cookie 中保存了个存取级别码，而不是只放进一个“OK”，你能用这个进行更细致的权限调控。

在一个 cookie 中存放明码数据会比较糟糕，因为用户可以打开 cookie 并看到它的内容，当然也可以很容易地修改它。因此使用 CI 加密 cookie 会是一个很好的办法。还有，你可以在数据库中建立一个用户表，在用户登录以后修改用户表的登录状态，在用户表中维护一些权限属性，每次访问时，从数据库的用户表中读取权限信息，也是一个比较好的做法，如果结合缓存技术，就会解决系统的性能问题。

在你的数据库中保存会话数据非常简单。首先，创建数据库表。如果你正在使用 MySQL，使用 SQL 语句：

```

MySQL 代码
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)
);

```

然后，修改在 system/application/config/database.php 文件中的连接参数告诉 CI 数据库在哪里。详细介绍参见第 4 章

如果工作正常的话，当你登录或退出时，在你的数据库表中增加了相关的数据。如果你在一张数据库表中储存会话，当用户登录到你的网站，网站会查找 cookie，如果它找到了，就会读取会话 ID，并用这个 ID 去匹配数据库表中保存的 ID。

你现在拥有一个强健的会话机制。而这一切只需要一行代码！

一点声明，PHP 本身的 Session 类在用户关闭 cookie 功能时能够应付。（取代生成 cookie，它把会话数据加入 URL）CI 类不会这样做，如果使用 CI，一旦用户禁止 cookie，他就不能登录到你的网站。这方面需要增强功能。希望 Rich 将会很快升级 CI。

## 6.4 安全

注意到：Session 类会自动地保存关于访问者的 IP 地址和使用者的资料。你可以使用一些增强的安全措施。

通过修改 config 文件的两处设置可增加安全性：

sess\_match\_ip：如果你设定这个参数为 TRUE，当它读取会话数据时，CI 将会尝试匹配使用者的 IP 地址。这将预防使用者使用“抢”来的 cookie 信息。但是，有些服务器（ISP 和大公司服务器）可能存在不同 IP 地址上的相同使用者登录的请求。如果你设定这



个参数为 TRUE，可能会给他们造成麻烦。

sess\_match\_useragent：如果你设定这为 TRUE，当读取会话数据的时候，CI 将会试着匹配使用者代理。这意味试着“抢夺”会话的人还需要匹配真正用户的 user\_agent 设置。它使“抢夺”变得稍稍有些困难。

CI 也有一个 user\_agent 类，你可以这样装载：

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

一旦装载，你能要求它返回访问你网站的浏览器和操作系统的各种信息，而不管它是一个浏览器，手机或机器人。比如，如果你想列出参观你的网站的机器人，你可以这样做：

```
$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
    is returning*/
}
```

类通过装入后开始工作，与用代理、浏览器、机器人和其它的类型的数组作比较，这个数组存放在 system/application/config/user\_agents。

如果你愿意，你可以容易开发出使你的网站锁定特定机器人，特定浏览器类型的功能。不过，记得一个攻击者很容易写出一个代理类型，并返回他想要扔给你的那种类型。因此，他们能容易地伪装成通常的浏览器。许多机器人，包括象 CI 的 user\_agents 数组已列出的 Googlebot，是‘品行端正的’。这意味着如果你设定你的 robots.txt 文件排除他们，他们将不会强行攻击。没有容易的方法去阻击一个不知名的机械手，除非你预先知道他们的名字！

在 CI 框架中，会话机制保存那发出请求的 IP 地址，因此你可以使用这个功能维护一个网站的黑名单。可以用如下代码取回来自 Session 的 IP：

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

因此你能针对黑名单作相应的安全处理，比如拒绝登录。

你也可以用 CI 的会话机制限制来自重复请求的损害—像是一个机器人通过重复地请求网页来使你的网站超载。你也可以使用这一个机制处理‘字典’攻击，即一个机器人不断重复尝试数百或者数以千计的密码/用户名组合直到它找到正确的一组。

因为 CI 的 Session 类保存每个会话的 last\_activity，所以你能做到这一点。每次，当网页被请求时，你能检查多久以前这个 IP 地址的用户发出一请求，两次请求的间隔如果不正常，你可以终止会话，或放慢响应的速度。

## 6.5 总结

我们已经概略介绍了我们想要建立的一个应用，和几乎所有应用都需要解决的问题：Session 管理和安全。

为了做到这一点，我们已经学习了 CI Session 类库的一些细节，而且见到了它如何生成 Session 记录和在访客的浏览器中生成 Cookie。

当后续的请求发生时，你可以读取 Cookie 对响应进行控制。



## 第七章 CodeIgniter 和对象

本章是“技术发烧友”的最爱。它讲述的是 CodeIgniter 的工作原理，也就是揭开 CI 头上的“神秘面纱”。如果你是 CI 新手，你可能会跳过本章。不过，迟早你会想要了解 CI 的幕后都发生了什么，为什么不真正的玩转它呢？

当我刚开始使用 CodeIgniter 的时候，对象使我迷惑。我是在使用 PHP4 的时候接触 CI 的，PHP4 并不是真正的面向对象 (OO) 语言。我在一大堆的对象和方法，属性和继承，还有封装等数据结构中转悠，总是被类似的出错信息包围“Call to a member function on a non-object”。我如此频繁地看到它们，因此我想到要印一件 T 恤衫，上写：神秘，无规律可循，而我仿佛正穿着它站在一个现代艺术展会的会场上。

这一章的内容包含 CI 使用对象的方法，和 OO 编程的方法。顺便说一下，术语“变量/属性”，“方法/函数”是等义的，CI 和 PHP 经常会混着使用它们。比如，你在控制器中写一个“函数”，纯 OO 程序员会称他们为“方法”。你称之为类的“变量”而纯 OO 程序员会叫它们“属性”。

### 7.1 面向对象编程

我正在假定你和我一样有 OOP 的基本知识，但如果只是在 PHP4 中尝试过可能还不太够。PHP4 不是一种 OO 语言，虽然具备了一些 OO 的特征。PHP5 会更好一些，它的引擎已经彻底改写成面向对象的了。

不过基本的 OO 特征 PHP4 也能实现，而且 CI 设法让你无论是使用 PHP4 还是 PHP5，都有一样的行为特征。

重要的是你要记住，当 OO 程序运行时，总会有一个或数个真实的对象存在。对象可能彼此调用，只有当对象处于运行状态的那一刻你才可以读取变量（属性）和方法（函数）。因此了解哪个对象当前在运行并控制它们很重要。当一个类没有实例化时，你不能对它内部的属性和方法操作，静态方法和属性除外。

PHP，作为一个过程式编程和 OO 编程的混合体，可以让你混合编写又是过程式又是 OO 的程序，你可以在一个过程式代码中实例化一个类，然后使用它的属性和方法，用完后把它从内存中释放掉。这些工作，CI 都可以为你代劳。

#### 7.1.1 CI “超级对象”的工作原理

CI 构建一个“超级对象”：它把你的整个程序当作一个大的对象。

当你开始运行 CI 程序的时候，将发生一连串复杂的事件。如果你设定你的 CI 允许记录日志，你将会见到类似下面这样的内容：

```
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
```

在启动时—每当通过 Internet 接收到一个页面请求—CI 都执行相同的程序。你可以通过 CI 的文件来跟踪日志：

index.php 文件收到一个页面请求。URL 指出哪一个控制器被调用，如果没有，CI 有一个默认控制器（第 2 行）。Index.php 开始一些基本检查然后调用 codeigniter.php 文件（\codeigniter\codeigniter.php）。

codeigniter.php 文件实例化 Config、Router、Input 和 URL（等等）类。（第 1 行和 3-9 行）这些被调用的叫做“基础”类：你很少直接与它们交互，但是 CI 做的每件事都与它们有关。

codeigniter.php 检测它正在使用的 PHP 版本，根据版本决定调用 Base4 还是 Base5（\codeigniter\base4（或 5）.php）。然后创建一个“singleton”实例：即一个类只能有一个实例。并且都有一个公共的 &get\_instance() 方法。注意符号 &：这是引用实例的符号。因此如果你调用 &get\_instance() 方法，它产生

类的单一实例。换句话说，整个应用中这个实例是唯一的，其中包含许许多多框架中其它类的实例。

在安全检查之后，codeigniter.php 实例化被请求的控制器、或一个默认控制器（第 10 行）。新的类叫做 \$CI。然后调用 URL 中指定的函数（或默认函数），类被实例化之后，相当于被唤醒了，实实在在的存在于内存中。然后，CI 会实例化你需要的任何其他类，并“include”你需要的功能脚本。因此，在上面的日志中，model 类被实例化。（第 16 行）“boilerplate”脚本，也被装载（第 13 行），这是我编写的包含标准代码的一个文件。它是一个.php 文件，保存在 scripts 目录中，但是它不是一个类：仅仅是一组函数。如果你正在写“纯粹的”PHP 代码，你可能会使用“include”或者“require”把这个文件放进命名空间，CI 会使用它自己的“装载”函数把它放入“超级对象”中。

“名字空间(namespace)”的概念或范围在这里是很重要的。当你声明一个变量、数组、对象等等的时候，PHP 把变量名保存在内存中，并为它们的内容分配一个内存块。如果你用相同的名字定义两个变量就会出现错误。（在一个复杂的网站中，容易犯这样的错误。）基于这个原因，PHP 有几条规则。举例来说：

每个函数有它自己的名字空间或者范围，而且定义在一个函数中的变量一般是一个“局部”变量。在函数外，它们是不可见的。

你可以声明“全局”变量，它们被放在特别的全局名字空间中，并且在整个程序中都可以调用。

对象有他们自己的名字空间：对象内的变量（属性）是与对象同时存在的，可以通过对象来引用。

因此 \$variable、global \$variable 和 \$this->variable 是三个不同的东西。

特别地，在 OO 之前，这可能导致各种混乱：你可能有太多的变量在同一个名字空间中（以至于许多冲突的变量互相覆盖），也可能发现有些变量在某个位置无法存取。CI 为此提供了一个解决办法。

假如现在你已经键入如下 URL：

www.mysite.com/index.php/welcome/index，你希望调用 welcome 控制器的 index 函数。

如果你想要了解，哪个类和方法在当前的名字空间中可用，试着在 welcome 控制器中插入下列“检测”代码：

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

试着运行它，它列出了 270 个已定义的类。大部分是 PHP 核心定义的。最后 11 个来自 CI：10 个是 CI 基础类（config、router 等等）而且都是我的控制器调用的类。下面列出这 11 个类，清单只保留了最后两个方法，其它的被省略了：

```
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 )
```

注意—看一下 Welcome 类括号中包含的内容（第 270 个：即我正在使用的控制器），它列出了 Controller 类的所有方法（第 269 个）。这就是为什么你总是需要从一个控制器类派生子类的原因—因为你需要你的新控制器保留这些函数。（而同样地，你的模型应该总是从 model 类继承。）Welcome 类有两个额外的方法：welcome() 和 index()。到目前为止，在 270 个类中，我写的只有这两个函数！

你可能还注意到了类的实例—即 object。有一个指向它的变量，注意到那个引用符号了吗？表明在整个系统中，CI\_Input 类只有一个实例，可以用类变量 input 调用它：

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

还记得我们何时装载了 input 文件并且创建了最初的 input 类吗？它包含的属性是：

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

你可以看到，他们现在在已经全部被包括在实例中了，“设计图纸”变成了房子，不是吗？

所有其它的 CI “基础”类（routers、output 等等）同样地被包含了。你不需要调用这些基础类，但是 CI 本身需要他们使你的代码正常工作。

### 7.1.2 引用复制

刚才提到，类变量 input 引用了 CI\_Input 类：

(["input"]=>object(CI\_Input))，加不加引用符号区别在于：加上引用符号，一变全变，不加引用符号，原始对象的内容不会改变。你可能会对此感到困惑，用一个简单的例子来说明：

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

显示 1，因为 \$two 是 \$one 的拷贝。然而，如果你再重新给 \$one 赋值：

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

仍然显示 1，因为在对 \$one 重新赋值前 \$two 已经赋为 1 了，而 \$one 和 \$two 是两个不同的变量，各自分配有一小块内存，分别存放它们的值。

如果在 \$one 改变的时候，\$two 也要相应地改变，我们就要使用引用了，这个时候，\$one 和 \$two 实际上是指向了同一个内存块，一变全变：

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

现在显示 5：我们改变变量 \$one，实际上也同时改变了 \$two。

把符号“=”改成“&”意味着“引用”。针对对象来说，如果你要复制一个对象，与原来的对象没有关联，用“=”，如果要使两个变量指向同一个对象，就使用“&”，这时候，一个变量做出的任何改变都会影响到其它变量。

### 7.2 在 CI “超级对象”中加入你自己的代码

你可以为 CI “超级对象”加上你自己的代码。假定你已经写了一个名为“Status”的模型，它有两个属性：\$one 和 \$two，构造函数给他们分配两个值：\$one = 1 和 \$two = 2。当你装载这个模型时，让我们来看看会发生什么。

“instance”类有一个变量叫做“load”，用来引用对象 CI\_Loader。因此，在你的控制器中的代码是：

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

换句话说，调用当前 CI “超级对象”的类变量 load 的 model 方法，装载一个模型，这个模型的名称存放在变量 \$status 中。让我们看一下保存在 /system/libraries/loader.php 中的 model 方法：

```
function model($model, $name = '')
{
    if ($model == '')
        return;

    $obj =& get_instance();
    $obj->_ci_load_model($model, $name);
}
```

```
}
```

（这个函数里的变量 \$name 是你想要装载的模型的一个别名。我不知道为什么要使用一个别名，也许它会用在其他的名字空间中。）

就像你看到的，模型是以引用的方式装载进来的。因为 get\_instance() 是一个 Singleton 实例的方法，你总是针对同一个实例进行操作。

如果你重新运行控制器，用我们的“检测”代码来显示类的属性，你将会发现这个实例包含了一个新的属性：

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

换句话说，CI “超级对象”现在包括一个对象叫做 \$status，它包含了我们刚定义的两个变量，并被分配了两个值。

因此我们正在逐渐地创建一个大的 CI “超级对象”，允许你使用它的某些方法和属性，而不必担心它们来自哪里，或处于哪个名字空间中。

这就是需要 CI 箭头语法的理由。为了要使用一个模型中的方法，你必须先在你的控制器中装载模型：

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

这使模型被装载进当前控制器的实例中，也就是 \$this-> 中。你随后可以调用控制器中的 model 对象中的方法，像这样：

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

就行了。

### 7.3 CI “超级对象”的问题

当 Rick 刚开始开发 CI 时，为了让 CI 在 PHP4 和 PHP5 下行为一致，他必须在 Base4 文件中使用比较“丑陋”的代码，不管丑不丑，我们不用关心，只要 CI 能够在 PHP4 环境下工作得和 PHP5 一样好就行了。

还有两个问题值得在这里提一下：

你可以尝试开发一个全新的对象并让它参与工作。

你必须小心地为你的网站设计架构，因为你不能在一个控制器里调用另一个控制器里的方法。

让我们一个一个地来分析这两个问题。你记得我提到的 T 恤衫的那件事吗？在调用一个成员函数时我一直收到“Call to a member function on a non-object”的出错信息，这个出错信息产生的原因一般是因为你调用了类方法，但是忘了装载这个类。换句话说，你写了下列语句：

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

但是忘记在它之前调用：

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

还有一些其它情况，比如，你在类的一个方法中装载了模型，然后你尝试在另一个方法里调用模型的方法，虽然在同一个对象中，这样做也不行。所以最好的方法是在类的构造函数中装载模型，然后可以在这个类的所有方法中使用。

问题也可能更严重。如果写你自己的类，举例来说，你可能想要使用这个类存取数据库，或在你的 config 文件中读取信息，换句话说，让这个类存取 CI 超级对象的某些部分。（如何装载你自己的类和类库会在第 13 章中讨论。）概括起来，除非你的新类是一个控制器，一个模型或视图，它不能在 CI 超级对象中被构造。因此你不能在你的新类中写这样的代码：

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

这不会工作的，因为对你的新类来说，\$this-> 意味着它本身，而不是 CI 超级对象。取而代之地，你必须通过调用 Instance 类用另一个变量名（通常是 \$obj）把你的新类载入 CI 超级对象：

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

现在你能像调用 CI 超级对象一样地调用它：

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

并且这次它能工作了。

因此，当你编写你的新类时，记得它有它自己的标识符。让我们使用一个较简短的例子来把这个问题讲得更清楚一点。

你想要写一个 library 类，向你的服务器发出页面请求的 URL 查找它的地理位置。这个 library 类有点像 netGeo 类，你可以在下列网址找到它：

<http://www.phpclasses.org/browse/package/514.html>

这个类使用一个 switch 函数，根据 URL 的地域分派不同的网页，比如来自英国和美国的 URL 请求，你就返回一个英语网页，德国和奥地利的 URL 请求就返回一个德语网页等等。现在，完整的 URL 会分成两个部分：基本 URL (www.mysite.com/index.php) 和附加的 URL 部分 (mypage/germanversion)。

你需要从 CI 的 config 文件中取得基本 URL 部分。后半段网址通过你的新类的构造函数中的 switch 语句生成，如果这个客户在德国，调用德国的页面函数，依次类推。当这个工作在构造函数中做完以后，你需要把结果放到一个类属性中，所以可以在同一个类的其它函数中使用，这意味着：

基本 URL 从 CI config 文件中取得，这个只能通过 CI 超级对象的引用获得，换句话说，你可以用

`$obj->config->item('base_url')`；获得

URL 的后半部分由你的新类的构造函数生成，并写到一个类属性中：`$base`。这个属性与 CI 超级对象无关，它属于你的新类，被引用为 `$this->base`。

装载时会用到两个关键词：`$this->` 和 `$obj->`，在同一段代码中被引用，举例来说：

```
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;
}
```

如果你不清楚这些概念，“Call to a member function on a non-object”就会频繁的出现。上例中，如果你试着调用 `$obj->base` 或 `$this->config->item()` 时，这个出错信息就出现了。

转到下一个问题，你无法在一个控制器的内部调用另一个控制器的方法。你为什么会想要这样做？这视情况而定。在一个应用中，我在每个控制器内部写了一系列自我测试函数，如果我调用 `$this->selftest()`，它将完成各种不同的测试。但是，在每个函数中写重复的代码似乎与 OO 编程的设计原则不符，因此我想在其中一个控制器中写一个函数，可以进入到其它的控制器的自我测试代码。当我这样做了，期望得到想要的结果。最后，当然不能如我所愿，因为在一个控制器内不能调用另一个控制器的方法。

作为一个准则，如果你有代码被超过一个控制器调用的话，把它放入一个模型或者其它什么分离的代码文件中，这样就可以使用了。

这些都是小问题。正如 Rick 告诉我的一样：

“我想要简化问题，所以，我决定创建一个大的控制器对象包含很多其它对象的实例：…当一个用户创建他们自己的控制器时，他们能够轻松地访问任何资源，不用担心作用域的问题。”

这样做相当不错，绝大多数情况下，高效的在幕后完成了所有工作。因此我不必去做那件 T 恤衫了。

## 7.4 总结

我们已经看到 CI 创建的“超级对象”确保了所有方法和属性可以自动地获取，而不用担心如何管理，并且也不需要为“作用域”操心。

CI 用引用实例的方法把一个一个的类实例组合成一个超级对象。大多数情况下，你不需要知道 CI 超级对象是如何工作的，只需要正确使用“`->`”符号就行了。

我们也学习了如何编写自己的类，并使它与 CI 很好地协同工作。



## 第八章 用 CI 测试你的代码

这一章介绍 CI 如何帮助你测试代码。测试是一个应用程序的核心。我们用它测试其他的远程应用程序；我们也想要测试它自己，因为它本身也是代码。CI 使测试变得很容易。

不过，“测试”的含义很广泛，因此我们从两种主要的测试类型之间的差异开始，分析一下你应该如何开展测试工作。

我们来看看 CI 类如何帮你测试代码：

单元测试  
基准测试  
“评测器”

CI 提供你在数据库还没数据的时候进行测试的方法

### 8.1 为什么测试，为谁测试？

有关测试已经有很多的文献。它已经变成一种必要的技术。复杂程序需要一支测试员组成的队伍来测试软件。而且‘测试驱动开发’的概念是你在写下你的第一行代码之前首先设计你的测试程序，然后把你编写的代码交给它们。

与此相反的另一个极端，许多程序员不做任何的系统测试。因为测试似乎太困难，烦人并且花费大量时间。也许我们会做几下测试，然后希望其它的都工作正常。

CI 提供一些方法使测试变得容易。或者说—更有乐趣。有两种主要的测试类型：

单元测试：采取“由下而上的”的方法。他们查看你的一个代码块，比如说一个函数，把一些变量放进去，看看它是否返回正确的结果。

端到端测试：这些是“由上而下的”。他们把重心集中在某件事上，看系统能不能做到。举例来说，他们试着登录到你的网站(使用一个有效的用户名和密码)看看系统是否正常工作。(甚至他们会试着使用一个无效的密码登录…)

如你看到的，它是一种不同的理论。单元测试，不关心测试结果；而完整测试只关心结果，不考虑代码是否正常工作。

重要的事情是要考虑你为什么测试。什么让你最担心？什么最有可能出毛病而且使你困惑？你希望从你的测试中获取什么信息？仅仅是好或不好，还是更多的细节？对每个应用，你能负担多少时间编写测试代码并进行测试？

我们正在开发我们的测试网站，不过当我们编程时，我们需要测试我们的代码。当然，我们试着预期用户要做的每件事，和可能出现的每种情形。单元测试在很多方面是有用：设计测试方法帮助你改善代码的设计。

一旦我们的代码上传到一个服务器上，它的数据量一天一天地增大超过我们的控制。最坏的事情是导致客户发现错误信息或者空白的荧屏，而且期待你在忙着其它事情时来解决问题。这就是为什么我们正在创建这个网站，去测试其他的远程网站。

CI 能帮助我们检测网站，看是否发生如下的事情：

第一，我们已经预期多种可能出问题的情况。举例来说，我可能通过 ID 来做一个数据库查询，删除记录。是的，它可以工作：我实际这样操作同时也是测试它。但是发生了什么事：如果—不知何故—代码在调用一张不存在的表？或是 ID 号有问题？或根本没有什么 ID 值？这是单元测试有帮助的地方。

第二，当我在别的地方写更多代码的时候，我的第一部分的代码是否按我的要求工作，或者我已经不经意地修改了一些第一部分代码所依赖的部分？再一次地，给单元一个测试。他们也能定期地帮助我们检查产品服务器(包括它的所有部件，例如：如果数据库在一个分离的服务器，将不需要做一般性的‘ping’检测!)

