MANUAL DE

Code Igniter

En español

Traducción
Pablo Martínez, Pablo Ruiz Díaz, Sebastián Waisbrot
Ver créditos

Recopilación y diagramación
Víktor Lozano
Conocimiento Virtual Academia, Colombia
www.conocimientovirtual.edu.co

Créditos

Code Igniter® es una marca registrada de EllisLab inc.
http://codeigniter.com/
http://ellislab.com/

URL ayuda de Code Igniter en inglés:
http://codeigniter.com/user_guide/

URL ayuda de Code Igniter en español:
http://neonetsi.com.ar/seppo/traduccion/

Traducción
Pablo Martinez (http://codeigniter.com/forums/member/61914/)
pablomartinez81@gmail.com

Pablo Ruiz Diaz (http://codeigniter.com/forums/member/61648/)
pablito.federico@gmail.com

Sebastian Waisbrot (http://codeigniter.com/forums/member/40301/)
seppo0010@gmail.com

Recopilación y Diagramación
Viktor Lozano
viktorlozano@gmail.com
Conocimiento Virtual Academia ltda.
Medellín, Colombia
www.conocimientovirtual.edu.co

Contenido

Esta página está en blanco de forma intencional

Información Básica

Esta página está en blanco de forma intencional

Requisitos del Servidor

- PHP (http://www.php.net) versión 4.3.2 o superior.
- Una base de datos es necesaria para la programación de la mayoría de aplicaciones web. Actualmente las base de datos soportadas con MySQL (4.1+), MySQLi, MS SQL, Postgre, Oracle, SQLite, y ODBC.

Acuerdo de Licencia de CodeIgniter

IMPORTANTE: La siguiente traducción es meramente informativa y no tiene ninguna validez legal. Para consultar la licencia de valor legal por favor diríjase a la licencia original (en inglés).

Esta licencia es un acuerdo legal entre usted y EllisLab Inc. para el uso de el Software CodeIgniter (el «Software»). Al obtener el Software usted accede a cumplir con los términos y condiciones de esta Licencia.

Uso Permitido

Usted puede usar, copiar, modificar y distribuir el Software y su documentación, con o sin modificaciones, para cualquier propósito, siempre y cuando las siguientes condiciones se cumplan:

1. Una copia de este acuerdo de licencia debe ser incluido con la distribución.
2. Las redistribuciones del código fuente debe retener la leyenda de copyright en todos los archivos de código fuente.

3. Las redistribuciones en forma binaria deben reproducir la leyenda de copyright en la documentación y/u otros materiales provistos con la distribución.
4. Cualquier archivo que haya sido modificado debe contener mensajes estableciendo la naturaleza del cambio y el nombre de aquellos que lo hayan cambiado.
5. Los productos derivadas del Software deben incluir un reconocimiento que ellos son derivados de CodeIgniter en su documentación y/u otros materiales provistos con la distribución.
6. Los productos derivados del Software no pueden ser llamado «CodeIgniter», ni «CodeIgniter» puede aparecer en su nombre, sin el consentimiento por escrito de EllisLab, Inc.

Indemnización

Usted accede a la indemnización y exención de responsabilidad a los autores del Software y cualquier contribuidor por cualquier directa, indirecta, incidental, o consecuente reclamo, acciones o juicios de tercero, así bien como cualquier gasto, exigible, daño, liquidación o tarifa que surja del uso o mal uso del Software, o una violación de cualquier término de esta licencia.

Responsabilidad de Garantía

ESTE SOFTWARE ES PROVISTO «COMO ESTÁ», SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUYENDO, PERO NO LIMITADO A, GARANTÍAS DE CALIDAD, DESEMPEÑO, NO-INFRACCIÓN, MERCANTIBILIDAD, O AJUSTE POR CUALQUIER PROPÓSITO PARTICULAR.

Limitaciones de Resposabilidad

USTED ASUME TODO RIESGO ASOCIADO CON LA INSTALACIÓN Y USO DEL SOFTWARE. EN NINGÚN CASO PUEDEN LOS AUTORES O POSEEDORES DE DERECHOS DE AUTOR DEL SOFTWARE SER RESPONSABLES DE REIVINDICACIÓN, DAÑOS, U OTROS EXIGIBLES QUE SURJAN DESDE, POR, O EN CONEXIÓN CON EL SOFTWARE. LOS POSEEDORES DE LOS DERECHOS DE AUTOR SON SOLAMENTE RESPONSABLES POR DETERMINAR LO APROPIADO DEL USO Y ASUMEN TODOS LOS RIESGOS ASOCIADOS A SU USO, INCLUYENDO

PERO NO LIMITANDOSE AL RIESGO DE ERRORES DEL PROGRAMA, DAÑO AL EQUIPAMIENTO, PÉRDIDA DE DATOS O PROGRAMAS DE SOFTWARE, O INDISPONIBILIDAD O INTERRUPCIÓN DE OPERACIONES.

Creditos

CodeIgniter fue desarrollado por Rick Ellis, quien en su otra vida es Director Ejecutivo de Ellislab, Inc. El Núcldeo del entero de trabajo fue escrito específicamente para esta aplicación, mientras que muchas de las clases de librerías, asistentes y sub-sistemas fueron tomados prestados desde el código básico de ExpressionEngine, un Sistema de Manejo de Contenido (CMS) escrito por Rick Ellis y Paul Burdick.

Un mención especial va para Ruby on Rails por inspirarnos a crear un entorno de trabajo en PHP, y por llevar a los entornos de trabajo a la conciencia general de la comunidad web.

El logo de CodeIgniter y los iconos fueron creados por Rick Ellis.

Esta página está en blanco de forma intencional

Instalación

Esta página está en blanco de forma intencional

Bajando CodeIgniter

La última versión de Code Igniter se puede descargar de la página web http://www.codeigniter.com

Servidor de Subversion

El acceso publico a la subversion está ahora disponible a través de http://dev.ellislab.com/svn/CodeIgniter/trunk tenga en cuenta que mientras se hace todo lo posible para mantener este código funcional, no podemos garantizar la funcionalidad del código tomado del repositorio.

Subversion es un sistema de control de versiones (http://subversion.tigris.org/).

Instrucciones de Instalación

CodeIgniter se instala en cuatro pasos:

1. Descomprima el paquete.
2. Suba los archivos y carpetas de CodeIgniter a su servidor. Normalmente el archivo index.php será su raiz.
3. Abra el archivo *application/config/config.php* con un editor de texto y establezca su URL de base.
4. Si tiene intención de utilizar una base de datos, abra el archivo *application/config/database.php* con un editor de texto y establezca su configuracion de base de datos.

Si desea incrementar la seguridad ocultando la localizacion de sus archivos CodeIgniter, puede renombrar la carpeta *system* a algo mas privado. Si decide renombrar, debe abrir su archivo principal `index.php` y establecer la variable

`$system_folder` en la parte superior de la página con el nuevo nombre que ha elegido.

Eso es todo!

Si usted es nuevo en CodeIgniter, por favor lea la seccion **Cómo Empezar** de la Guia de Usuario para empezar a aprender cómo construir aplicaciones dinámicas en PHP. Disfrutá!

Actualizando Desde una Versión Anterior

Actualmente (26 de junio de 2008) Code Igniter se encuentra en la versión 1.6.2. Si usted está leyendo este manual en una fecha muy posterior (a la indicada entre paréntesis) o si usted tiene una versión previa y desea actualizarse, por favor lea las notas de actualización correspondientes a la de versión desde la que está actualizando en la página http://codeigniter.com/downloads/

Solución de problemas

Si encuentra que, no importa lo que ponga en su URL, sólo se carga su página por defecto, puede ser que el servidor no soporte la variable PATH_INFO necesaria para servir a los motores de busqueda de URL amigables. Como primer paso, abra su archivo *application/config/config.php* y busque la informacion `Protocolo URI`. Se recomienda que pruebe un par de ajustes alternativos. Si aun no funciona después de que ha intentado esto, tendrá que añadir a la fuerza en CodeIgniter un signo de pregunta a su URL. Para hacer esto abra su archivo `application/config/config.php` y cambie esto:

```
$config['index_page'] = "index.php";
```

por esto:

```
$config['index_page'] = "index.php?";
```


Introducción

Esta página está en blanco de forma intencional

CodeIgniter a primera vista

CodeIgniter es un Entorno de Trabajo para Aplicaciones

CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación web usando PHP. Su objetivo es permitirle desarrollar proyectos mucho más rápido de lo que podría si lo escribiese desde cero, proveyendole un rico juego de librerías para tareas comunmente necesarias, así como una interface simple y estructura lógica para acceder a esas librerías. CodeIgniter le permite creativamente enfocarse en su proyecto minimizando la cantidad de código necesaria para una tarea dada.

CodeIgniter es Libre

CodeIgniter se encuentra bajo una licencia open source Apache/BSD-style, así que lo puede usar donde le parezca. Para más información por favor lea el acuerdo de licencia.

CodeIgniter Corre en PHP 4

CodeIgniter está escrito para ser compatible con PHP 4. Aunque nos hubiese encantado tomar ventaja del mejor manejo de objetos en PHP 5 ya que hubiese simplificado algunas cosas, hemos tenido que buscarle soluciones creativas (ver su camino, herencia múltiple), al momento de este escrito PHP 5 no es de uso extendido, lo que significa que estaríamos aislando a la mayoría de nuestra potencia audencia. Los principales proveedores de Sistemas Operativos como RedHat se están moviendo lentamente a soportar PHP 5, y es poco probable que lo hagan en el corto plazo, así que sentimos que no servía a los mejores intereses de la comunidad de PHP escribir CodeIgniter en PHP 5.

> Nota: CodeIgniter correrá en PHP 5. Simplemente no toma ventaja de cualquiera de las características nativas que sólo están disponibles en esa versión.

CodeIgniter es Liviano

Verdaderamente Liviano. El núcleo del sistema sólo requiere unas pocas pequeñas librerías. Esto es en duro contraste a muchos entornos de trabajo que requieren significativamente más recursos. Las librerías adicionales son cargadas dinámicamente a pedido, basado en sus necesidades para un proceso dado, así que el sistema base es muy delgado y bastante rápido.

CodeIgniter es Rápido

Realmente rápido. Le desafiamos a encontrar un entorno de trabajo que tenga mejor desempeño que CodeIgniter.

CodeIgniter Usa M-V-C

CodeIgniter usa el acercamiento Modelo-Vista-Controlador, que permite una buena separación entre lógica y presentación. Esto es particularmente bueno para proyecto en los cuales diseñadores están trabajando con sus archivo de plantilla, ya que el código en esos archivos será mínimo. Describimos MVC en más detalle en su propia página.

CodeIgniter Genera URLs Limpias

Las URLs generadas por CodeIgniter son limpias y amigables a los motores de búsqueda. En vez de usar el acercamiento estándar «query string» a las URLs que es sinónimo de sistemas dinámicos, CodeIgniter usa un acercamiento basado en segmentos:

Nota: Por defecto el archivo index.php es incluido en su URL pero puede ser removido usando un simple archivo .htaccess.

CodeIgniter trae un Puñado de Paquetes

CodeIgniter viene con un rango lleno de librerías que le permiten realizar las tareas de desarrollo web más comunmente necesarias, como acceder a una base de datos, mandar un email, validar datos de un formulario, mantener sesiones, manipular imagenes, trabajando con datos XML-RPC y mucho más.

CodeIgniter es Extensible

El sistema puede ser fácilmente extendido a través del uso de plugins y librerías asistentes, o a través de extensión de clases o ganchos del sistema.

CodeIgniter No Requiere un Motor de Plantillas

Aunque CodeIgniter *SI* viene con un motor de plantillas simple que puede ser opcionalmente usado, no le fuerza a usarlo. Los motores de plantilla simplemente no pueden igualar el desempeño del nativo PHP, y la sintaxis que debe ser aprendida para usar un motor de plantilla es usualmente sólo marginalmente más fácil que aprender la base de PHP. Considere este bloque de código PHP:

```
<ul>
    <?php foreach ($libretadedirecciones as $nombre):?>
    <li><?=$nombre?></li>
    <?php endforeach; ?>
</ul>
```

Contraste con el pseudocódigo usado por el motor de plantillas:

```
<ul>
    {foreach from=$libretadedirecciones item="nombre"}
    <li>{$nombre}</li>
    {/foreach}
</ul>
```

Sí, el ejemplo del motor de plantillas es un poco más claro, pero viene con el precio del desempeño, ya que el pseudocódigo debe ser convertido de vuelta en PHP para correr. Ya que nuestra meta es *máximo desempeño*, hemos optado por no requerir el uso de un motor de plantillas.

CodeIgniter está Minuciosamente Documentado

Los programadores aman hacer código y odian escribir documentación. No somos diferentes, por supuesto, pero ya que la documentación es **tan importante** como el código mismo, estamos comprometidos a hacerlo. Nuestro código fuente es extremadamente limpio y bien documentado también.

CodeIgniter tiene una Amigable Comunidad de Usuarios

Nuestra creciente comunidad de usuarios puede ser vista participando activamente en nuestros Foros Comunitarios (en inglés).

Características de CodeIgniter

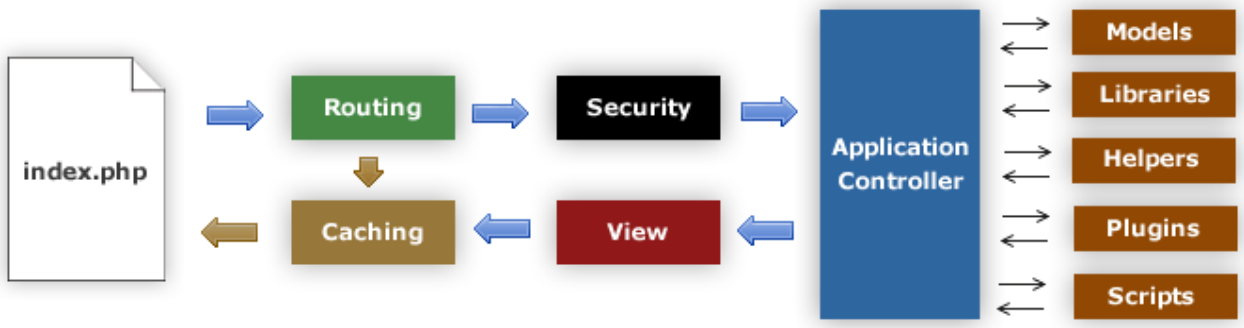
Las características en y de ellas son una forma muy pobre de juzgar una aplicación ya que no dicen nada acerca de la experiencia del usuario, o cuan intuitivamente o inteligentemente es diseñado. Las características no revelan nada acerca de la calidad del código, o la performance, o la atención a los detalles, o las prácicas de seguridad. La única forma de realmente juzgar una aplicación es probarla y llegar a conocer el código. Instalando CodeIgniter es un juego de niños así que alentamos que haga eso. Mientras tanto aquí hay una lista de las características principales de CodeIgniter.

- Sistema Basado en Modelo-Vista-Controlador
- Compatible con PHP 4
- Extremadamente Liviano
- Clases de base de datos llenas de características con soporte para varias plataformas.
- Soporte de Active Record para Base de Datos
- Formulario y Validación de Datos
- Seguridad y Filtro XSS
- Manejo de Sesión
- Clase de Envío de Email. Soporta Archivos Adjuntos, email de texto/HTML, múltiples protocolos (sendmail, SMTP, and Mail) y más.
- Librería de Manipulación de Imagen (cortar, redimensionar, rotar, etc.). Soporta GD, ImageMagick, y NetPBM
- Clase de Carga (upload) de Archivo
- Clase de FTP
- Localización
- Paginación
- Encriptación de Datos
- Puntos de referencia
- Cacheo de páginas enteras
- Historial de Errores
- Perfilando la Aplicación
- Scaffolding
- Clase de Calendario
- Clase de Agente del Usuario
- Clase de Codificación Zip
- Clase de Motor de Plantillas

- Clase de Trackback
- Librería XML-RPC
- Clase de Prueba de Unidad
- URLs amigables a motores de búsqueda
- Ruteo de URI Flexible
- Soporte para Ganchos, Extensiones de Clase y Plugins
- Larga librería de funciones "asistentes"

Diagrama de Flujo de la Aplicación

El siguiente gráfico ilustra como fluyen los datos a través del sistema:



1. El index.php sirve como controlador frontal, inicializando los recursos básicos necesarios para correr CodeIgniter.
2. El Router examina la petición HTTP para determinar que debe ser hecho con él.
3. Si un archivo de caché existe, es enviado directamente al explorador, sobrepasando el sistema de ejecución normal.
4. Seguridad. Antes que el controlador sea cargado, la petición HTTP y cualquier dato suministrado por el usuario es filtrado por segurdad.
5. El controlador carga los modelos, librerías, plugins, asistentes y cualquier otro recurso necesario para procesar la petición específica.
6. La Vista finalizada es presentada entonces envíada al exploreador web para ser vista. Si el cacheo está habilitado, la vista es cacheada primero para que las peticiones subsecuentes puedan ser servidas.

Modelo-Vista-Controlador

CodeIgniter está basado en el patrón de desarrollo Modelo-Vista-Controlador. MVC es una aproximación al software que separa la lógica de la aplicación de la presentación. En la práctica, permite que sus páginas web contengan mínima codificación ya que la presentación es separada del código PHP.

> El **Modelo** representa la estructura de datos. Típicamente sus clases de modelo contendrán funciones que lo ayudarán a recuperar, insertar y actualizar información en su base de datos.
>
> La **Vista** es la información que es presentada al usuario. La Vista normalmente será una página web, pero en CodeIgniter, una vista también puede ser un fragmento de una página como un encabezado o un píe de página. También puede ser una página RSS, o cualquier otro tipo de "página".
>
> El **Controlador** sirve como un *intermediario* entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la petición HTTP y generar una página web.

CodeIgniter tiene un enfoque bastante flexible del MVC, ya que los Modelos no son requeridos. Si no necesita agregar separación, o descubre que mantener los modelos requiera más complejidad que quería, puede ignorarlos y construir su aplicación mínimamente usando Controladores y Vista. CodeIgniter también le permite incorporar sus código existentes, o incluso desarrollar librerías de núcleo para el sistema, habilitandolo a trabajar en una forma que hace que tenga más sentido para usted.

Diseño y Metas Arquitectónicas

Nuestra meta para CodeIgniter es *máxima performance, capacidad, y flexibilidad en el más pequeño, liviano paquete posible*.

Para llegar a este objetivo estamos comprometidos a utilizar tiempos de referencia, refactorizar, y simplificar en cada paso del proceso de desarrollo, rechazando cualquier cosa que no avance en el objetivo establecido.

Desde un punto de vista técnico y arquitectonico, CodeIgniter fue creado con los siguientes objetivos:

- **Instanciación Dinámica.** En CodeIgniter, los componentes son cargados y las rutinas ejecutadas sólo cuando se pide, en vez de globalmente. No se hacen

suposiciones por el sistema acerca de que puede necesitar además de los recursos mínimos del núcleo, así que el sistema es muy liviano por defecto. Los eventos, que se desencadenan por la petición HTTP, y los controladores y vistas que diseñe determinarán que es invocado..

- **Poco Acomplamiento.** Acomplamiento es el grado que los componentes de un sistema dependen entre ellos. Mientras menos componentes dependan de otro, más reusable y flexible el sistema se vuelva. nuestra meta era un sistema con muy poco acoplamiento.
- **Singularidad del Componente.** Singularidad es el grado que más componentes tienen un proposito en el que enfocarse más estrecho. En CodeIgniter, cada clase y sus funciones son altamente autónomas para permitir máxima utilidad.

CodeIgniter es un sistema instanciado dinámicamente, poco acoplado con gran singularidad de componente. Se esfuerza para la simplicidad, flexibilidad y alta performance en un paquete de pequeña pisada.