CI 给你许多帮助，无论你处在什么情况下。它不提供一个功能测试的类，但是你能使用其他 做这个测试。但是让我们先看一下 CI 如何捕获你代码中的错误。

### 8.2 CI 的错误处理类

CI 有它自己的检测和报告错误的系统。一方面，这些是最简单和最普通的测试：他们是那些有帮助的(或者令人发怒的)信息：当你正在编写代码时并且它不正常工作的时候，你会看到这些信息。

默认地，CI 在屏幕上显示所有的错误。另一个选择是不报错；不显示出错信息会让你无法处理，因此，报错对调试是必要

的。全部的行为由 index.php 文件控制，代码如下：

```
/*
|-----
|
| 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);
```

这是一个 PHP 指令，表示报告所有的错误。为了要关掉错误报告，把最后一行改为：

```
error_reporting(0);
```

这会适合在产品网站上使用，它抑制所有的出错信息。

CI 有三个函数、show\_error()、show\_404() 和 log\_message()，控制错误如何在你的系统上被显示。(不同寻常地，这些函数是全局性的：你不需要装载就能使用他们，用就行了!)。事实上，show\_error() 和 show\_404() 通常默认产生：前一个在屏幕顶端的一个整洁的小 HTML 格式的框子中显示你的错误；后一个在你企图请求一个不存在的网页时显示一个‘404’页。

第三个函数，log\_message()，更有趣。你可能想要开发你自己的错误日志，原因有多种多样的，其中一个也许是也许因为你不能访问在你的 ISP 的 Apache 上的日志文件。首先，你需要设定权限确保 /system/logs 目录是可读写的。然后你在 config 文件中设定 logging 的级别：

```
/*
|-----
|
| 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;
```

这样会开启日志。

如果你修改 index.php 关闭错误信息显示并不会使错误日志不工作。因此你能看到信息，但你的用户看不到。

当你开启日志后，CI 每天产生新的记录文件，并把信息写入这个文件。但是小心，这些记录文件能快速地变得很大。

在实际使用过程中，你可能需要开发在某件事发生时显示特定出错信息的错误处理代码。



```

33 DEBUG - 2006-12-30 19:58:04 --> Language Class Initialized
34 DEBUG - 2006-12-30 19:58:04 --> Language file loaded: language/english/db_lang.php
35 DEBUG - 2006-12-30 19:58:22 --> Config Class Initialized
36 DEBUG - 2006-12-30 19:58:22 --> Hooks Class Initialized
37 DEBUG - 2006-12-30 19:58:22 --> No URI present. Default controller set.
38 DEBUG - 2006-12-30 19:58:22 --> Router Class Initialized
39 DEBUG - 2006-12-30 19:58:22 --> Output Class Initialized
40 DEBUG - 2006-12-30 19:58:22 --> Input Class Initialized
41 DEBUG - 2006-12-30 19:58:22 --> Global POST and COOKIE data sanitized
42 DEBUG - 2006-12-30 19:58:22 --> URI Class Initialized
43 DEBUG - 2006-12-30 19:58:22 --> Language Class Initialized
44 DEBUG - 2006-12-30 19:58:22 --> Loader Class Initialized
45 DEBUG - 2006-12-30 19:58:22 --> Session Class Initialized
46 DEBUG - 2006-12-30 19:58:22 --> Encrypt Class Initialized
47 DEBUG - 2006-12-30 19:58:23 --> Database Driver Class Initialized
48 DEBUG - 2006-12-30 19:58:23 --> A session cookie was not found.
49 DEBUG - 2006-12-30 19:58:23 --> Controller Class Initialized
50 DEBUG - 2006-12-30 19:58:23 --> Model Class Initialized
51 DEBUG - 2006-12-30 19:58:23 --> Helpers loaded: form
52 DEBUG - 2006-12-30 19:58:23 --> Table Class Initialized
53 DEBUG - 2006-12-30 19:58:24 --> Model Class Initialized
54 DEBUG - 2006-12-30 19:58:24 --> FTP Class Initialized
55 DEBUG - 2006-12-30 19:58:24 --> Helpers loaded: url
56 DEBUG - 2006-12-30 19:58:24 --> Model Class Initialized
57 DEBUG - 2006-12-30 19:58:24 --> Helpers loaded: date
58 DEBUG - 2006-12-30 19:58:24 --> Model Class Initialized
59 DEBUG - 2006-12-30 19:58:24 --> Model Class Initialized
60 DEBUG - 2006-12-30 19:58:24 --> Helpers loaded: form
61 DEBUG - 2006-12-30 19:58:24 --> Helpers loaded: url
62 DEBUG - 2006-12-30 19:58:25 --> Unit Testing Class Initialized
63 DEBUG - 2006-12-30 19:58:25 --> Validation Class Initialized

```

### 8.3 CI 的单元测试类

现在让我们开始做一些适当的测试工作：检测你的代码能否在不同的环境之下工作。

CI 使单元测试类和它的其它类一样简单。你以这装载它：

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

然后，为每个测试准备三个变量：

\$test—实际的测试内容，一般是一个 PHP 表达式

\$expected\_result—你期待的结果

\$test\_name—你想要显示的测试名称

针对 PHP 函数 floor() 的两个测试列在下面。(floor() 是 PHP 的取整函数) 注意到第一个预期的结果是正确的；第二个是错误的。（一个故意的错误）：

```

$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);

```

增加：

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

显示结果作为带格式的 HTML：

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

如果你想要你的系统分析或者保存它，使用：

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

返回一个你能使用的二维数组：

```

TEXT 代码
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 )
)

```

因此现在我们有一个得到测试结果的简单方法。

除了简单的等不等于这样的测试 (floor(1.56) 是不是等于 1?) CI 的单元测试类也测试数据类型 (is\_string、is\_bool、is\_true, 等等。-完整的清单在用户手册中)。你可以把下面的表达式：

```
$expected_result = 1;
```

替换成：

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

测试过程和前面的一样。

如果你在你的代码各处放上这样的测试代码，它可能会运行得很慢，而且将会在你的屏幕上显示所有的诊断。但是你能让它停下来。只要简单地把下列的代码加入你的构造函数：

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

而且（令人惊喜地）如果你将 FALSE 改成 TRUE，信息将再度显示，你甚至能动态地这样做。

#### 8.3.1 什么时候使用单元测试

事实上很少有人会去测试一个 PHP 的内置函数。但是用来测试你自己的函数是有价值的。观察它们是否可以返回正确的结果，需要担心的主要有：

他们的表现完美吗？

但是如果用户想在其他环境下运行，它还能正常工作吗？

或你将会写更多的代码，或修改现有的代码，造成你自己的函数不能正常地工作。

有时，出错是由于编程问题引起的，所以我们可以用编程来捕捉和修改错误。你能在用不同参数进行测试的过程中获得乐趣。

让我们回到我们那个运行一个数据库查询删除指定 ID 值的记录的那个例子。如果下面情况出现它会做什么：

ID 是 NULL，或者没有给出值？（特别地重要地，你可能偶然地删除表中所有的数据。）

ID 不是一个整数？（“x”，举例来说？）

身份证是一个整数，但是超出范围（你在你的表中有 1000 个记录，但是 ID 是 1001?）

ID 是一个负整数？

诸如此类，想出不同测试条件是有趣的。

在单元测试中把这些参数放进函数，并看看结果。当然，结果可能和你预料的一样。第一个情形和第二个情形会报错。你应该修改以阻止它发生。因此执行后，单元测试不能通过。

我们定义我们要从每个测试中得到的结果，如果结果和我们设定的一样，测试就通过。但是如果在测试过程中，程序抛出一个异常，后面的代码不再执行，我们如何能够让单元测试完成呢？这就要求我们必须先保证程序没有语法上的错误，让函数能够执行所有的代码。毕竟，单元测试不是用来对付语法错误的，这是 PHP 环境的工作。

上述假设的第三个情况是 ID 超出范围，这不是一个代码错误，数据库能安全地处理这种情况。但是，你可以在把查询发给数据库之前做一定的检测工作。或者你也可以让它运行一下，因为有可能使数据库返回一个出错信息，因此你需要用自己的错误信息来代替系统给出的信息，比如“对不起，现在系统正忙，无法提供服务”。

#### 8.3.2 单元测试的示例

让我们编写一些代码测试这个‘删除’函数。我已经建立了一个‘删除’函数（放在一个 model 中）以便我们来测试它，如果单元测试失败返回 \$dbvalue 上。

```

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

```

```
}
```

如果测试是成功的，一个简单的循环后在\$dbvalue 返回'OK'。测试代码很简单。首先，我们建立一个 ID 值的数组和我们期待的结果。换句话说，如果我们试着删除一个 ID 值为 '' 或者 'abc' 的身份证，系统应该抛出异常，如果 ID 是 1，或 9999，系统应该接受它作为一个有效的 ID，它将返回 'OK'。

因此数组的键名是你给定的测试条件而值是你期望函数返回的结果。

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

现在使用下列的代码循环传递\$numbers 数组的每个元素给 CI 的单元测试类来做所有的测试。

测试将运行 \$this->delete() 函数，记录你要删除的 'fred' 表中的记录和 ID 值 (\$testkey)。

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

记住，CI 单元测试允许你提供三个参数：

\$test：对于每个数组的键，我们把\$testkey 作为参数调用删除函数，数组的键就是给出的 ID 值。函数返回一个值。（在这里称为\$dbvalue）。我们的\$test 将使用 regex 比较那个值，我们期待它是\$testvalue，数组中键对应的值。（它包括 'OK' 或 'exception'？）

\$expectedresult 是 '1'，因为如果我们的代码是正确的，我们期待 regex 找到一个匹配。我们希望 'NULL' 返回一个 "exception" 而 '1' 返回 "OK"。

\$testname：这个参数是可选的：它是测试返回的字符串，用来解释我们进行什么测试，用什么参数

你可以从结果中看到，所有的测试都返回 'passed'，因此我们可以对我们的代码有信心了。（测试结果和预期结果的数据类型都是整数，即使我们的输入可能不是整数，因为测试实际上是一个正则表达式的比较，返回 1 或者 0）。

Test Name	exception at: E:\xampp\lite\htdocs\pack2\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:\xampp\lite\htdocs\pack2\system\application\models\crud.php
Line Number	615
Test Name	exception at: E:\xampp\lite\htdocs\pack2\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:\xampp\lite\htdocs\pack2\system\application\models\crud.php
Line Number	615
Test Name	OK at E:\xampp\lite\htdocs\pack2\system\application\models\crud.php:441 : doing delete on id of 9999
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampp\lite\htdocs\pack2\system\application\models\crud.php
Line Number	615
Test Name	exception at: E:\xampp\lite\htdocs\pack2\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:\xampp\lite\htdocs\pack2\system\application\models\crud.php
Line Number	615

充满乐趣，实际上也相当有用，测试结果存放在数组中。

例如，如果 ID 是一个数字，如果不是一个整数会有什么结果，用上面的测试，试验：

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

看看结果是什么？

结果会让你感到惊讶(我也是)：这个测试没有得到预期的结

果，它显示你的函数将把 3.5 当作一个函数。理由是 PHP 做一种松散的 '相等' 测试；如果你在这种情况下要得出正确的结果，你需要把比较设为 '严厉' 模式。为了要设定这个模式，使用：

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

## 8.4 CI 的基准测试类

这个类让你测试从一点运行到另一点所需要的时间。你在开始测试的地方插入一行代码：

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

在代码的另一行插入：

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

然后你插入第三行，告诉你执行的时间：

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

然后你能打印结果，\$fred，或做任何你想做的事情。

基准测试可以使用你喜欢的任何的名字，只要它们是不同的，而且你能设定很多组。你能使用这些测试看看是否你的代码执行需要太多的不正常的时间。如果你的页面装载时间太长，你可以插入一些基准测试识别符来测试引起延迟的代码块。

对于我们的网站监控应用的测试，我们对执行时间不是太感兴趣。当我们在英特网上登录进网站时，我们希望他们的速度是可接受的。每次执行时间略有长短其实没有多大意义。然而，如果我们在一些连续地测试同一个追踪基点，我们可能注意到它会有变化：这会为我们找到问题的原因提供一些线索。一个数据库查询可能费时很多；可能是我们的主机工作状态不是很稳定。因此为了达到我们的目标，我们将会采集 \$fred 的内容，并存放在数据库中。

## 8.5 CI 的评测器类

profiler 类非常精彩，你在你的类的一个函数中插入一行代码（它在构造函数中工作，因此放在那里才有意义。）这行代码是：

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

如果你改变主意了，你可以修改它：

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

插入这行代码得到的回报，就是你能在你的屏幕上得到一个完整的报告。显示 CI 超级对象和你的控制器的时间花销以及 \$\_POST 数组和数据库查询等等所用的时间。在开发阶段，这个特别有用。

如果你加上你自己的基准测试，它也会显示这些。你必须使用特别的名字命名你的基准测试-他们必须包括在 "\_start" 和 "\_end" 中，每一组的命名都要不一样：

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

并且，还有：

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

You are now logged out. Goodbye!

Username:

Password:

0

**BENCHMARKS**

Loading Time Base Classes	0.2237
Fred	0.0054
Controller Execution Time ( Start / Logout )	0.6031
Total Execution Time	0.8274

**POST DATA**

No POST data exists

**QUERIES**

INSERT INTO ci\_sessions (session\_id, ip\_address, user\_agent, last\_activity) VALUES ('b6406ba0eb6609446d663b17dccaab', '127.0.0.1', 'Mozilla/5.0 (Windows; U; Windows NT 5.0; en-GB; rv:1.166196520)')  
UPDATE ci\_sessions SET session\_id = ' WHERE session\_id = 'b6406ba0eb6609446d663b17dccaab'

你能见到，在这两个基准点之间的时间间隔被表示为 'fred'。

## 8.6 用“模拟”数据库来做测试

动态网站是围绕着数据库工作的。如果你正在测试它们，你应该测试你的代码是否能真正修改一个数据库。端到端（双向测试）测试这样做：举例来说，如果你的测试是你能否用正确的用户名/密码组合登录，你可能需要读一个数据库并实际这样操作。

但是测试你能否更新，插入，或是删除一个生产数据库上的数据是危险的，因为它会破坏你的真实数据！

记得：CI 可以声明超过一个数据库，而且可以容易地在他们之间切换，见第 4 章。使用这个功能，你可以很容易地建立一个模拟数据库，然后增加，改变，并且在它里面删除数据。

你也能使用 CI 建立并且删除表、或可能根据你的主机和权限建立和删除数据库。CI 的：

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

函数让你运行任何 SQL 查询，包括有点像这的：

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

能创建一个新表，或者象：

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

将插入一行数据到 fred 表中。

因此，通过几行代码，CI 让你建立模拟数据进行完整的测试，在不需要的时候删除这个模拟数据库。你可能针对 delete() 函数进行几次单元测试，看看这个函数在有不同 ID 值时是否正常工作。

现在你正在超越简单的单元测试。如果我们进行在我们的表中删除数据的单元测试，我们需要检查这些数据是否真的删除了。这容易地用下面的代码实现，再使用 CI 的单元测试类。和它的 AR 类：

```
$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` 把计算表中所有的结果，而且我们知道那里现在应该没有任何结果。你能容易地使用这种代码检查‘插入’操作，看看是否在操作后会有一记录保存在表中。

因为这是虚拟数据，我们特别地为测试而生成的，我们完全地知道该期待什么，而且做什么都没有关系。只是记得在测试后要把系统恢复到产品数据库上，否则会得到奇怪的结果。

## 8.7 控制和时间安排

测试是应用程序的心脏，因此重要的不是要不要测试，而是如何让测试有效。

你应该记得在第 4 章我们有一个建表的 SQL 文件，有一个表名叫 tests，另一个表叫 events，每次网站在进行一次测试运行时，它会在 tests 表中寻找两个字段：frequency 和 last\_done。如果 frequency 中的值是“hourly”，我们就再检查 last\_done，如果当前这个小时内没有做过测试，我们就启动测试，然后更新它的 last\_done 字段为本次测试的时间。

测试完成后，程序在 events 表中插入一条记录，这里会插入测试过的网站 ID，和各种各样其它的测试结果信息。这一张表提供我们网站和我们的客户网站的统计数据，挑出个体测试，或一个给定网站的所有测试，等等。

我们在这一章稍早的时候讨论的 benchmark 类还有一点没有涉及到：当你针对一个类似上面谈到的函数进行测试的时候，放入基准点进行计时是一个好主意，你可以得到测试花费多少时间的数据。把时间保存到 events 表中，那里有一个字段 timetaken，这是一个浮点类型放入测试花费的秒数。如果定期地评估测试用时可能会得出很多结论，比如如果花时较长，有可能是 ISP 超负荷运行，或者你的网页有太多的内容，还或者是你的网站变得十分出名，流量大增，你需要增加你的带宽。

不管哪种方法，经常地测试会让你及早地发现事故苗子，及早地排除故障。

## 8.8 总结

我们已经在测试上花费了很多时间。它不是最让人激动的内容，但是如果你们正在开发网站，测试是保证你晚上睡个安稳觉的好方法。

我们已经见到 CI 如何处理错误，如何显示出错信息，但是当你的网站正式开始运行后你可以关闭它们(或者停止日志操作)。

我们看了如何操作 CI 的单元测试工具。我们也看了 benchmark，它能简单地统计任一代码块之间的运行时间。

profiler 是一个在你编写代码时用来显示许多信息的相当棒

的工具。CI 提供了一系列开发测试你代码的好工具。

我们还学习了使用模拟数据库调试数据库交互部分代码的方法，它不会破坏你的生产数据库。

我们还能 CI 整合一些外部代码，让我们创建网页机器人进行远程网站的完整功能测试。



## 第九章 用 CI 通信

Internet 的主要优点是它的通信能力。本章介绍 CI 在三个方面使通信更容易。

首先,我们将会通过使用 CI 的 FTP 类存取远程目录中的文件,来增加测试工具包。

然后,我们将会使用电子邮件类在某些条件满足时自动地给我们发邮件。

最后,我们将会尝试进入 web 2.0 的领域,使用 XML-RPC 创建一个个人 WEB 服务,允许我们的远程网站响应并返回来自我们测试网站的信息。

### 9.1 使用 FTP 类测试远程文件

FTP 是在英特网上传输文件的方法。它通常使用一个特别的 FTP 程序来上传或下载网站的文件。对我们大多数人来讲,它只是在创建一个新的网站时偶尔用到。

不过你能使用 CI 轻松地实现全部过程。一种用途是检查你远程网站的完整性:文件还在那里吗?作为一个网站拥有者,你必须面对某人针对你的网站文件的一些特别企图。你的 ISP 或服务器管理员可能误删你的文件。(我本人就遇到过一次,我的 ISP 重建服务器时忘记重新安装我的应用文件)。这个文件不经常被用到,但是它与系统有很大的关系。这导致我花了一些时间去追踪这个有趣的错误!)

作为一个体现 CI 的 FTP 类威力的例子,让我们建立一个常规测试程序,用来检查远程网站的文件。有些代码是我们每个人都需要的:

```
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;
}
```

首先,如果还没有装载请先装载 FTP 类。然后,定义设置参数:主机名称(比如, www.mysite.com)、访问 FTP 使用的用户名和密码。

一旦连接建立,CI 的 FTP 类给你一些选项。在这种情况下,我们使用 list\_files() 返回在/my\_directory/目录下的文件清单。函数返回一个数组。而且你能容易地检查包含文件的数组以发现你要查找的文件。在这之前,我们正在尝试在一个数据库中列出我们所有的测试。因此这次我们需要列出 FTP URL 的清单(或者主机名称)、用户名和密码,没有使用正则表达式而是一个用于检出的存放文件的数组。为了确保数组的完整性,如果你要把它保存在数据库中,你需要在保存前对它进行序列化,取出后需要反序列化。

比较 getremotefiles() 函数返回的 \$remotearray 和从你的数据库返回的反序列化的 \$referencearray:

```
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;
}
```

PHP 的 array\_diff 函数比较两个数组。因此,它将会列出在第一个数组存在但第二个数组中不存在的文件,因此,运行这个函数两次,颠倒参数的次序:那样,你能得到两个清单,一个是不在你网站上的文件清单(但是应该在),一个是在你网站上文

件清单(但是不应该在)。你一个目录显示了你的 ISP 可能误删的那些文件。第二个清单是那些我们应该加入的文件。

CI 的 FTP 类应该允许你上传,移动更名和删除文件。假如你的测试显示你的参考文件数组中的一个文件丢失了(让我们假定它的文件名是 myfile.php),你可以 FTP 类来上传它:

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

在这个例子中,第一个参数是本地的路径,第二个参数远程网站的路径。可选择地,你能在第三个参数中指定文件应该如何被上传(作为 ASCII 或 binary)如果你不选,CI 会根据文件的后缀名自己决定。如果你正在运行 PHP5,你能增加第四个参数设定文件权限,这第四个参数用在你正在上传到一个 Linux 服务器。

要非常小心那个删除选项。就象用户手册中说的那样,“它将会递归地删除指定路径中的每样东西,包括子目录和所有的文件”,甚至写这一个段落都已经使我紧张。

使用 FTP 的删除函数和上传函数的一个组合,你能自动地在你的远程网站上更新文件。列出你需要更新的文件,而且依次访问每个网站,首先划除旧的文件,然后上传每个新的。

还有一有趣的 'mirror' 函数,让你在另外的一个服务器上建立网站的一个完全副本。

如果你正在运行 PHP5,FTP 类还有一个让你改变文件权限的函数。

正如你见到的,从测试你的远程网站到实际上维护或更新他们涉及到你的应用的许多方面。你可以,举例来说,写代码自动地进行更新。

### 9.2 机器之间的对话—XML-RPC

WEB 2.0 带来的革命是巨大的,体现在建立电脑对电脑的接口,允许 mashup 和 APIs 以及其它的好东西。

这是'网络服务'的基础。你能提供一个接口给你的网站,允许其它人用它为他们服务。举一个简单的例子,如果你建立一个'网络服务'对华氏温度进行转换,变成摄氏温度,客户用一个参数(要转换的温度)送出一个请求,而服务器返回被转换的值。因此,任何人都能增加一个温度转换功能在他自己的网站上,但是实际上是调用你的服务。

XML-RPC 让二部电脑直接地对话。接收网站创建一个简单的 API(应用程序接口)。任何一个想要与它交互的人需要知道那个 API-什么方法是可调用的,它们需要什么参数,语法是什么-为了访问它们。举例来说,许多主要的网站使用这个系统:Google,让你通过公布的 API 直接调用它的搜索引擎或 Google 地球。

建立你自己的个人 API 相对来说很容易,对 CI 说声谢谢吧。你需要建立两个网站来测试它,让它成为一件比大多数事情复杂一点的事情吧。一个网站(让我们称它为'接收'网站)提供 API,倾听请求,并回答它们。(在我们的例子中,这是我们正在尝试测试并管理的远程网站之一。)另一个网站使用 API 取回答案。(在我们的例子中,这就是测试网站本身。)

在 XML-RPC 协议,这两个网站通过高度结构化的 XML 进行对话。(名字 XML-RPC 由此而来-它是 XML Remote Procedure Call 的宿略词)客户将一个 XML 包送到'接收网站'服务器,告诉它想要使用的函数并传入一些参数。服务器解码 XML,如果它符合 API 的要求,调用函数并返回一个结果,封装成客户可以解码和使用的 XML。

你的 API 由接收网站提供的函数组成,并指导如何使用他们-举例来说,他们需要什么参数,这些参数应该是什么数据类型,等等。

在接收方,我们创建一个 XML-RPC 的服务器,确保内部的函数能为外部网站所用。这些'内部方法'实际上是你的控制器里面的正常的函数:服务器的角色是处理外部的调用和内在的函数之间的接口。

当你建立一个 XML-RPC 流程的时候,有二组问题:

获取需要对话的这两个网站

确定数据以一个适当的格式被传输

两者都充分利用多维数组,电脑能够十分容易地理解他们,虽然人脑反而会对多维数组感到有一点迷惑。CI 使它变得比较容易-虽然要做得正确还需要相当地睿智。

#### 9.2.1 使 XML-RPC 的服务器与客户端交互



首先，你必须在远程网站上建立一个服务器，和在请求网站上的一个客户机。这些通过几行代码就可以做到。让我们假定我们正在一个叫做 mycontroller 的控制器（在接收网站上）中建立服务器和在叫做 'xmlrpc\_client' 的控制器中建立客户（在请求网站上）。

在每个网站上，通过在构造函数中初始化 CI 类。分二步：对于客户端，你只需要装载第一行，在服务器端你需要装载这两行：

```
$this->load->library('xmlrpc');  
$this->load->library('xmlrpcs');
```

现在，在服务器一方，关闭你的构造函数，并且在 'mycontroller' 控制器的 index() 函数里面，定义供外部调用的函数。你通过建立一个“functions”子数组（在 CI 的 \$config 主数组中），映射进入的请求名和你实际要使用的函数：

```
$config['functions']['call'] = array('function' =>  
    'mycontroller.myfunction');  
$config['functions']['call2'] = array('function' =>  
    'mycontroller.myfunction2');
```

在这个例子中，有两个被命名的函数叫做“call”和“call2”。这就是 request 所请求的。（它不需要函数的名称，而是需要调用名。如果你愿意，当然，你能使用相同的名字。）对于每个调用，你定义一个子数组给控制器里面的函数-例如：分别是 'myfunction' 和 'myfunction2'。

你然后通过设定它初值并且实例化完成服务器的设定：

```
$this->xmlrpcs->initialize($config);  
$this->xmlrpcs->serve();
```

而且现在它已准备好侦听请求。

现在你需要去另一个网站-客户网站-并且建立一个 XML-RPC 的客户来发出请求。这应该在你的客户网站上的一个单独的控制器的。它相当短：

```
$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');
```

你定义接收网站的 URL，指定包含你想要的 XML-RPC 的服务器的控制器。你装载 XML-RPC 的类，定义服务器，和你想要调用的方法-这是你定义的调用的名字，不是函数的真实名称，如果你正在调用的函数需要参数，你这样传递它们：

```
$request = array('optimisation', 'sites');
```

就象你见到的，我们正在传递两个参数。

然后，如果一个结果已经返回，处理它：

```
if (! $this->xmlrpc->send_request())  
{  
    echo $this->xmlrpc->display_error();  
}  
else  
{  
    print_r($this->xmlrpc->display_response());  
}
```

最简单的做法是显示它；但是在一个真正的应用中你更有可能想要机器分析它，比方说，通过使用正则表达式，然后计算出结果。举例来说，如果结果包含一个错误信息，你可能想要在你的数据库中记录错误，而且采取行动向某个用户报告此事。

### 9.2.2 格式化 XML-RPC 数据交换

让我们使用一个真实的，也比较简单的例子。在这一节中，我们将会创建一个 XML-RPC 调用/响应，让你远程启动一个数据库的优化操作。

我们在上面写的客户端，正在申请调用一个方法名为 'call' 并带有两个参数：'optimisation' 和 'sites'。

接收网站的服务器把这个名为 'call' 的请求和一个实际的控制器中的函数 'myfunction' 建立起一个映射。

让我们大致看一下这个函数。它基本上是在控制器里面的一个平常的函数。它尝试优化一张 MySQL 数据库的表、并根据优化结果返回 'success' 或 'failure'。

```
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);  
}
```

注意 \$request，设定作为函数的参数。这包含来自客户的 \$request 的数组，它有两个值，'optimisation' 和 'sites'。CI 已经把数组转换成一个对象，\$request。因此你不能把它当做数组来处理，你必须使用 \$request 对象的方法 \$request->output\_parameters()。这返回一个原先的数组。

利用这个，我们已经告诉接收网站上的函数我们需要优化的表名，'sites' 表。我们也已经告诉它该调用函数 'optimisation'。它还带有一个参数叫做 'result'，获得所有的值后返回这三个值。

返回到客户网站的结果看起来有点像这：

```
XML 代码  
<?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>
```

```
</param>
</params>
</methodResponse>
```

(实际上没有缩进:我这样做是为了使结构变得更清楚。)

正如你看到的,我们的简单的三个词的结果(optimisation, exercises, success) 已经被装进如此复杂的分层标签中,在一定程度反应出 XML 可悲之处,精确地地告诉一部机器收到了什么。有三个<member></member> 标签对。每个有一个 <name></name> 对。('function','table','results') 而且每个都有 <value></value>对,包含了我们实际需要的信息(连同数据类型)-也就是 'optimisation','sites','success'。

不必介意我是否喜欢它。计算机因这种东西而有收获:一部机器需要精确、不含糊、和容易的数据特性。这一个章节是讲解计算机之间如何交谈的事的,不是有关如何方便真正的人类的。

现在,在你的客户网站上的 XML-RPC 客户函数能够解析出收到的数据。它可以方便地使用正则表达式来做到这一点,因为每个答案都清楚地被 XML 标签标示着。

注意 CI 如何让你免写许许多多的尖括号。

### 9.2.3 调试

一旦你开始测试你的客户/服务器组合,你或许会得到这样的信息:

#### TEXT 代码

```
The XML data received was either invalid or not in the correct
form for XML-RPC. Turn on debugging to examine the XML data
further.
```

加入下列代码打开调试功能:

```
$this->xmlrpc->set_debug(TRUE);
```

在你的客户端。这让你完整地看到你接收的信息。被警告,这是调试功能获取失败原因的地方。

有几个地方容易出错:

远程网站没有正确回应。(你可能暂时设定它显示错误,目的是为了要找出它为什么没有回应的原因。如果它是一个活动的网站,这是令人感到懊恼的。另外的,如果捕获 22 是它将会显示 HTML 格式的错误信息,返回不是你的客户端网站期望的 XML 格式信息,因此,你们将会收到由第一组所引起的第二组错误信息...) 对这个情况除错这可能涉及许多 FTP 来回传输,直到你达成正确的结果。

客户端代码可能工作不正常。

你把网址弄错了。(XML\_RPC 服务器上要求以 CI 的方式来定位控制器-也就是

http://www.mysite.com/index.php/mycontroller。如果你把所有的服务器代码放入控制器的构造函数而非在 index() 函数中,它会正常运行,但是你需要给出你要调用的函数的名称-比如:

http://www.mysite.com/index.php/mycontroller/myfunction)

XML 信息交换可能不完全正确。set\_debug 函数让你了解什么正在被返回,但是你能需要花好长时间盯着屏幕试着找出哪里出了问题。(相信我...)

然而,一旦调试正确了,你就会很有成就感。你已经在远程网站上创建了一个函数,可以让你远程调用它。

换句话说,你已经建立一个能维护和操作远程网站的应用程序。如果你有一些元程网站需要维护,你能容易地复制这些去访问它们,允许你(举例来说)每天一次在本地网站上优化你的数据库中的表。

### 9.2.4 XML-RPC 带来的问题?

安全当然是一个问题。你应该密码保护你的函数调用,所以客户在收到网站响应前必须发送一个密码作为一个参数。这能通过发送密码作为请求的一个另外的参数来做到,并且让被调用的函数在响应之前检查它是否正确。

如果你正在暴露重要的函数,你可能想要所有的事情在 SSL 层后面发生。我们的例子看起来无害处-如果一个电脑黑客反复地闯入到你的网站,你可能不介意,每次他所能做的就是让你的数据库表变得更“整洁”。另一方面,它会为“拒绝服务攻击”创造条件。

必须承认即使有 CI 的帮助,XML-RPC 容易出错,不易设置和调试。你需要一次性针对两个网站编程和调试,而且在它们之间传输数据的 XML 格式要求非常苛刻。甚至最小的错误不都可以犯。

有人主张 XML-RPC 是一种必须被替代的技术,可以使用其它高级语言编写的更新的接口或者 APIs,像是 SOAP(创建更费时)。

然而,以我们的观点,XML-RPC 是理想的。它让我们没有细节烦恼地运行远程网站中复杂的内部函数。

### 9.3 与人交流的工具: Email 类

我们已经在我们的测试网站中放入了许多代码。我们有一个测试的数据库,并且我们已经构建了许多函数进行不同类型的测试。我们能存取我们的网站和检查我们浏览的网页;我们能检查所有的文件是不是如我们所希望的正常地存放在远程服务器上。我们能自动地在网站上运行函数并让它优化自身。写使用这些工具进行测试的代码非常简单,如果我们需要,我们可以登入或可以使用一些定期执行软件,以适当的时间间隔运行我们的程序。

但是只是充分地进行测试并且仅仅在一个数据库中的保存结果是不够的。如果发生了故障,我们需要尽快地知道。

这就需要 CI 的电子邮件类上场了,每当特定的条件满足,它允许我们编制程序,把电子邮件寄给我们。你可能想要为每个失败的测试寄一封电子邮件,或者你可能想要进行一系列的测试,收集结果,然后再寄一封电子邮件报告。

使用电子邮件类,首先(一如往常)你必须装载它。

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

然后我们必须设定一些 config 参数。因为这个类仰赖它运行的服务器能够提供电子邮件的收发功能,所以我们可能在这里陷入麻烦。再一次,我们需要检查 ISP 是否提供这个功能。(因为 Xampplite, 举例来说,可能不能够提供你一个邮件服务器,也因此在本本地测试也会很困难。)

然而,如果我们已经确认 ISP 提供相应的电子邮件服务,我们能容易地配置电子邮件类。有许多选项,在用户手册有详细的列表。主要是:

protocol: 你的系统使用 mail, sendmail 还是 SMTP 发送电子邮件?

mailpath: 你系统的邮件程序保存在哪里?

你应该这样设定他们:

```
$config['protocol'] = 'sendmail';
$config['mailpath'] = '/usr/sbin/sendmail';
$this->email->initialize($config);
```

其他的选项,全部都有有意义的默认值,包括像字自动换行,字符集,个性组, text/HTML, 等等。设定可选项并使用它们正常工作是使用这个类唯一有点困难的部分。

一旦你已经装载类并且进行了初始化,使用它简单得不可思议:

```
$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();
```

将发送一封电子邮件寄给我,并抄送给我的客户,报告我想要的信息。

如果你需要发送超过一封电子邮件,在开始新的电邮前:

```
$this->email->clear();
```

只是确定你每次从一个干净的版式开始。

你也能使用电子邮件类发送附件。记住:附件一定要已经保存在发送电子邮件的服务器中,而且你必须指定路径,而且是绝对路径.给出路径和文件名,如同下列:

```
$path = $this->config->item('server_root');
$file = $path.'/my_subdirectory/myfile.htm';
```

然后仅仅增加这一行:

```
$this->email->attach($file);
```

放在这一行的前面

```
$this->email->send();
```

这简单的 CI 函数比直接用 PHP 编写容易得多，它处理所有牵涉到的协议，你甚至不需要知道他们的存在。

如果你包含这行：

```
$result = $this->email->print_debugger();
```

在你的代码中，打印出\$result，你将会得到一个有用的屏幕信息，如：

TEXT 代码

```
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: <462614219cla6@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-8
Content-Transfer-Encoding: 8bit
test message
hello world
--B_ATC_462614219d14d
Content-type: text/html; name="myfile.html"
Content-Disposition: attachment;
Content-Transfer-Encoding: base64
```

(等等)

如果哪里出了问题，那么侦错数据也将会返回任何的服务器错误信息。举例来说，如果在使用 SMTP 发送方式没有设定适当的主机或权限：

```
$config['protocol'] = 'smtp';
```

它不能发送信息，并且它会告诉我：

TEXT 代码

```
You did not specify a SMTP hostname Unable to send email using
PHP SMTP.
Your server might not be configured to send mail using this
method.
```

不过请记住，'sendmail'可能正在产生误导-如果它已经发送邮件到服务器上并返回一个成功信息，但是这并不表示邮件实际上被发送。（因为如果你设定错误的'mailpath'选项，'sendmail'可能报告它已经发出电子邮件，但它实际上没有。）CI 依赖邮件改善程序返回的信息，因此，它可能被愚弄。对电子邮件来说，检查它们已发出的唯一的方法是确定他们已经到达-这可不是这里讨论的主题了。

CI 的电子邮件类中还包括一些有用的选项，全部在用户手册中有说明。举例来说，你能设定它发送本文或者 HTML 格式邮件-如果你选择 HTML 格式，这个函数还允许你设定一个备用的文本信息给那些不接受 HTML 电子邮件的人。

你也能设定它使用不同的字符集，而且处理单词回绕。你能设定批处理的尺寸，如果你想要群发电子邮件到一个长长的电子邮件清单中，你的服务器将不能接受。（或者你的 ISP 会认为你是个垃圾邮件制造者。）

## 9.4 总结

我们现在已经用 CI 为我们的网站提供丰富功能建立了一些非常复杂的工具。

首先，我们用 CI 的 FTP 类简化了文件传输。最初，我们用

这个类检查指定的文件是不是正常存放在指定的网站上，或者意外地增加了文件，这是很有价值的检查，因为网站可能因为文件的变更而遇到问题，有时候是因为网站管理员造成的，有时候是由黑客造成的。这函数将定期地执行检查。CI 的 FTP 类也提供远程维护并且更新网站信息的可能性。

然后我们用 CI 的 XML-RPC 类开发我们自己的'web services'。这样允许我们调用一个远程网站上的函数，如有需要就传以参数，并且把结果返回给我们-就好像我们登录到远程网站而不是我们的测试网站。我们把它用在了优化远程网站的数据库上并返回报告给我们。再一次，我们已经超越我们最初只是要监控远程网站的目标。现在我们能够教他们检查和优化他们自己。

最后，我们使用了 CI 的电子邮件类，这让我们的测试网站能生成电子邮件。CI 的代码极其简单明了容易使用，如果它认为出现了问题，我们的网站就会发电子邮件通知我们。CI 使得建立并且发送电子邮件很简单，包括发送附件。



## 第十章 CI 如何帮助提供动态的数据

我们已经把许多想法放入我们正在构建的测试网站中，而且 CI 已经让这么一件复杂的事变得很容易。我们已经建立数据库，使用了 FTP 协议，构建了测试模块，并且开始用电邮发送测试结果。但是有的时候我们容易注重技术的东西而忘记评判一个网站好坏的标准是它们在处理数据上有多好，以及它们适合用户操作的程度。

当你正在构建一个网站的时候，有几个类是经常用来帮助解决问题的，特别地当它用来传递动态的数据给你的用户：

日期辅助函数转换不同的时间格式以及帮助你处理时区问题。

文本和 Inflector 辅助函数提供有用的功能操纵并且转换字符串。

语言类使编写以不同的语言显示相同的数据的网站比较容易。

表格类一帮助编写表格，不用写许多沉闷的 `<tr><td>`。

你能自动地为一个频繁存取的网页建立缓存。

它们中的每一个都能节约你的编程时间，并使你的网站看上去更专业(而且使它比较容易更新)。

### 10.1 日期辅助函数：转换和本地化日期

你知道哪些网站希望用户搞得懂机器日期？比如说 MySQL 的本地 'timestamp' 格式是非常有用的，但是它看起来非常粗糙，让你的网站用户看到它好像：

```
TEXT 代码
20070124161830
```

或者像：

```
TEXT 代码
2007-01-24 16:18:30
```

当然，大多数的人能猜出它的意思，但是它给你的网站一种不专业和未完成的感觉。CI 使用它的日期辅助函数解决这个问题，老规矩，需要装载：

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

而且立刻提供给你许多有用的函数。可以参考用户手册得到完整的资料。

日期能在许多不同的方法中被指定。CI 的 `standard_date()` 功能显示同一个日期给你十个方法：

```
TEXT 代码
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
```

你需要做的就是从中选择一个。举例来说：

```
$time = now();
echo standard_date('DATE_RFC822', $time);
```

也有函数可以在不同格式的日期/时间值之间转换。他们的名字是自明的，而且精确的语法在用户手册中有介绍。他们使你能够非常简单地转换。

举例来说：

```
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;
}
```

生成这样的结果：

```
TEXT 代码
Start with MySQL time
20070101120000
now convert to unix timestamp
1167652800
then back to 'human' time
2007-01-01 12:00 PM
now convert unix stamp to local time in Tehran
1167661800
and say that in human time
2007-01-01 02:30 PM
```

你可以得到在这些函数后面的许多非常有用的代码，而且他们使设置时区非常容易。

日期辅助函数还有 `timezone_menu()`，一个产生时区下拉菜单的函数。你能让它与数据库一起工作，让网站的用户在使用时选择一个时区，稍后可以显示出他们所在时区的准确时间。而实现这个只需要：

```
echo timezone_menu();
```

你会看到这样的界面：



看起来好像 CI 产生提供一个自动的方法去处理时区，在日期辅助函数中的 `now()` 函数。用户手册建议你在你的 `config` 文件中设置 '主时间参数' 为 'local' 或 'gmt'，使用：

```
$config['time_reference'] = 'local';
```

'local' 是默认值。如果你将它设定为 'gmt'，代码似乎会返回基于 PHP `mktime()` 函数为基础的系统时间；如果这样做无效，你可将 `config` 文件设定为 'local'，它返回以 `now()` 函数为基础的时间。

然而，这两个做法都依赖你的服务器：它必须被设定成正确的时间，而且它的默认时区一定要设定。(你能用 `phpinfo()` 检查这一项。)但是时区也可能没有设置，而且你的服务器可能也不与你在相同的时区中：举例来说，这对大的公司来说相当普遍。

所以 CI 本身不知道你的时区差是什么，虽然它可能取得到



你服务器的时区差。因此，如果你用了 `timezone_menu()` 取得用户所在区域的时区差，你可以以 `now()` 函数为基础，通过 CI 的日期函数将格林威治标准时间转换成他们的当地时区。你将需要找出他们的偏差，并且写出单独的代码来转换他们想要显示的时间。

## 10.2 使用文本和 Inflector 辅助函数

文本辅助函数有一系列的函数，可以帮助你用各种各样的方法操作文本。在用户手册上查阅更详细的介绍。我只是列举几个有用的东西。

`word_limiter()` 函数合理地截断你设定的长度的字符串。  
`word_wrap()` 按你指定的长度回绕文本。而 `word_censor()` 用指定字符串替换掉你不想要见到的字符串。

还有函数 `ascii_to_entities()` 可以帮助做到，在转换诸如 MSWORD 文本到 HTML 格式时出现奇怪字符时进行抑制。

Inflector 辅助函数提供函数将单数转换成复数，反之亦然，虽然它们处理不了不规则的单词像 'sheep/sheep' 和 'child/children'，也会犯一些错误，举例来说，将 'day' 变成 'daies'。他们还能处理 '骆驼' 风格、或处理在多个字之间的底线连接，然后再把它们转换回来。

你从这些处理中找到乐趣，比如：

```
function converttext()
{
    $this->load->helper('text');
    $this->load->helper('inflector');
    $mytext = "Mr Bill Gates is a man I like. He is a very
clever man and writes superb software";
    echo "$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";
}
```

将会给你这结果：