Esta página está en blanco de forma intencional

Tópicos generales

Esta página está en blanco de forma intencional

Cómo Empezar Con CodeIgniter

Cualquier aplicación de software requiere un poco de esfuerzo para aprender. Hemos hecho todo lo posible para minimizar la curva de aprendizaje y al mismo tiempo hacer que el proceso sea lo más agradable posible.

El primer paso es instalar CodeIgniter, entonces lea todos los temas en la sección **Introducción** de la Tabla de Contenidos.

A continuación, cada una de las páginas de los **Temas Generales** en órden. Cada tema se basa en la anterior, e incluye ejemplos de código que le animamos a probar.

Una vez que usted entienda los conceptos básicos, estará dispuesto a estudiar la **Referencia de Clases** y las páginas de la **Referencia de Archivos de Ayuda (Helpers)** para aprender a utilizar las librerias nativas y los archivos de ayuda (helpers).

No dude en aprovechar nuestra Comunidad de Foros si tiene preguntas o problemas, y nuestra Wiki para ver ejemplos de código enviados por otros usuarios.

URLs en CodeIgniter

Por defecto, las URLs en CodeIgniter están diseñadas para ser amigable a los motores de búsqueda y amigables a los humanos. En vez de usar el estándar acercamiento "query string" para URLs que es sinónimo con sistemas dinámicos, CodeIgniter usa un aproximamiento **basado-en-segmentos**:

```
www.su-sitio.com/noticias/articulo/mi_articulo
```

Nota: Las URLs con "query string" pueden ser habilitadas opcionalmente, como se describe abajo.

Segmentos URI

Los segmentos en la URL, en consecuencia con la aproximación Modelo-Vista-Controlador, usualmente representa:

```
www.su-sitio.com/clase/funcion/ID
```

1. El primer segmento representa la **clase** controlador que debe ser invocado.
2. El segundo segmento representa la **función** de la clase, o método, que debe ser llamado.
3. El tercer, y cualquier segmento adicional, representa el ID y cualquier variable que será pasada al controlador.

La Clase de URI y el Asistente de URL contiene funciones que hacen fácil trabajar con datos de URI. En adición, sus URLs pueden ser remapeados usando la característica de Routeo de URI para más flexibilidad.

Remover el archivo index.php

Por defecto, el archivo **index.php** será incluido en sus URLs:

```
www.su-sitio.com/index.php/noticias/articulo/mi_articulo
```

Puede fácilmente remover este archivo usando un archivo .htaccess con algunas simples reglas. Aquí hay un ejemplo de tal archivo, usando el método "negativo" en el cual todo es redireccionado excepto los items especificado:

```
RewriteEngine on
RewriteCond $1 !^(index\.php|images|robots\.txt)
RewriteRule ^(.*)$ /index.php/$1 [L]
```

En el ejemplo anterior, cualquier pedido HTTP distinto del que sea para index.php, images, y robots.txt es tratado como un pedido para el archivo index.php.

Agregando un sufijo a la URL

En su archivo *config/config.php* puede especificar un sufijo que será agregado a todas las URLs generada por CodeIgniter. Por ejemplo, si una URL es esta:

```
www.su-sitio.com/index.php/productos/ver/zapatillas
```


Puede opcionalmente agregar un sufijo, como `.html`, haciendo la página aparecer ser de un cierto tipo:

```
www.su-sitio.com/index.php/productos/ver/zapatillas.html
```

Habilitando "Query Strings"

En algunos casos, puede preferir usar URLs con query strings:

```
index.php?c=productos&m=ver&id=345
```

CodeIgniter opcionalmente soporta esta capacidad, la cual puede ser habilitada en su archivo *application/config.php*. Si abre su archivo de configuración verá estos items:

```
$config['enable_query_strings'] = FALSE;
$config['controller_trigger'] = 'c';
$config['function_trigger'] = 'm';
```

Si cambia "enable_query_strings" a TRUE esta característica se activará. Sus controladores y funciones serán entonces accesibles usando las palabras "trigger" que haya establecido para invocar sus controladores y métodos:

```
index.php?c=controlador&m=metodo
```

Por favor note: Si está usando query strings tendrá que construir sus propias URLs, en vez de utilizar los asistentes de URL (y otros asistentes que generan URLs, como alguno de los asistentes de formularios) ya que ellos están diseñados para trabajar con URLs basadas en segmentos

Controladores

Los controladores son el corazón de tu aplicación, es el encargado de determinar, como las solicitudes HTTP deben ser manejadas.

Qué es un controlador?

Un controlador es simplemente un archivo de clase que es llamado en una forma que puede ser asociado con un URI.

considere esta URI:

```
www.your-site.com/index.php/blog/
```

En el ejemplo anterior, Codeigniter trataría de encontrar un controlador llamado *blog.php* y lo cargaría.

Cuando el nombre de un controlador coincide con el primer segmento de una URI, esta será cargada.

Vamos a intentarlo: Hola mundo!

Vamos a crear un controlador sencillo, así lo podrás ver en acción. Usando tu editor de texto, cree un archivo llamado *blog.php*, y coloca dentro el siguiente código:

```
<?php
class Blog extends Controller {
    function index()
    {
        echo 'Hello World!';
    }
}
?>
```

Luego guarda el archivo en tu carpeta *application/controllers/*.

Ahora vista tu sitio usando una URL similar a esto:

```
www.your-site.com/index.php/blog/
```

Si lo has hecho correctamente, deberías ver `Hola mundo!`.

Nota: Los nombres de Clases deben empezar con una letra mayúscula. En otras palabras, esto es válido:

```
<?php
class Blog extends Controller {

}
?>
```

Esto **no** es válido:

```
<?php
class blog extends Controller {

}
?>
```

También, siempre debes asegurarte de que tu controlador *herede* de la clase padre "Controller" así podrá heredar todas sus funciones.

Funciones

En el ejemplo anterior el nombre de la función es *index()*. La función "index" es siempre cargada por defecto si el **segundo segmento** de la URI está vacía. Otra forma de mostrar tu mensaje "Hola mundo!" podría ser esta:

```
www.your-site.com/index.php/blog/index/
```

El segundo segmento de la URI determina que función en el controlador será llamada.

Vamos a probarlo. Agrega una nueva función a tu controlador:

```
<?php
class Blog extends Controller {
    function index()
    {
        echo 'Hola mundo!';
    }
    function comments()
    {
        echo 'Mira esto!';
    }
}
?>
```

Ahora carga la siguiente URL para ver la función *comment*:

```
www.your-site.com/index.php/blog/comments/
```

Deberias ver el siguiente mensaje. `Mira esto!`

Pasando Segmentos URI a tus funciones

Si tu URI contiene mas de dos segmentos, ellos seran pasados a tu función como parámetros.

Por ejemplo, digamos que tienes una URI como esta:

```
www.your-site.com/index.php/products/shoes/sandals/123
```

A tu funcion se le pasaran los segmentos URI 3 y 4 ("sandals" y "123"):

```
<?php
class Products extends Controller {
    function shoes($sandals, $id)
    {
        echo $sandals;
        echo $id;
    }
}
?>
```

Importante: Si estas usando la característica URI Routing, los segmentos pasados a tus funciones seran re-ruteadas.

Definiendo un Controlador por Defecto

Se le puede decir a CodeIgniter que cargue un controlador por defecto, cuando una URI no esta presente, ese sería el caso cuando solamente la raíz del sitio es llamada. Para especificar un controlador por defecto abre tu archivo *application/config/routes.php* y coloca este valor a la variable:

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


Donde *Blog* es el nombre de la clase del controlador que tú quieres usar. Si ahora cargas tu archivo principal index.php sin especificar ningún segmento URI podrás ver el mensaje Hola Mundo! por defecto.

Remapeando llamadas a Funciones

Como señalamos anteriormente, el segundo segmento de la URI típicamente determina cual función en el controlador sera llamado. CodeIgniter te permite anular este comportamiento a través del uso de la función `_remap()`:

```
function _remap()
{
    // Algún codigo por aquí...
}
```

Importante: Si tu controlador contiene una función llamada _remap(), esta será **siempre** llamada independientemente de que contenga tu URI. Esto anula el normal comportamiento en el cual la URI determina que función es llamada, permitiéndote definir tus propias reglas de ruteo.

La llamada a la función anulada (típicamente el segundo segmento de la URI) será pasada como parámetro a la función `_remap()`:

```
function _remap($method)
{
    if ($method == 'algun_metodo')
    {
        $this->$method();
    }
    else
    {
        $this->metodo_por_defecto();
    }
}
```

Procesando la Salida

CodeIgniter tiene una clase de salida que se encarga de enviar tus datos finales al navegador web automáticamente. Más información sobre esto puede ser encontrado en las páginas Views (vistas) y Output class (Clase Salida). En algunos casos, sin embargo, tu podrías querer post-procesar los datos finalizados en alguna forma para enviarlo al navegador por tu cuenta. CodeIgniter te lo permite agregando una función llamada *_output()* al controlador que recibe la salida de datos finalizada.

> **Importante:** Si tu controlador contiene una función llamada _output(), esta sera **siempre** llamada por la clase "Output" en lugar de imprimir el dato finalizado directamente. El primer parametro de la función deberá contener la salida finalizada.

Aqui tenemos un ejemplo:

```
function _output($output)
{
    echo $output;
}
```

Por favor note que tu funcion _output() deberá recibir el dato en su estado finalizado. El punto de referencia y el uso de memoria, los datos serán prestados, los archivos cache serán escritos (si tu tienes habilitado el cacheo), y las cabeceras serán enviadas (si usa esa caraterística) antes de que esta sea entregada a la función _output(). Si estas usando esta característica el tiempo de ejecucíon de la página y el uso de memoria podría no ser preciso ya que no toma en cuenta ninguno de los procesos que tu haces. Para una alternativa a la forma de controlar la salida antes de que ninguno de los procesos finales estén hechos, por favor vea los métodos disponibles en la clase Output.

Funciones Privadas

En algunos casos usted podría querer esconder ciertas funciones del acceso publico. Para hacer una funcion privada, simplemente agregue un guion bajo como prefijo y esta no podrá ser accedida via pedido URL. Por ejemplo, si tuvieras una función como esta:

```
function _utility()
{
  // algún codigo aquí
}
```

Tratar de accederlo via la URL, como esta, no funcionará:

```
www.your-site.com/index.php/blog/_utility/
```

Organizando Tus Controladores en sub-carpetas

Si tu estas construyendo una gran aplicación usted podría encontrar conveniente organizar tus controladores en sub-carpetas. CodeIgniter te permite hacer esto.

Simplemente debes crear las carpetas dentro de tu directorio *application/controllers* y ubicar tus clases de controladores dentro.

Nota: Cuando usamos esta característica usando esta característica el primer segmento de tu URI debe especificar la carpeta. Por ejemplo, digamos que tienes un controlador ubicado aqui:

```
application/controllers/products/shoes.php
```

Para llamar al controlador anterior tu URI deberia ser algo similar a esto:

```
www.your-site.com/index.php/products/shoes/123
```

Cada una de tus sub-carpetas deberan contener un controlador por defecto el cual sera llamado si la URL contiene solamente la sub-carpeta. Simplemente nombra a tu controlador por defecto como lo especificaste en tu archivo *application/config/routes.php*

CodeIgniter tambien te permite remapear tus URIs usando su caracteristica URI Routing.

Constructores de Clase

Si tu piensas usar un constructor en cualquiera de tus Controladores, tu **DEBES** ubicar la siguiente linea de codigo dentro de el:

```
parent::Controller();
```

La razón de esta linea, es que es necesaria porque tu constructor local anularia al constructor de la clase padre Controller, por lo tanto necesitamso llamarlo manualmente.

Si no estas familiarizado con los constructores, en PHP4, un *constructor* es simplemente una funcion que tiene exactamente el mismo nombre que que la clase:

```
<?php
class Blog extends Controller {

    function Blog()
    {
    parent::Controller();
    }
}
?>
```

En PHP 5, los constructores siguen la siguiente sintaxis:

```
<?php
class Blog extends Controller {

    function __construct()
    {
    parent::Controller();
    }
}
?>
```

Los Constructores son utiles si necesitas colocar valores por defecto, o correr algon proceso cuando la clase es instanciada. Los constructores no pueden retorna un valor, pero pueden hacer algun trabajo por defecto.

Nombres de Funciones Reservadas

Desde que tus clases controlador heredan de la aplicacion principal controlador debes tener cuidado de no llamar a tus funciones de una forma identica a las que son usadas por las clases, de otro modo tus funciones locales las anularían. A continuación una lista de palabras reservadas. No llames a ninguna de tus funciones de igual forma a alguna de estas:

- Controller
- CI_Base
- _ci_initialize
- _ci_scaffolding

Si está usando PHP 4 hay algunos nombres adicionales más. Estos ÚNICAMENTE se aplican si está corriendo PHP 4.

- CI_Loader
- config
- database
- dbutil
- dbforge
- file
- helper
- helpers
- language
- library
- model
- plugin
- plugins
- scaffolding
- script
- view
- vars
- _ci_assign_to_models
- _ci_autoloader
- _ci_init_class
- _ci_init_scaffolding
- _ci_is_instance
- _ci_load
- _ci_load_class
- _ci_object_to_array

Funciones

- is_really_writable()
- load_class()
- get_config()
- config_item()
- show_error()
- show_404()
- log_message()
- _exception_handler()
- get_instance()

Variables

- $config
- $mimes
- $lang

Constantes

- EXT
- FCPATH
- SELF
- BASEPATH
- APPPATH
- CI_VERSION
- FILE_READ_MODE
- FILE_WRITE_MODE
- DIR_READ_MODE
- DIR_WRITE_MODE
- FOPEN_READ
- FOPEN_READ_WRITE

- FOPEN_WRITE_CREATE_DESTRUCTIVE
- FOPEN_READ_WRITE_CREATE_DESTRUCTIVE
- FOPEN_WRITE_CREATE
- FOPEN_READ_WRITE_CREATE
- FOPEN_WRITE_CREATE_STRICT
- FOPEN_READ_WRITE_CREATE_STRICT

Eso es todo!

Eso, en pocas palabras, es todo lo que necesita saber sobre controladores.

Vistas

Una *vista* es simplmenete una página web, o un fragmento de ella, como un encabezado, un píe de página, una barra lateral, etc. De hecho, las vistas pueden ser flexiblemente embebidas dentro de otras vistas (dentro de otras vistas, etc., etc.) si necesita este tipo de jerarquía.

Las vistas nunca son llamadas directamente, ellas deben ser cargadas por un controlador. Recuerda que en un entorno de trabajo MVC, el Controlador actua como el "policia de tránsito", así que es responsable de traer una vista en particular. Sino ha leído la página del Controlador debería hacerlo antes de continuar.

Usando el controlador que creó en la página de controlador, le permite agregar una vista a él.

Crear una vista

Usando su editor de texto, cree un archivo llamado *vistablog.php*, y ponga esto en él:

```
<html>
    <head>
        <title>My Blog</title>
    </head>
    <body>
        <h1>Welcome to my Blog!</h1>
    </body>
</html>
```

Entonces guarde el archivo en su carpeta *application/views/*.

Cargando una Vista

Para cargar un archivo de vista en particular, usará la siguiente función:

```
$this->load->view('nombre');
```

Donde *nombre* es el nombre de su archivo de vista. Nota: no es necesario especificar la extensión .php del archivo a menos que use una distinta de `.php`.

Ahora, abra el archivo controlador que hizo previamente llamado *blog.php*, y reemplace la sentencia "echo" con la función de carga de vista:

```
<?php
class Blog extends Controller {
    function index()
    {
        $this->load->view('blogview');
    }
}
?>
```

Si visita su sitio usando la URL que usó antes, debería ver su nueva vista. La URL era similar a esta:

```
www.su-sitio.com/index.php/blog/
```

Cargando múltiples vistas

CodeIgniter manejará inteligentemente múltiples llamadas a $this->load->view desde dentro de un controlador. Si más de una llamada ocurre serán agregados juntos. Por ejemplo, puede querer tener una vista de encabezado, una vista de menu, una vista de contenido, y una vista de píe de página. Eso puede verse más o menos así:

```
<?php
class Pagina extends Controller {
    function index()
    {
        $datos['titulo_pagina'] = 'Su titulo';
        $this->load->view('encabezado');
        $this->load->view('menu');
        $this->load->view('contenido', $datos);
        $this->load->view('pie_de_pagina');
    }
}
?>
```

En el ejemplo anterior, estamos usando "datos agregados dinámicamente", el cual verá debajo.

Guardando Vistas dentro de Sub-carpetas

Sus archivos de vista pueden ser guardados dentro de sub-carpetas si prefiere ese tipo de organización. Cuando lo haga, necesitará incluir el nombre de la carpeta al cargad la vista. Ejemplo:

```
$this->load->view('nombre_carpeta/nombre_archivo');
```

Agregar Datos Dinámicos a la Vista

Los datos son pasados desde el controlador a la vista de la forma de un **arreglo** o un **objeto** en el segundo parámetro de la función de carga de vistas. Aquí hay un ejemplo usando un arreglo:

```
$datos = array(
    'titulo' => 'Mi Titulo',
    'encabezado' => 'Mi Encabezado',
    'mensaje' => 'Mi Mensaje'
        );

$this->load->view('vistablog', $datos);
```

Y aquí hay un ejemplo usando un objeto:

```
$datos = new Someclass();
$this->load->view('vistablog', $datos);
```

Nota: si usa un objeto las variables de clase serán convertidas en un arreglo de elementos.

Probémoslo con su archivo controlador. Ábralo y agregue este código:

```
<?php
class Blog extends Controller {
    function index()
    {
        $data['title'] = "My Real Title";
        $data['heading'] = "My Real Heading";
```


```
        $this->load->view('blogview', $data);
    }
}
?>
```

Ahora puede abrir su archivo de vista y cambiar el texto a variables que correspondan a las claves del arreglo de sus datos:

```
<html>
    <head>
        <title><?php echo $title;?></title>
    </head>
    <body>
        <h1><?php echo $heading;?></h1>
    </body>
</html>
```

Entonces cargue la página en la URL que viene usando y debería ver las variables reemplazadas.

Creando Iteraciones

El arreglo de datos que le pasa a su archivo de vista no está limitado a variables simples. Puede pasar arreglos multidimensionales, los que serán iterados para generar múltiples filas. Por ejemplo, si trajo datos de su base de datos tipicalmente será en la forma de un arreglo múltidimensional.

Aquí hay un ejemplo simple. Agregue esto a su controlador:

```
<?php
class Blog extends Controller {
    function index()
    {
        $data['todo_list'] = array('Clean House', 'Call Mom', 'Run Errands');
        $data['title'] = "My Real Title";
        $data['heading'] = "My Real Heading";
        $this->load->view('blogview', $data);
    }
}
?>
```

Ahora abra su archivo de vista y cree la iteración:

```
<html>
    <head>
        <title><?php echo $title;?></title>
    </head>
    <body>
        <h1><?php echo $heading;?></h1>
        <h3>My Todo List</h3>
        <ul>
            <?php foreach($todo_list as $item):?>
            <li><?php echo $item;?></li>
            <?php endforeach;?>
        </ul>
    </body>
</html>
```

Nota: Notará que en el ejemplo de arriba estamos usando la sintaxis alternativa de PHP. Si no está familiarizado con ella, puede leer acerca de ella en la sección de sintaxis alternativa.

Modelos

Los modelos están disponibles **opcionalmente** para los que deseen utilizar un enfoque más tradicional MVC.

Qué es un modelo?