```
TEXT 代码
Mr Bill Gates is a man I like. He is a very clever man and
writes superb software
Censored, this might read: Mr Bill Gates is a man I ####.
He is a very #####
man and writes ##### software
His name could be written like this mr_bill_gates...or like
this mrBillGates...
```

如果你正在接受其它来源的文本并且需要转换它或检查它，这些函数可能极其有用。他们可能替你省下许多时间去写正则表达式。

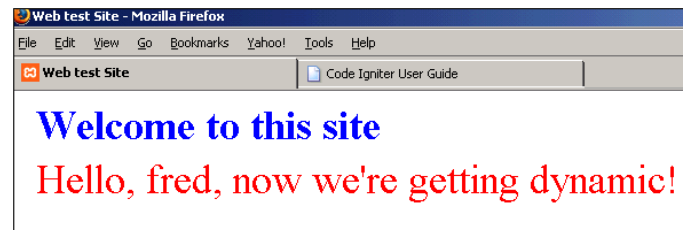
## 10.3 国际化：语言类

如果你正在编写一个可能被许多国家的用户访问的网站，那么，CI 能以多种语言为你显示网页。它像这样工作：

首先，你标记显示给你的用户的文本需要翻译。让我们回到我们在这一本书中讨论的动态的数据第一个例子。你的 welcome 页可能被在编写一个模型中的代码调用：

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

被赋给 \$data 数组的字符串是显示给用户的信息



但是你可能知道使用者是一位德国人，可以通过他或者她的服务器的位置，或可能地因为他或者她在登录时，表明了语言偏爱。如果你可以以德语向他或她问好，会很好。CI 提供一个容易的方法做到这一点。

首先，你需要建立一个语言文件。如果你曾经查看过系统文件夹，你将会看到那里有一个语言目录里面有一个英文子目录。依次包含一系列的文件-举例来说，`unit_test_lang.php`。这是一个 PHP 文件，很简单地定义了显示给用户的表达式数组：

```
<?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 etc
?>
数组的值是你想要显示的表达式，数组的键是你想要使用的识别他们的标记。文件名一定以 '_lang' 结束。
我们需要建立我们自己的，我们想要表示的每一种语言。让我们调用第一个 welcome_lang.php，并把它保存到 system/language/English 子目录，它看起来像这样：
```

```
<?php
$lang['welcome_title']     = 'Welcome to this site';
$lang['welcome_text1']    = 'Hello';
$lang['welcome_text2']    = "now we're getting
dynamic";
?>
```

数组的键名可以是你喜欢任何字符串；但是给他们加个前缀是一个好主意，为测试语言数组单元加上 'ut'，和 'welcome' 为我们刚才写的数组。他们都放入同一个基本数组，因此，如果你不注意地用同一键名对应两个值，第二个将会覆盖第一个。

最初的函数建立需要修改的网页。首先，你需要装载语言文件。在这个例子中，我已经把它包含在函数中，但是通常，把它放在类的构造函数中比较有意义。注意虽然文件名以 `_lang(welcome_lang.php)` 结尾，当你装载它的时候，你需要省略这个后缀（也就是你装载 'welcom'，而不是 'welcome\_lang'）。第二，你使用数组的键代码实际的文本-那就是说：

```
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);
}
```

但是这还是只给了我们和以前一样的页：它仍然是英语。如果我们想要翻译成德语，我们需要另外的一个语言文件。首先，我们建立一个新的子目录：在 `system/language/english` 的同一级，我们建立 `system/language/german`。在新的文件夹中，我们保存一个与英语版本一样的文件：`welcome_lang.php`。

这一个文件和英语的那个内容基本一样，至少在数组的左边。键名是相同的，但是数组现在在右边会放上德语。

```
<?php
```

```
$lang['welcome_title']      = 'Willkommen auf dieser Web
Seite';
$lang['welcome_text1']      = 'Guten tag ';
$lang['welcome_text2']      = 'jetzt sind wir
dynamisch!';
?>
```

(恐怕你必须自己做翻译-CI 不为你做那个!)

剩下一件事要做。当最初的 'hello' 函数装载语言文件:

```
$this->lang->load('welcome');
```

它没有指定哪一个语言, 因此默认值是英语的。你可以想象, 默认语言在 'config' 文件中被指定:

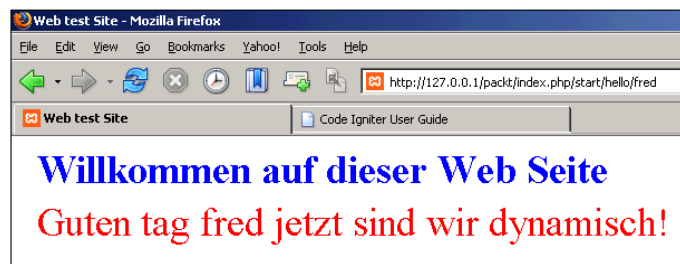
```
$config['language'] = "english";
```

为了要变成德语, 在 'hello' 函数的语言载入表达式应该附加指定为保存德语数组的目录。(非常合乎逻辑地, 这是 'german'。)

因此函数现在是:

```
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);
}
```

而且产生的页看起来就是个德文网站:



我们现在需要做的将确定我们的函数动态地装载正确的语言。

假定我们已经检测到用户的语言偏爱, 并且把它保存到变量 \$user\_language\_pref, 我们需要点东西来有条件地装载语言文件, 像这:

```
if ($user_language_pref == 'german')
{$this->lang->load('welcome', 'german');}
elseif ($user_language_pref == 'french')
{$this->lang->load('welcome', 'french');}
// etc etc
```

要训练自己编写这样的代码。你必须记住永远不要放真正的文本到你的代码中, 每次用语言文件产生一个输入。但是一旦你已经这样做了, 你就必须将语言文件让人转换成你的目标语言, 而且你的网站在翻译中神奇给出对应的语言显示。如果你改变网站的语种, 你只需要改变语言文件。如果使用某个表达式不止一次, 你不需要从整个网页查找需要需要的地方。

如果你的网站使用复杂文本的较长伸展, 用这种方法不太好处理。但是这样的长文本不会散布在每个网站的各处, CI 的语言类应该工作得很好, 并使你的网站更加令人印象深刻。

## 10.4 编写 HTML 表格的好方法: 表格类

我已经使用 CI 有好几个月了, 但是我还在发现使生活变得更容易的函数。

这里有一个好例子, 对任何一个花费许多时间写这种东西的人来说是个好消息:

```
echo "<tr><td>$value1</td><td>$value2</td></tr>";
```

CI 的表格类允许你自动生成 HTML 标记。让我们显示我们已

经运行的一些测试的细节。你需要装载这个类, 跟其它类一样。然后你能指定表格数据作为数组, 像这样:

```
$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);
```

但是当你从一个数据库查询中返回数据时你能自动地直接生成函数需要的数据。举例来说, 下面这段代码:

```
function dotable()
{
    $this->load->database();
    $this->load->library('table');
    $query = $this->db->query("SELECT name,type,time FROM
events");
    echo $this->table->generate($query);
}
```

给你的查询结果直接生成 HTML 表格。真是省事, 虽然默认的格式不那么好看!

name	type	time
not set	test	1166627335
not set	test	1166627882
not set	test	1166628621
not set		1166700269
ete	test	1166700394
ete	test	1166700928
etc	test	1166701193
ete	test	1166702238
ete	test	1166702361
etc	test	1166702400
ete	test	1166703348
ete	test	1166703451
etc	test	1166703835
ete	test	1166703851
ete	test	1166703928
etc	test	1166704104
ete	test	1166712129
ete	test	1166712176
etc	test	1166712209

代码只有四行, 就完成了把数据从数据库表中读出并以 HTML 表格形式显示。事实上, 当我想到我过去一直花费大量时间写实现类似功能代码的时候, 眼泪都要下来了:

### HTML 代码

```
<table>
<tr><td>$variable1</td><td>$variable2</td></tr> //etc.
```

同时, 你能看到, CI 的默认输出格式不是太精彩, 你能设定你自己的模板, 可以使用 CSS 风格做出你想要的效果, 而且函数会忠实实现它。模板是放在表格类里的一个数组, 因此你需要重新设定它, 在每一次调用这个类的时候。

```
$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);
```

有默认的模板数组, 看起来像这个一样, 函数基于这个数组而设计。注意有两组行定义 (row 和 row\_alt), 以防万一你想要进行颜色轮换。

如果你提交对模板的修改, 函数会作出改变, 产生不同的 HTML 表格效果。

你将会注意到模板只是一个数组, 而且你只是通过修改每个键对应的值来改变输出效果, 举例来说, 如果你有一个 CSS 文件, 里面定义了一个叫做 mytable 的类, 你能这样做:

```
$tmpl = array ( 'table_open' => '<table
class="mytable">' );
```

你不一定要改变每个值: 那些你不改变的键保留为默认值。现在你的表格神奇地以你新设定的格式显示。

## 10.5 缓存网页

到现在为止, 我们正在写一些相当复杂的代码。服务器必须停下来分析每个动态生成的网页。你可以很简单地编写一个函数象上面的 dotable(), 但是, 可怜的老服务器不得不做更多的工作来生成结果。

有时候, 这能导致你的页面显示起来比较慢。对这点可能没有什么好办法。如果你正在写报告, 每次写得都不一样, 你只能等待。然而, 你可能生成将会保持一段时间内容不变的网页。比如说一篇博客, 在你提交下一篇之前, 你没有什么变化, 如果有一千个读者来看你的博客, 每个视图都是一样的, 而你却需要浪费时间去动态生成同样的页面, 一次又一次。

解决之道是缓存网页。你一次生成网页, 然后把生成的 HTML 文件保存在缓存目录中, 加上一个时间戳, 然后被显示到用户的浏览器上。然后, 当下一个读者请求页面时, 系统检查离上次生成和保存有多长的间隔, 如果还在你设定的时间范围内, 它继续返回缓存页面, 如果不是, 它将更新页面。

听起来这里像需要一些复杂的编程工作, 其实如果你使用 CI, 你只需要做两件事情:

找到你网站中的 /system/cache 目录, 它应该是空的, 除了一个 index.html 文件。确定此目录是可读写的-也就是如果在 Linux 系统上, 权限设为 666。

插入, 在一个产生 HTML 页的控制器的函数中加上:

```
$this->output->cache(5);
```

5 是你想要你的缓存持续的分钟数, 即在页面被重新生成前会持续多长时间使用静态 HTML 文件。

搞定。如果你现在装载函数, 你将会像往常一样见到页面被装载。如果你现在观察你的 /system/cache 目录, 当然, 你将会在那里里面见到一个新的文件, 有一个无意义的名称。

把它在一个文本编辑器里打开, 你将会见到它包含 HTML 代码, 加上一个时间戳。如果你请求相同的页, 在五分钟之内, 你将会得到被缓存的页面。超过这个时间段, 你的下一次请求将会自动地删除它而且用一个较新的版本替代它。

如果你不想缓冲页面, 从你的控制器中删除 this->output->cache(5) 这一行, 你的页面将每次刷新。(最后一个被保存的文件将会留在你的 /system/cache 目录, 直到你手工删除它。) 如果你想要继续缓冲, 但是偶然地删除一个缓存文件, 不要紧张; 当那一页下一次被调用时, 系统将会产生新的。

CI 让这一切变得快和简单, 它正在试着缓存每一页! 只是要记住这一点: 要缓存内容不频繁改变的页面, 其它情况就不要使用缓存功能。

## 10.6 总结

CI 给你提供了很多好东西, 使你编程更容易并且让你的网站更专业。本章介绍了下面五个东西:

文本和 Inflector 辅助函数提供有用的函数操作并且转换字符串。

日期辅助函数让你在不同的日期格式之间转换以及应付时区。

语言类使你编写多语言网站比较方便, 满足使用者语言偏爱比较容易。唉, 你仍然必须自己做翻译!

表格类让你输出适当的 HTML 表格, 直接地从一个数据库查询中产生的数据可以输出到漂亮的 HTML 表格中。

自动地缓存高负荷动态网页提供一个较快速的响应。



## 第十一章 使用 CI 处理文件和图片

本章看几个有用的 CI 功能和辅助函数。他们中的每一个都是如何用几行 CI 代码无缝存取一系列应用和动作的好例子，如果从头开始编码的话你需要具备许多专业知识。在许多情况下，CI 提供一个对已有类的简单接口，这些类可能是你从 PEAR 或者其他源代码中提取的。但 CI 给你一个标准接口：你只需把它作为本地 CI 代码，并且框架向你提供所有接口的内容。

让我们看看本章中的五个内容：

文件辅助函数让我们很容易的读写文件。

下载辅助函数让用户直接下载你网站上的文件，而不是显示他们为 HTML。

文件上传类以其他方式工作，他允许用户存储文件到你的网站，并有内建的安全措施以限制用户的行为。

图像处理类允许你给图片做几个有用的处理，我们将看看如何调整图片的大小和给图片加水印。

最后，Zip 类允许你的用户下载文件之前压缩它。

这里的每一个例子都暗藏着许多聪明的代码实现，并且允许你以最小的代价写出实际的应用。在很多例子中，增加了一些额外的代码以使其更健壮。

让我们一个接一个的看看：

### 11.1 文件辅助函数

第一次学习 PHP 的读写文件的语法是不容易掌握的。CI 的文件辅助函数包含许多有用的函数，他封装了 PHP 自己的文件处理操作。开始一如既往的装载这个辅助函数：

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

然后生活得到了很多简化。例如，写文件，你需要知道的是：你的文件的位置。

你想写入的文本。

你想以什么模式打开文件。模式定义在 PHP 手册中（看“fopen”这页）。它们包含“r”为读，“w”为写（写入文件，覆盖已存在的数据），“a”为追加（写入文件，在已存在的数据后添加）。每种情况下，添加一个“+”，例如“a+”，打开文件以进行读写操作。“a”和“w”，而不是“r”或“r+”，如果没有的话，总是创建文件。

然后你使用这三个信息作为 write\_file() 函数的参数：

```
write_file('e:/filetest.txt', 'hello world', 'a');
```

这比 PHP 的两步代码更简单、更直观：

```
if ( $fp = fopen('e:/filetest.txt','r'))
{
    fwrite($fp, 'hello world');
}
```

其次，CI 代码添加了一点额外功能：它在写入之前自动锁定文件，写入后再解锁。如果没有发生文件操作，则辅助函数返回“FALSE”，所以你可以用它来报告成功或失败。你需要为你的文件指定一个文件名，但如果你不指定文件路径，它将保存在你站点的 web 根目录下，就是你的主 index.php 文件所在的位置。

当然，在任何文件夹中你创建或写入文件必须具有写入权限。记住，如果你运行于一个 Windows 系统，你必须使用正斜杠 — / — 来描述你的文件路径。

在我们的应用程序中，我们可以把数据库工具类与这个辅助函数组合起来。这允许我们创建、备份、修复和优化数据库和表，虽然只适用于 MySQL 和 MySQLi 数据库。与文件辅助函数一起使用，你就可以创建一个不错的备份例程。

```
$this->load->dbutil();
$backup =& $this->dbutil->backup();
$this->load->helper('file');
write_file('e:/mybackup.gz', $backup);
```

上面的代码将保存服务器上的最新版本的数据到文件。

再次读取文件同样简单：

```
$content = read_file('e:/filetest.txt');
```

还有个功能是，返回指定目录中的所有文件和文件夹，并以数组形式表示：

```
$filenames = get_filenames('e:/');
```

不过，如果你在有很多文件的文件夹中使用它的话，你可能会发现 PHP 在把目录读取完毕前就超时了。你可以使用这个简单的代码片段，来检查你想要的文件或文件夹是否存在。首先，使用 CI 的函数查找文件，并定义一个要查找文件的数组，然后使用 array\_diff() 比较它们。给出两个数组，array\_diff() 告诉你在第一个数组中，但在任何其它数组中的值，所以，你需要两次调用它，把每个数组都放在前面。

```
//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, $files_expected);
echo "<br />Missing files are:";
print_r($difference);
// any expected that we didn't find?
$difference = array_diff($files_expected, $files_there);
echo "<br />Extra files are:";
print_r($difference);
```

最后，有个比较“恐怖”的函数 - delete\_files()。删除指定目录里的所有文件：

```
delete_files('c:/mydirectory/');
```

将删除“mydirectory”中的所有内容。如果你添加可选参数为“TRUE”：

```
delete_files('c:/mydirectory/', TRUE);
```

这同时会删除目录下的所有子文件夹，多加小心。想象下面发生什么？

```
delete_files('c:/', TRUE);
```

要不你试试？！

### 11.2 下载辅助函数

下载辅助函数库里只有一个函数，但他是文件辅助函数很好的补充。你可能创建了一个文件，而希望将之以文本文件的方式展现给访问者，而非一个网页。

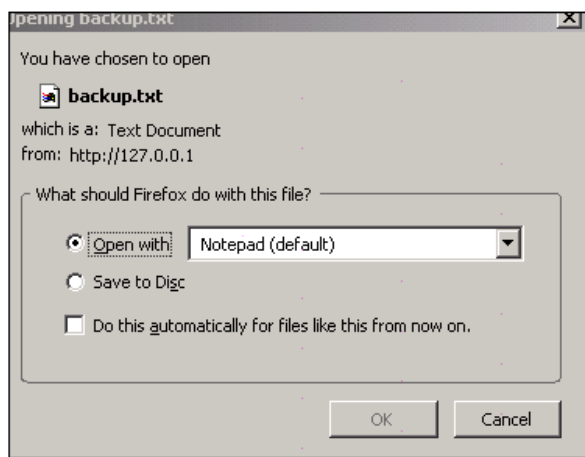
一个很好的例子是一个数据库备份文件，就像我们刚刚创建的那段代码。

为了在数据库崩溃的时候重建它，我们需要一个 MySQL 格式的文本文件。屏幕上的东西对我们来说没多大用处：

```
## TABLE STRUCTURE FOR: ci_sessions # DROP TABLE IF EXISTS
ci_sessions; CREATE TABLE `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; INSERT INTO
ci_sessions (session_id, peopleid, ip_address, user_agent, last_activity, left,
name, status) VALUES ('0ee2555fd8f6e6714cab86365f5c6712', 2, '127.0.0.1',
'Mozilla/5.0 (Windows; U; Windows NT 5.0; en-GB; rv:1168785860, 0,
'David Upton', 9); ## TABLE STRUCTURE FOR: domains # DROP TABLE IF
EXISTS domains; CREATE TABLE `domains` ( `id` int(10) NOT NULL
auto_increment, `url` varchar(100) NOT NULL, `name` varchar(100) NOT
NULL, `registrar` varchar(100) NOT NULL, `datereg` 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`
```

我们需要找到一种下载文件的方式。换句话说，在 Windows 系统中，我们想看到这个对话框：





为了在 Internet 连接下编码这些内容，你必须在 HTTP 头中指定页面类型。CI 的下载辅助函数帮助你在后台完成这些事情。装载辅助函数：

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

像下面这样使用其唯一的方法：

```
force_download($name, $data);
```

\$name 是被下载文件的名字，\$data 是文件的内容。如果你想下载已存在的文件，那么你需要先把它读到字符串中：

```
$data = file_get_contents("e:/mybackup.txt");
$name = 'backup.txt';
force_download($name, $data);
```

文件的内容可以直接重建 MySQL 数据库。你也可以使用这个辅助函数直接下载报告，无需强迫用户把他们从屏幕上摘抄下来。

在幕后，辅助函数帮助识别 MIME 类型并设置 HTTP 头。他依赖于其中的“配置”文件，system/application/config/mimes，下一节将看到上传类也使用它。这个配置文件存储 MIME 类型和对应 HTTP 扩展名的数组—例如：

```
'rtf' => 'text/rtf',
'text' => 'text/plain',
```

这个方法能节省你去记忆他们的时间！

如果你经常使用的文件类型没有包括在 CI 的列表中，那么你可以很容易的将它们添加到“配置”文件中。

### 11.3 文件上传类

有时候，你想允许用户在你的网站上传文件。这些可能是文本、图像或更多其他文件类型，如 MP3 音频或 MPEG 视频。这是一个比我们刚刚讨论过的文件下载更复杂的过程，但 CI 的文件上传类可以帮你完成大多数的工作。它也能处理一些安全问题。不过，你应该总是三思而后行允许任何人上传文件到你的网站，你可能想要保护上传页面，以防止未经授权的用户上传。

首先，您需要在服务器上分配空间（文件夹），以存放上传的文件。这个文件夹必须已设置了正确的权限，以允许用户写入。（例如 Unix/Linux 系统上的 777）。让我们假设你新建一个名为“uploads”的文件夹，并将它放在 WEB 根目录文件夹中。

装载 CI 的文件上传类：

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

然后你需要做三件事情：

设置默认值

创建一个控制器来处理上传。

给你的用户提供一个上传表单和“成功”表单。

让我们按这个顺序设置他们。首先，设置一系列默认值。创建一个 \$config 数组以完成这件事。

让我们将你刚创建的目录设置为上传目录。这样写：

```
$config['upload_path'] = 'uploads';
```

这行代码可以写在控制器中，也可以在 config 文件夹中创建一个包含这行代码的 upload.php 文件。

(system/application/config/upload.php)。

```
<?php if (!defined('BASEPATH')) exit('No direct script
access allowed');
$config['upload_path'] = 'uploads';
?>
```

理解这两种设置默认值方式之间的区别很重要。如果你在 config/upload.php 文件中设置了默认值，你就不需要专门进行初始化文件上传类的工作。只要装载他，他自己就能找到默认值。

但是，你可以在加载类的时候在第二个参数中指定该控制器的默认值，像这样：

```
$this->load->library('upload', $config);
```

\$config 是默认值数组的名字。（不要试图在 config/upload.php 文件和控制里设置默认值！）

这种处理方式显得不够优雅，所以 CI 推荐使用更合理的方式组织配置文件。下面是几个要点：

上传文件的位置：CI 不做任何假设，你必须告诉他。

允许用户上传的文件类型。像这样设置：

```
$config['allowed_types'] = 'gif|jpg|png';
```

使用管道操作符 (|) 分隔允许的文件类型。上面的设置将允许在你的网站上传大多数的图像类型的文件，但他将不允许上传音频文件。设置这个参数是一个基本的安全措施：如果你只想让用户上传图片，而不允许上传可执行文件或大型 MP3 文件。注意：你必须在允许上传文件之前设置一个值：默认设置（例如：不设置）不允许上传任何文件。

Max\_size：如果你不想恶意用户填满你所有的空间，那就明确的设置可上传文件的最大文件大小（KB）。默认值是 0，表示没有限制。

覆盖：如果用户正在上传的文件，和上传文件夹里的文件同名的话，旧文件将被覆盖并永远丢失吗？这取决于网站的实现和允许用户上传的原因。CI 的默认值是“FALSE”，意思是不覆盖旧文件，并用一个新名字保存文件。如果你想新文件覆盖旧文件，就明确设置这个默认值为“TRUE”。

注意：CI 不会自动通知用户他或她的文件已被重命名，所以可能会让用户产生困惑：阅读下面的内容，来学习如何取得上传过程的报告。

你还可以设置图片尺寸，宽和高的默认值；加密文件；清理标题上的空白。

现在你已经确定默认设置了，接下来你需要一个上传控制器。这非常简单。他的作用是初始化上传类、从用户表单接收上传，然后确定上传是否成功。如果成功则显示一个报告；如果失败则返回一个带错误信息的上传表单。非常简单，他只包括一个方法：do\_upload()，像这样：

```
<?php
/*constructor function to initialize controller and load the
file upload 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);
    }
}

```

此功能需要一个 upload\_form 视图和 upload\_success 视图。先使用表单辅助函数创建一个表单，并指向“upload”控制器的 do\_upload 方法：不要用下面的函数“打开”：

```
echo form_open('upload/do_upload');
```

（第 5 章里介绍的表单“打开”方式），而是用表单辅助函数的 multipart 函数“打开”：

```
echo form_open_multipart('upload/do_upload');
```

（记住：我们是在用代码来产生 HTML 标签，所以需要显示到屏幕上。）

然后，使用表单辅助函数的 form\_upload 函数替换 form\_input：

```
echo form_upload($data);
```

这两行代码可以帮你完成很多乏味的工作。

添加一个提交按钮，并“关闭”表单。

```
echo form_submit('mysubmit', 'Submit Post!');
```

传递 \$view 变量到视图并装载他。当 do\_upload 函数传递 \$error 到视图中时，你的视图也应该显示 \$error 变量。

```
echo $error;
```

你现在看到的页面应该是这样的：

单击“浏览”查看用户电脑上的文件（本地），而不是服务器上的。一旦他（她）选择了一个文件并单击上传按钮，将调用上传控制器，然后文件将被传输到服务器上的上传文件夹中。

当我们尝试上传一个文本文件时，（记住，我们只允许“gif | jpg | png”文件类型。）我们将看到：

CI 返回一个类型错误：这是控制器中的 \$this->upload->display\_errors() 方法在起作用，他在视图中添加了一个错误消息变量。

你也可以让 CI 报告上传已成功。正如你看到的，如果上传成功，控制器就装载一个名为 upload\_success 的视图。\$this->upload->data 方法的内容将被传递到这个视图中。他返回上传过程的所有信息的数组：可能多于你想要显示的信息。

我上传一个名为 waltzer.jpg 的文件：默认报告看起来是这样的：

## Your file was successfully uploaded!

- ◆ file\_name: waltzer1.jpg
- ◆ file\_type: image/jpeg
- ◆ file\_path: E:/xampplite/htdocs/packt2/uploads/
- ◆ full\_path: E:/xampplite/htdocs/packt2/uploads/waltzer1.jpg
- ◆ raw\_name: waltzer1
- ◆ orig\_name: waltzer.jpg
- ◆ file\_ext: .jpg
- ◆ file\_size: 562.08
- ◆ is\_image: 1
- ◆ image\_width: 1600
- ◆ image\_height: 1200
- ◆ image\_type: jpeg
- ◆ image\_size\_str: width="1600" height="1200"

[Upload Another File!](#)

如果你想创建一个像 Flickr 那样的站点的话，那这些信息可能会影响用户上传他们的图片！然而，你可以很容易的去掉上传控制器中你不需要的任何信息。

请注意，顺便说一下，在写这个例子的时候，我设置文件上传类的“overwrite”值为“FALSE”，然后上传了两次 waltzer.jpg 文件。

上面的截图显示的是 CI 关于第二次上传成功的报告。你将看到文件已被重命名为 waltzer1.jpg。如果查看上传文件夹，你就会看到两个文件，一个是原来的 waltzer.jpg 文件，另一个是新的 waltzer1.jpg 文件。在你的应用里，你可能想比较 raw\_name 和 orig\_name 的值，并告诉用户文件名已被更改。

CI 不比较两个文件的内容，只比较文件名。如果你允许多人上传文件，那么，其中的两个人，将很有可能在无意中，使用相同的文件名来上传不同的文件，并且，你可能不希望丢失第一个文件。另一方面，如果你通过网站上传名字总是相同的文件，你也许宁愿只在网站上保存最新的文件，在这种情况下，覆盖文件是一个简单的节省空间的方法。

顺便说一下这个图片，我们将在下一节用到它。



## 11.4 CI 的图像类

如果你允许用户上传图片，你还需要看看 CI 的图像处理类。他支持 PHP 最流行的三个图像类库：GD/GD2、NetPBM 和 ImageMagick。（使用 phpinfo() 查看你的服务器是否支持这些类库。）虽然图像水印效果只支持 GD/GD2：

图像处理类允许你完成图像的 4 个基本功能：

缩放：在你的屏幕上，你可能想把它们的大小设置为标准尺寸；或者你想设置成“缩略图”的大小。

裁剪

旋转

水印（只可用在 GD/GD2）：它经常用于在图片上放置版权标志，这样可以防止别人从你的网站上下载图片，然后占为己有，当作他们自己的原创作品。

这些功能中最有用的可能就是缩放了，那么一会儿我们来详

细看看吧。裁剪和旋转用处不大，除非你能在屏幕上看到图片。要做到这一点，你需要某种形式的用户界面，使用户可以指定她或他想做的事，并控制 CI 实现这些功能，并且，你需要自己编写这些功能代码！

让我们假设你已经使用刚才讨论过的文件上传类，上传了 wltzer.jpg 图片到 /uploads 文件夹中。（上传和处理图片需要设置这个文件夹的权限为 777，因为 CI 需要保存处理结果到这个文件夹中。）

首先，装载类库：

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

然后，你需要设置几个配置信息。（与文件上传类一样，你可以写在代码里，也可以放到单独的 system/application/config/image\_lib.php 文件中。）

这里有一些配置，并且他们已列在《用户指南》里了。或许最重要的是：

选择你使用的图像库。默认是 GD2，所以，如果你的 PHP 安装的不是这个库的话，你需要指定一个，例如，  
`$config['image_library'] = 'ImageMagick'`；（你也应该提供 ImageMagick 库的路径：`$config['library_path'] = '/mypath'`；）

你要处理的图片。应该提供路径（相对于网站根目录）和文件名。

处理后图片的尺寸—“x”表示像素数，设置宽度：

`$config['width'] = x`；，设置高度 `$config['height'] = x`；。

要调整图片尺寸，只这些还不够，需要用已调整尺寸的图片覆盖旧图片文件。代码如下：

```
function do_image($image_name)
{
    $this->load->library('image_lib');
    $config['image_library'] = 'GD';
    $config['source_image'] = "$image_name";
    $config['width'] = 140;
    $config['height'] = 104;
    $this->image_lib->initialize($config);
    if(!$this->image_lib->resize())
    {echo "failed";}
    else{echo 'success!';}
}
```

这个库可以做其他一些聪明的事情。如果你不想覆盖原始图片，可以为新版本指定新名称和文件路径，添加：

```
$config['new_image'] = 'newfolder/newname.png';
```

或者，如果你想创建一个图片的缩略图，简单地添加：

```
$config['create_thumb'] = TRUE;
```

执行的效果是：用默认的后缀（\_thumb）重命名新调整的文件，把 waltzer.jpg 改成 waltzer\_thumb.jpg。（你可以很容易的修改后缀--参考《用户指南》。）所以，现在有两个文件在同一个文件夹中：原图和缩略图。

注意：缩略图设置什么也不做，你还是需要设置你想要的尺寸。

这个图片被缩小到 140x104 像素：



图像类的附加功能可以给你的图片加水印。所以，如果你把自己的精彩照片放到网站上，你也可以添加一个版权声明。

虽然添加水印功能有很多选项，在《用户指南》里有详细说明，但基本的代码很简单。初始化类，告诉它想要加水印的图片和水印的内容，然后调用 watermark 方法。

```
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';}
}
```

（wm\_type 选项设置为 text，则允许你添加文本水印。否则，设置这个选项为 overlay，并提供一个叠加在你原图上的图片。）现在图片看起来像这样。



我的实际代码比上面的例子复杂的多，以便让我能控制大小和水印的位置，使之更清楚的显示在这个页面上。默认的代码如上所示，将足以应付大多数的用途，但水印太小，打印出来后无法看清。要了解更多关于使用外部字体的信息，请参考 CI 的《用户指南》。

这个类使用起来非常的简单。只有当你查看类的代码（在 system/libraries/Image\_lib.php 文件中）后，才能体会到 CI 帮你节省了多少时间！

## 11.5 用 CI Zip 类压缩文件很容易

如果你要移动像图片这样的大文件的话，你可能需要压缩它们。CI 包含一个实现这个功能的，并且便于使用的类库。

让我们拿出刚用过的照片：waltzer.jpg。它在我们的 /uploads 文件夹中。

与以往一样，你要先初始化 Zip 类。然后，你要告诉 CI 你想压缩成什么文件，并创建要压缩的文件。接下来，使用 read\_file 方法读取这些文件并压缩它，最后用 download 方法下载到你的桌面上。

```
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');
}
```

CI Zip 编码类比上面说的更复杂，并有数个选项。一如以往，他们都刊载于《用户指南》中。但也应该给了你一个怎样用 CI 简单制作 zip 文件，并从网站上下载它的例子，以便尽量减少带宽消耗并为用户节省时间。

## 11.6 总结

本章集中说明了一些 CI 辅助函数和类库的功能，他们可以提供：

使用最少的代码进行文件读写操作，并且可以自动处理锁定和解锁文件。

自动处理 HTTP 头，而不用担心如何处理是否要显示页面或



者下载文件。

上传文件到服务器，并进行一些安全上的设置，比如说上传文件的尺寸和类型。

更方便的图片处理，如缩放和添加水印。

处理下载请求前先对文件进行压缩以节省带宽。

对我来说框架就是这样，代替你完成许多乏味的编码工作，他会给你一个标准且易用的接口，并且为你考虑细节问题。

## 第十二章 产品版本、升级和重大决定

伟大的一天终于到来了！你的网站在本地环境中已经运行得足够好，是时候把它上传到远程服务器上使之成为一个正式运行的网站了。这件事按说很容易：上传所有的文件，包括系统文件夹的全部、更新 config 设置，复制完成后连接到数据库，以及其他。有时候，这的确很容易。

在你把重要的一切呈现在风投或公众视野的前夜，也就是在还没完成时。此时，为了防止出现意外，这一章将告诉你：

你该在 config 文件中设置什么

出现问题需要的一些诊断工具

可能出现问题的服务器与本地服务器的隐性差异

几点安全提示，你将身处于大千世界

然后，这一章还包括了升级以及 CI 在最近几年的一些更新。它稳定吗？当要提交一个重要网站时你该如何选择？你该如何操作？一旦你的网站开始运作，而 CI 推出了新版本你又该如何应对？

最后，我们简短地讨论如何针对 CI 的核心做你的自定义修改。它就在那里，它是开放源代码的，是可以修改的——但是否可行是另一码事。

### 12.1 连接：检查 Config 文件

系统错误通常是在接口上的。这就是为什么需要 config 文件：给你一个地方放那些接口。如果你还没这么做，你已经错过 CI 最主要的优点之一。

主要的接口问题可能是：

#### 12.1.1 URL

CI 通过查找文件进行工作。当用户连接到 index.php，然后开始运行整个程序。至少，它应该运行。确定你在 config 文件中已经正确地设置好 Web 地址和其它服务器地址。Web 地址是你的网站根目录。而服务器地址则可能要去问你的 ISP，通常在他们提供的“文件管理”菜单中也有。

我尝试设置子域时曾经遇到过一些特殊的问题。虽然许多主机允许这么做，但将域直接映射到目录也许并不是你所希望看到的。

#### 12.1.2 数据库

设置并连接到数据库是一个主要议题。查看你的 config 文件与 config/database 文件。你需要确定有正确的网站地址与服务器地址，正确的数据库名、地址、用户名与密码。小心前缀——它有时会自动添加的，比如你的网站叫“fred”，你的数据库名为“mydata”，而你的用户名是“mylogin”，但是在服务器上它们会叫做“fred\_mydata”、“fred\_mylogin”等等。

有时，它会在数据库中创建一个新的用户，并设定了登录用的用户名和密码，即使你已经有一个了。我也不知道为什么，但就是会这样。

在 config 文件中，你可以设定 CI 接受不同的 URL 协议类型，以决定服务器如何处理 URI 字符串。默认值是：

```
$config['uri_protocol'] = 'auto';
```

如果它不能正常工作，还有其他四个选项可以试试看。如果没有选择合适选项的话，你可能会发现你的网站不能完全正常的运作，（例如）表单不能调用目标页。

#### 12.1.3 其它 config 文件

如果用户没有指定明确的 controller/method（比如他们仅仅登录到 www.mysite.com），config/routes 文件则会设定程序执行的默认路径。其默认值可能是：

```
$route['default_controller'] = 'index';
```

如果你重新命名过系统文件夹，就必须记得你还需要在网站的根目录中修改 index.php 文件。该值默认为：

```
$system_folder = "system";
```

### 12.2 找出 PHP 4/5 和操作系统间的差异

CI 应该能够兼容 PHP 4.3 及更高版本。但这并不意味着你写的任何的 PHP 程序都能正常工作——如果你在使用 PHP 5，但正在向一个 PHP 4 服务器迁移时，就可能会遇到由于版本不同引发的问题。

（无论哪一个版本的）PHP 环境都可以不同的方式搭建。值得

一做的是在你本地和远程服务器上运行 phpinfo()，以找到它们的差异。

微软和 Linux 服务器上的大小写敏感是不同的。所以当你在一个 Windows 操作系统的 PC 上开发你的网站，然后上传到一台 Linux 服务器上，可能会收到出错信息，报告找不到一些你要装载的模型或者类库。如果你检查了并确定它们已被上传，你仍需要确定大小写是正确的。因为在 CI 的类定义和构造函数命名时，是要求以一个大写字母开头的，以一个大写字母开头的文件名名也很容易。因此在控制器内装载一个模型，应冠以一个大写字母开头的名字，如：(\$this->load->Mymodel)。而 Windows 和 Linux 则可能通过 \$this->mymodel 调用不同的视图。

作为服务器差异的一个极端例子，我有一次写一个控制器，后决定将其修改为一个模型，我把它保存在 model 目录中，却没有意识到我在开头已经写了几行代码：

```
class Myclass extends Controller {
    function Myclass()
    {
        parent::Controller();
    }
}
```

而不是把它改成：

```
class Myclass extends Model {
    function Myclass()
    {
        parent::Model();
    }
}
```

在本地的 Xampplite 上运行时并没有出错。但迁移到一个远程的 Linux 服务器上时，立刻报错。（你应该能想像调试它花费了多长时间……）

一些 PHP 函数在不同的操作系统上也有不同的表现：举例来说，include\_once() 在 windows 上是大小写敏感的，但在其它系统上不是。虽然这与 CI 本身没什么关系。

同时，你的数据库也可能不是同一个的版本——许多 ISP 还在使用 MySQL 3.23！这也许引起一些不兼容，这意味着，通过 SQL 上传一个数据库不是很容易实现的。（例如，它可能不接受数据表的提交。）

Linux 与 Windows 相比，有一个不同的文件权限管理系统。请确定你有对应的文件和目录的权限。CI 有几个目录文件的权限在它能够正常运行前必须正确设置。

#### 12.2.1 诊断工具

index.php 文件的第一行是：

```
error_reporting(E_ALL);
```

如下图所示，这会在屏幕上显示所有的 PHP 错误。

```
A PHP Error was encountered

Severity: Warning

Message: Cannot modify header information - headers already sent by (output started at E:\xampplite\htdocs\packt2\system\application\libraries\errors.php:35)

Filename: libraries/Session.php

Line Number: 282
```

很显然，这种错误报告很糟糕，可能会给黑客太多信息，所以应在产品服务器上你应该改成：

```
error_reporting(0);
```

但是，很多问题都可能只是造成一个白屏，没有可供参考的诊断信息。这时你可能必须把错误报告再打开，直到你已经使网站可以正常运行。一个折中的做法是把它设定为一个中间状态，像是：

```
error_reporting(E_ERROR);
```

这将会避免“warnings”，但仍可给你关于严重问题的信息。“warnings”通常不会中止程序的执行，但可能会引发其它你没有考虑到的问题。

CI 的 Profiler 类——详见第 8 章——也非常有用：它可以显示你正在做什么查询，以及 POST 数组的内容是什么等。

当上述工具都不起作用时，就需要使用其它的工具，下面列出了我找到的一些工具：

设定 CI 开启日志文件。（可通过修改 config 文件实现，详

见第8章)：

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

“4”显示所有的信息，包括 notices 和 warnings，有时这会显示出潜在的问题。然后查看日志文件（保存在/system/logs中，依日期排序），这将会告诉你CI的哪个部分已经被调用，因此，你能至少能看到程序中中止的地方。（把值设定回“0”会停止日志记录。请记得要这样做，调试完毕后删除日志文件：它非常占用空间。）

如果你能取得PHP的服务器和会话变量，请输出它们：

```
print_r($_SERVER);
```

以及：

```
print_r($_SESSION);
```

并使用它们检查 document\_root 和 script\_filename 的值是不是你所预期的。如果不是，你可能还要调整 config 文件中的 base\_url、server 的设置，你可以看一下是否有 [HTTP\_COOKIE] 的设置，它会显示是否你的 session 类能够工作。

用PHP的方法检查CI把什么装载进它的“超级对象”：

```
get_declared_classes();
```

以及：

```
get_class_methods();
```

CI 自己的 show\_error() 函数只是格式化生成的错误报告。因此在代码中加入下面的这一行，可以显示出程序没有运行哪一个命令分支：

```
show_error('test of error function');
```

会在屏幕上显示：

```
An Error Was Encountered
test of error function
```

我并没有发现这多有用。如果希望系统在必要的时间和位置给我一个完整的有帮助的错误报告，且当我不想要的时候就不显示它们，就得编写我自己的函数了，如下：

```
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);
    }
}
```

把它放在一个叫做 errors 的 library 文件中。我需要装载这

个 library，然后，每当我对某一段代码不是很确定时，我就会 include 这个函数：

```
$this->errors->reportme(__FILE__, __LINE__, 'if the code has reached here it is because.....');
```

我还可以设定 reportme() 函数是否屏幕上显示，或是否保存在日志文件中。

这个简单方法有几个优点：第一，我能容易地修改 reportme() 函数，让它将错误信息写到一个文件中，或什么也不做：因此我只要修改一行代码，就可以立刻使所有的错误信息从屏幕上消失，或再显示出来。

第二，比如我生成了一个包含特定值的变量。（如一个 ID，类型为整型。）然后生成一个尽可能完整且有帮助的信息。我预期会找到一个整数，也包含了这个值，而我确实找到了。该函数会用“魔术常数”PHP\_\_FILE\_\_和\_\_LINE\_\_完整地告诉我它发生的地方。

因此，如果我把程序迁移到另一个服务器后出现了一个问题，我能立刻通过这段代码找到问题所在，且文字信息可以帮助我记起它是一个什么问题。在你编程完成六个月后，你不可能马上弄明白，尤其是当深夜一位客户在电话中要求你给他做情况解释时！更有帮助的错误文本，将帮你更容易地作出反应。

第三，如果网站的完整性真的至关重要，可以设定一个函数以电子邮件形式将错误报告发送给我。在开发阶段这可能会造成邮箱爆满，但一旦网站正式运行后，当遇到问题时，有一个即时的警告邮件发给我是非常有用的。你将会在你的用户知道之前就发现问题。

## 12.3 应对 CI 新版本带来的变化

在2006年2月28日到2006年10月30日之间，CI从它的一个Beta版升级到了1.5版。这个升级过程令人印象相当深刻。

在那期间，Rick Ellis 做了一些非常激进的更新，特别地在网站的结构上。大致上，他已经注意到需要向后兼容——但并没有完全做到。如果你刚使用CI并下载了最新版本，你可以跳过这一段。但如果你也使用了较早版本的程序，或在用其他人写的CI libraries 或 plug-ins，你就需要确认一些变化。

Rick 已经努力处理了两个主要问题：

### 12.3.1 如何装载模型，以及如何调用它们

起先，没有模型，只是通过目录来管理脚本和 libraries。并没准备自动地初始化它们作为CI超级对象的一部分。结果有了一个缺少“模型”的MVC系统，这很让人迷惑。

不仅这样，还有两个 libraries 目录：

/system/application/libraries 保存你为自己编写的一些文件，而/system/libraries 则保存系统自己的操作文件。这可能会让人糊涂：这二者之间完全不同！你应该增加或改变前一目录中的文件，但你也许不用改变后者，这很容易搞错。（而且如果你这样做了，当升级CI版本时，你将会面临不兼容的危险：在下面可以看到。）

1.3版带来了一个新的“Model”类。用户手册中将模型定义为“设计用来与你的数据库中的数据合作的PHP类”。第一次使用时，CI模型自动地与数据库连接。但从1.3.3版起，你就必须要在模型或控制器中显式地连接数据库。

或者，当你从控制器调用模型时以如下格式实现：