Los modelos son clases PHP que se han diseñado para trabajar con la información en su base de datos. Por ejemplo, digamos que usted usa CodeIgniter para administrar un blog. Puede que tenga una clase de modelo que contiene funciones para insertar, actualizar y recuperar datos de su blog. Aquí está un ejemplo de lo que podría ser la clase del modelo:

```
class Blogmodel extends Model {
    var $title   = '';
    var $content = '';
    var $date    = '';

    function Blogmodel()
    {
        // Llamando al contructor del Modelo
        parent::Model();
    }
```


```
    function get_last_ten_entries()
    {
        $query = $this->db->get('entries', 10);
        return $query->result();
    }

    function insert_entry()
    {
        $this->title   = $_POST['title'];
        $this->content = $_POST['content'];
        $this->date    = time();

        $this->db->insert('entries', $this);
    }

    function update_entry()
    {
        $this->title   = $_POST['title'];
        $this->content = $_POST['content'];
        $this->date    = time();

        $this->db->update('entries', $this, array('id', $_POST['id']));
    }

}
```

Nota: Las funciones en el ejemplo anterior usan funciones de base de datos Active Record.

> Importante: Con el objetivo de la simplicidad, en este ejemplo estamos usando $_POST directamente. Esto es generalmente una mala práctica pues debería usarse la clase Input (más adelante en este manual) usando: **`this->input->post('title')`**

Anatomia de un Modelo

La clases de Modelo estan almacenadas en su carpeta *application/models/*. Se puede anidar dentro de sub-carpetas si desea esta organización.

El prototipo básico para una clase de Modelo es esta:

```
class Model_name extends Model {

    function Model_name()
    {
        parent::Model();
    }
}
```

Cuando *Model_name* es el nombre de su clase. Los nombre de clase **debe** tener la primera letra en mayúscula con el resto del nombre en minúscula. Asegúrese de que su clase extiende el modelo de base de la clase.

El nombre del archivo será la versión en minúscula de su nombre de clase. Por ejemplo, si su clase es esta:

```
class User_model extends Model {

    function User_model()
    {
        parent::Model();
    }
}
```

su archivo será este:

```
application/models/user_model.php
```

Cargando un Modelo

Sus modelos suelen ser cargados y llamados desde sus funciones de controlador. Para carga un modelo debe usar la siguiente función:

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

Si su modelo esta localizado en una sub-carpeta, incluya la ruta relativa de su carpeta de modelos. Por ejemplo, si tiene un modelo localizado en *application/models/blog/queries.php* cargará usando esto:

```
$this->load->model('blog/queries');
```

Una vez cargado, tendrá que acceder a su funciones de modelo utilizando un objeto con el mismo nombre que su clase:

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

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

Si desea que su modelo este asignado a un nombre de objeto diferente, ud. lo puede especificar por medio del segundo parámetro de la función de carga:

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

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

Este es un ejemplo de un controlador, que carga un modelo, entonces sirve a una vista:

```
class Blog_controller extends Controller {

    function blog()
    {
        $this->load->model('Blog');

        $data['query'] = $this->Blog->get_last_ten_entries();

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

Auto-carga de Modelos

Si usted encuentra que necesita un modelo particular a nivel global a lo largo de su aplicación, le puede decir a CodeIgniter que lo cargue automáticamente durante la inicialización del sistema. Esto se realiza abriendo el archivo application/config/autoload.php y agregando el modelo al arreglo de auto-carga.

Conexión a su base de datos

Cuando se carga un modelo, éste **NO** se conecta automáticamente a su base de datos. Las siguientes opciones de conexión están habilitadas para ud:

- Puede conectarse usando los métodos de base de datos standard descriptos aquí, ya sea dentro de su clase Controlador o su clase Modelo.
- Puede decirle a la función de autocarga del modelo para auto-conectarse pasándole TRUE (boolean) a través del tercer parámetro, y la configuración de conectividad, tal como se define en su archivo de configuración de la base de datos a ser utilizado:

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

- Puede pasar manualmente la configuración de la conectividad de base de datos a través del tercer parámetro:

```
$config['hostname'] = "localhost";
$config['username'] = "myusername";
$config['password'] = "mypassword";
$config['database'] = "mydatabase";
$config['dbdriver'] = "mysql";
$config['dbprefix'] = "";
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

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

Funciones Asistentes

Los asistentes, como el nombre sugiere, le ayudan con tareas. Cada archivo de asistente es simplemente una colección de funciones de una categoría particular. Existe un *Asistente de URL*, que ayuda en la creación de hipervínculos, está el *Asistente de Formularios* que ayuda a crear elementos de formularios, , *Asistente de Texto* que realiza varias rutinas de formato,*Asistente de Cookie* establece y lee cookies, *Asistente de Archivos* ayuda a trabajar con archivos, etc.

A diferencia de muchos otros sistemas en CodeIgniter, los Asistentes no están escritos en un formato Orientado a Objetos. Son simples funciones de procedimiento. Cada función de asistente realiza una tarea específica, sin dependencia en otras funciones.

CodeIgniter no carga Archivos Auxiliares por defecto, así que el primer paso en el uso de Asistentes es cargarlos. Una vez cargados, se vuelven disponibles globalmente en su controlador y vistas.

Los asistentes son típicamente guardados en su directorio *system/helpers*. Alternativamente puede crear una carpeta llamada `helpers` dentro de su carpeta `application` y guardarlos allí. CodeIgniter primero buscará en su direcotrio *system/application/helpers*. Si el directorio no existe o el asistente especificado no está ubicado allí, CI buscará entonces en su carpeta global *system/helpers*.

Cargando un Asistente

Cargar un archivo asistente es bastante simple usando la función siguiente:

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


Dónde *nombre* es el nombre del archivo del asistente, sin la extensión .php o la parte "helper".

Por ejemplo, para cargar el archivo *Asistente de URL*, que es nombrado *url_helper.php*, haría así:

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

Un asistente puede ser cargado en cualquier lugar dentro de su función controlador (o incluso dentro de sus archivos Vista, aunque no es una buena práctica), siempre que lo cargue antes de usarlo. Puede cargar sus asistentes en el constructor de su controlador, para que esté disponible automáticamente en cualquier función, o puede cargarlo en una función específica que lo necesite.

> Nota: La función de carga de Asistente no devuelve un valor, así que no intente asignarselo a una variable. Simplemente úselo como se muestra.

Cargando Múltiples Asistentes

Si necesita cargar más de un asistente puede especificarlos en un arreglo, así:

```
$this->load->helper( array('asistente1', 'asistente2', 'asistente3') );
```

Auto-cargando Asistentes

Si descubre que necesita un asistente particular globalmente a lo largo de su aplicación, puede decirle a CodeIgniter que lo auto-cargue durante la inicialización del sistema. Esto se hace abriendo el archivo *application/config/autoload.php* y agregando el asistente al arreglo autoload.

Usando un Asistente

Una vez que haya cargado el Archivo Asistente conteniendo la función que tiene intención de usar, puede llamarla de la forma estándar de PHP.

Por ejemplo, para crear un hipervínculo usando la función *anchor()* en uno de sus archivos de vista, haría esto:

```
<?=anchor('blog/comentarios', 'Cliquee aquí');?>
```

Donde "Cliquee aquí" es el nombre del hipervínculo, y "blog/comentarios" es la URI al controlador/función al que desee dirigir.

"Extendiendo" Asistentes

Para "extender" Asistentes, creer un archivo en su carpeta *application/helpers/* con nombre idéntico al Asistente existente, pero con préfijo con `MY_` (este item es configurable. Vea abajo.).

Si todo lo que necesita hacer es agregar alguna funcionalidad a un asistente existente - tal vez agregar una función o dos, o cambiar como una función asistente en particular opera - entonces es exagerado reemplazar el asistente entero con su versión. En este caso, es mejor simlpemente "extender" el Asistente. El término "extender" es usado como aproximación ya que las funciones Asistentes son procedimientos discretos y no pueden ser extendidos en el sentido tradicional de programación. Bajo este velo, se da la habilidad para agregar funciones a los Asistentes proveidos, o modificar como una función de Asistente operante.

Por ejemplo, para extender el nativo Asistente de Arreglo creará un archivo llamado *application/helpers/*MY_array_helper.php, y agrega o sobreescribe funciones:

```
// cualquiera_en_arreglo() no es en el Asistente de Arreglo,
// así que define una nueva función
function cualquiera_en_arreglo($aguja, $pajar)
{
    $aguja = (is_array($aguja)) ? $aguja : array($aguja);
    foreach ($aguja as $item)
    {
        if (in_array($item, $pajar))
        {
            return TRUE;
        }
    }
    return FALSE;
}

// random_element() está incluido en el Asistente de Arreglo,
// así que sobreescribe la función nativa

function random_element($arreglo)
{
    shuffle($arreglo);
    return array_pop();
}
```


Estableciendo Su Propio Prefijo

El prefijo del nombre de archivo "extendiendo" Asistentes es el mismo usado para extender librerías y clases de núcleo. Para establecer su propio prefijo, abra su archivo *application/config/config.php* y busque este item:

```
$config['subclass_prefix'] = 'MY_';
```

Por favor note que todas las librerías nativas de CodeIgniter son prefijadas con `CI_` así que NO use esto como prefijo.

Ahora qué?

En la Tabla de Contenidos encontrará una lista de todos los Archivos Asistentes disponibles. Navegue cada uno para ver que hacen.

Plugins

Los plugins funcionan casi idénticamente a los Asistentes. La diferencia principal es que el plugin usualmente provee una sola función, mientras que los Asistentes usualmente son colecciones de funciones. Los asistentes también son considerados una parte del sistema; los plugins intentan ser creados y compartidos por nuestra comunidad.

Los plugins deben ser guardados en su directorio *system/plugins* o puede crear una carpeta llamada `plugins` dentro de su carpeta `application` y guardelos allí. CodeIgniter buscará primero en su directorio *system/application/plugins*. Si el directorio no existe o el plugin especificado no está ubicado allí, CI buscará en su carpeta global *system/plugins*.

Cargando un Plugin

Cargando un archivo plugin es tan simple como usar la siguiente función:

```
$this->load->plugin('nombre');
```

Donde *nombre* es el nombre del archivo del plugin, sin la extensión .php del archivo o la parte "plugin".

Por ejemplo, para cargar el plugin *Captcha*, que es llamado *captcha_pi.php*, hará lo siguiente:

```
$this->load->plugin('captcha');
```

Un plugin puede ser cargado desde cualquier lugar dentro de sus funciones de controlador (o incluso dentro de sus archivos de Vista, aunque eso no es buena práctica), mientras que lo cargue antes de usarlo. Puede cargar sus plugins dentro del constructor del controlador para que esté disponible automáticamente en cualquier función, o puede cargar el plugin en una función específica que lo necesite.

Nota: La función de carga de Plugin de arriba no devuelve una valor, así que no intente asignarselo a una variable. Simplemente úselo como se mostró.

Cargando Múltipls Plugins

Si necesita cargar más de un plugin puede especificarlos como un arreglo, así:

```
$this->load->plugin( array('plugin1', 'plugin2', 'plugin3') );
```

Auto-cargando Plugins

Si encuentra que necesita un plugin en particular a lo largo de toda la aplicación, puede decirle a CodeIgniter que lo auto-cargue durante la inicialización del sistema. Esto se hace abriendo el archivo *application/config/autoload.php* y agregando el plugin al arreglo de autoload.

Usando un Plugin

Una vez cargado el Plugin, puede llamarlo de la misma forma que a una función de PHP estándar.

Usando Librerías de CodeIgniter

Todas las librerías disponibles están ubicadas en su carpeta system/libraries. En la mayoría de los casos, para usar uno de esas clases involucra inicializarla dentro de un controlador usando la siguiente función de inicialización:

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

Donde *nombre de clase* es el nombre de la clase que quiere invocar. Por ejemplo, para cargar la clase de validación haría esto:

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

Una vez inicializada puede usarlo como se indica en la página de la guía del usuario correspondiente a esa clase.

Creando Sus Propias Librerías

Por favor lea la sección de la guía del usuario que discute como crear sus propias librerías

Creación de Librerias

Cuando usamos el termino "Librerias" normalmente nos referimos a las clases que se localizan en el directorio `libraries` y descriptas en la Referencia de Clases de su Guia de Usuario. En este caso, sin embargo, en lugar de ello describiremos como puede crear sus propias librerias dentro del directorio *application/libraries* con el fin de mantener la separación entre sus recursos locales y los recursos del marco de trabajo global.

Como un agregado extra, CodeIgniter permite `extender` sus clases nativas a sus librerias si simplemente necesita agregar alguna funcionalidad a una libreria existente. O puede incluso sustituir a las librerias nativas colocando nombres idénticos de versiones en su carpeta *application/libraries*.

En Resumen:

- Puede crear librerias totalmente nuevas.
- Puede extender librerias nativas.
- Puede reemplazar librerias nativas.

La siguiente página explica estos tres conceptos en detalle.

> **Nota:** Las clases de Base de Datos no pueden ser extendidas o reemplazadas con sus propias clases, ni puede la clase Loader en PHP 4. Todas las otras clases estan habilitadas para ser reemplazadas/extendidas.

Almacenamiento

Su libreria de clases debe almacenarce dentro de su carpeta *application/libraries,* pues es allí donde CodeIgniter las buscará cuando sean inicializadas.

Convenciones de Nombre

- Los nombres de archivos deben ser capitalizados. Por ejemplo: *Myclass.php*
- Las declaraciones de clases deben ser capitalizadas. Por ejemplo: class Myclass
- Los nombre de las clases y los nombres del archivo deben coincidir.

El Archivo de Clase

Las clases deben tener este prototipo básico (Nota: Estamos utilizando el nombre Someclass puramente como un ejemplo):

```
<?php if (!defined('BASEPATH')) exit('No permitir el acceso directo al
script');

class Someclass {
    function some_function()
    {
    }
}
?>
```


Usando su Clase

Desde cualquiera de sus funciones de Controller puede inicializar su clases utilizando el estándar:

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

Donde *someclass* es el nombre del archivo, sin la extensión ".php" del archivo. Puede enviar el nombre del archivo en mayúscula o minúscula. A CodeIgniter no le importa.

Una vez cargado puede acceder a su clase utilizando la versión en `minúsculas`:

```
$this->someclass->some_function();
// Las instancias de objetos serán siempre en minúsculas
```

Pasando Parámetros Cuando Inicializa Su Clase

En la función que carga la libreria puede pasar dinámicamente datos a travéz del segundo parámetro y ellos serán pasados a su constructor de clase:

```
$params = array('type' => 'large', 'color' => 'red');

$this->load->library('Someclass', $params);
```

Si utiliza esta función debe configurar su constructor de la clase para esperar los datos:

```
<?php if (!defined('BASEPATH')) exit('No permitir el acceso directo al
script');

class Someclass {

    function Someclass($params)
    {
        // Hacer algo con $params
    }
}

?>
```

También puede pasar parámetros almacenados en un archivo de configuración. Simplemente cree un archivo de configuración que se llame igual al `nombre del`

archivo de la clase y almacénelo en su carpeta *application/config/*. Tenga en cuenta que si pasa el dinámicamente los parámetros descritos anteriormente, la opción en el archivo de configuración no estará disponible.

Utilizando los Recursos de CodeIgniter dentro de su Libreria

Para acceder a los recursos nativos de CodeIgniter dentro de su libreria use la función `get_instance()`. Esta función retorna el objeto super de CodeIgniter.

Normalmente desde sus funciones del controlador llamará a cualquiera de las funciones habilitadas en CodeIgniter usando el constructor `$this`:

```
$this->load->helper('url');
$this->load->library('session');
$this->config->item('base_url');
etc.
```

`$this`, sin embargo, sólo trabaja directamente dentro de sus controladores, sus modelos, o sus vistas. Si desea utilizar las clases de CodeIgniter dentro de sus propias clases puede hacerlo de la siguiente manera:

Primero, asigne el objeto de CodeIgniter a una variable:

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

Una vez que se ha asignado el objeto a una variable, va a utilizar esa variable *en lugar* de `$this`:

```
$CI =& get_instance();

$CI->load->helper('url');
$CI->load->library('session');
$CI->config->item('base_url');
etc.
```


Nota: Se dará cuenta de que la anterior función get_instance() esta siendo pasada por referencia:

$CI =& get_instance();

Esto es muy importante. La asignación por referencia le permite utilizar el objeto original de CodeIgniter en lugar de crear una copia del mismo.

Además, tenga en cuenta: Si está corriendo PHP 4 es usualmente mejor evitar que se hagan llamadas a *get_instance()* dentro de sus constructores de clase. PHP 4 tiene problemas de referencias al objeto super CI dentro de los constructores de la aplicación ya que los objetos no existen hasta que la clase es totalmente instanciada.

Reemplazando Librerias Nativas con Sus Versiones

Simplemente por asignar un nombre a sus archivos de clase idéntico a una libreria nativa causará que CodeIgniter los utilice en lugar de los nativos. Para usar esta característica debe nombrar el archivo y la declaración de la clase exactamente como la libreria nativa. Por ejemplo, para reemplazar la libreria nativa `Email` deberá crear un archivo llamado *application/libraries/Email.php*, y declarar su clase con:

```
class CI_Email {

}
```

Tenga en cuenta que la mayoría de las clases son con prefijo **`CI_`**.

Para cargar su libreria puede ver la función de carga estándar:

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

Nota: En este momento las clases de base de datos no puede ser reemplazadas con sus propias versiones.

Extendiendo las Librerias Nativas

Si todo lo que necesita hacer es añadir alguna funcionalidad a una libreria existente - quizás añadir una función o dos - entonces es excesivo sustituir toda la libreria con su

versión. En este caso es mejor simplemente extender la clase. La extensión de una clase es casi igual que la sustitución de una clase con un par de excepciones:

- La declaración de la clase debe extender la clase padre.
- Su nuevo nombre de la clase y el nombre del archivo se debe prefijar con MY_ (Este tema es configurable. Véase a continuación.).

Por ejemplo, para extender la clase nativa `Email` deberá crear un archivo llamado *application/libraries/*`MY_Email.php`, y declarar su clase con:

```
class MY_Email extends CI_Email {

}
```

Nota: Si necesita utilizar un constructor en su clase, asegúrese de extender el padre constructor:

```
class MY_Email extends CI_Email {

    function My_Email()
    {
        parent::CI_Email();
    }
}
```

Cargando Su Sub-Clase

Para cargar su sub-clase debe usar la sintaxis estándar que normalmente se utiliza. NO incluya su prefijo. Por ejemplo, para cargar el ejemplo anterior, que extiende la clase Email, se utiliza:

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

Una vez cargado se utiliza la variable de clase como siempre lo ha hecho para la clase de la que se extiende. En el caso de la clase Email utilizaremos todas las llamadas:

```
$this->email->some_function();
```


Estableciendo Su Propio Prefijo

Para establecer su propio prefijo de sub-clase, abra su *application/config/config.php* y busque este tema:

```
$config['subclass_prefix'] = 'MY_';
```

Tenga en cuenta que todas las librerias nativas de CodeIgniter están prefijadas con `CI_` NO USE esto como su prefijo.

Creación de Clases de Sistema de Núcleo

Cada vez que CodeIgniter corre hay varias clases base que son inicializados automáticamente como parte del núcleo del entorno de trabajo. Es posible, sin embargo, intercambiar cualquiera de las clases de sistema de núcleo con sus propias versiones o incluso extender las versiones de núcleo.

La mayoría de los usuarios nunca necesitará hacer nada de esto, pero la opción de reemplazar o extenderla existe para quienes qioerem significantemente alterar el núcleo de CodeIgniter.

> **Nota:** Meterse con una clase de sistema de núcleo tiene varias implicaciones, así que esté seguro que sabe lo que está haciendo antes de intentarlo.

Lista de Clases de Sistema

La siguiente es una lista de archivos de sistema de núcleo que son invocados cada vez que CodeIgniter corre:

- Benchmark
- Config
- Controller
- Exceptions
- Hooks
- Input

- Language
- Loader
- Log
- Output
- Router
- URI

Reemplazar las Clases de Núcelo

Para usar una de sus propias clases de sistema en vez de una de las predeterminadas simplemente ubique su versión dentro de su directorio local *application/libraries*:

```
application/libraries/alguna-clase.php
```

Si el directorio no existe, puedes crearlo.

Cualquier archivo llamado idénticamente a uno de la lista de arriba será usado en vez del normalmente usado

Por favor note que su clase debe usar `CI` como prefijo. Por ejemplo, si su archivo es llamado`Input.php` la clase será llamada:

```
class CI_Input {

}
```

Extendiendo las Clases de Núcleo

Si todo lo que necesita hacer es agregar alguna funcionalidad a una librería existente - quizás agregar una función o dos - entonces es exagerado reemplazar la librería entera con su versión. En este caso es mejor simplemente extender la clase. Extender la clase es casi idéntico a reemplazar la clase con un par de exceptiones:

- La declaración de clase debe extender la clase padre.
- Su nuevo nombre de clase y de archivo debe ser prefijado con MY_ (este item es configurable. Vea abajo.).

Por ejemplo, para extender la clase nativa `Input` creará un archivo llamado dfn>application/libraries/`MY_Input.php`, y declara su clase con:

```
class MY_Input extends CI_Input {

}
```

Nota: Si necesita usar un constructor en su clase, asegurese que extiende al constructor del padre:

```
class MY_Input extends CI_Input {

    function My_Input()
    {
        parent::CI_Input();
    }
}
```

Consejo: Cualquier función en su clase que es nombrada idénticamente a las funciones en la clase padre serán usadas en vez de las nativas (esto es conocido como "sobreescritura de método"). Esto le permite alterar sustancialmente el núcleo de CodeIgniter.

Estableciendo Su Propio Prefijo

Para establecer su propio prefijo de sub-clase, abra su archivo *application/config/config.php* y busque este item:

```
$config['subclass_prefix'] = 'MY_';
```

Por favor note que todas las librerías nativas de CodeIgniter están prefijadas con `CI_` para NO usar ese prefijo.

Ganchos (hooks)
Extendiendo el Núcleo del Entorno de Trabajo

Los Ganchos de CodeIgniter proveen un funcionamiento para meterse y modificar el funcionamiento interno del entorno de trabajo sin "hackear" los archivo del núcleo. Donde CodeIgniter corre sigue un proceso específico de ejecución, diagramado en la página Flujo de Aplicación. Hay muchas instancias, sin embargo, donde querría causar que alguna acción tome lugar en un escenario particular en el proceso de ejecución. Por ejemplo, puede querer correr un script justo antes que su controlador sea cargado, o justo después, o puede querer activar uno de sus scripts en otra ubicación.

Habilitando Ganchos

La característica de ganchos puede ser habilitada/deshabilitada estableciendo el siguiente item en su archivo `application/config/config.php`:

```
$config['enable_hooks'] = TRUE;
```

Definiendo un Gancho

Los ganchos son definidos en el archivo *application/config/hooks.php*. Cada gancho es especficiada como un arreglo con este prototipo:

```
$hook['pre_controller'] = array(
                'class'=> 'MiClase',
                'function' => 'MiFuncion',
                'filename' => 'Miclase.php',
                'filepath' => 'hooks',
                'params'   => array('cerveza', 'vino', 'snacks')
                );
```

Notas:
El indice del arreglo correlata el nombre del gancho en particular que quiere usar. En el ejemplo anterior el gancho apuntado es `pre_controller`. Una lista de puntos de ganchos es encontrada debajo. Los siguientes items pueden ser definidos en su arreglo asociativo de gancho:

- **class** El nombre de la clase que desea invocar. Si prefiere usar una función de procedimiento en vez de una clase, debe este ítem en blanco.

- **function** El nombre de la función que desea llamar.
- **filename** El nombre del archivo que contiene su clase/función.
- **filepath** El nombre del directorio que contiene su script. Nota: Su script debe estar ubicado en un directorio DENTRO de la carpeta application, así que la ruta al archivo es relativa a esa carpeta. Por ejemplo, si su script está ubicado en *application/hooks*, simplemente usará hooks como su ruta al archivo. Si su script es ubicado en *application/hooks/utilities* usará hooks/utilities como su ruta al archivo. Sin barra al final.
- **params** Cualquier parámetro que desee pasarle a su script. Este ítem es opcional.

Múltiples Llamadas a el Gancho

Si quiere usar el mismo gancho apuntado con más de un script, simplemente haga la declaración de su arreglo multi-dimensional, así:

```
$hook['pre_controller'][] = array(
                'class'=> 'MiClase',
                'function' => 'Mifuncion',
                'filename' => 'Miclase.php',
                'filepath' => 'hooks',
                'params'   => array('cerveza', 'vino', 'snacks')
                );

$hook['pre_controller'][] = array(
                'class'=> 'MiOtraClase',
                'function' => 'Miotrafuncion',
                'filename' => 'Miotraclase.php',
                'filepath' => 'hooks',
                'params'   => array('rojo', 'amarillo', 'azul')
                );
```

Note los corchetes después de cada índice del arreglo:

```
$hook['pre_controller'][]
```

Esto le permite que el mismo gancho apunte a múltiples scripts. El orden que le define a su arreglo será el orden de ejecución.

Puntos de Gancho

La siguiente es una lista de los puntos de gancho disponibles.

- **pre_system**
  Llamado muy al principio durante la ejecución de sistema. Sólo las clases de

punto de referencia (benchmark) y ganchos han sido cargadas en este punto. Ningún proceso de ruteo u otro ha ocurrido.

- **pre_controller**
  Llamado inmediatamente antes a cualquiera de que los controladores sean llamados. Todas las clases, ruteo y verificaciones de seguridad han sucedido.
- **post_controller_constructor**
  Llamado inmediatamente después de que su controlador sea instanciado, pero antes de que ocurra cualquier llamada a un método.
- **post_controller**
  Llamado inmediatamente después de que su controlador esté completamente ejecutado.
- **display_override**
  Sobreescribe la función *_display()*, usada para mandar la página finalizada al explorador web al final de la ejecución del sistema. Esto le permite usar su propia metodología de visualización. Note que los datos finalizados estarán disponibles llamando a *$this->output->get_output()*
- **cache_override**
  Habilita el llamar a su propia función en vez de la función *_display_cache()* en la clase de salida. Esto le permite usar su propio mecanismo para mostrar el cache.

  scaffolding_override
  Permite que una petición de scaffolding dispare su propio script en su lugar.

  post_system
  Llamado después de que la presentación final de la página es enviada al explorador, al final de la ejecución del sistema después de que los datos finalizados son enviados al explorador.

Auto-Carga de Recursos

CodeIgniter viene con una característica de "Auto-carga" que permite a librerías, asistentes(helpers), y complementos ser inicializados automáticamente cada vez que el sistema arranque. Si necesita ciertos recursos a nivel global a lo largo de su aplicación, debe considerar la posibilidad de Auto-carga para su conveniencia.

Los siguientes elementos pueden ser cargados automáticamente:

- Clases Core encontradas en la carpeta "libraries"
- Asistentes (Helpers) encontrados en la carpeta "helpers"

- Complementos encontrados en la carpeta "plugins"
- Archivos de configuración encontrados en la carpeta "config"
- Archivos de lenguaje encontrados en la carpeta "system/language"
- Modelos encontrados en la carpeta "models"

Para autocargar recursos, abra el archivo *application/config/autoload.php* y agregue el elemento que desea cargar al arreglo `autoload`. Encontrará las instrucciones en el archivo correspondiente a cada uno de los tipos de elementos.

Nota: No incluya la extensión del archivo (.php) cuando agregue elementos al arreglo de autocargado.

Scaffolding

La característica Scaffolding de CodeIgniter provee una forma rápida y muy conveniente de agregar, editar o borrar información de su base de datos durante el desarrollo.

Muy Importante: Scaffolding tiene intenciones de uso sólo para desarrollo. Provee poca seguridad además de una palabra "secreta", así que cualquiera que tiene acceso a su sitio en CodeIgniter puede potencialmente editar o borrar su información. Si usa scaffolding asegurece de deshabilitarlo inmediatamente después de que lo haya uasdo. NO lo dehe habilitado en un sitio en vivo. Y por favor, establezca una palabra secreta antes de usarlo.

Por qué alguien usaría scaffolding?

Aquí hay un escenario típico: Crea una nueva table de base de datos durante el desarrollo y quiere una forma rápida de insertar algo de datos en ella con la que trabajar. Sin scaffolding sus opciones son escribir un insert usando una línea de comando, o usar una herramienta de manejo de base de datos como phpMyAdmin. Con la característica de scaffolding de CodeIgniter puede rápidamente agrgar algo de datos usando la interface de su explorador. Y cuando esté en medio del uso de datos, puede borrarlo fácilmente.

Establecer una Palabra Secreta

Antes de habilitar scaffolding por favor tome un momento para establecer una palabra secreta. Esta palabra, cuando se encuentre en su URL, lanzará la interface de scaffolding, así que por favor elija algo oscuro que nadie adivine.

Para establecer una palabra secreta, abra su archivo `application/config/routes.php` y busque este ítem:

```
$route['scaffolding_trigger'] = '';
```

Una vez que lo haya encontrado, agréguele su única y propia palabra.

> **Nota:** La palabra scaffolding **no** puede empezar con guión bajo.

Habilitando Scaffolding

Nota: La información en esta página asume que ya sabe como funcionan los controladores, y que tiene uno disponible trabajando. También asume que ha configurado CodeIgniter para autoconectar a su base de datos. Si no, la información de aquí no le será muy relevante, así que es alentado a if por esas secciones primero. Por último, asume que entienda que es un constructor de clase. Si no, lea la última sección de la página de controladores.

Para habilitar scaffolding lo inicializará en su constructor así:

```
<?php
class Blog extends Controller {

    function Blog()
    {
    parent::Controller();

    $this->load->scaffolding('nombre_tabla');
    }
}
?>
```

Donde `nombre_tabla` es el nombre de la tabl (tabla, not base de datos) con la que desea trabajar.

Una vez que haya inicializado scaffolding, accederá a él con este prototipo de URL:

```
www.su-sitio.com/index.php/clase/palabra_secreta/
```

Por ejemplo, usando un controlador llamado *Blog*, y *abracadabra* como la palabra secreta, puede acceder a scaffolding así:

```
www.su-sitio.com/index.php/blog/abracadabra/
```

La interface scaffolding debe ser auto.explicatoria. Puede agregar, editar o borrar registros.

Una Nota Final:

La característica scaffolding sólo trabajará con tablas que contengan una clave primera, ya que esta información es necesaria pare realizar varias funciones de base de datos.

Ruteo URI

Tipicamente hay una relacion uno-a-uno entre una cadena URL y su correspondiente controlador clase/metodo. Los segmentos en una URI normalmente siguen este patron:

```
www.your-site.com/class/function/id/
```

En algunas instancias, sin embargo, querras remapear esta relación para que una clase/funcion diferente pueda ser llamada en lugar de la correspondiente a la URL.

Por ejemplo, digamos que quieres que tus URLs tengan este prototipo:

```
www.tu-sitio.com/product/1/
www.tu-sitio.com/product/2/
www.tu-sitio.com/product/3/
www.tu-sitio.com/product/4/
```

Normalmente el segundo segmento de la URL es reservada para el nombre de funcion, pero en este ejemplo en lugar de eso tiene el ID del producto. Para superar esto, CodeIgniter te permite remapear el manejador URI.

Creando tus propias reglas de ruteo

Las reglas de ruteo son definidos dentro de tu archivo *application/config/routes.php*. Dentro de el deberias poder ver un arreglo llamado *$route* que te permite especificar tus criterios de ruteo. Las rutas podrian ser especificadas utilizando *comodines* o *Expresiones Regulares*

Comodines

Un tipico comodidin de ruteo deberia parecerse a esto:

$route['product/:num'] = "catalog/product_lookup";

En una ruta, la llave (key) del arreglo contiene la URI que debe coincidir, mientras que el valor del arreglo contiene el destino a donde debe ser re-ruteado. En el ejemplo anterior, si el literalmente la palabra "product" es encontrada en el primer segmento de la URL, y el numero es encontrado en el segundo segmento, la clase "catalog" y el metodo "product_lookup" seran usados.

Puedes hacer que coincidan los valores literales, o puedes usar dos tipos de comodines:

:num
:any

:num coincidira con un segmento que unicamente contiene numeros.
:any coincidira con un segmento que contenga cualquier caracter.

> **Nota:** Las Rutas correran en el orden que sean definidas. Rutas superiores siempre tomaran precedencia sobre las que tienen menos precedencia.

Ejemplos

He aqui algunos ejemplos de ruteo:

```
$route['journals'] = "blogs";
```

una URL conteniendo la palabra "journals" en el primer segmento sera remapeada a la clase "blogs".

```
$route['blog/joe'] = "blogs/users/34";
```

una URL conteniendo el segmento blog/joe sera remapead a la clase "blogs" y el metodo "users". El ID sera seteado a "34".

```
$route['product/:any'] = "catalog/product_lookup";
```

una URL con "product" como el primer segmento, y cualquier cosa en el segundo sera remapeado a la clase "catalog" y el metodo "product_lookup".

Importante: No use barras al comienzo/final.

Expresiones Regulares

Si lo prefieres, puedes utilizar expresiones regulares para definir tus reglas de ruteo. Cualquier expresion regular es permitida, como lo son las back-references.

Note: Si usa back-references debes usar la sintaxis de dolares antes que la sintaxis de doble barra diagonal.

Una tipica Expresion Regular deberia parecerse a esto:

$route['products/([a-z]+)/(\d+)'] = "$1/id_$2";

En el ejemplo anterior, una URI similar a *products/shirts/123* llamara en su lugar a la clase controlador *shirts* y la funcion *id_123*.

Tambien puedes mesclar comodines con expresiones regulares.

Rutas Reservadas

Hay dos rutas reservadas:

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

Esta ruta indica cual clase controlador debera ser cargado si la URI no contiene datos, caso que ocurrira cuando la gente cargue la URL raiz. En el ejemplo anterior, la clase

"welcome" seria la que se cargue. Te animamos a que siempre tengas una ruta por defecto, caso contrario una pagina 404 apareceria por defecto.

```
$route['scaffolding_trigger'] = 'scaffolding';
```

Esta ruta te permite colocar un palabra secreta, la cual debe estar presente en la URL, disparando la caracteristica scaffolding. Por favor lea la pagina Scaffolding para mas detalles.

Important: Las rutas reservadas deben aparecer antes que cualquier comodin o expresion regular de ruteo.

Manejo de Errores

CodeIgniter le permite crear un reporte de errores en sus aplicaciones usando las funciones descritas abajo. Además, tiene una clase de historial de errores que le permite que los mensajes de error y depuración sean guardados en archivos de texto.

> **Nota:** Por defecto, CodeIgniter muestra todos los errores PHP. Puede querer cambiar este comportamiento una vez que su desarrollo esté completo. Encontrará la función *error_reporting()* ubicada al principio de su archivo index.php principal. Deshabilitar el reporte de errores NO prevendrá que el archivo de historial sea escrito si hay errores.

A diferencia de la mayoría de los sistemas en CodeIgniter, las funciones de error son interfaces de simple procedimiento que están disponibles globalmente a lo largo de la aplicación. Este aproximamiento le permite que los mensajes de error sean activados sin tener que preocuparse de el ámbito de la clase/función.

La siguientes funciones le permiten generar errores:

show_error('*mensaje*')

Esta función mostrará el mensaje de error suministrado usando la siguiente plantilla de error:

`application/errors/error_general.php`

show_404('*pagina*')

Esta función mostrará el mensaje de error 404 suministrado usando la siguiente plantilla de error:

```
application/errors/error_404.php
```

La función espera que la cadena pasada sea la ruta del archivo a la página que no se encontró. Note que CodeIgniter automáticamente mostrará mensajes 404 si los controladores no serán encontrados.

log_message('*nivel*', 'mensaje')

Esta función le permitirá escribir mensajes de error en sus archivos de historial. Debe suministrar uno de tres "niveles" en el primer parámetro, indicando que tipo de mensaje es (depuración, error, info), con el mensaje mismo en el segundo parámetro. Ejemplo:

```
if ($alguna_variable == "")
{
    log_message('error', 'Alguna variable no contenía un valor.');
}
else
{
    log_message('debug', 'Alguna variable era correctamente establecida');
}

log_message('info', 'El proposito de alguna variabl es proveer algún valor.');
```

Hay tres tipos de mensajes:

1. **Mensajes de Error**. Estos son mensajes de error, tal como errores de PHP o del usuario.
2. **Mensajes de Depuración**. Estos son mensajes que le asisten al depurar. Por ejemplo, si una clase ha sido inicializadam podría guardar en el historial esta información de depuración.
3. **Mensajes Informacionales**. Estos son los mensajes de menor prioridad, simplemente dan información acerca de algún proceso. CodeIgniter no genera nativamente ningún mensaje de informaciójn pero puede quererlo en su aplicación.

Nota: Para que el archivo de historial sea realmente escriturable, la carpeta "logs" debe ser escriturable. Además debe establecer el "threshold" del historial. Puede, por ejemplo, sólo querer que los mensajes de errore sean guardados, y no los otros dos tipos. Si lo establece a cero el historial será deshabilitado.

Almacenamiento en Cache de Páginas Webs

CodeIgniter le permite hacer caché de sus páginas con el fin de lograr el máximo rendimiento.

Aunque CodeIgniter es bastante rápido, la cantidad de información dinámica que se muestren en sus páginas se correlaciona directamente a los recursos del servidor, la memoria y los ciclos de procesamiento utilizados, que afectan a su velocidad de carga de páginas. Por cachear sus páginas, ya que se guardan en su estado plenamente renderizadas, puede alcanzar el rendimiento que se acerca a la de las páginas web estáticas.

Cómo Funciona el Trabajo de Almacenar en Caché ?

Se puede habilitar el almacenamiento en caché para cada página, y puede establecer el tiempo que debe permanecer una página en caché antes de ser refrescada. Cuando una página se carga por primera vez, el archivo de caché se escribirá en su carpeta *system/cache*. En posteriores cargas de la página el archivo de caché se recibirá y se enviará a la solicitud del navegador del usuario. Si ha caducado, será eliminado y actualizado antes de ser enviado al navegador.

Nota: La etiqueta Benchmark no fue cacheada para que pueda ver la velocidad de carga de páginas cuando está permitido el almacenamiento en caché.

Habilitar el Almacenamiento en Caché

Para habilitar el almacenamiento en caché, poner la siguiente etiqueta en cualquiera de sus funciones de controlador:

$this->output->cache(*n*);

Donde *n* es el número de **minutos** que desea que la página permanezca en caché entre refrescos.

La etiqueta puede ir a cualquier parte dentro de una función. No se ve afectada por la orden en la que aparece, de modo que la puede poner en el lugar donde le parezca mas lógico para usted. Una vez que la etiqueta esté en su lugar, sus páginas comenzarán a ser cacheadas.

Advertencia: Debido a la forma en CodeIgniter almacena el contenido de la producción, el almacenamiento en caché sólo funcionará si está generando vistas para su controlador con una vista.

Nota: Antes de que los archivos de caché puedan ser escritos, debe configurar los permisos de archivo en su carpeta *system/cache* de tal modo que se pueda grabar.

Borrar Caches

Si ya no desea un archivo de caché, puede quitar la etiqueta de cacheo y ya no será refrescado cuando expire. Nota: La eliminación de la etiqueta no implica la supresión de la memoria caché inmediatamente. Tendrá que expirar normalmente. Si quiere eliminar lo anterior tendrá que eliminarla manualmente de la carpeta de caché.