```
$this->load->model('Mymodel', '', TRUE);
```

然后“TRUE”指定当与默认的数据库连接时才装载模型，正象你在 config 文件中所配置的那样。（第二个空白参数是模型中一个可选择的别名。）

如果你把(MVC 意义上的)“模型”功能放到一个“library”或者一个（声明：不赞成这这样做）脚本中，CI或许还可以工作。早期版本没有“model”目录：你必须用其它方式存取CI资源——详见下一节！

### 12.3.2 如何初始化你自己的类库

本来，你不能让你自己的类成为CI超级对象的一部分。这里有一个问题，因为它意味着你的 library 代码不能通过AR读写数据库，或者使用其他的CI library，这样就过于受限了。

1.2版本增加了 get\_instance() 函数，允许你读写CI的超级对象（详见第7章），你可以在“library”或者脚本中 include



它，然后使用 CI 资源。（除非你的新文件是一个函数脚本而不是一个 OO 类。当然，脚本形式可能是用来编写简单的底层函数的最佳选择。）

1.4 版引进了一个新的系统。你必须为每个“library”类创建两个文件。第一个是类本身，比如 Newclass.php，保存在 application/libraries 目录中；第二个则保存在 application/init 目录中，必须叫做 init\_newclass.php，它必须包含几行标准代码，以进行初始化，使其成为 CI 超级对象的一部分，但你仍要使用 get\_instance() 函数存取 CI 资源。

在 1.5 版中，已经不鼓励使用 init 文件夹了，初始化将自动地进行。每个“library”只需要一个文件。

旧的脚本目录也不鼓励使用了。“不鼓励使用”通常意味着相关的实现方法能够工作，但是请开发者尽量不要这么做，因为 CI 不能保证在未来的版本中还支持它。如果你还有一个 system/application/scripts 目录，无需紧张——但是请不要再用了。

如果你正在计划使用由 CI 社区编写的 libraries 或者 plug-ins，请首先检查这些资源是否是为 CI 的最新版开发的。有相当多的是专为 1.4.1 版本开发的，仍包含独立的“init”文件。更新它们不困难，但需小心行事，以确保它们正常工作。

12.4 如果有了新版 CI，我需要更新吗？

新版 CI 会不时推出，它们会带有更新指南。通常，这包括一组新的文件需要拷贝到你的 system 目录中。有时，你还需要更新 config 文件，或 index.php 文件。这不会带来巨大改变，因为目录结构已经将你的应用保存在他们自己的位置，这样在升级系统时并不会涉及到应用程序。

假如说你已基于 1.5 版完成了一个优秀作品，它被上传到产品系统中且运行得很好。此时如果 Rick 推出了 CI 1.6 版（或 2.8，或其它版本……），它有一些有趣的新功能、还有一些 Bug 修正。你是否需要对它进行升级呢？

我会说：“是的。”如果它只是一个较小的升级，比如在 1.5.2 和 1.5.3 之间，你应该升级。但如果它是一个主要的版本变化，而你现有的系统正在工作，暂缓升级是个比较明智的选择。你可能从数字部分分辨出版本升级变化大小的程度，也可以从新版本附带的“更新列表”中得出结论。从去年开始，CI 划分了三种类型来表示不同的变化：

Bug 修正：令人惊讶的少，CI 有优良的代码，大多数基础类已经被数以千计的使用者精心地测试了上百遍。

新功能：经常出现，但如果你不使用这些新功能来开发你的应用程序，它们并不会对你有什么帮助。

敏感更新：就象我说过，CI 经过了一个内部升级的过程，而且它理所应当会继续这么做。你可以通过下面的列表看出来，其中一些更新会向后兼容，否则它们可导致你将重写部分代码。

CI 版本更新过程中的一些变化：

版本	变更记录
1.2	增加了一个名为 get_instance() 的全局函数，允许你的自定义类很容易地读写 CI 的主对象。
1.3	增加了对模型的支持。
1.3	增加了传递自定义初始化参数到常规核心 library 的功能，你可以这样做：\$this->load->library()
1.3	增加了较好的类与函数的命名空间——避免冲突。所有的 CodeIgniter 类开始冠以 CI_ 前缀，所有的控制器方法均以 _ci 为前缀，从而避免了控制器命名冲突。
1.3.3	该版本模型不自动连接数据库。
1.4	增加了用你自定义的类替换核心系统类的功能。
1.4	升级了模型的装载函数，允许对同一模型多次装

	载。
1.4.1	更新了 plugins、helpers 和 language 类，允许你的应用程序目录包含你自己的 plugins、helpers 和 language 类。之前它们在安装时总是被当作全局的。而现在你的应用目录如果包含了它们中的任何一个，自定义的将会替代系统中的这些资源。
1.4.1	声明了不鼓励使用 application/script 目录。但它将会为以前的使用者继续保留，但建议你改为创建你自己的 library 或 model。在 CI 支持用户自定义 library 或者 model 之前，它本来就存在，但是它已不再是必须的了。
1.5	增加了扩展 library 和扩展核心类的功能，而且也能替换它们。
1.5	声明了不鼓励使用 init 文件夹。现在的初始化开始自动进行了。

不要嫌我啰嗦，这些都是重要的更新与改良。如果你启动了一个新项目，请使用最新的 CI 版本。但是如果你正在使用 1.3 版开发应用，你会发现 scripts 目录被声明不鼓励使用，而且模型也不会自动连接数据库。就个人而论，我曾经坚持在 CI 的 1.3 版而不去升级它，浪费了很多时间。

12.5 如何修改 CI 的基础类

一般使用者可能无需改变 CI 基础类。它具备相当好的框架，能做出许多东西来，使用框架不是会让事情更容易吗？当然，如果你一定要做的话……

CI 是开源产品，下载后你就能看到所有的代码。这包括使 CI 工作的基本 library（保存在 system/libraries 中）与你自己的 libraries（保存在 system/application/libraries 中），所以你可以改变 CI，使其以你喜欢的方式工作。

修改系统 libraries 文件存在两个问题，就是：谁也不能保证你的新代码与 CI 的其他部分或更新的版本兼容。这可能会导致不易跟踪的、敏感的或奇怪的错误。

如果你稍后升级你的 CI 版本，系统目录也可能会随之改变。你修改过的 library 将被新的文件覆盖或更新，因此你需要自行修改并将其更新到升级的版本中。

不过，对可能有人会胡乱修改 CI 类库的情况，从 1.5 版起增加了两个明确的“工作区”。（不包括基本的“数据库”和“控制器”类，这两个都很危险，请不要自行修改。）

第一，当你在/system/application 目录中创建一个与系统基础类同名的文件，系统会优先使用这一个；如果该文件不存在或不可用时，则使用/system 目录中的那个。该操作有明确的命名限制——详见用户手册。它还需要你复制所有在现有的类中存在的功能，让它们和你改过的那些类一起工作。

第二，更方便地，你可以从现有系统类中派生出一个新类。（派生一个子类可能是更好的做法。）这当然也有命名限制，详见用户手册。继承一个系统类意味着你的新子类潜在地继承了 CI 类中的所有资源，并增加了几个你的额外方法。这也许意味着，如果你升级了 CI 版本，祖先类将会被替换，但是你的新子类（应该将其放在 system/application 目录中）将安然无恙。

然而，上述两种方式都不能保证你的代码与 CI 的其它部分完全兼容。

去看看 CI 社区，上面有对各种扩充验证类、单元测试和 Session 类的不同建议。以单元测试为例，只有两个函数和比较有限的字符数做比较。如果你希望有一个函数，当测试结果返回时，以比较醒目的红色标识错误信息？

或者你希望扩充一些其他的测试函数，比较简单的做法是通过一个子类将其加入。扩展到单元测试上，即当你每次调用单元测试时，将其写入控制器中。

如果你想这样做，可以这样开始新子类：

```
class MY_Unit_test extends CI_Unit-test {
    function My_Unit_test()
```

```
{  
    parent::CI_Unit_test();  
}  
function newfunction()  
{  
    //new code here!  
}  
}
```

这里需要注意三件事：

单元测试类的名称是 CI\_Unit\_test，即使类代码的文件名为 system/libraries.unit\_test。

如果你需要在你的子类中使用一个构造函数，请先确定在里面调用父类的构造函数。

该新子类名称应该为 MY\_前缀并保存为 application/libraries/MY\_unit\_test.php。（不像系统中的主要类，是以 CI\_为前缀，而不是文件名，在这里 MY\_前缀其实是二者的一部分。）

如果你已经创建了你的子类，你可以这样装载它：

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

换句话说，新子类与你以前编写的子类完全相同，而且会以同样的方式调用函数，不过此时你不但能调用原来的单元测试函数，还能调用你自己编写的新函数：

```
$this->unit_test->newfunction();
```

在你以后升级 CI 时，在系统文件夹中的单元测试类库将会覆盖，但在 application 目录中的那个不会被覆盖，所以你的代码还会在那里。当然，你还需要检查更新的系统类是否依然与你自己的代码兼容。

## 12.6 总结

在这一章里，当你尝试迁移一个本地应用到远程服务器时，可能会出现一些问题。这包括：

PHP 或 MySQL 的版本差异

操作系统的差异

特别地，我们分析了大小写敏感问题、PHP 差异与 MySQL 问题。我们还讨论了几个诊断工具。

然后我们分析了 CI 的升级。对这些重要的进步，我的建议是，如果你在现在的 CI 版本上工作得很好，当 CI 有新的版本推出时，请仔细评估是否需要升级以及如何升级。

最后，我们分析了修改 CI 基础类的正反面因素。其实大多数的使用者无需这样做。但如果你确信的确需要这样做，我强烈建议：实现它的最好方式是从一个现存的 library 类中派生一个子类。

## 第十三章 快捷的 CRUD 及其配合使用

编写任何一个动态网页时，最基本的，也是最令人厌烦的部分是 CRUD。你有一个或多个数据表时，你需要去对它们每一个实体进行建立、读取、更新与删除。后来，你将对数据处理会变得灵活，但在此之前也有一些界面友好的方式来处理它，并保留下来，但对你的网站是不能用的。

不过，这牵涉到写各种各样的 CRUD 函数，虽然在理念上相当容易，但这些都是相当复杂且费时的的工作。所以对于我们的网站，我写了一个通用的 CRUD 模型，使用 CI 的类及辅助函数，使之更加容易。在这一章中，你会看到这个模型如何工作，如何将它集成到我们的应用程序中来。

CRUD 模型不只是 CRUD。它验证用户输入的数据，并用多种方式检查它。如：当你在特定行做“delete”操作时，或者通过返回进入到表单并且在你的浏览器中重新加载它，你不小心重复“create”操作时。

最后，CRUD 模型包含了它自己的自检框架，因此你能够把它作为你的架构或适应你的代码，进行开发测试。

CRUD 除此之外，还能做更全面，更优秀的代码（参见第 15 章）。然而，我们所要做的是对我们以前所学的课程进行一个好的总结与提高。

此章对模型所要展示的代码如下：

设计原理。

一个标准的控制器与模型配合使用。

数据库表必须是有条理的。

模型本身：保存数据库信息的数组、分离函数。

自检函数。

### 13.1 CRUD 模型：设计原理

在这个 CRUD 模型背后的想法是它能被任何表的任何控制器所调用。对于数据表的数据，在一个数组中，如何从过去显示的数据去更新它，所有都是标准：控制器恰好识别它自己（并且在表中起作用）并且需要对每条记录给一个 ID 号。因此我们不得不写一些简单的控制器，连接到数据库去取数据，并展现它到表单中去。

记得我们不能直接使用一个模型，因此每次我们不得不需要通过一个控制器不使用它。你可以放一些代码到控制器中，但你不得不复制它到其它新的控制器中去。有一种方法，只通过在模型中的一套 CRUD 代码，我们就能去更新，去维护了。代价是你必须去保持控制器与模型之间来去通畅，如何使代码更简洁，更多的困难在后面。

为了简单，我们使用在代码中没有定义的两个扩展函数：

failure()，当报告有错误时，我们仍然想继续执行。

被调用的模型显示一建立菜单与设置基准 URL 等。因而 CRUD 函数建立了一大堆数据，放在 \$data 变量中，可以简单的调用：

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

我想让 CRUD 模型能够自检，因此它包含了一个自检组件。如果我们愿意的话，可以在设计处理期间允许我们去调用它。（编写测试套件会让你发现很多错误，起初不经意的地方会出现错误让你很是意外，但是现在发现错误，总比以后你的客户访问网站发现错误要好。）

请以一个折衷的方法记住每一个模型。你要求它的越多，它要求你的也越多。例如，除非你的数据库表以某种特殊方式下，这个模型它不工作。它以完全的高级方式显示在表格中，但它不能灵活处理。它没有使用 Javascript 来更好的适应用户的习惯。它不能用它自己的规则来处理异常。在其它方面，如果你只想实现一系列标准事件（它是很通用的），它是很简单达到的。

### 13.2 标准的控制器格式

首先，对每一个数据表，你需要有一个标准的控制器。它是怎样让用户与你的数据表接口，如：新增记录，更新记录等。对于新增一个新的人员，用户将与人员数据表对接，因此需要一个不同的控制器：但它与其它的控制器的控制器大多是相同的。

下面是我们一个 Sites 数据表的控制器：

```
<?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 the
table/controller (First letter in upper case for the Class
name and
constructor function, lower case for the file and
variable. lower
case!)*//
    var $controller = 'sites';
/*constructor function*/
    function Sites()
    {
        parent::Controller();
        $this->load->model('crud');
    }
/*function to update an entry (if an ID is sent) or to insert
a new
one. Also includes validation, courtesy of CI */
    function insert($id)
    {
        $this->crud->insert($this->controller, $id);
    }
/*interim function to pass post data from an update or insert
through
to Crud model, which can't receive it directly*/
    function interim()
    {
        $this->crud->insert2($this->controller,
$_POST);
    }
/*function to delete an entry, needs table name and id. If
called
directly, needs parameters passed to function; if not, from
Post
array*/
    function delete($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
any alterations*/
    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 made
on a temporary test table so that your real data is not
affected*/
    function test()
    {
```



```

$this->crud->test();
}
}
?>

```

你领会之后，你会发现它是优美苗条并且十分通用的。如果你想让 people 控制器去代替 Sites 控制器——换句话说，允许你在 people 表中去建立，读取，更新或删除记录，等等，你需要做如下事情：

更改 Sites 为 People (首字母大写!)

更改 \$controller 变量 sites 为 people (小写)。

更改构造函数名 Sites 为 People (首字母大写)。

保存新的控制器为：

system/application/controllers/people.php.

控制器名必须严格的与数据表名一样，如对于 people 表，它必须是 people。在类定义行与构造函数中，名字的首字母必须是大写，但是其他地方不一定要这样。

### 13.3 数据库表

对于你的数据表有三个简单的规则：

最主要是第个表有 ID 字段为主键并且为自增字段（这是一个标准的 MySQL 字段类型。新增记录时，自动建立一个新的不重复的数字）。

如果你想使用它在一个动态的下拉列表，你有一个 NAME 字段。

你也必须有一个 SUBMIT 字段来存储状态，和类似的东西等等。

除此之外，你可拥有任何你想要的字段，并随意对其命名，其他所有都由 CRUD 模块处理，适用于设计针对与这些行一起配合成对的任何控制器/数据表。

### 13.4 模型的心脏：数组

准备工作完成后。让我们开始创建 CRUD 模型吧。

首先你需要定义 CRUD 模型与一个构造函数。标准写法如下：

```

<?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');
    }
}

```

保存它到 system/application/models/crud.php。

继续钻研，但你仅需要做一次下面的事。你要写一个多维数组(我刚开始的时候从一本书上学的 PHP——它是非常好的——上面说“多维数组不会经常遇到，因此我们以后都没有必要深入的学习它”。看来现在开始就要用它了)。

我们数组的一维是数据表列表(sites, people, 等)。

二维是每个表的字段列表。对于 sites 表，就是 id, name, url 等。

三维是描述每个字段与提供将在插入/更新表单中处理控制的一组变量。如下：

在输入框显示你想希望被用户看的文本：这个字段是如何向人们描述的。(所以它的第一个字段是网站 ID 而不仅仅是 ID)。这个会让你的表单更加用户友好。

你可以在你的插入/更新表单上显示表单类型的字段：这可以是一个输入框、一个文本域或者一个下拉框。(CRUD 模型定制了一些设置但不是全部。)

当用户填写表单的时候，你可以加入 CI 的验证规则。这个可以留空。

如果你希望动态下拉框显示这个字段，它的表名将会显示出来。看下面的这个解释，它也同样可以留空。

我们已经声明了类里面的一个变量 \$form 为数组，所以以后任何时候，我们必须这样用它 \$this->form。它被定义在构造函数数组里，它直接跟随前面的代码。

```

$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',
            'numeric'),
        'domains' => array
        (
            'id' => array('ID number of this domain',
                'hidden', 'numeric'),
            //etc etc etc!!
        )
    )
)

```

你会发现 \$form 数组里每一个表(这里指站点和域名，虽然由于空间的原因我只开通了后者)都有一个二级数组，每个二级数组都包含了他们自有的三级数组，每一个字段(“id”、“name”等等)对应一个。每个三级数组都是依次排列在数组中，包含了三个或者四个在前面描述的值。

你并不能很容易的理解这种数组，但是从概念上看很简单。

为完成我们应用程序中的表的设置，创建该数组约需 120 行。但是，你只需要去做一次！这是你的模型的心脏。用括号‘}’关闭这个构造函数，并继续创建 CRUD 模型的其他函数。

如果你需要改变你的数据库表(例如添加一个新的字段)，或者你要改变你的验证规则，那么你只需要改变数组里的数值。他们将会自动更改：举例来说，当您下次尝试增加新的条目时，你应该看到在表上反映的变化。

### 13.5 CRUD 模型

以下各个函数组成了 CRUD 模型：

#### 13.5.1 Showall 函数

这是一个用户最常用的函数。它可以作为一个切入点，对输入所有其他的操作，添加，更新，或删除。它显示你表里已有内容。网站表上的一些测试数据，它看起来是这样的：

sites		
<a href="#">New entry</a>		
<a href="#">Show all entries in the table</a>		
1 upton.cc	<a href="#">Update this entry</a>	<a href="#">Delete</a>
2 simulation.cc	<a href="#">Update this entry</a>	<a href="#">Delete</a>
5 Testsite1	<a href="#">Update this entry</a>	<a href="#">Delete</a>
9 Testsite2	<a href="#">Update this entry</a>	<a href="#">Delete</a>
10 777aaa	<a href="#">Update this entry</a>	<a href="#">Delete</a>
11 test sitenumber 11	<a href="#">Update this entry</a>	<a href="#">Delete</a>
12 new insert	<a href="#">Update this entry</a>	<a href="#">Delete</a>
14 Testsite3	<a href="#">Update this entry</a>	<a href="#">Delete</a>
17 test	<a href="#">Update this entry</a>	<a href="#">Delete</a>
18 mysite	<a href="#">Update this entry</a>	<a href="#">Delete</a>
26 TestSite3	<a href="#">Update this entry</a>	<a href="#">Delete</a>
24 TestSite1	<a href="#">Update this entry</a>	<a href="#">Delete</a>
25 Testsite2	<a href="#">Update this entry</a>	<a href="#">Delete</a>
27 TestSite4	<a href="#">Update this entry</a>	<a href="#">Delete</a>
28 TestSite5	<a href="#">Update this entry</a>	<a href="#">Delete</a>

如同你看到的，在这个页面中你可以更新或删除整个网站。你可以添加新的内容，或者从表中读取所有数据。

顺便说一句，请大家不要忘记，该模型并不包括任何安全规定。在一个真实的网站中，你也许要微调用户的选项-例如，允许更新而不能删除。你要确保黑客无法通过输入 URL (例如: `www.example.com/index.php/sites/delete/18`) 访问 CRUD 模型的功能。CI 的基于 URL 的结构使得它比较容易推断出系统如何访问这些命令，所以你可能希望在 CRUD 模型激活以前，确保用户已登录到网站中。

回到 CRUD 机制。记住，用户不能直接调用模型。每一个操作（删除，更新等）都是通过控制器来调用的。控制器用下面这行代码调用 `showall` 函数：

```
$this->crud->showall($this->controller);
```

换句话说，就是用 `showall` 函数取代网站的 `$this->controller`，并传递一个参数到 CRUD 函数中，就是要告诉它，它在取代哪个控制器的功能。

我们现在来看看 `showall` 函数。我们已经把第一个参数传递给它了。我们把 `$message` 留到后面。集中看标为高亮的行。

```
/*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 to
create, 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 success
or 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,
$this->status);
}
else
{
    {$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');
}
}
```

```
}
```

它列出了一张表，展示了关于每个条目的一些数据（ID 和名称）。你也可以选择更新或删除功能，以更新或删除该条目：这是利用 CI 的 `anchor` 函数创建超链接，并链接到适当控制器中的适当函数。

这也有一行代码，为你提供了创建一个新站点的机会，再通过超链接连接到控制器的 `insert` 函数。（注：我将添加新的条目和更新旧的条目都称为 `insert` 函数，这是因为，模型假设如果插入的是已存在的 ID 号码，它将更新相应的条目，如果没有对应的 ID，它会创建一个新的条目）。

很多代码都采用了异常处理：例如该表不存在，例如查询没有返回信息。异常会传递到 `failure` 函数中。此外，还有两个测试模块，允许我运行自我测试。

此外，还有一行代码，允许你读取（但不修改）表中的所有数据。让我们先看看最简单的 `read` 函数。

### 13.5.1.1 读取数据

我用 CI 的 HTML 表格类（见第 10 章）和 `ActiveRecord` 类（见第 4 章）简单演示了这个功能片段。我想用数据库中的所有数据把页面格式化成 HTML 表格。它不允许有任何的更改：这就是用于“读取”的页面。

首先，控制器中必须有一个函数来调用模型，并告诉该模型要显示哪个控制器/表。`read()` 是标准控制器中的函数。

在 CRUD 模型中调用下列函数：

```
/*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);
}
```

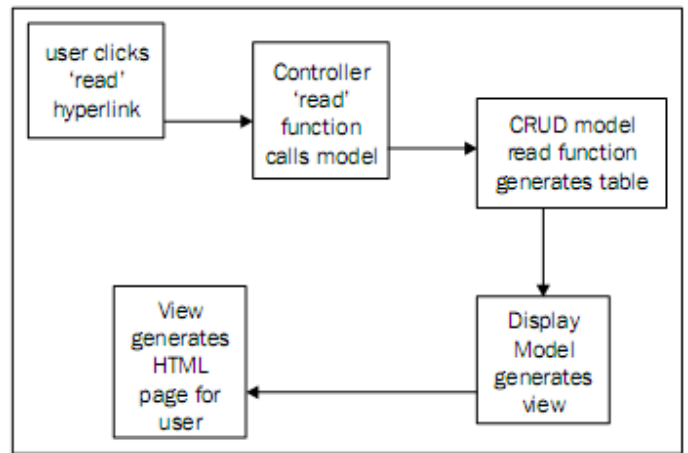
高亮显示的两行代码处理所有的数据库查询工作，并格式化结果。

我用 `display` 类中的 `mainpage` 函数来格式化该页：这里的 `read` 函数组织数据并把它变成数组的一部分。

其结果是由 `test` 文件的所有数据组成的页面：

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	alloysius	1	1	1	1	1	1	

让我们再次提醒自己控制行为是如何在控制器、CRUD 模型和程序其他部分之间传递的。



### 13.5.1.2 Delete 和 Trydelete 函数

删除是不可逆的操作！基于这个原因，我们的 `delete` 函数将确认两件事：

在“submit”字段中的状态变量已被设置为“yes”：如果不是，传递请求到 `trydelete` 函数中。即询问用户，她或他是否真的想删除。如果她或他确认，`trydelete` 函数设置状态变量为“yes”，并发送请求到 `delete` 函数，然后删除将被执行。

执行删除查询以前，它会检查 ID 号是否已设置（否则可能删除所有内容）。然后，它使用 CI 的 `ActiveRecord` 执行删除，并确定此行已从数据库表中删除。如果此行已删除，则返回到 `showall` 函数中。你会发现它传回两个参数—控制器名称和一个报告删除已成功执行的消息。（这是 `showall` 的第二个参数。如果设置它，则会在表格的顶部显示一个红框，以便让用户知道怎么回事。）

首先，这是 `delete` 函数。你会发现这个代码也因为“test block”代码行而变得复杂了很多。先忽略这些：只看高亮的代码。

```
/*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 $outcome;
        }
        else
        {
            /*end test block*/
            {$this->trydelete($controller, $idno,
'no')};
        }
    }
    else{
        /*'yes' flag is set, so now make sure there is an id number*/
        if(isset($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 $outcome;}
        else
/*end test block: if I failed, report me*/
        {$this->failure($outcome);}
    }
}

```

我答应过要解释一下 showall 函数的 \$message 参数。你看它在这：如果这个功能执行成功，它将用一个合适的消息调用 showall，并返回该页：

```
$this->showall($controller, "Entry no. $idno deleted.");
```

重要的不仅仅是操作的完成，还要让用户知道操作已完成。

现在，回到防止意外删除的话题。如果没有用 state=yes 参数调用 delete 函数，它将重新分配请求到 trydelete 函数中——“第二次机会”。实际上，只有 trydelete 函数会设置这个参数为 yes，所以，表单删除会一直提示你是否确认删除操作。

让我们看看 trydelete 函数。它创建一个简单的表单，看起来像这样：

Are you sure you want to delete this entry?

[No, don't delete](#)

单击“yes”重新调用 delete 函数。（注意：表单不能直接返回到 crud/delete，因为表单不能指向一个模型。它已经指向控制器中的 sites/delete 函数，该函数只是简单的将所有参数直接传递给模型中的 crud/delete 函数。）

这个微妙的改变是，如果用户确认删除，trydelete 表单将添加（作为一个隐藏域）submit=yes 参数，该参数将保存在 post 数组中，然后，返回到控制器的 delete 函数。控制器的 delete 函数从 post 数组中取出 submit=yes 参数，并把 state=yes 作为参数来调用 crud/delete 函数，然后 delete 函数转至下一个步骤。

如果用户不想执行删除操作，用户单击 CI anchor 函数创建的超链接，然后被传回 showall 函数——用户最有可能的点击来源。

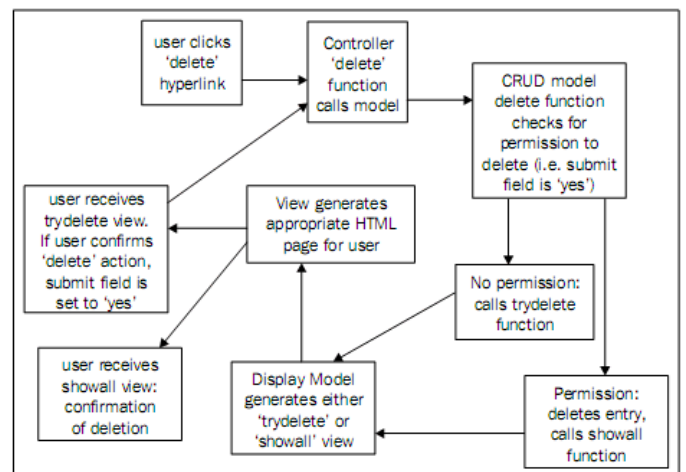
这里是完成这个功能的代码：

```

/*TRYDELETE FUNCTION: 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 to delete 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);
    }
}

```

只是为了清楚起见，这里有一张如何在删除操作中传递控制行为的图表。



你可以看到，这比我们先前的例子复杂的多。模型会处理所有的工作，但用户只能调用控制器，所以如果你需要后退并向用户重新显示问题，你需要再次调用控制器。

不过，一旦你整理出来，它运作良好，而且还具有高度的逻辑性。CI 强加这个框架给你，但长远来说，这就是一个优势。你的代码是一致的，模块化的。注意模型和视图每次被调用时的相同处：他们向用户展示什么依赖于调用他们的 CRUD 模型里的函数。

### 13.5.2 Insert 函数

这是最复杂的函数，因为它生成一个让用户填写的表单。（和

人相关的接口总是最复杂的东西……)

与其写两个独立的函数，一个用来插入、一个更新，而且需要建立两次表单，我写一个函数来完成这两个工作。如果你提供了一个合法的 ID 号，就更新相应的记录；如果未提供，则插入一条新记录。

简便起见，我没有加入我们在 delete 函数中的测试模块。

以下就是我们使用本章开头定义的那个数组的地方。这个函数生成了一个表格，通过使用 CI 的表单辅助函数，基于数组中特定的表单元素（下拉，文本域等等）。函数的核心是一个状态切换来实现如下工作。

这段代码使用 CI 的验证类来帮助我们检查输入的数据：记住我们在初始数组中设置了验证规则。

```
/*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).*/
                    $dropdown = array();
                    if(isset($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)
                            {
                                $dropdown[$row->id]
= $row->name;
                            }
                        }
                    }
            }
        }
    }

```



```

    }
    $myform .= "<tr><td
valign='top'>$value[0]</td><td>";
    $myform .= form_dropdown($key,
$dropdown, $currentvalue[$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>$currentvalue[$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);
}

```

有很多东西需要在这里解释。所有的表单域类型都是标准的，除非是只读——就是说，一个只允许看而不允许改变的隐藏表单域。这并不安全，当然：一个聪明的用户可以轻易的获取这个值。它只是设计用来区分不同的用户。

你将注意到表单指向一个函数叫做 interim，所有的控制器都需要调用它。重申一下，这是因为你无法直接通过 URL 来调用一个模型 (model)。因此，如果 URL 由“site”控制器来处理，则表单的值传给“site/interim”，数据由用户输入或者已经存在，通过 \$\_POST 数组被传递到哪儿。如果你在此调用本章开始部分，那个函数就会调用 crud 函数 insert2，通过 \$\_POST 数组以变量形式传递。

### 13.5.3 insert2 函数

Insert2 接收 \$\_POST 数组作为参数并检测其中是否含有“id”域。如果有，就更新该纪录；否则就插入一条新纪录。

为了使用 CI 的验证类，需要 \$\_POST 数组，我们的函数将接收 \$\_POST 为参数。

```

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__);
            }
        }
        else{
            $this->db->insert($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__);
            }
        }
    }
}

```

这就是全部。百余行的代码，让在任何表格上实现 CRUD 。

### 13.5.4 测试套件

还记得 delete 函数中的“测试模块”吗？它们的目的在于简单的测试函数是“实际”运行还是只是一个测试，然后确定返回一个容易测试的数值。

这是因为，在 CRUD 模型的最后，我们还有一个“自测试（self-test）”套件。它在任一控制器中（不管是哪一个）都会被调用并通过使用一个虚拟的表格实现全面地 CRUD 测试。

在 CRUD 类中首先有一个控制函数“test”，仅是用来访问其他函数。

```

/*now a suite of self-test functions.*/
/*first function just calls all the others and supplies any
formatting
you want. Also it builds/ destroys temporary data table
before/ after
tests 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);
}

```

这里集中了所有你想要的测试，并运行这些函数。

然而，不用运行所有这些函数，我们只要展示一个：一个叫做 testdelete() 的函数。

首先，虽然，我们需要两个函数：一个用来建立，一个用来销毁我们的虚拟表格，“fred”。第一个函数销毁所有存在的“fred”表格，建立另外一个，将一系列测试数据存入其中：

```

/*this function builds a new temporary table. 'fred', in your
database
so 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,
'blogs')");
}

```

如果需要，你可以设置得更复杂一些——比如，通常更多表单域，或者跟多行数据。

第二部分是销毁表格我们才能再次运行。那需要没有任何数据在里面，在我们删除测试之后，除非失败了或者写入了其他测试，让我们确认一下。

```

/*this function destroys the temporary table, to avoid any
confusion later on*/
function testdestroy()
{
    $this->db->query("DROP TABLE IF EXISTS fred");
}

```

现在我们开始测试 delete 函数：

```

function testdelete()
{
    $result = '<p>Deletion test</p>';
}

```

第一个测试中我们要确认 delete 函数拦截了所有 \$state 变量不为 yes 的删除动作，并将其发送给 trydelete 函数来询问“确定？”

记住我们希望程序正确地处理所有可能的情况并测试能返回“OK”，如果不是可能性本身的“right”与“wrong”。因此，“state”变量是“haggis（这里只是一个 right 和 wrong 不同的字符串，不用具体考究 haggis 的实际意义）”，这明显是“wrong”，在程序多次对测试说“not yes”之后，测试需要说“OK”。我们只是希望一小段高亮的错误提示：如果测试成功，我们不需要了解细节。

首先我们建立一个数组，它所有的舰只都是测试中可能用到的表达式，他们对应的值就是我们需要的结果：

```

$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);
}

```

假设我们的代码运行顺利，它将会返回：

Test Name	exception at E:\xampp\lite\htdocs\pack12\system\application\models\crud.php657: sent state value no to trydelete function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampp\lite\htdocs\pack12\system\application\models\crud.php
Line Number	876

Test Name	exception at E:\xampp\lite\htdocs\pack12\system\application\models\crud.php657: sent state value 1 to trydelete function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampp\lite\htdocs\pack12\system\application\models\crud.php
Line Number	876

Test Name	exception at E:\xampp\lite\htdocs\pack12\system\application\models\crud.php657: sent state value yyyes to trydelete function
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampp\lite\htdocs\pack12\system\application\models\crud.php
Line Number	876

我们下面将要进行的测试是看如何，在正确的条件下，delete 函数对一系列的 ID 值做出反应——包括非整数值、负数值等等。注意测试的颗粒性，比如大于 0 的整数 9999 就是是一个非法 ID，在我们只有一条记录的情况下不会导致任何删除的操作。你需要明确你在测试的阶段。

```

/*given $state set to 'yes', test another array of values
for the id number. 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;
}

```

一切顺利的话，它将返回如下：

Test Name	OK at E:\xampp\lite\htdocs\pack12\system\application\models\crud.php675: doing delete on id of 1
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	E:\xampp\lite\htdocs\pack12\system\application\models\crud.php
Line Number	876

Test Name	exception at E:\xampp\lite\htdocs\pack12\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:\xampp\lite\htdocs\pack12\system\application\models\crud.php
Line Number	901

Test Name	exception at E:\xampp\lite\htdocs\pack12\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:\xampp\lite\htdocs\pack12\system\application\models\crud.php
Line Number	901

你可以增加很多其他的你想要的测试。

测试有助于开发过程。如果你在测试数组中放入不同值得时候，你必须考虑你的代码真的能够得体的处理他们。

以后它将有助于你，在你改变代码突然获得错误提示时；一旦代码放入产业环境，经常性的运行测试将让你宽心。

## 13.6 总结

本章很长，但集中描绘了很多内容。我们看到了：

如何概括 CRUD 操作使你能够通过两个类完成它们：一个是控制器，还有一个是 CRUD 模型。表单在各个表格中重复，但后者总是一样。

我们加入各种检查和安全保护，比如测试，我们可以确认完成 CRUD 操做。

使用 CI，可以让我们将所有功能用几百行（相对的）简单的代码实现，我们可以多次用于几乎所有我们所架设的网站，为我们提供了一个遵守简单命名和布局的规则。对我而言，框架就是要这样。



## 第十四章 审视 CI

这本书以一些特定例子开始，这些例子演示了 CodeIgniter 如何能节约你的时间，让你在用 PHP 开发网站时更有效率。在开发这个能够定期测试其它网站的网站应用中，CI 成为该网站的一些基础部分，我们也花了一些时间来演示 CI 所能做的那些事情。我希望这些例子已经显示了 CI 如何在宏观上使编码更容易。

在这一章中，我想要稍微往回走一点，来看一下使用 CI 框架带来的全面影响。它使写一个完整的应用变得容易吗？它能产生专业的结果吗？

当你写一本像这样的书时，把它分成几部分，并且每次把重心集中在一个新的战术上是很重要的。这意味着把各部分的东西组合在一起有时候是很困难的。我希望在上一章的“实现 CRUD”的代码提供了把 AR 和单元测试以及表格等不同部分组合起来的一些方法。在这一章里，我想要演示如何把所有的部分组合在一起最终交付一个完成的项目。换句话说，我们的测试网站能够正常工作吗？

由上而下地查看一些来自网站的样本代码应该有助于我们拟订一个平衡表：

CI 提供帮助的地方

组织网站

使代码变成更简单

为你完成一些工作（数据校验等）

CI 不能提供帮助的地方。

### 14.1 一些代码：“do\_test”模型

我们来仔细查看我们网站上一个模型中的一部分，这个模型与‘tests’表和‘events’表一起工作。它的目的是控制我们网站上的中间函数。例如在远程网站上做的测试。这个模型：

连接到 sites 表列出的网站数据库表并进行测试

更新另外一张表，也就是 events 表（也就是，每次一个测试被实施）

提供用户接口，让他或者她选择一个测试样式并且以各种不同的格式得到数据

因为它是一个模型，它需要被一个控制器调用，并且返回结果给视图。如果视图包含超链接，它们依次调用一个控制器函数，扮演另一个模型函数的前端的角色。

这是 do\_test 模型的主要功能之一：从一个数据库查询直接生成一张数据表的信息。这样做将列出可用的网站和选择其中一张表进行测试。如果你还没有从其中选择一个，函数生成一张网站列表让你从中选择一个。代码如下：

```
/*this function prepares a report on existing tests and
allows you to choose 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
work on</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 generate
hyperlinks*/
$text .=
anchor("tests/report/$row->id","Select");
$text .= "</td></tr>";
}
}
}
```

```
}
}
$response['mytext'] = $text;
$response['message'] = $message;
$this->display->mainpage($response);
}
```

作为结果，当它通过一个控制器被调用的时候，它会生成一个页面：



它列出了一个供测试的网站清单，并提供一个与每个网站对应的超链接。因此你能选择那个网站。我喜欢使用 HTML “表格”类自动地完成它，但是如果我通过这个超链接来选择需要测试的网站，我找不到一个容易的方法来包含结果行的超链接。但是：

CI 的 AR 使数据库查询非常容易编写

CI 的 anchor 函数（URL 助手的一部分）使超链接编写起来比较容易

并且，理所当然地，当我们把我们的网站迁移到生产服务器时，我们知道这两个 CI 函数会自动更新数据库连接信息和 URL 的设置。

CI 也帮助我们组织代码。当模型从菜单中被调用时，\$siteid 变量将作为 URL 的一部分。

注意真实的网页是如何通过 display 模型的 mainpage 函数创建的。模型 do\_test 要做的一切就是从数据库获取我们需要的特定的信息，并把它送走。display 模型把它用 CSS 文件进行格式化，把它放到一个网页上作为一个菜单项。

当然，因为表格式放在 do\_test 模型中，这样做并不是很合适。那里有太多的尖括号了，理想地，他们应该在一个视图文件中。因此，这需要我们为这些数据专门设计一个‘视图’文件。这样做似乎比较简单。至少数据没有和 CSS 文件放在一起。

现在我们移至这函数的下半部份。如果你已经选择一个网站，视图现在让我们看到针对那个特别网站所做的测试结果。

代码看起来像这样—是下面这个条件语句的 else 部分：

```
if (!$siteid > 0)
{
/*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>";
$report .= "<tr class='header'><td width
='25%'>Test</td><td
width='15%'>Type</td><td width='10%'>Frequency</td><td
width
='40%'>Result</td></tr>";
foreach ($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);
$query = $this->db->get('frequencies');
if ($query->num_rows() > 0)
{
    $frow = $query->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 all outstanding tests now');
$report .= "</table>";
}
else
$report = "no tests for this site yet.";
$report .= "<table class='table'><tr
class='header'><td>Other options</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", 'Update
remote file list for this site');
$report .= "</td></tr><table>";
$response['mytext'] = $report;
$this->display->mainpage($response);
}
}

```

这个代码的运行结果是：

Run tests   Sites   Domains   People   Hosts   Tests   Events   Frequencies   XMLRPC   Logout			
Test report on ATestSite1			
Test	Type	Frequency	Result
ping test	ping	every 15 minutes	Overdue: <a href="#">do it now</a>
run all outstanding tests now			
Other options			
<a href="#">Get a written report for last week</a>			
<a href="#">Get a written report for last month</a>			
<a href="#">Update remote file list for this site</a>			

你能看到它已经查询了数据库以找出针对我们选择的网站要做哪些测试。只有一个测试，一个简单的 'ping'，这个测试被建议需要每 15 分钟做一次。

因为在最近的 15 分钟内还没有这样做，因此，它已经显示为过期，而且一个超链接让我们可以做这一个测试。如果我点击超链接，我调用了 tests 控制器的 runtest 函数，提供它我想要的测试的 ID 值。

结果是：

<a href="#">Run tests</a>   <a href="#">Sites</a>   <a href="#">Domains</a>   <a href="#">People</a>   <a href="#">Hosts</a>   <a href="#">Tests</a>   <a href="#">Events</a>   <a href="#">Frequencies</a>   <a href="#">XMLRPC</a>   <a href="#">Logout</a>			
Test took 0.1942		Result was OK	
Statistics for this test			
Data for ping test			
Time	Problem?	Time taken	Result, if not successful
18 Apr 2007 15:52	n	0.1942	

测试完成后，系统会告诉我什么时候进行的测试，有没有问题，和花了多长时间。

如果我现在返回到 report 函数，你可以重新调用针对我选择的这个网站的上一周，上一月的测试报告。我只是把这个网站作为一个例子，因此，它没有真正的测试历史：但是如果我执行了连续的一系列的测试，报告中将会增这些测试的结果。

Run tests | Sites | Domains | People | Hosts | Tests | Events | Frequencies | XMLRPC | Logout |

Site name: ATestSite1

- Location: <http://www.example1.com>.
- Client is 1
- Report at 18:4:2007 16:46

Tests on this site are:

Name	Last done	Last status
ping test	18:4:2007 16:36	OK
check front page content	18:4:2007 16:45	OK

Test history:

Time	Done	Time taken	Status
18:4:2007 16:45	check front page content	0.1625	OK
18:4:2007 16:36	ping test	0.1592	OK
18:4:2007 16:36	ping test	0.1578	OK
18:4:2007 16:35	ping test	0.167	OK
18:4:2007 16:35	ping test	0.1775	OK

正如你看到的一样，我们正在建立一项报告，它能在不同的时间以不同的测试方式得到某位客户的网站正在运行着并且能作出回应的结果。测试的任一次都不会比另外一次来得更有趣，但是可以帮助我们得到一个比较长的时间段的图表。

实际上进行测试的函数被构造成一个 switch 语句。它有两个参数：

测试的 ID 值，它给我们提供了我们需要运行的网站的基本数据：URL，域名/地址，测试完成后需要的文字信息

用户类型。（如果用户真正的人，程序会以对人更友好的格式返回更多的信息-换句话说，如果你想要在屏幕上显示结果，调用时把用户参数设置为人。如果你想要系统处理结果，将这个参数设定为其它。）

在这个代码片段中，我们已经定义了测试的一些类型。两个例子是：

“ping”测试，只是简单地调用 URL。如果他们得到一个结果，他们或是用一个预设的格式来分析它（在数据库中被称为 '正则表达式'）或是在没有设置正则表达式时作为一个一般性的 HTML 术语。

“ete”测试，用一些我们开发的代码来对一个被保护的网页进行一个 '全面' 的测试，登录后查找一个预期的片段。这段代码不在这一本书中进行解释，因为它属于 CI 的函数。

每个测试返回一个 \$result 变量和一个 \$timetaken 变量，这些作为一个记录被保存在数据库的 'events' 表中，连同一些来自数据库的其它的信息。下面列出代码，这些代码很大程度上依靠 CI 的 AR 模型读写数据库，并且使用 benchmark 类统计每次测试需要的时间。

```

/*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 = $this->httpost($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;}
    }
}

```

我们如何实际上打印出测试报告是一个 CI 能提供较大帮助的有趣例子。你可以有多种的选择。你能用硬编码的方式打印出你的报告，在你的代码中使用 HTML 就象我们早些时候所做的，象这样：

```

/*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>";
    }

```

另一方面，你可以使用 HTML 的表格辅助函数来使生活变得更容易：

```

/*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*/
$tpl = array ('table_open' => '<table border="1",
class="table">');
$this->table->set_template($tpl);
if ($query->num_rows() > 0)
{
    $this->table->set_heading('Time of test',
'Name', 'Time taken', 'Result');
    $report .= $this->table->generate($query3);
}
$this->table->clear();

```

明显地，更短而且更简单。结果和使用 HTML 的格式完全一样。不过，让我们来看这些代码带来的两个问题。

当你看得更仔细一些的时候，有一个与方便俱来的妥协。前一段代码，比较长的版本，格式化日期是可能的：