Perfilando Su Aplicación

La Clase de Perfil mostrará resultados de tiempos de referencia, consultas que ha corrido, y datos $_POST en el final de sus páginas. Esta información puede ser útil durante el desarrollo para ayudar a depurar y optimizar.

Inicializando la Clase

Importante: Esta clase NO necesita ser inicializada. Es cargada automáticamente por la Clase de Salida si el Perfil está habilitado como se muestra debajo.

Habilitando el Perfil

Para habilitar el perfil ubique la siguiente función en cualquier lugar dentro de sus funciones de Controlador:

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

Cuando esté habilitado, un reporte se generará e insertará en el final de sus páginas.

Para deshabilitar el perfil use:

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

Estableciendo tiempos de referencia

Para que el Perfile compile y muestra sus datos de puntos de referencia debe nombrar sus puntos de marca usando una sintaxis específica. Por favor lea la información acerca de establecer puntos de referencia en la página de Clase de Referencia (benchmark).

Manejando sus Aplicaciones

Por defecto es asumido que sólo intentará usar CodeIgniter para manejar una aplicación, el cual construirá en su directorio *system/application/*. Es posible, sin embargo, tener múltiples juegos de aplicaciones que compartan una sola instalación de CodeIgniter, o incluso renombrar o reubicar su carpeta *application*.

Renombrando la Carpeta Application

Si quiere renombrar su carpeta *application* puede hacerlo mientras que abra su archivo principal `index.php` y establezca el nombre usando la variable `$application_folder`:

```
$application_folder = "application";
```

Reubicando la Carpeta Aplicación

Es posible mover su carpeta *application* a una diferente ubicación en su servidor de la carpeta `system`. Para hacerlo abra su archivo principal `index.php` y establezca una *dirección absoluta en el servidor* en la variable `$application_folder`.

```
$application_folder = "/Ruta/a/su/aplicacion";
```

Corriendo Múltiples Aplicaciones con una Instalación de CodeIgniter

Si quiere compartir una instalación común de CodeIgniter para manejar diferentes aplicaciones simplemente ponga todos los directorios ubicados en su carpeta `application` en su propia sub-carpeta.

Por ejemplo, digamos que quiere crear dos aplicaciones, "foo" y "bar". Estructurará su carpeta de aplicación así:

```
system/application/foo/
system/application/foo/config/
system/application/foo/controllers/
system/application/foo/errors/
system/application/foo/libraries/
system/application/foo/models/
system/application/foo/views/
system/application/bar/
system/application/bar/config/
system/application/bar/controllers/
system/application/bar/errors/
system/application/bar/libraries/
system/application/bar/models/
system/application/bar/views/
```

Para seleccionar una aplicación en particular para usar requiere que abra si archivo principal y `index.php` establezca la variable *$application_folder*. Por ejemplo, para seleccionar la aplicación "foo" usará esto:

```
$application_folder = "application/foo";
```

Nota: Cada una de sus aplicaciones necesitará su propio archivo *index.php* el cual llama a la aplicación deseada. El archivo index.php puede ser nombrado como desee.

Sintaxis Alternativa PHP para Archivos de Vista

En caso de no utilizar el motor de plantilla de CodeIgniter, estará usando PHP puro en su archivos de Vista. Para minimizar el código PHP en estos archivos, y para que sea más fácil identificar los bloques de código se recomienda que utilice sintáxis alternativa PHP para las estructuras de control y etiquetas cortas para las declaraciones "echo". Si no está familiarizado con esta sintaxis, ésta le permite eliminar las llaves de su código, y eliminar las declaraciones "echo".

Soporte Automático para Etiquetas Cortas

Nota: Si encuentra que la sintáxis que se describe en esta página no funciona en su servidor podría ser que las "etiquetas cortas" esten desactivadas en su archivo ini de PHP. CodeIgniter, opcionalmente, reescribirá etiquetas cortas sobre la marcha, lo cual le permite utilizar la sintáxis, incluso si su servidor no lo soporta. Esta característica puede ser habilitada en su archivo *config/config.php*.

> Tenga en cuenta que si hace uso de esta función, si encuentra errores de PHP en sus **archivos de vista**, el mensaje de error y el número de línea no se muestran correctamente. En lugar de ello, todos los errores que se mostrará como errores eval().

Alternativas Echos

Normalmente para "echo", o imprimir una variable se haría esto:

```
<?php echo $variable; ?>
```

Con la sintáxis alternativa lo puede hacer de esta manera:

```
<?=$variable?>
```

Estructuras de Control Alternativas

Las estructuras de control, como *if*, *for*, *foreach*, y *while* pueden ser escritas en un formato más simple también. Aquí tiene un ejemplo de uso de foreach:

```
<ul>
    <?php foreach($todo as $item): ?>
    <li><?=$item?></li>
    <?php endforeach; ?>
</ul>
```

Note que no hay llaves. En lugar de ello, la llave que cierra es reemplazada con un *endforeach*. Cada una de las estructuras de control mencionadas anteriormente tiene una sintaxis similar de cierre: *endif*, *endfor*, *endforeach*, y *endwhile*

Observe también que en lugar de utilizar un punto y coma después de cada estructura (excepto la última), hay dos puntos. Esto es importante!

Aquí hay otro ejemplo, usando if/elseif/else. Nótese los dos puntos:

```
<?php if ($username == 'sally'): ?>
    <h3>Hola Sally</h3>
<?php elseif ($username == 'joe'): ?>
    <h3>Hola Joe</h3>
<?php else: ?>
    <h3>Hola usuario desconocido</h3>
<?php endif; ?>
```

Seguridad

Esta página describe algunas "buenas prácticas" acerca de segurdad web, y detalles de características de seguridad interna de CodeIgniter.

Seguridad en URI

CodeIgniter es justamente restrictivo sobre que caracteres permitir en las cadenas URI para ayudar a minimizar la posibilidad de que datos maliciosos puedan ser pasados a su aplicación. Las URIs sólo pueden contener lo siguiente:

- Texto alfanumérico
- "Tilde": ~
- Punto: .
- Dos puntos: :
- Guion bajo: _
- Guion: -

Datos GET, POST, y COOKIE

Los datos GET son simplemente anulados por CodeIgniter ya que el sistema utiliza segmentos URI en vez de las tradicionales query strings de URL (a menos que la opción query string esté habilitada en su archivo config). El arreglo global GET es **destruido** por la clase de Entrada (input) durante la inicialización del sistema.

Register_globals

Durante la inicialización del sistema, todas las variables globales son destruidas, excepto aquellas encontradas en los arreglos $_POST y $_COOKIE. La rutina de eliminación es efectivamente lo misma que register_globals = off.

magic_quotes_runtime

La directiva magic_quotes_runtime es apagada durante la inicialización del sistema para que no tenga que remover las barras cuando se recuperen datos de la base de datos.

Buenas prácticas

Antes de aceptar cualquier dato en su aplicación, ya sean datos POST desde el envío de un formulario, datos COOKIE, datos URI, datos XML-RPC, o incluso datos desde el arreglo SERVER, está alentado a practicar estos tres pasos de acercamiento:

1. Filtrar los datos como si fueran contaminados.
2. Validar los datos para asegurarse que conforman el tipo correcto, largo, tamaño, etc. (a veces, este paso puede reemplazar al paso uno)
3. Escapar los datos entes de enviarlo a su base de datos.

CodeIgniter provee las siguientes funciones para asistirlo en este proceso

Filtro de XSS

CodeIgniter viene con un filtro de XSS (Cross Site Scripting). Este filtro busca técnicas comunmente usadas para embeber Javascript malicioso a sus datos, u otro tipo de código que intente "secuestrar" (hijack) cookies o hacer otra cosa maliciosa. El Filtro XSS es descripto aquí.

Validar los datos

CodeIgniter tiene una Clase de Validación que le asiste en la validación, filtro y preparación de datos.

Escapar todos los datos antes de insertarlo a la base de datos

Nunca inserte información a su base de datos sin escaparla. Por favor vea la sección que trata consultas para más información.

Esta página está en blanco de forma intencional

Referencia de Clases

Esta página está en blanco de forma intencional

Clase de Puntos de Referencia (Benchmarking)

CodeIgniter tiene una clase de punto de referencia que está siempre activa, permitiendo que se calcule la diferencia de tiempo entre dos puntos que sean marcados.

Nota: Esta clase es inicializada automáticamente por el sistema, por esto, no hay necesidad de hacerlo manualmente.

Adicionalmente, el punto de referencia siempre es iniciado cuando el framework es invocado, y es finalizado por la clase de salida(ouput) justo antes de que se envie la vista final al navegador, permitiendo que sea muy precisa la medicion del tiempo de ejecucion de todo el sistema para que sea mostrado.

Usando la clase Puntos de referencia

La clase Puntos de referencia puede ser usado con tus controladores, vistas, o tus Modelos. El proceso para usarlo es este:

1. Marcando un punto inicial
2. Marcando un punto final
3. Ejecuta la funcion "elapsed time"(tiempo transcurrido) para ver los resultados

Aqui un ejemplo utilizando codigo real:

```
$this->benchmark->mark('code_start');
    // Algún codigo por aquí
$this->benchmark->mark('code_end');
echo $this->benchmark->elapsed_time('code_start', 'code_end');
```

Nota: Las Las palabras "code_start" y "code_end" son arbitrarios. Son solamente palabras que se usan para setear dos marcas. Puedes usar cual cualquier palabra que quieras, y puedes poner multiples juegos de marcas. Considere este ejemplo:

```
$this->benchmark->mark('perro');
    // Algun codigo aqui
$this->benchmark->mark('gato');
    // Más codigo aqui
$this->benchmark->mark('pajaro');
```

```
echo $this->benchmark->elapsed_time('perro', 'gato');
echo $this->benchmark->elapsed_time('gato', 'pajaro');
echo $this->benchmark->elapsed_time('perro', 'pajaro');
```

Perfilando tus Puntos de Referencia

Si quieres hacer disponible los datos de tus puntos de referencia disponible para tu Profiler todos los puntos que marques deben ser seteado en pares, y cada punto de marca debe finalizar con _start y _end. Cada par de puntos deben ser llamados de igual forma. Por ejemplo:

```
$this->benchmark->mark('mi_marca_start');
    // Some code happens here...
$this->benchmark->mark('mi_marca_end');
$this->benchmark->mark('otra_marca_start');
    // Agun codigo por aqui...
$this->benchmark->mark('otra_marca_end');
```

Por favor lea la Pagina Profiler para mas informacion.

Desplegando el tiempo total de ejecucion

Si quieres desplegar el tiempo total transcurrido desde el momento en el que CodeIgniter comienza hasta el momento en el que la salida es enviada al navegador, simplemente ubica esto en una de las vistas de tu plantilla:

```
<?=$this->benchmark->elapsed_time();?>
```

Ud. se dara cuenta de que es la misma funcion usada en los ejemplos anteriores, para calcular el tiempo entre dos puntos, excepto que **no** estas usando ningun parametro. Cuando el parametro esta ausente, CodeIgniter no detiene el punto de referencia ahsta justo antes de que la salida final sea enviada al navegador. No importa donde se use la llamada a la funcion, el temporizador continuara corriedon hasta el final.

Una forma alternativa de mostrar el tiempo transcurrido en tus archivos de vista es usar esta pseudo-variable, si prefieres no usar el PHP puro:

```
{elapsed_time}
```


Nota: Si quieres algo de referencia en tus funciones de controlador deberas setear tus propios puntos de inicio/fin.

Desplegando el consumo de memoria

Si tu instalacion PHP esta configurada con el —enable-memory-limit, puedes desplegar la cantidad de memoria utilizada por sistema entero usando el siguiente codigo en tu archivo vista:

```
<?=$this->benchmark->memory_usage();?>
```

Nota:Esta funcion puede ser solamente usad en los archivos vista. El consumo de memoria reflejara el uso total de memoria por la aplicacion completa.

Una forma alternativa de mostrar el uso de memoria en tus archivos de vista, es usar esta pseudo-variable, si prefieres no usar el PHP puro:

```
{memory_usage}
```

Clase Calendar

La clase Calendar permite crear dinamicamente calendarios. Sus calendarios pueden formatearse a través del uso de una plantilla calendario, permitiendo un 100% de control sobre todos los aspectos de su diseño. Además, puede pasar datos a las celdas de su calendario.

Inicializando la clase

Similar a la mayoría de las clases en codeIgniter, la clase calendar es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez cargado, el objeto Calendario estará disponible usando: *$this->calendar*

Mostrando un calendario

Éste es un ejemplo muy simple que enseña como puede mostrar un calendario:

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

echo $this->calendar->generate();
```

El código anterior generará un calendario para el mes/año actual basado en la hora de su servidor. Para mostrar un calendario para un mes y año específicos deberá pasar esta infromación a la función de generación de calendario:

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

echo $this->calendar->generate(2006, 6);
```

El código de arriba generará un calendario que muestra el mes de junio del año 2006. El primer parámetro especifica el año, el segundo parámetro especifica el mes.

Pasando datos a las celdas de tu calendario

Agregar datos a las celdas de tu calendario involucra crear un arreglo asociativo en el cual los índices corresponden a los das que deseas poblar y el valor del arreglo contiene los datos. El arreglo es pasado al tercer parámetro de la función generar calendario. Considere este ejemplo:

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

$data = array(
    3  => 'http://your-site.com/news/article/2006/03/',
    7  => 'http://your-site.com/news/article/2006/07/',
    13 => 'http://your-site.com/news/article/2006/13/',
    26 => 'http://your-site.com/news/article/2006/26/'
    );

echo $this->calendar->generate(2006, 6, $data);
```

Usando el ejemplo arriba citado, los das número 3, 7, 13 y 26 se convertirán en links que apuntan a las URLs provistas.

Nota: Por defecto se adume que el array contiene links. En la sección que explica la plantilla calendario verá como puede personalizar el modo en que los datos son pasados, de manera que puede pasar tipos diferentes de información

Ajustando las Preferencias de Visualización

Hay siete preferencias que puede ajustar para controlar varios aspectos del calendario. Las preferencias son ajustadas pasando un arreglo de preferencias en el segundo parámetro de la funcion loading. Este es un ejemplo:

```
$prefs = array (
    'start_day'     => 'saturday',
    'month_type'    => 'long',
    'day_type'  => 'short'
    );

$this->load->library('calendar', $prefs);

echo $this->calendar->generate();
```

El código citado anteriormente haría el calendario comenzar en sábado, usa la cabezera de mes "largo", y los nombres de días "pequeños". Más información con respecto a las preferencias a continuación.

Preferencia	Valor por defecto	Opciones	Descripción
template	None	None	Un cadena que contiene su plantilla calendario. Ver la seccion plantilla debajo.
local_time	time()	None	Un timestamp Unix que corresponde al tiempo actual.
start_day	sunday	Any week day	(sunday, monday, tuesday, etc.) Ajusta el día de la semana con que el calendario debería iniciar.
month_type	long	long, short	Determina que versión del nombre del mes usar en la cabecera. long = January, short = Jan.
day_type	abr	long, short, abr	Determina que versión de los nombre de los días de la semana usar en la cabecera de la columna. long = Sunday, short = Sun, abr = Su.
show_next_prev	FALSE	TRUE/FALSE	Determina si se mostrarán los links que permiten ir al mes siguiente/previo. Ver informacin sobre esta característica debajo.
next_prev_url	None	A URL	Ajusta el basepath usado en los links del calendario siguiente/previo.

Mostrando los Links Mes Siguiente/Previo

Permitir a su calendario incrementar/decrementar dinámicamente a través de los links next/previous requiere que establezca su código de calendario similar a este ejemplo:

```
$prefs = array (
    'show_next_prev'  => TRUE,
    'next_prev_url'   => 'http://www.your-site.com/index.php/calendar/show/'
    );

$this->load->library('calendar', $prefs);

echo $this->calendar->generate($this->uri->segment(3), $this->uri->segment(4));
```

Advertirá algunas cosas acerca del ejemplo citado anteriormente:

- Debe ajustar el "show_next_prev" a TRUE.
- Debe proveer la URL al controlador que contiene su calendario en la preferencia "next_prev_url" preference.
- Debe proveer el "Año" y el "Mes" a la función de generación de calendario mediante los segmenteos URI donde aparecen (Nota: La clase calendario automáticamente agrega el año/mes a la URL base provista.).

Creando una Plantilla Calendario

Creando una plantilla calendario puede tener 100% de control sobre el diseño de su calendario. Cada componente de su calendario estará contenido dentro de un par de pseudo-variables como se muestra aquí:

```
$prefs['template'] = '

    {table_open}<table border="0" cellpadding="0" cellspacing="0">{/
table_open}

    {heading_row_start}<tr>{/heading_row_start}

    {heading_previous_cell}<th><a href="{previous_url}">&lt;&lt;</a></th>{/
heading_previous_cell}
    {heading_title_cell}<th colspan="{colspan}">{heading}</th>{/
heading_title_cell}
    {heading_next_cell}<th><a href="{next_url}">&gt;&gt;</a></th>{/
heading_next_cell}

    {heading_row_end}</tr>{/heading_row_end}

    {week_row_start}<tr>{/week_row_start}
    {week_day_cell}<td>{week_day}</td>{/week_day_cell}
    {week_row_end}</tr>{/week_row_end}
```


```
    {cal_row_start}<tr>{/cal_row_start}
    {cal_cell_start}<td>{/cal_cell_start}

    {cal_cell_content}<a href="{content}">{day}</a>{/cal_cell_content}
    {cal_cell_content_today}<div class="highlight"><a href="{content}">{day}</
a></div>{/cal_cell_content_today}

    {cal_cell_no_content}{day}{/cal_cell_no_content}
    {cal_cell_no_content_today}<div class="highlight">{day}</div>{/
cal_cell_no_content_today}

    {cal_cell_blank} {/cal_cell_blank}

    {cal_cell_end}</td>{/cal_cell_end}
    {cal_row_end}</tr>{/cal_row_end}

    {table_close}</table>{/table_close}
';

$this->load->library('calendar', $prefs);

echo $this->calendar->generate();
```

Clase Configuracion(config)

La Clase Config provee un medio de recuperar las preferencias de configuracion. Esas preferencias pueden provenir del archivo por defecto (`application/config/config.php`) o de tus propios archivos de configuración.

> **Nota:** Esta clase es inicializada automáticamente por el sistema, por tanto, no es necesario hacerlo manualmente.

Anatomía de un archivo de Configuracion

Por defecto, CodeIgniter tiene un archivo primario de configuracion, ubicado en `application/config/config.php`. Si abres un archivo usando tu editor de texto verás que los items son almacenados en un arreglo llamado *$config*.

Puedes agregar tus propios intems de configuración a este archivo, o si lo prefieres, puedes mantenerlos en forma separada (asumiendo que necesitas items de configuración), simplemente debes crear tu archivo de configuración y guardarlo en la carpeta *config*.

Nota: Si creas tu propio archivo de configuración con el mismo formato del archivo primario de configuración, y guardándolo en un arreglo llamado *$config*. CodeIgniter lo manejará de una forma inteligente esos archivos, así, no existirá un conflicto inclusive si el arreglo tiene el mismo nombre (asumiendo que el índice del arreglo NO es llamado de la misma forma que en el otro)..

Cargando un archivo de Configuración

Nota: CodeIgniter automáticamente carga el archivo primario de configuración (`application/config/config.php`), por lo tanto, solo necesitarás cargar un archivo de configuración, si lo has creado por tu cuenta.

Hay dos formas de cargar un archivo de configuración:

1. Carga Manual

Para cargar uno de tus archivos de configuración, usarás la siguiente función con el controlador que lo necesita:

```
$this->config->load('nombre_archivo');
```

Donde *nombre_archivo* es el nombre de tu archivo de configuración, sin la extensión .php

Si necesitas cargar multiples archivos de configuración, normalmente, ellos deberan ser fusionados dentro de un arreglo maestro de configuración. Las colisiones de nombres pueden ocurrir, sin embargo, si tienes nombres de indices de arreglos identicos en diferentes archivos de configuración. Para evitar colisiones puedes poner en `TRUE` y cada archivo de configuración será almacenado en un arreglo correspondiente al nombre del archivo de configuración. Por ejemplo:

```
// Guardado en un arreglo con este prototipo: $this->config['blog_settings'] =
$config
$this->config->load('blog_settings', TRUE);
```

Por favor vea mas abajo la seccion titulada *Recuperando Items de Configuración para aprender como recuperar elementos de configuración seteados de esta forma.*

El tercer parámetro te permite suprimir los errores en el caso de que un archivo de configuracion no exista:

```
$this->config->load('blog_settings', FALSE, TRUE);
```


2. Auto-carga

Si encuentras que necesitas un archivo de configuración en particular globalmente, puedes hacer que este se cargue automáticamente por el sistema. Para hacer esto, abre el archivo **autoload.php**, ubicado en `application/config/autoload.php`, y agrega tu archivo de configuracion como se indica en el archivo .

Recuperando elementos de configuración

Para recuperar un elemento de tu archivo de configuración, use la siguiente función:

```
$this->config->item('nombre del item');
```

Donde *item name* es el indice del arreglo $config que quieres recuperar. Por ejemplo, para recuperar la opción de lenguaje debes hacer esto:

```
$lang = $this->config->item('language');
```

La función retorna FALSE (booleano) si el item que tratas de recuperar no existe.

Si estas usando el segundo parámetro de la funcion `$this->config->load` con el fin de asignar tus elementos de configuración a un indice específico puedes hacerlo especificando el nombre del indice en el segundo parametro de la funcion `$this->config->item()`. Ejemplo:

```
//Cargando un archivo de configuración llamado blog_settings.php y asignándolo
a un índice llamado "blog_settings"
$this->config->load('blog_settings', 'TRUE');

//Recuperando un elemento de configuración llamado site_name contenido en el
arreglo blog_settings
$site_name = $this->config->item('site_name', 'blog_settings');

//Una forma alterna de especificar el mismo elemento:
$blog_config = $this->config->item('blog_settings');
$site_name = $blog_config['site_name'];
```

Estableciendo un Elemento de Configuración

Si desea establecer un elemento de configuración de forma dinámica, o cambiar una ya existente, puede hacerlo del siguiente modo:

```
$this->config->set_item('nombre_elemento', 'valor_elemento');
```

Where *item_name* is the $config array index you want to change, and *item_value* is its value.

Funciones de Ayuda

La clase de configuración tiene las siguientes funciones de ayuda:

```
$this->config->site_url();
```

Esta función recupera la URL de tu sitio, junto con el valor del índice que has especificado en el archivo de configuración.