```

$report .= gmDate("j:n:Y H:i", $row->time);

```

这提供给我一个对人更友好的日期，而不是我实际保存在数据库中的 Unix 格式日期。换句话说，在表中如果表示为'24 Apr 2007 09 04' 会比'1177405479'好。但是，要用 CI 的 HTML 表格函数做到这一点不容易。(你可能能够在一些数据库系统的数据库查询中做到这一点；但是 MySQL 的日期函数只能操作和保存 MySQL 的独特的日期格式的数据，我们选择改为使用 Unix 格式。)我们在早些时候当我们生成超链接时碰到了相同的麻烦。你不



能这样做。

这于这点有另外一个大问题。格式应该存放在哪里？先前写的所有的代码应该出现在一个控制器中或一个模型中。（我已经把它写在一个模型中，当用户点击一个链接来生成测试报告时被控制器调用，把它放在模型中的理由是这样在我需要的时候可以从几个控制器中调用它。）

MVC 纯粹论者会说你不应该在一个模型中存放格式。它应该放在视图中，当然，的确他们有他们的道理。我可能想要重写代码生成一个文本报表，而且使用 CI 的下载助手以文本的格式下载它而不是屏幕上显示它。（关于下载助手的内容请参考第 11 章。）当我这样做时，我必须改写所有的代码以生成文本控制符而不是 `<tr></tr>` 之类的，甚至更糟糕的是，生成 .rtf 格式或一些类似的富文本格式。

另一个选择是可以使用其它可能的 PHP 语法或模板语法分析器类，把变量或占位符号，放在视图中。（针对这些内容在第 5 章有些简要的论述。）然后，你把真正的，未格式化的数据传递给视图。这可能使 MVC 纯粹论者满意，我的观点是这种什么是把一个复杂的额外的层加入到代码中。但是这仅是我个人的观点，而且多数人会同不同意这一点。

重要的一点是 CI 提供多种选择：它让你自由选择你感到愉快的那一种。没有绝对对或者错的方法：确实有一些方法比另一些表现更好，适合你本人的方法的是最好的方法。

## 14.2 一个平衡表

让我们回顾我们已经在这一本书中涉及的内容。CI 有帮助吗？

### 14.2.1 CI 能提供帮助的地方：结构

即使只观察我们网站一部分比如说一个模型，很明显发现一些有实用价值的应用正在变得很复杂。CI，通过建议或者一定程度上的强迫，构建了一个 MVC 结构，来帮助你使复杂的东西变得更有条理。虽然你还可能忘记你把一段代码放到哪里去了，或者会在不同的控制器或模型中重复编写一个相似的函数：但是它更多的是让你的代码更合乎逻辑。

CI 的 URL 机制帮助你快速从一个代码文件连接到别一个。

CI‘超级对象’没有 namespace 冲突，能够使代码互相调用并互相传递数据。因为每一个变量都有自己的作用域，因此即使同名也都相安无事，不会引起混乱。同时，你能容易地在从你的代码中上存取所有的 CI 资源。

CI 的 ‘config’ 文件鼓励你们建立针对你网站参数的集中设置。

所有的这些好处使你在开发网站时更容易，维护更方便，也更有利于别的程序员看懂代码。

### 14.2.2 CI 能提供帮助的地方：简化

CI 在许多方面帮助你简化代码。也许最好的例子来自 AR 类，但是还有很多其它的方面。CI 把复杂的代码隐藏在函数或一些类库中，做得很优秀，允许你使用一个简单的函数来调用它们。

### 14.2.3 CI 能提供帮助的地方：额外功能

当你使用 CI 的时候，许多 CI 函数带给你许多额外的好处。象 URL 类和 AR 类能够自动读取 config 文件的设置参数，因此你不需要重复数据，因此，针对网站的修改只需要在一个地方进行就可以在整个程序得到应用。

有许多小的例子-容易的方法使你能屏蔽机器人来抓取你的电子邮件住址，举例来说（见第 3 章），又或者 AR 类也能为你准备数据。

事实上，CI 主要的学习曲线（至少对我来说）就是去发现可以走什么捷径，并且请记住，使用它们来代替艰苦漫长的 PHP 编程。如果这一本书能指导你更方便地找到捷径，它就达到了目的。

## 14.3 CI 的问题

CI 并不十全全美。这意味着它本身是一个平衡体：轻巧便捷，而不是复杂和全面。正如有人曾经说过：‘轻量级’意味着“我想要的东西都包含，我不想要的都不包含”。

### 14.3.1 完整性

CI 函数几乎包含了你在开发一般网站时所需要的全部内容。但也有一些 CI 主要函数库中没有包含的例外，而它们之中的绝大多数被 CI 社区中的用户提供的类库所覆盖。（详见下一章），或者你可以通过 PEAR 得到。最明显的省略包括：

AJAX 类

编写网络机器人的类

一个创建加密网站的类，处理登入，网页保护，以及其它的基本会话维护

一个改进的‘脚手架’类，能提供面向外部网络功能，而不仅仅面向开发都。

CI 也从 Rails 书中摘了一页，制作了一个代码生成器，能够帮助开发者建立客制化类。

### 14.3.2 易用性

CI 提供你期待的学习效果。但是，假如你已经了解一些 PHP 知识，会变得相当容易。事实上，我发现 CI 主要的‘学习曲线’是，如果你知道如何用标准 PHP 方法实现某个功能，你就会很容易地用 CI 实现。只有稍后，当你从用户手册中找到更多的内容时，你才会了解，做同一件事，使用 CI 类和助手会变得多么快捷。

私下里讲，我发现有两个 CI 类较难理解和掌握。一个是 XMLRPC 类，另一个是 Validation 类。主要是它们都需要不同网页或不同网站之间的接口。为了使它们工作，正确地设置它们有时候会很困难（分别参见第 8 和第 9 章）

我也发现使用 CI 的“超级对象”一开始也比较困难-见第 7 章。可能是最陡峭的 CI 学习曲线，错误的做法会让你迷惑，有挫败感，直到你掌握它。

其余的：都挺容易的。如果有任何疑问，你总是可以去研究源代码。

## 14.4 总结

这一章我们看到一些编码的例子，还有许多我们已经在以前各章中一点一点地讨论过的函数。

我们还看到 CI 可以：

组织你的网站

简化编程

增加功能

我希望这一本书已经说服你在使用 PHP 开发动态网站时选择 CI。

它还是对开源运动和支持开源的人的贡献，源代码如此丰富，可以容易地，自由地和广泛地得到源代码。谢谢，Rick！

更为慷慨的一个主题，在这本书的最后一章，会提供一些让你更好地开发 CI 应用的资源-CI 用户社区提供进一步的帮助，支持和其它的代码资源。

第十五章 资源和扩展

好了，我们已经相当完整的讲解了 CI，并且，我希望能给你留下深刻的印象。在这个过程中，我们自己也编写了一些代码。我敢肯定，当你看过我的一些代码后，你会开始思考：“我可以写这么好...” 。每个人都有自己的风格，并且 CI 给你很多的自由。

在 CI 社区里有很多可以写出好代码的人，幸好他们中的许多人正准备利用业余时间做这件事情。所以，有很多代码能为您节省大量的工作。拿一个例子来说，你要创建一个数据取自数据库的动态图形，你可以坐下来好好写自己的代码，但实际上，至少有 3 个人已经解决了这个问题，并且他们都把其代码提供给了你。

最后一章介绍一些你可以借鉴的资源，使你的编码快捷又方便。CI 有一个繁荣和活跃的社区，并且可用的资源一直都在发生变化，所以，我也没有试图制作一个完整的清单，只是让你知道这里有什么，并且到哪里可以看到。

这是一个警告提示，太多的混乱代码很容易让人头晕。一些人写项目的时候，只注重代码的漂亮。大多数人都喜欢写代码，而没有写评论和注释的习惯。正因为这样，很多人就不容易理解一些类库和插件的使用方法。

接下来，让我们看看这本书的最后一章所能给我们提供的帮助吧。

首先，让我们看看源代码。  
然后，让我们看几个主题，并比较可用的代码。  
最后，让我们看看更普遍的帮助源：在 PHP、MySQL 和 Apache 上。

15.1 CI 的用户论坛

CI 有两个主要的资源：  
用户论坛的地址是 <http://www.codeigniter.com/forums/>，这里对大多数 CI 问题进行活跃的、几乎连续不断的讨论。评论和建议并不一定总有用（或准确），但也有一些“资深会员”，他们经常贡献很多智慧。它同样是一个非常友好的论坛：人们问一些非常明显的“菜鸟”问题，也会得到耐心并有价值的回复。有时 Rick Ellis 自己也会被一些东西所吸引，但他并没有去涉及所有领域确实是明智之举。

Wiki 的地址是 <http://www.codeigniter.com/wiki/>。这是一个集提示、技巧、Hack、插件和功能强化为一体的知识库，它包含很多有用的代码，尽管涉及的范围还不是很系统化。

论坛和 Wiki 使用起来都很简单：你只需为自己（免费）建立会员账号，然后登陆就可以做自己的事情了。  
如果你认真的使用 CI，那么有必要设置你的 RSS 阅读器来订阅 Wiki 的“近期更改” Feed。

然而，请记住：  
并不是所有的插件编写者象 Rick Ellis 那样技术精湛，他们的作品可能会有一些 BUG 或问题。  
一些在 CI 1.5 版本出现之前编写的早期插件可能需要修改，因为类库的初始化方式已经改变（见第 12 章）。这并不是很难修改，但是也证实了这些类库超出一定范围之后就难以稳定的运行。

15.2 视频教程

如果你想有人手把手的教你完成第一个 CI 应用程序的话，那么，在 CI 的网站上有 3 个优秀的视频教程。  
对 CI 的介绍。  
20 分钟建立一个 Blog。Derek Jones 建立了基本的 Blog 页面，向你展示怎样去设置这个网站，编写数据库查询，以及在视图中展示结果。  
由 Derek Allard 制作的视频（详见 <http://video.derekallard.com/>），它描述了除其他事项外，如何利用 Scriptaculous 类库去集成 AJAX 和 JavaScript 特效。在下面的图片中，它向你展示如何去构建一个用 Ajax 更新的，有自动完成功能的文本输入下拉框。

15.2.1 可用的插件和类库

Rick Ellis 的目的和希望是：CI 的用户能够把“插件”或类库贡献出来用于帮助其他的 CI 用户。这个框架刚出现大约一年，但已经有了很多有趣的代码。  
插件和类库的数量在稳步的增长，已有的插件和类库也在不断的更新。因此下一节并不是一个全面的介绍：只有一部分相关的说明是你觉得有用的。我很抱歉，一直以来我漏掉了很多好资料：请务必亲自查看 Wiki。

15.2.2 AJAX/JavaScript

在 Wiki 里包含了 2 个 AJAX 包：其中一个使用 XAJAX，另外一个使用 prototype.js/scriptaculous.js 类库。

名称	Ajax for CI 1.5.1
URL	<a href="http://www.codeigniter.com/wiki/AJAX_for_CodeIgniter/">http://www.codeigniter.com/wiki/AJAX_for_CodeIgniter/</a>
使用 prototype.js 和 scriptaculous.js 类库	
下载包括了 .js 文件和 .php 文件，以及一个完整的用户指南。（如果你没有很好的掌握 AJAX 和 DOM 的话，这是不容易理解的，不过一些例子有助于你理解它。）简单的安装：把 .php 文件放到你的 application/libraries 文件夹下，把 .js 文件放到你的根目录下。因为刚刚正式发布，所以在 CI 的论坛上只有很少的讨论。	
作者	siric

名称	XAJAX
URL	<a href="http://www.codeigniter.com/wiki/XAJAX/">http://www.codeigniter.com/wiki/XAJAX/</a>
XAJAX 类库的一个 CI “前端”。包括其自己的 Javascript 文件，xajax.js	
作者	Greg McLellan—基于 xajax 的 php 类库（详见 <a href="http://www.xajaxproject.org/">http://www.xajaxproject.org/</a> ）

15.2.3 身份验证

Wiki 上的用户也困于安全问题：这 3 个包用于验证你的用户，并避免保存 Session 数据在 Cookie 中所可能出现的陷阱。

名称	FreakAuth_light
URL	<a href="http://www.4webby.com/freakauth/">http://www.4webby.com/freakauth/</a>
它包括一个类库 用户登录/退出 用户注册 记住密码 修改密码 锁定网站的保留区域 一个后台管理程序： 管理用户 管理管理员 它允许你设置 4 个访问级别（从超级管理员到来宾），然后在控制器中设置一个“check”方法。它既能被设置在控制器的构造函数中，也能被设置在某个指定的函数中。如果用户调用控制器（或指定的函数），这段代码将检查他（她）是否登录。	
在编写这段代码的时候，CI 论坛上正在对其进行广泛的讨论。一些错误已被发现，但是这段代码已发布到第三版了，这些错误看起来已经解决了。	
作者	danfreak

名称	Auth
URL	<a href="http://www.codeigniter.com/wiki/auth">http://www.codeigniter.com/wiki/auth</a>
这个包提供了登录/退出功能，带激活的注册，甚至忘记密码功能。它搭建起来非常复杂：你必须建立一个数据库表，引入一些核心的库文件和帮助文件，然后还要作一些配置。	
在 CI 1.5 下运行。	
作者	(匿名/未知)

名称	DB Session
URL	<a href="http://www.codeigniter.com/wiki/DB_Session/">http://www.codeigniter.com/wiki/DB_Session/</a> <a href="http://dready.jexiste.fr/dotclear/index.php?2006/09/13/19-reworked-session-handler-for-code-igniter">http://dready.jexiste.fr/dotclear/index.php?2006/09/13/19-reworked-session-handler-for-code-igniter</a>
修改 CI 的 Session 类（见第 6 章），它把 Session 数据存储在 Cookie 里。（当然 Cookie 可加密。）这个类只存储 Session 标识符：你在数据库中添加一个额外的表，它将在那里查找所有的 Session 信息。	
在 CI 1.5 下运行。	
作者	dready

#### 15.2.4 外部网站

有一些 CI 的“强大用户”，他们贡献他们自己的代码。一个很好的例子是 Glossopteris，它是一个美国网页设计公司经营的网站。他们提供了一些自己的类库，例如（地址是 <http://www.glossopteris.com/journal/post/table-relations-hips-in-ci>）另外一个 CRUD 类库，他们声称“将允许指定复杂的表内关系，并完成简单的 CRUD 操作。”它遵循 Rails 的惯例：你可以在表之间定义象“一对一”和“一对多”的关系。虽然代码是现成的，但还应该多写一些注释，或者制作一个用户指南。

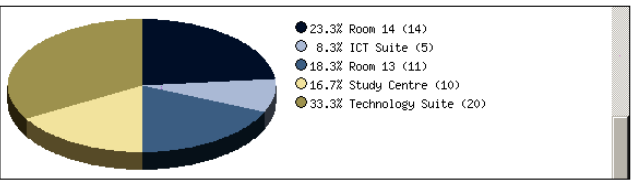
另外一个网站是 CI Forge (<http://www.ciforge.com/>)，它是“一个用于项目设计，为了增强和扩展轻量级 PHP 框架 CodeIgniter 的地方。”它提供 Subversion 和 Trac 主机、一个 Wiki 和一个问题跟踪程序并且支持变更记录。这虽然是一个新的应用，但是（在 2007 年 7 月）已经托管了 20 个项目。

#### 15.2.5 比较：使用哪个图表类库？

图表类库可选择范围很大。有时会有非常多的选择。

为了演示它，让我们看一下做同一件事情的三种选择，并看看他们有什么不同。自己制作动态数据图表不是一件简单的事情，但它确实使你的网站变漂亮了。

让我们来看 3 个 CI 的插件，然后像看他们生成的结果一样尝试去比较他们的强项和弱点。

名称	3d-pie-chart
URL	<a href="http://codeigniter.com/wiki/3d-pie-chart/">http://codeigniter.com/wiki/3d-pie-chart/</a>
由两个数组（标签和值）生成一个饼图，然后把它保存在你的网站上。看起来很棒，但这是它所能做的一切。	
	
简单设置：把 piechart.zip 文件放在你的	

application/libraries 文件夹中，然后写一个基于例子的控制器。需要一个字体文件，然后你需要修改一个视图来显示结果。在 CI 1.5 版本下运行。	
作者	Craig

名称	Panaci
URL	<a href="http://bleakview.orgfree.com/">http://bleakview.orgfree.com/</a> 或 <a href="http://codeigniter.com/wiki/Charting/">http://codeigniter.com/wiki/Charting/</a>

动态生成图表和图形，包括：条，线，区域，阶梯，以及脉冲图（而不是饼图）。在其 Wiki 页面中声明：“请注意，这不是一个像 jgraph 或 chartdirector 的商业级类库，但是它足够能胜任基本的绘图需要”。下面的代码示例和样本图展示了它的样子，以及怎样去使用它。

在 CI 1.5 版本下运行。和 3d-pie-chart 一样，你复制文件到你的 application/libraries 文件夹中，然后从控制器中调用它（提供基本的参数和一个数组数据）。在 CI 论坛上的讨论很简短，所以，在编写本书时没有发现大的错误。

作者	Oscar Bajner
----	--------------

名称	JP Graph
URL	<a href="http://codeigniter.com/wiki/JP_Graph/">http://codeigniter.com/wiki/JP_Graph/</a>

严格的讲，这不是一个插件：它是一段在 CI 和外部 JP Graph 类库之间提供接口的代码。你需要下载 JP Graph 类库，为每个你需要使用的图形类型建立一系列插件，然后当你需要他们时再从控制器中调用。正如其网站上的这些例子图表：<http://www.aditus.nu/jpgraph/features.php> 所显示的，JP Graph 提供了范围更加广阔的图表，并且他们看起来很不错。

JP Graph 有两个缺点。正如其 Wiki 页面中所说：“请记住 JpGraph 有一个非常庞大的代码库，所以请确保为每个图表只引入你需要的类库。”其次，JP Graph 对个人使用是免费的，但是商业使用并不免费。

作者	Aditus Consulting
----	-------------------

关于这三个选择：前两个相对简单，第三个复杂一些。这个取决于你的需求（如果你准备购买的话）。

#### 15.2.6 CRUD：新领域

几乎在所有的应用程序中你都要用到 CRUD。它能简单、逻辑化的自动完成页面的生成。它们是最基础的，虽然它们存在有千百万种的形式。如果不按你的方式制订规则并让用户来遵守，要写出一个应用程序几乎是不可能的。因此，你需要在考虑尽可能多的可能性方面和在简化使用方面做出权衡。你考虑到的特殊情况的可能性越多，代码就会变得越复杂，需要下载的也会越多。

所以，很多人已经开始尝试使 CRUD 的基本操作更简洁。

在第 13 章中，我们已经尝试开发了我们自己的 CRUD 应用程序。这是一个相当简单的模型，它截掉了很多细枝末节，仅仅允许你使用 HTML 表单对象的部分功能；但它确实拥有验证功能。

在本章中，我们已经提到了 Glossopteris 类库。

另外一个有趣的实现是“CodeCrafter”，它被列在 CI 的 Wiki 上面，同时也发表在南非的 Datacraft 软件咨询网站上（<http://www.datacraft.co.za/index.php?contents=codecraft>



[r/codecraft](#))。该网站声称：“CodeCrafter 将会帮助你在仅仅几秒钟内生成你的全部 CodeIgniter 应用程序。”它有一个 26 页的在线手册，该手册向你展示了如何使用它的接口去生成 CI 代码。与大多数其他方法不同的评论如下：它使用图形化的界面为你构建 CI 代码，而不是提供供你调用的类库或代码。

SuperModel (详见

<http://codeigniter.com/wiki/SuperModel/>) 就是：

“SuperModel 类库是模型的一个扩展，使一般的表单生成和验证工作自动化。想一下脚手架 (scaffolding) 吧。”

在 SuperModel 作者的一篇评论中阐述了编写这类代码所面临的困难以及用户所承担的风险。他说：“请注意这个类库是一个正在开发的产品。我目前正在做很多修改，包括对 API 的修改，这些修改将不兼容于旧的程序。在我写此文 (2006 年 5 月 30 日) 的同时，我正在努力实现一对多和多对多的关联查询。不幸的是，这个类库迫使你以某种方式来工作。我试图让它尽可能的小巧灵活，但是同时，在小巧灵活和复杂臃肿之间必须找到一个平衡点，以上就是其作为一个外部第三方类库出现的原因—你可以自由的编写你需要的模型，或者使用其他类似功能的第三方类库。”

### 15.2.7 其它编程资源，例如 Xampplite、MySQL 和 PHP

有许多对 PHP 有用的资源。来让我们略微涉及一下他们。

1. PHP 可以从 [www.php.net](http://www.php.net) 上免费下载，也包括了完整的手册。

2. 廉价的 PHP 编辑器可以从 MP Software <http://www.mpsoftware.dk/> 那里购买。

有许多关于 PHP 的好书，包括《PHP Programming with PEAR》，作者是 Carsten Lucke、Aaron Wormus、Stoyan Stefanov 和 Stephan Schmidt，出版社是 Packt。

在你自己的机器上运行本地 Web 服务器，尝试看一下 <http://www.apachefriends.org/en/index.html>——一个免费提供 XAMPP 包下载的站点。它将安装一个 Apache Web 服务器、MySQL、PHP 和 Perl。如果 XAMPP 包对你来说太全面，请尝试这个站点上的 Minixampp，本书的代码就是在这个环境下编写的。

MySQL 同样拥有自己的网页——<http://www.mysql.com/>——不过，如果你想免费下载最新版本，请到 <http://dev.mysql.com/>。

(请记住虽然很多 ISP 都没有使用最新的版本。虽然 MySQL 的最新版本是 5，但是大多数 ISP 仍旧使用版本 4。这阻止了你使用某些更有趣的新特性，例如存储过程。) 更多内容请参考

《Creating your MySQL Database: Practical Design Tips and Techniques》，作者是 Marc Delisle，出版社是 Packt。

虽然 MySQL 有它自己的工具，但是最流行 (也是最常见) 的工具是 PHPMyAdmin。(更多内容请参考《Mastering phpMyAdmin 2.8 for Effective MySQL Management》，作者也是 Marc Delisle，出版社是 Packt。)

### 15.3 总结

在本章中，我们为你展示了一些当你开始用 CI 编程时要用到的资源。有很多现成的代码可以使用，你需要在使用之前好好的看看它们：不要一看到满足你需要的插件或类库就直接开始使用。你需要好好的研究这些代码，看看他们是如何工作的，这对你整体把握代码很有帮助，并且有利于你更好的理解它。不管怎样，只要你决定使用 CI 框架，你就能找到各种不同层次和复杂程度的类库，它们将完成很多要由你手工完成的任务。

我们详细介绍了这些类库：

AJAX 和 JavaScript

身份验证

图表

CRUD

最后，我们介绍了一些关于 PHP、MySQL 和运行一个本地 Web 服务器所需要的资源。