```
$this->config->system_url();
```

Esta función recupera la URL de tu *system folder (carpeta de sistema)*.

La Clase de Base de Datos

CodeIgniter viene con una clase de base de datos llena de "features" y una muy rápido abstracta clase de base de datos que soporta tanto estructuras tradicionales y el patrón Active Record. Las funciones de base de datos ofrecen clara, simple sintaxis.

Comienzo Rápido: Código de Ejemplo

Las páginas siguientes contienen código de ejemplo mostrando como la clase de base de datos es usada. Para detalles completos por favor lea las páginas individuales describiendo cada función.

Inicializando la Clase de Base de Datos

El código siguiente carga e inicializa la clase de base de datos basado en tu configuración:

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

Una vez cargada la clase, está lista para ser usada como se describe a continuación.

Nota: si todas tus páginas requieren acceso a la base de datos, puedes conectarla automáticamente. Vea la página de conexión para detalles.

Consulta Estándar con Múltiples Resultados (Versión Objeto)

```
$consulta = $this->db->query('SELECT nombre, titulo, email FROM mi_tabla');

foreach ($consulta->result() as $fila)
{
    echo $fila->titulo;
    echo $fila->nombre;
    echo $fila->email;
}

echo 'Resultados Totales: ' . $consulta->num_rows();
```

La función *result()* de arriba devuelve un arreglo de **objetos**. Por ejemplo: $fila->titulo

Consulta Estándar con Múltiples Resultados (Versión Arreglo)

```
$consulta = $this->db->query('SELECT nombre, titulo, email FROM mi_tabla');

foreach ($consulta->result_array() as $fila)
{
    echo $fila['titulo'];
    echo $fila['nombre'];
    echo $fila['email'];
}
```

La función *result_array()* de arriba devuelve una arreglo de arreglos con índices comunes. Por ejemplo: $fila['titulo']

Probando los Resultados

Si ejecutas consultas que pueden **not** producir un resultado, eres animado a probar el resultado antes usando la función *num_rows()*:

```
$consulta = $this->db->query("YOUR QUERY");

if ($consulta->num_rows() > 0)
{
    foreach ($consulta->result() as $fila)
    {
    echo $fila->titulo;
    echo $fila->nombre;
    echo $fila->body;
    }
}
```

Consulta Estándar Con Un Solo Resultado

```
$consulta = $this->db->query('SELECT nombre FROM mi_tabla LIMIT 1');

$fila = $consulta->row();
echo $fila->nombre;
```

La función de arriba *row()* devuelve un **objeto**. Por ejemplo: $fila->nombre

Consulta Estándar Con Un Solo Resultado (Versión Arreglo)

```
$consulta = $this->db->query('SELECT nombre FROM mi_tabla LIMIT 1');

$fila = $consulta->row_array();
echo $fila['nombre'];
```

La función de arriba *row_array()* devuelve un **arreglo**. Por ejemplo: $fila['nombre']

Insertar Estándar

```
$sql = "INSERT INTO mitabla (titulo, nombre)
       VALUES (".$this->db->escape($titulo).", ".$this->db-
>escape($nombre).")";

$this->db->query($sql);

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

Consulta Active Record

El Patrón Active Record otorga una forma simplificada de recuperar datos:

```
$consulta = $this->db->get('nombre_tabla');

foreach ($consulta->result() as $fila)
{
    echo $fila->titulo;
}
```

La función *get()* de arriba devuelve todos los resultados de la tabla. La clase Active Record contiene un completo complemento de funciones para trabajar con datos.

Insertar con Active Record

```
$data = array(
    'titulo' => $titulo,
    'nombre' => $nombre,
    'fecha' => $fecha
    );

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

// Produce: INSERT INTO mitabla (titulo, nombre, fecha) VALUES ('{$titulo}',
'{$nombre}', '{$fecha}')
```

Configuración de Base de Datos

CodeIgniter tiene un archivo de configuración que permite almacenar los valores de conexión (usuario, contraseña, nombre de base de datos). El archivo de configuración se localiza en:

```
application/config/database.php
```

Los parametros de configuración son guardados en un arreglo multidimensional con este prototipo:

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "database_name";
$db['default']['dbdriver'] = "mysql";
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = FALSE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
```

La razon por la que usamos un arreglo multidimensionla y no uno más sencillo es permitir el almacenamiento de múltiple parámetros de conexión. Por ejemplo si queremos trabajar con múltiples entornos (desarrollo, producción, prueba, etc.) bajo la misma instalación, podemos un grupo de parametro de configuración para cada uno, luego cambiar entre cada uno según sea necesario. Por ejemplo, para configurar un entorno de prueba podemos hacer esto:

```
$db['test']['hostname'] = "localhost";
$db['test']['username'] = "root";
$db['test']['password'] = "";
$db['test']['database'] = "nombre_de_base_de_datos";
$db['test']['dbdriver'] = "mysql";
$db['test']['dbprefix'] = "";
$db['test']['pconnect'] = TRUE;
$db['test']['db_debug'] = FALSE;
$db['test']['cache_on'] = FALSE;
$db['test']['cachedir'] = "";
$db['test']['char_set'] = "utf8";
$db['test']['dbcollat'] = "utf8_general_ci";
```

Luego para usar este grupo de parámetros (del entorno de prueba) cambiamos el valor de la variable que se encuentra en el archivo de configuración, así:

```
$active_group = "test";
```

Nota: El nombre "test" es arbitrario. Puede ser cualquiera que querramos. Por defecto usamos el valor "default" para la primer conexión, pero esto también puede ser cambiado por algo más relevante para el proyecto.

Active Record

La Clase Active Record es globalmente activada o desactivada estableciendo la variable $active_record en el archivo de configuración para la base de datos, en TRUE/FALSE. Si no vamos a usar la Clase Active Record, establecerla en FALSE hará que se utilicen menos recursos cuando se inicialicen las clases de base de datos.

```
$active_record = TRUE;
```

Note: that some CodeIgniter classes such as Sessions require Active Records be enabled to access certain functionality.

Explanation of Values:

- **hostname** - The hostname of your database server. Often this is "localhost".
- **username** - The username used to connect to the database.
- **password** - The password used to connect to the database.
- **database** - The name of the database you want to connect to.
- **dbdriver** - The database type. ie: mysql, postgre, obdc, etc. Must be specified in lower case.
- **dbprefix** - An optional table prefix which will added to the table name when running Active Record queries. This permits multiple CodeIgniter installations to share one database.
- **pconnect** - TRUE/FALSE (boolean) - Whether to use a persistent connection.
- **db_debug** - TRUE/FALSE (boolean) - Whether database errors should be displayed.
- **cache_on** - TRUE/FALSE (boolean) - Whether database query caching is enabled, see also Database Caching Class.
- **cachedir** - The absolute server path to your database query cache directory.

- **char_set** - The character set used in communicating with the database.
- **dbcollat** - The character collation used in communicating with the database.
- **port** - The database port number. Currently only used with the Postgre driver. To use this value you have to add a line to the database config array.

```
$db['default']['port'] = 5432;
```

Note: Depending on what database platform you are using (MySQL, Postgre, etc.) not all values will be needed. For example, when using SQLite you will not need to supply a username or password, and the database name will be the path to your database file. The information above assumes you are using MySQL.

Conectando a una Base de datos

Hay dos formas de conectarse a una base de datos:

Conectando automáticamente

La característica de "auto conexión" cargará e instanciará la clase de base de datos en cada página cargada. Para permitir "auto conexión", agregue la palabra *database* al arreglo "library", como se indica en el siguiente archivo:

```
application/config/autoload.php
```

Conectando Manualmente

Si sólo algunas páginas requieren conexión a la base de datos, puedes conectar manualmente agregando esta línea de código en cualquier función donde sea necesario, o en el constructor de tu clase para hacer a la base de datos disponible globalmente en esa clase.

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


> Si la función de arriba **no** contiene ninguna información en el primer parámetro, conectará al grupo especificado en tu archivo de configuración de base de datos. Para la mayoría de las personas, este es el método preferido de uso.

El primer parámetro de esta función puede **opcionalmente** ser usado para especificar un grupo en particular de tu archivo de configuración, o puedes envíar valores de conexión para uan base de datos que no esté especificada en tu archivo de configuración. Ejemplos:

Para seleccionar un grupo específico de tu archivo de configuración, puede hacer esto:

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

Donde `nombre_grupo` es el nombre del grupo de la conexión de tu archivo de configuración.

Para conectar manualmente a una base de datos deseada, puedes pasar un arreglo de valores:

```
$config['hostname'] = "localhost";
$config['username'] = "miusername";
$config['password'] = "mipassword";
$config['database'] = "midatabase";
$config['dbdriver'] = "mysql";
$config['dbprefix'] = "";
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

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

Para información en cada uno de esos valores por favor vea la página de configuración.

O puedes envíar valores de tu base de datos como un "Data Source Name". DSNs debe tener este prototipo:

```
$dsn = 'dbdriver://username:password@hostname/database';

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

Note que si usa un DSN no podrá especificar algunos valores por defecto que sí podría hacerlo a través de un arreglo de conexión.

Conectando a Múltiples Bases de Datos

Si necesita conectar a más de una base de datos simultáneamente puede hacer como sigue:

```
$DB1 = $this->load->database('grupo_uno', TRUE);
$DB2 = $this->load->database('grupo_dos', TRUE);
```

Nota: Cambié las palabras "grupo_uno" y "grupo_dos" al nombre del grupo específico al que se está conectando (o puede pasar los valores de conexión como se indico antes).

Cuando se establece el segundo parámetro como TRUE (booleano) la función devolverá el objeto de base de datos.

Cuando se conecta de esta forma, deberá usar el nombre del objeto para ejecutar comandos en vez de la sintaxis usada a través de esta guía. En otras palabras, en vez de ejecutar comandos con:

$this->db->query();
$this->db->result();
etc...

En vez use:

$DB1->query();
$DB1->result();
etc...

Consultas

$this->db->query();

Para realizar una consulta, utilice la siguiente función:

```
$this->db->query('SU CONSULTA AQUI');
```


La función *query()* devuelve un **objeto** resultante de la base de datos cuando se ejecutan consultas de tipo "lectura", las cuales se pueden usar para mostrar sus resultados. Cuando se ejecutan consultas de tipo "escritura" simplemente devuelve VERDADERO o FALSO según el éxito o el fracaso. Para la recuperación de datos se suele asignar típicamente la consulta a su propia variable, como esto:

```
$query = $this->db->query('SU CONSULTA AQUI');
```

$this->db->simple_query();

Esta es una versión simplificada de la función *$this->db->query()*. Esto SOLO devuelve VERDADERO/FALSO según el exito o el fracaso. NO retorna un conjunto de resultados de la base de datos, ni establece el tiempo de la consulta, o compila los datos obligatorios, o almacena su consulta para depuración. Simplemente le permite realizar una consulta. La mayoría de los usuarios raramente suelen usar esta función.

Agregando prefijos manualmente a la Base de Datos

Si ha configurado un prefijo a la base de datos y desea añadirlo manualmente, puede usar lo siguiente.

```
$this->db->dbprefix('tablename');
// Salida prefix_tablename
```

Protegiendo Identificadores

En muchas bases de datos es recomendable proteger las tablas y los nombres de los campos - por ejemplo con backticks en MySQL. Las consultas de Active Record estan automáticamente protegidas, sin embargo si necesita proteger manualmente un identificador puede usar:

```
$this->db->protect_identifier('table_name');
```

Escapando Consultas

Es una muy buena práctica de seguridad para escapar de sus datos antes de presentarlos en su base de datos. CodeIgniter tiene dos funciones que lo ayudan a hacer esto:

1. **$this->db->escape()** Esta función determina el tipo de datos a fin de que sólo pueda escapar de cadenas de datos. También añade automáticamente comillas simples alrededor de los datos para que no tenga que:

```
$sql = "INSERT INTO table (title) VALUES(".$this->db->escape($title).")";
```

2. **$this->db->escape_str()** Esta función escapa a los datos pasados a la misma, independientemente de su tipo. La mayoría de las veces debe usar la función anterior en lugar de ésta. Utilice la función de esta manera:

```
$sql = "INSERT INTO table (title) VALUES('".$this->db-
>escape_str($title)."')";
```

Anidando Consultas

Anidar le permite simplificar la sintaxis de la consulta dejando que el sistema ponga las consultas juntas en lugar de usted. Considere el siguiente ejemplo:

```
$sql = "SELECT * FROM some_table WHERE id = ? AND status = ? AND author = ?";

$this->db->query($sql, array(3, 'live', 'Rick'));
```

Los signos de interrogación en la consulta son automáticamente reemplazados con los valores en el arreglo en el segundo parámetro de la función de consulta.

El beneficio secundario de la utilización de anidar es que los valores son automáticamente escapados, produciendo consultas mas seguras. No tiene que acordarse de escapar los datos manualmente, el motor lo hace automáticamente para usted.

Generación de Resultados de una Consulta

Hay varias formas de generar los resultados de una consulta:

result()

Esta función retorna el resultado de la consulta como un arreglo de **objetos**, o **un arreglo vacío** en el caso de fracaso. Normalmente puede utilizar esto en un bucle foreach, de esta manera:

```
$query = $this->db->query("SU CONSULTA");

foreach ($query->result() as $row)
{
    echo $row->title;
    echo $row->name;
    echo $row->body;
}
```

La anterior *función* es un alias de *result_object()*.

Si ejecuta consultas que podrían no producir un resultado, lo animamos a probar el resultado de esto primero:

```
$query = $this->db->query("SU CONSULTA");

if ($query->num_rows() > 0)
{
    foreach ($query->result() as $row)
    {
    echo $row->title;
    echo $row->name;
    echo $row->body;
    }
}
```

result_array()

Esta función retorna el resultado de la consulta como un arreglo puro, o un arreglo vacío cuando no se producen resultados. Normalmente puede utilizar esto en un bucle foreach, de esta manera:

```
$query = $this->db->query("SU CONSULTA");

foreach ($query->result_array() as $row)
{
    echo $row['title'];
    echo $row['name'];
    echo $row['body'];
}
```

row()

Esta función retorna una única fila como resultado. Si su consulta tiene mas de una fila, ésta solo retorna la primera fila. El resultado es retornado como un **objeto**. Aquí hay un ejemplo de su uso:

```
$query = $this->db->query("SU CONSULTA");

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

    echo $row->title;
    echo $row->name;
    echo $row->body;
}
```

Si desea retornar una fila específica puede presentar el número de la fila como un dígito en el primer parámetro:

```
$row = $query->row(5);
```

row_array()

Idéntica a la función *row()* anterior, excepto que retorna un arreglo. Por ejemplo:

```
$query = $this->db->query("SU CONSULTA");

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

    echo $row['title'];
    echo $row['name'];
    echo $row['body'];
}
```


Si desea retornar una fila específica puede presentar el número de la fila como un dígito en el primer parámetro:

```
$row = $query->row_array(5);
```

Además, se puede caminar hacia adelante / atrás / primera / última fila a través de sus resultados utilizando estas variaciones:

```
$row = $query->first_row()
$row = $query->last_row()
$row = $query->next_row()
$row = $query->previous_row()
```

Por defecto, retorna un objeto a menos que ponga la palabra "array" en el parámetro:

```
$row = $query->first_row('array')
$row = $query->last_row('array')
$row = $query->next_row('array')
$row = $query->previous_row('array')
```

Resultado de Funciones de Ayuda

$query->num_rows()

El número de filas retornado por la consulta. Nota: En este ejemplo, *$query* es la variable que el objeto resultante de la consulta es asignado a:

```
$query = $this->db->query('SELECT * FROM my_table');

echo $query->num_rows();
```

$query->num_fields()

El número de CAMPOS (columnas) retornado por la consulta. Asegúrese de llamar a la función utilizando su objeto resultante de la consulta:

```
$query = $this->db->query('SELECT * FROM my_table');

echo $query->num_fields();
```

$query->free_result()

Se libera la memoria asociada con el resultado y elimina los recursos ID resultantes. Normalmente PHP libera la memoria automáticamente al finalizar la ejecución de scripts. Sin embargo, si está ejecutando una gran cantidad de consultas en un script en particular, puede que desee liberar el resultado después de cada resultado de consulta que se ha generado, con el fin de reducir los consumos de memoria. Por ejemplo:

```
$query = $this->db->query('SELECT title FROM my_table');

foreach ($query->result() as $row)
{
    echo $row->title;
}
$query->free_result(); // El objeto resultante $query dejará de estar
disponible

$query2 = $this->db->query('SELECT name FROM some_table');

$row = $query2->row();
echo $row->name;
$query2->free_result(); // El objeto resultante $query2 dejará de estar
disponible
```

Funciones Auxiliares de Consulta

$this->db->insert_id()

El número de ID de cuando se realizo una inserción a la base de datos.

$this->db->affected_rows()

Muesta el número de filas afectadas, cuando se realiza una consulta de tipo de "escritura" (insert, update, etc.).

Nota: En MySQL "DELETE FROM TABLE" devuelve 0 filas afectadas. La clase de base de datos tiene un pequeño hack que le permite devolver el número correcto de filas afectadas. Por defecto este hack está habilitado pero se puede apagar en el archivo de driver de la base de datos.

$this->db->count_all();

Permite determinar el número de filas en una tabla en particular. El primer parámetro es el nombre de la tabla. Ejemplo:

```
echo $this->db->count_all('my_table');

// Produce un entero, como 25
```

$this->db->platform()

Imprime la plataforma de base de datos que está corriendo (MySQL, MS SQL, Postgre, etc...):

```
echo $this->db->platform();
```

$this->db->version()

Imprime la versión de base de datos que está corriendo:

```
echo $this->db->version();
```

$this->db->last_query();

Devuelve la última consulta que fue ejecutada (la cadena de la consulta, no el resultado). Example:

```
$str = $this->db->last_query();

// Produce: SELECT * FROM sometable....
```

Las dos funciones siguientes ayudan a simplificar el proceso de escritura de INSERTs Y UPDATEs de base de datos.

$this->db->insert_string();

Esta función simplifica el proceso de escritura de inserciones de base de datos. Devuelve una cadena SQL de inserción correctamente formada. Ejemplo:

```
$data = array('nombre' => $nombre, 'email' => $email, 'url' => $url);

$str = $this->db->insert_string('nombre_tabla', $data);
```

El primer parámetro es el nombre de la tabla. El segundo un arreglo asociativo con los datos que serán insertados. El ejemplo anterior produce:

```
INSERT INTO nombre_tabla (nombre, email, url) VALUES ('Rick', 'rick@your-
site.com', 'www.your-site.com')
```

Nota: Los valores son automáticamente escapados, produciendo consultas más seguras.

$this->db->update_string();

Esta función simplifica el proceso de escritura de actualizaciones de base de datos. Devuelve una cadena SQL de actualización correctamente formada. Ejemplo:

```
$data = array('nombre' => $nombre, 'email' => $email, 'url' => $url);

$condicion = "autor_id = 1 AND estado = 'activo'";

$str = $this->db->update_string('nombre_tabla', $data, $condicion);
```

El primer parámetro es el nombre de la tabla, el segundo un arreglo asociativo con los datos que serán insertados, y el tercer parámetro es la cláusula "where". El ejemplo anterior produce:

```
UPDATE nombre_tabla SET name = 'Rick', email = 'rick@your-site.com', url =
'www.your-site.com' WHERE autor_id = 1 AND estado = 'activo'
```

Nota: Los valores son automáticamente escapados, produciendo consultas más seguras.

Clase Active Record

CodeIgniter usa una versión modificada del Patrón de Base de Datos Active Record. Este patrón permite obtener, insertar y actualizar información in tu base de datos con mínima codificación. En algunos casos, sólo una o dos líneas de código son necesarias para realizar una acción de base de datos. CodeIgniter no requiere que cada tabla de la base de datos sea un propio archivo de clase. Se permite una interface más simplificada

Más allá de la simplicidad, un beneficio mayor de usar la Active Record es que te permite crear una aplicación independiente de la base de datos que usa, ya que la sintaxis de consulta es generada por cada adaptador de base de datos. También permite queries más seguras, ya que los valores son escapadas automáticamente por el sistema.

> **Nota:** Si tienes intenciones de usar tus propias consultas, puedese deshabilitar esta clase en tu archivo de configuración de base de datos, permitiendo a la librerí de la base de datos y adaptadores usar menos recursos

Seleccionando Datos

Las siguientes funciones permiten construir una sentencia **SELECT** SQL.

Nota: Si está usando PHP 5, puede usar métodos en cadena para una sintaxis más compacta. Esto es descripto al final de la página.

$this->db->get();

Ejecuta la consulta de selección y devuelve el resultado. Puede ser usado solo, para traer todos los registros de una tabla:

```
$consulta = $this->db->get('mitabla');

// Produce: SELECT * FROM mitabla
```

El segundo y tercer parámetro permiten establecer cláusulas de límite y principio:

```
$consulta = $this->db->get('mitabla', 10, 20);

// Produce: SELECT * FROM mitabla LIMIT 20, 10 (en MySQL. Otras bases de datos
tienen pequeñas diferencias en la sintaxis)
```

Notará que la funcion de arriba es asignada a la variable llamada `$consulta`, que puede ser usada para mostrar los resultados:

```
$consulta = $this->db->get('mitabla');

foreach ($consulta->result() as $fila)
{
    echo $fila->titulo;
}
```

Por favor visite la página de funciones de resultados para una completa discusión acerca de la generación de resultados.

$this->db->get_where();

Identica a la función de arriba, excepto que permite agregar una cl´usula "where" en el segundo parámetro, en vez de usar la función db->where():

```
$consulta = $this->db->get_where('mitabla', array('id' => $id), $limite,
$principio);
```

Por favor lea sobre la función where más abajo para más información

> Nota: get_where() era anteriormente conocida como getwhere(), que ha sido deprecada

$this->db->select();

Permite escribir la porción de SELECT de tu consulta:

```
$this->db->select('titulo, contenido, fecha');

$consulta = $this->db->get('mitabla');

// Produce: SELECT titulo, contenido, fecha FROM mitabla
```


Nota: Si está seleccionando todo (*) de una tabla, no es necesario usar esta función. Cuando se omite, CodeIgniter asume que desa SELECT *

$this->db->select() acepta un opcional segundo parámetro. Si se establece como FALSE, CodeIgniter no intentará proteger los nombres de campos u tablas. Esto es útil si necesita una consulta de selección compuesta.

```
$this->db->select('(SELECT SUM(pagos.monto) FROM pagos WHERE
pagos.factura_id=4') AS monto_pagado', FALSE);
$consulta = $this->db->get('mitabla');
```

$this->db->select_max();

Escribe una porción "SELECT MAX(campo)" de tu consulta. Opcionalmente, puedes incluir un segunda parámetro para renombrar el campo resultante.

```
$this->db->select_max('edad');
$consulta = $this->db->get('miembros');
// Produce: SELECT MAX(edad) as edad FROM miembros

$this->db->select_max('edad', 'edad_miembro');
$query = $this->db->get('miembros');
// Produce: SELECT MAX(edad) as edad_miembro FROM miembros
```

$this->db->select_min();

Escriba una porción "SELECT MIN(campo)" de tu consulta. Como en *select_max()*, puedes opcionalmente incluir un segundo parámetro para renombrar el campo resultante.

```
$this->db->select_min('edad');
$consulta = $this->db->get('miembros');
// Produce: SELECT MIN(edad) as edad FROM miembros
```

$this->db->select_avg();

Escriba una porción "SELECT AVG(campo)" de tu consulta. Como en *select_max()*, puedes opcionalmente incluir un segundo parámetro para renombrar el campo resultante.

```
$this->db->select_avg('edad');
$consulta = $this->db->get('miembros');
// Produce: SELECT AVG(edad) as edad FROM miembros
```

$this->db->select_sum();

Escriba una porción "SELECT SUM(campo)" de tu consulta. Como en *select_max()*, puedes opcionalmente incluir un segundo parámetro para renombrar el campo resultante.

```
$this->db->select_sum('edad');
$consulta = $this->db->get('miembros');
// Produce: SELECT SUM(edad) as edad FROM miembros
```

$this->db->from();

Permite escribir la porción FROM de la consulta:

```
$this->db->select('titulo, contenido, fecha');
$this->db->from('mitabla');

$consulta = $this->db->get();

// Produce: SELECT titulo, contenido, fecha FROM mitabla
```

Nota: Como se mostró antes, la porción FROM de tu consulta puede ser especificada en la función *$this->db->get()*, así que use el método que prefiera

$this->db->join();

Permite escribir una porción JOIN de la consulta:

```
$this->db->select('*');
$this->db->from('blogs');
$this->db->join('comentarios', 'comentarios.id = blogs.id');

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

// Produce:
// SELECT * FROM blogs
// JOIN comentarios ON comentarios.id = blogs.id
```


Multiples llamados a la función pueden ser hechos si necesita multiples joins en una consulta.

Si necesita algo distinto al natural JOIN puede especificarlo a través del tercer parámetro de la función. Las opciones son: left, right, outer, inner, left outer, and right outer.

```
$this->db->join('comentarios', 'comentarios.id = blogs.id', 'left');

// Produce: LEFT JOIN comentarios ON comentarios.id = blogs.id
```

$this->db->where();

Está función habilita establecer cláusulas **WHERE** usando uno de cuátro métodos:

Nota: Todos los valores pasados a esta función son escapados automáticamente, produciendo consultas más seguras.

1. Método simpre de clave/valor:

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

// Produce: WHERE nombre = 'Jose'
```

Note que el signo igual es agregado para uested.

Si utiliza multiples llamadas a la función, ellos serán encadenados juntos con un *AND* entre ellos:

```
$this->db->where('nombre', $nombre);
$this->db->where('titulo', $titulo);
$this->db->where('estado', $estado);

// WHERE nombre = 'Jose' AND titulo = 'jefe' AND estado = 'activo'
```

2. Método especial de clave/valor:

Se puede incluir un operador en el primer parámetro con el objeto de controlar la comparación:

```
$this->db->where('nombre !=', $nombre);
$this->db->where('id <', $id);

// Produce: WHERE nombre != 'Jose' AND id < 45
```

3. **Método de arreglo asociativo:**

```
$arreglo = array('nombre' => $nombre, 'titulo' => $titulo, 'status' =>
$status);
$this->db->where($arreglo);

// Produce: WHERE nombre = 'Joe' AND titulo = 'boss' AND status = 'active'
```

Puede incluir operadores propios cuando se usa este método también:

```
$array = array('nombre !=' => $nombre, 'id <' => $id, 'date >' => $date);

$this->db->where($array);
```

4. **Cadena especial:**

Se puede escribir cláusulas propias manualmente:

```
$where = "nombre='Jose' AND estado='jefe' OR estado='activo'";

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

$this->db->where acepta un tercer parámetro opcional. Si se le pasa FALSE, CodeIgniter no intentará proteger los nombres de sus campos o tablas con backticks.

```
$this->db->where('MATCH (field) AGAINST ("value")', NULL, FALSE);
```

$this->db->or_where();

Está función es idéntica a la de arriba, excepto que multiples instancias son unidas por OR:

```
$this->db->where('nombre !=', $nombre);
$this->db->or_where('id >', $id);

// Produce: WHERE nombre != 'Joe' OR id > 50
```

Nota: or_where() era anteriormente conocida como orwhere(), la cual ha sido deprecada.

$this->db->where_in();

Genera WHERE campo IN ('item', 'item') unido con AND si corresponde

```
$nombres = array('Frank', 'Todd', 'James');
$this->db->where_in('username', $nombres);
// Produce: AND WHERE username IN ('Frank', 'Todd', 'James')
```

$this->db->or_where_in();

Genera WHERE campo IN ('item', 'item') unido con OR si corresponde

```
$nombres = array('Frank', 'Todd', 'James');
$this->db->or_where_in('username', $nombres);
// Produce: OR WHERE username IN ('Frank', 'Todd', 'James')
```

$this->db->where_not_in();

Genera WHERE campo NOT IN ('item', 'item') unido con AND si corresponde

```
$nombres = array('Frank', 'Todd', 'James');
$this->db->where_not_in('username', $nombres);
// Produce: AND WHERE username NOT IN ('Frank', 'Todd', 'James')
```

$this->db->or_where_not_in();

Genera WHERE campo NOT IN ('item', 'item') unido con OR si corresponde

```
$nombres = array('Frank', 'Todd', 'James');
$this->db->or_where_not_in('username', $nombres);
// Produces: OR WHERE username NOT IN ('Frank', 'Todd', 'James')
```

$this->db->like();

Está función permite generar cláusulas **LIKE**, útiles para realizar búsquedas.

Nota: Todos los valores pasados a esta función son escapados automáticamente.

1. Método simple de clave/valor:

```
$this->db->like('titulo', 'match');

// Produce: WHERE titulo LIKE '%match%'
```

Si utiliza múltiples llamados a la función, ellos serán encadenados con *AND* entre ellos:

```
$this->db->like('titulo', 'match');
$this->db->like('body', 'match');

// WHERE titulo LIKE '%match%' AND body LIKE '%match%
```

Si desea controlar donde la wildcard (%) es ubicada, puede utilizar un tercer parámetro opcional. Las opciones son 'before', 'after' y 'both' (por defecto).

```
$this->db->like('titulo', 'match', 'before');
// Produces: WHERE titulo LIKE '%match'

$this->db->like('titulo', 'match', 'after');
// Produces: WHERE titulo LIKE 'match%'

$this->db->like('titulo', 'match', 'both');
// Produces: WHERE titulo LIKE '%match%'
```

2. Método de arreglo asociativo:

```
$array = array('titulo' => $match, 'pagina1' => $match, 'pagina2' => $match);

$this->db->like($array);

// WHERE titulo LIKE '%match%' AND pagina1 LIKE '%match%' AND pagina2 LIKE
'%match%'
```

$this->db->or_like();

Está función es idéntica a la anterior, excepto que multiples instances son unidas por OR:

```
$this->db->like('titulo', 'match');
$this->db->or_like('body', $match);

// WHERE titulo LIKE '%match%' OR body LIKE '%match%'
```

Nota: or_like() era anteriormente conocida como orlike(), la cual ha sido deprecada.

$this->db->not_like();

Esta función es idéntica a **like()**, excepto que genera sentencias NOT LIKE:

```
$this->db->not_like('titulo', 'match');

// WHERE titulo NOT LIKE '%match%
```

$this->db->or_not_like();

Esta función es idéntica a **not_like()**, excepto que muchas instancias son unidas por OR:

```
$this->db->like('titulo', 'match');
$this->db->or_not_like('body', 'match');

// WHERE titulo LIKE '%match% OR body NOT LIKE '%match%'
```

$this->db->group_by();

Permite escribir porciones GROUP BY de la consulta:

```
$this->db->group_by("titulo");

// Produces: GROUP BY titulo
```

También se puede pasar un arreglo de multiples valores:

```
$this->db->group_by(array("titulo", "fecha"));

// Produce: GROUP BY titulo, fecha
```

Nota: group_by() era anteriormente conocida como groupby(), la cual ha sido deprecada.

$this->db->distinct();

Agrega la palabra clave "DISTINCT" a la consulta

```
$this->db->distinct();
$this->db->get('tabla');

// Produce: SELECT DISTINCT * FROM tabla
```

$this->db->having();

Permite escribir la porción HAVING de la consulta:

```
$this->db->having('user_id = 45');

// Produce: HAVING user_id = 45
```

También puede pasar un arreglo de multiples valores:

```
$this->db->having(array('titulo =' => 'Mi titulo', 'id <' => $id));

// Produce: HAVING titulo = 'Mi titulo', id < 45
```

$this->db->or_having();

Idéntica a having(), sólo que separa múltiples cláusulas con "OR".

$this->db->order_by();

Permite establecer una cláusula de ORDER BY. El primer parámetro contiene el nombre de la columna por la que desea ordenar. El segundo parámetro establece la dirección del resultado. Las opciones son `asc` or `desc`, or `random`.

```
$this->db->order_by("titulo", "desc");

// Produce: ORDER BY titulo DESC
```


También puede pasar su propia cadena en el primer parámetro:

```
$this->db->order_by('titulo desc, nombre asc');

// Produce: ORDER BY titulo DESC, nombre ASC
```

O múltiples llamados a la función pueden ser hechos si necesita múltiples campos.

```
$this->db->order_by("titulo", "desc");
$this->db->order_by("nombre", "asc");

// Produces: ORDER BY titulo DESC, nombre ASC
```

Nota: order_by() era anteriormente conocida como orderby(), la cual a sido deprecada.

Note: ordernamiento aleatorio no es soportado actualmetne en los "drivers" Oracle o MSSQL. Estos serán predeterminados como 'ASC'.

$this->db->limit();

Permite limitar el número de filas que desea que devuelva la consulta:

```
$this->db->limit(10);

// Produce: LIMIT 10
```

El segundo parámetro permite establecer el inicio del resultado.

```
$this->db->limit(10, 20);

// Produce: LIMIT 20, 10 (in MySQL. Otras bases de datos tienen pequeñas
diferencias de sintaxis.)
```

$this->db->count_all_results();

Permite determinar el número de filas de una consulta Active Record en particular. Las consultas aceptan las restricciones de Active Record tales como where(), or_where(), like(), or_like(), etc. Ejemplo:

```
echo $this->db->count_all_results('mi_tabla');
// Produce un entero, como 25

$this->db->like('titulo', 'match');
$this->db->from('mi_tabla');
echo $this->db->count_all_results();
// Produce un entero, como 17
```

$this->db->count_all();

Permite determinar el número de filas de una tabla en particular. Permits you to determine the number of rows in a particular table. Envíe el nombre de la tabla como primer parámetro. Por ejemplo:

```
echo $this->db->count_all('mi_tabla');

// Produce un entero, como 25
```

Insertando Datos

$this->db->insert();

Genera una cadena de inserción basado en los datos que se suministren, y ejectua la consulta. Se puede pasar una **arreglo** o un **objeto** a la función. Aquí hay un ejemplo, usando un arreglo:

```
$data = array(
    'titulo' => 'Mi titulo' ,
    'nombre' => 'Mi nombre' ,
    'fecha' => 'Mi fecha'
    );

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

// Produces: INSERT INTO mitabla (titulo, nombre, fecha) VALUES ('Mi titulo',
'Mi nombre', 'Mi fecha')
```

El primer parámetro contiene el nombre de la table, la segunda es un arreglo de valores.

Aquí hay un ejemplo usando un objeto:

```
/*
    class Myclass {
        var $titulo = 'Mi titulo';
        var $contenido = 'Mi Contenido';
        var $fecha = 'Mi Fecha';
    }
*/

$objeto = new Myclass;

$this->db->insert('mitabla', $objeto);

// Produce: INSERT INTO mitabla (titulo, contenido, fecha) VALUES ('Mi
titulo', 'Mi Contenido', 'Mi Fecha')
```

El primer parámetro contendrá el nombre de la tabla, el segundo es un arreglo asociativa de valores.

> **Nota:** Todos los valores son escapados automáticamente produciendo consultas más seguras.

$this->db->set();

Esta función habilita permite establecer valores para *insertar* o *actualizar*.

Puede ser usado en vez de pasar un arreglo de datos directamente a las funciones de insert o update:

```
$this->db->set('nombre', $nombre);
$this->db->insert('mitabla');

// Produce: INSERT INTO mitabla (nombre) VALUES ('{$nombre}')
```

Si utiliza múltiples llamados a la función, ellos serán ensamblados apropiadamente basados en si está insertando o actualizando:

```
$this->db->set('nombre', $nombre);
$this->db->set('titulo', $titulo);
$this->db->set('estado', $estado);
$this->db->insert('mytable');
```

set() también aceptará un opcional tercer parámetro ($escape), que prevendrá datos de ser escapado si es establecido como FALSE. Para ilustrar la diferencia, aquí está set() usado con y sin el parámetro de escape.

```
$this->db->set('campo', 'campo+1', FALSE);
$this->db->insert('mitabla');
// resulta INSERT INTO mitabla (campo) VALUES (campo+1)

$this->db->set('campo', 'campo+1');
$this->db->insert('mitabla');
// resulta INSERT INTO mitabla (campo) VALUES ('campo+1')
```

También puede pasar un arreglo asociativo a esta función:

```
$arreglo = array('nombre' => $nombre, 'titulo' => $titulo, 'estado' =>
$estado);

$this->db->set($arreglo);
$this->db->insert('mitabla');
```

O un objeto:

```
/*
    class Miclase {
        var $titulo = 'Mi titulo';
        var $content = 'Mi Content';
        var $date = 'Mi Date';
    }
*/

$objeto = new Miclase;

$this->db->set($objeto);
$this->db->insert('mitabla');
```

Actualizando Datos

$this->db->update();

Genera una cadena de actualización y corre la consulta basado en los datos suministrados. Puede pasar una **arreglo** o un **objeto** a la función. Aquí hay un ejemplo, usando un arreglo:

```
$data = array(
    'titulo' => $titulo,
    'nombre' => $nombre,
    'fecha' => $fecha
    );

$this->db->where('id', $id);
$this->db->update('mitabla', $data);

// Produces:
// UPDATE mitabla
// SET titulo = '{$titulo}', nombre = '{$nombre}', fecha = '{$fecha}'
// WHERE id = $id
```

O puede suministrar un objeto:

```
/*
    class Miclase {
        var $titulo = 'My titulo';
        var $contenido = 'My Contenido';
        var $fecha = 'My Fecha';
    }
*/
$objeto = new Miclase;
$this->db->where('id', $id);
$this->db->update('mitabla', $objeto);
// Produce:
// UPDATE mitabla
// SET titulo = '{$titulo}', nombre = '{$nombre}', fecha = '{$fecha}'
// WHERE id = $id
```

Nota: Todos los valores son escapados automáticamente produciendo consulta más seguras.

Notará el uso de la función *$this->db->where()*, permitiendo establecer una cláusula WHERE. Opcionalmente, puede pasar información directamente como una cadena:

```
$this->db->update('mitabla', $data, "id = 4");
```

O como un arreglo:

```
$this->db->update('mitabla', $data, array('id' => $id));
```

También se puede usar la función *$this->db->set()* descripta anteriormente, cuando se efectúan actializaciones.

Borrando Datos

$this->db->delete();

Genera una cadena de eliminación SQL y ejecuta la consulta.

```
$this->db->delete('mitabla', array('id' => $id));

// Produce:
// DELETE FROM mitabla
// WHERE id = $id
```

El primer parámetro es el nombre de la tabla, el segundo la cláusula WHERE. También puede usar las funciones *where()* o *or_where()* en vez de pasar los datos como segundo parámetro de la función:

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

// Produce:
// DELETE FROM mitabla
// WHERE id = $id
```

Un arreglo de nombres de tablas puede ser pasada a delete() si desea eliminar datos de más de una tabla.

```
$tablas = array('tabla1', 'tabla2', 'tabla3');
$this->db->where('id', '5');
$this->db->delete($tablas);
```

Si desea eliminar todos los datos de una tabla, puede usar la función *truncate()*, o *empty_table()*.

$this->db->empty_table();

```
Genera una cadena SQL de eliminación y ejecuta la consulta. $this->db-
>empty_table('mitabla');

// Produce:
// DELETE FROM mitabla
```


$this->db->truncate();

Genera una cadena SQL de truncado y ejecuta una consulta.

```
$this->db->from('mitabla');
$this->db->truncate();
// or
$this->db->truncate('mitabla');

// Produce:
// TRUNCATE mitabla
```

Nota: Si el comando TRUNCATE no está disponible, truncate() ejecutará "DELETE FROM tabla".

Métodos en cadena

Métodos en cadena permite simplificar tu sintaxis conectando múltiples funciones. Considere este ejemplo:

```
$this->db->select('titulo')->from('mitabla')->where('id', $id)->limit(10, 20);

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

Note: Métodos en cadena sólo funciona con PHP 5.

Active Record Caching

Mientras no es un verdadero cacheo, Active Record permite salvar (o "cache") ciertas partes de tus consultas para reusar después. Normalmente, cuando una llamada a Active Record es completada, toda la información es reinizializada para el próximo llamado. Con cacheo, puede prevenir esta reinización, y reusar la información fácilmente.

Las llamadas a cache son acumulables. Si ejecuta 2 llamadas a cached select(), y luego 2 llamadas a no cached select(), esto ersultará en 4 llamadas a select(). Hay tres funciones de cacheo disponibles:

$this->db->start_cache()

Esta función debe ser llamada para comenzar a cachear. Todas las consultas Active Record del tipo correcto (ver debajo para las consultas soportadas) son guardadas para uso posterior.

$this->db->stop_cache()

Esta función puede ser llamada para parar el cacheo.

$this->db->flush_cache()

Esta función elimina todos los items cacheados por la Active Record.

Aquí hay un ejemplo de uso:

```
$this->db->start_cache();
$this->db->select('campo1');
$this->db->stop_cache();
$this->db->get('tabla');
// Results in:
// SELECT `campo1` FROM (`tabla`)

$this->db->select('campo2');
$this->db->get('tabla');
// Results in:
// SELECT `campo1`, `campo2` FROM (`tabla`)

$this->db->flush_cache();

$this->db->select('campo2');
$this->db->get('tabla');
// Results in:
// SELECT `campo2` FROM (`tabla`)
```

Nota: Los siguientes campos pueden ser cacheados: 'select', 'from', 'join', 'where', 'like', 'groupby', 'having', 'orderby', 'set'

Transacciones

La clase de abstracción de base de datos de CodeIgniter permite usar *transactions* con bases de datos que soporten tipos de tablas de transacciones seguras. En MySQL necesitará correr tipos de tablas InnoDB o BDB en vez de la más común MyISAM. La mayoría de las otras bases de datos soportan transacciones nativamente.

Si no está familiarizado con transacciones recomendamos buscar ub buen recurso en línea para aprender acerca de su base de datos en particulare. La información debajo asume que usted tiene un básico entendimiento de transacciones.

El Acercamiento de CodeIgniter a las Transacciones

CodeIgniter utiliza un aproximamiento para transacciones que es muy similar al proceso usado por la popular clase de base de datos ADODB. Hemos elegido ese acercamiento porque simplifica excelentemente el proceso de ejecución de transacciones. En la mayoría de los casos todo lo que es requerido son dos líneas de código.

Tradicionalmente, las transacciones han requerido una justa cantidad de trabajo para implementar, debido a que demandan que mantenga un rastro de sus consultas y determinar si desea *commit* o *rollback* basado en el éxito o fallo de sus consultas. Esto es particularmente engorroso con consultas en cadena. En contraste, hemos implementado un sistema de transacciones inteligeuten que hace todo esto automáticamente por ti (también puede hacerlo manualmente, si lo desea, pero no existe beneficio real).

Ejecutando Transacciones

Para ejecutar tus consultas usando transacciones, usará las funciones *$this->db->trans_start()* y *$this->db->trans_complete()* de esta manera:

```
$this->db->trans_start();
$this->db->query('UNA CONSULTA SQL...');
$this->db->query('OTRA CONSULTA...');
$this->db->query('Y TODAVÍA OTRA CONSULTA...');
$this->db->trans_complete();
```

Puede ejecutar tantas consultas como desee entre las funciones start/complete y ellas serán todas cometidas o canceladas basado en el éxito o fallo de una consulta dada.

Manejando Errores

Si tiene reporte de errores habilitado en el archivo *config/database.php* verá un mensaje de error estándar si nos fue éxitosa la transacción. Si debug está deshabilitado, se pueden manejar los propios errores de esta forma:

```
$this->db->trans_start();
$this->db->query('UNA CONSULTA SQL...');
$this->db->query('OTRA CONSULTA...');
$this->db->trans_complete();

if ($this->db->trans_status() === FALSE)
{
    // genera un error... o usa la función log_message() para guardar un
registro del error
}
```

Habilitando Transacciones

Las transacciones son habilitadas automáticamente al momento en que se usa *$this->db->trans_start()*. Si desea deshabilitar transacciones se puede hacer usando *$this->db->trans_off()*:

```
$this->db->trans_off()

$this->db->trans_start();
$this->db->query('UNA CONSULTA SQL...');
$this->db->trans_complete();
```

Cuando las transacciones son deshabilitadas, las consultas serán auto-cometidas, de la misma forma que ejecutando consultas sin transacciones.

Modo de Prueba

Opcionalmente, se puede poner al sistema de transacciones en "modo de prueba", el cual causará que tus consultas sean canceladas — incluso si las consultas producen un resultado valido. Para usar el modo de prueba simplemente establezca el primer parámetro en la función *$this->db->trans_start()* como `TRUE`:

```
$this->db->trans_start(TRUE); // La consulta será cancelada
$this->db->query('UNA CONSULTA SQL...');
$this->db->trans_complete();
```


Corriendo Transacciones Manualmente

Si desea correr transacciones manualmente, puede hacerlo como sigue

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

$this->db->query('UNA CONSULTA SQL...');
$this->db->query('OTRA CONSULTA...');
$this->db->query('Y TODAVÍA OTRA CONSULTA...');

if ($this->db->trans_status() === FALSE)
{
    $this->db->trans_rollback();
}
else
{
    $this->db->trans_commit();
}
```

Nota: Asegurese de usar $this->db->trans_begin() cuando corre transacciones manuales, **NO** *$this->db->trans_start()*.

Datos de Tabla

Estas funciones le permite obtener información de tablas.

$this->db->list_tables();

Devuelve un arreglo que contiene todos los nombres de todas las tablas en la base de datos a la que actualmente está conectado. Ejemplo:

```
$tablas = $this->db->list_tables();

foreach ($tablas as $tabla)
{
   echo $tabla;
}
```

$this->db->table_exists();

A veces es útil saber si una tabla en particular existe antes de correr una operación en ella. Devuelve un booleano TRUE/FALSE. Ejemplo de uso:

```
if ($this->db->table_exists('nombre_tabla'))
{
   // algo de código...
}
```

Nota: Reemplace *nombre_tabla* con el nombre de la tabla que está buscando.

Campo de Datos

$this->db->list_fields()

Devuelve un arreglo que contiene los nombres ed los campos. Esta consulta puede ser llamada de dos formar:

1. Se puede suministrar el nombre de la tabla y ser llamado desde el objeto *$this->db->*:

```
$campos = $this->db->list_fields('nombre_tabla')

foreach ($campos as $campo)
{
    echo $campo;
}
```

2. Se puede reunir los nombres de los campos asociados a cualquier consulat que ejecute llamando a la función desde el objeto de resultado de la consulta:

```
$consulta = $this->db->query('SELECT * FROM some_table')

foreach ($consulta->list_fields() as $campo)
{
    echo $campo;
}
```


$this->db->field_exists()

A veecs es útil saber si un campo particular existe antes de realizar una acción. Devuelve un booleano TRUE/FALSE. Ejemplo de uso:

```
if ($this->db->field_exists('nombre_campo', 'nombre_tabla'))
{
   // some code...
}
```

Nota: Reemplace *nombre_campo* con el nombre de la columna qu está buscando, y reemplace *nombre_tabla* con el nombre de la table que está buscando.

$this->db->field_data()

Devuelve un arreglo de objetos conteniendo información del campo.

A veces es útil recolectar los nombres de los campos u otra metainformación, como el typo de columna, el máximo largo, etc.

> Nota: No todas las bases de datos proveen meta-data.

Ejemplo de uso:

```
$campos = $this->db->field_data('nombre_tabla')

foreach ($campos as $campo)
{
    echo $campo->name;
    echo $campo->type;
    echo $campo->max_length;
    echo $campo->primary_key;
}
```

Si ya ha ejecutado una consulta, puede usar el objeto de resultado en vez de proveer el nombre de la tabla:

```
$consulta = $this->db->query("YOUR QUERY")
$campos = $consulta->field_data()
```

Los siguientes datos están disponible desde esta función, si es soportado por su base de datos:

- name - nombre de la columna
- max_length - máximo largo de la columna
- primary_key - 1 si la columna es una clave primeria
- type - el tipo de columna

Llamados a Funciones Especiales

$this->db->call_function();

Esta función permite llamar a funciones de la base de datos de PHP que no son nativamente incluidas en CodeIgniter, en una forma independiente de la plataforma. Por ejemplo, digamos que quieres llamar a la función mysql_get_client_info(), la cual no es nativamente soportada por CodeIgniter. Puede hacer algo así:

```
$this->db->call_function('get_client_info');
```

Debe suministrar el nombre de la función, sin el prefijo mysql_, en el primer parámetro. El prefijo es agregado automáticamente basado en el driver de base de datos que actualmente es usado. Esto permite que ejecutes la misma función en diferentes plataformas de base de datos. Obviamente no todas las llamadas a la función son idénticas entre plataformas, entonces estos son los límites de cuán útil esta función puede ser en términos de portabilidad.

Cualquier parámetro necesitado por la función que estás llamando será agregado al segundo parámetro.

```
$this->db->call_function('some_function', $param1, $param2, etc..);
```

A menudo, necesitarás suministrar un ID de conexión a la base de datos o ID del resultado de base de datos. El ID de conexión puede ser accedido usando:

```
$this->db->conn_id;
```


El ID del resultado puede ser accedido desde el objeto de resultado, de esta forma:

```
$consulta = $this->db->query("SOME QUERY");

$consulta->result_id;
```

Clase Almacenamiento en Caché de Base de Datos

La Clase Almacenamiento en Caché de Base de Datos le permite cachear sus consulas como archivos de texto para reducir la carga de la base de datos.

Importante: Esta clase es inicializada automáticamente por el controlador de la base de datos cuando el cacheo es habilitado. NO cargar esta clase manualmente.

Tenga en cuenta también: No todas las funciones de resultados de consultas están disponibles cuando se utiliza la memoria caché. Por favor, lea esta página cuidadosamente.

Habilitar el almacenamiento en caché

El almacenamiento en caché se habilita en tres pasos:

- Crear un directorio de escritura en su servidor donde los archivos de caché puedan ser almacenados.
- Establecer la ruta de su carpeta de caché en su archivo *application/config/database.php*.
- Habilitar la función de caché, ya sea globalmente mediante el establecimiento de la preferencia en su archivo *application/config/database.php*, o manualmente como se describe a continuación.

Una vez activado, el almacenamiento en caché sucederá de forma automática cada vez que se carga una página que contiene consultas a la base de datos.

Cómo trabaja el almacenamiento de caché?

El sistema de almacenamiento en caché de consultas de CodeIgniter ocurre dinámicamente cuando sus páginas son visitadas. Cuando la memoria caché está

habilitada, la primera vez que una página web se carga, el objeto resultante de la consulta será serializado y almacenado en un archivo de texto en su servidor. La próxima vez que la página se cargue, el archivo de caché se utilizará en lugar de acceder a su base de datos. El uso de la base de datos puede, efectivamente, ser reducido a cero para cualquiera de las paginas que haya sido cacheada.

Sólo consultas del *tipo-Lectura* (SELECT) pueden ser cacheadas, ya que éstas son el único tipo de consultas que producen un resultado. Las consulta del *tipo-Escritura* (INSERT, UPDATE, etc.), ya que no generan un resultado, no serán cacheadas por el sistema.

La caché de archivos NO caduca. Todas las consultas que se han almacenado en caché permanecerán guardadas hasta que usted las borre. El sistema de almacenamiento en caché permite borrar cachés asociadas con páginas individuales, o puede eliminar toda la colección de archivos de caché. Típicamente usted utilizará las funciones de limpieza que se describen a continuación para borrar archivos de la memoria caché, después de que algunos eventos tengan lugar, como cuando se añade nueva información a su base de datos.

Cómo mejora el rendimiento de su sitio el almacenamiento en caché?

Obtener una ganancia de rendimiento como resultado de la memoria caché depende de muchos factores. Si dispone de una base de datos altamente optimizada con muy poca carga, probablemente no verá un incremento del rendimiento. Si su base de datos está con bastante uso probablemente verá una mejor respuesta, en el supuesto de su sistema de archivos no esté excesivamente taxed. Recuerde que el almacenamiento en caché simplemente cambia como la información es recuperada, pasando de ser una operación de base de datos a una del sistema de archivos.

En algunos entornos de servidores en cluster, por ejemplo, el almacenamiento en caché puede ser perjudicial, debido a que las operaciones del sistema de archivo son intensas. En servidores únicos en ambientes compartidos, el almacenamiento en memoria caché será probablemente beneficioso. Lamentablemente no hay una sola respuesta a la pregunta de si se debe hacer caché de la base de datos. Realmente depende de su situación.

Cómo se almacenan los archivos de caché?

CodeIgniter almacenada el resultado de CADA consulta en su propio archivo de caché. Los conjuntos de archivos de caché son almacenados en las sub-carpetas correspondientes a sus funciones de controlador. Para ser precisos las sub-carpetas se

nombran idénticamente a los primeros dos segmentos de su URI (el nombre de la clase controlador y el nombre de la función).

Por ejemplo, digamos que tiene un controlador llamado *blog* con una función llamada *comments* que contiene tres consultas. El sistema de almacenamiento en caché una carpeta de caché llamada `blog+comments`, en la se escribirá los tres archivos de caché.

Si usa consultas dinámicas que cambian basadas en la información en su URI (cuando se usa paginación, por ejemplo), cada instancia de la consulta producirá su propio archivo de caché. Es posible, por lo tanto, terminar muchas veces con mas archivos de caché que tengan consultas.

Gestión de sus Archivos de Caché

Como los archivos de caché no caduca, tendrá que crear rutinas de eliminación en su aplicación. Por ejemplo, digamos que usted tiene un blog que permite a los usuarios comentar. Siempre que un nuevo comentario se presenta, querrá borrar los archivos de caché asociados con la función del controlador que sirve sus comentarios. Encontrará dos funciones de eliminación descriptas a continuación que le ayudarán a limpiar los datos.

No Todas las Funciones de Base de Datos Trabajan con almacenamiento en Caché

Por último, hay que señalar que el objeto resultante que es almacenado en caché es una versión simplificada del objeto resultante completo. Por esa razón, algunas de las funciones de resultado de consulta no están disponibles para su uso.

Las siguientes funciones `NO ESTÁN` disponibles cuando se utiliza un objeto resultado de la caché:

- num_fields()
- field_names()
- field_data()
- free_result()

Además, los dos recursos de base de datos (result_id y conn_id) no están disponibles cuando se usa almacenamiento en memoria caché, ya que los resultados de los recursos solo se refieren a operaciones en tiempo de ejecución.

$this->db->cache_on() / $this->db->cache_off()

Habilita/Deshabilita manualmente el almacenamiento en caché. Esto puede ser útil si desea mantener algunas consultas despues de haber sido cacheadas. Por ejemplo:

```
// Habilito caché
$this->db->cache_on();
$query = $this->db->query("SELECT * FROM mytable");

// Deshabilito caché solo para esta onsulta
$this->db->cache_off();
$query = $this->db->query("SELECT * FROM members WHERE member_id =
'$current_user'");

// Habilito caché nuevamente
$this->db->cache_on();
$query = $this->db->query("SELECT * FROM another_table");
```

$this->db->cache_delete()

Elimina archivos de caché asociados con una página en particular. Esto es útil si necesita borrar la memoria caché después de actualizar su base de datos.

El sistema de almacenamiento en caché guarda los archivos de la memoria caché a las carpetas que corresponden a la URI de la página que está viendo. Por ejemplo, si está viendo una página en *www.your-site.com/index.php/blog/comments*, el sistema de almacenamiento en caché pondrá todos los archivos de caché asociado a ella en una carpeta llamada *blog+comments*. Para eliminar estos archivos de caché utilizará:

```
$this->db->cache_delete('blog', 'comments');
```

Si no utiliza todos los parámetros de la actual URI, se utilizará a la hora de determinar que es lo que debe ser borrado.

$this->db->cache_delete_all()

Borra todos los archivos de caché existentes. Por ejemplo:

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


Clase Database Forge

La Clase Database Forge Class contiene funciones que ayudan a manejar to base de datos.

Inicializando la Clase Forge

Importante: Para inicializar la clase Forge, tu cliente de base de datos debe estar corriendo, ya que la clase forge depende de él.

Se carga la Clase Forge de esta manera:

```
$this->load->dbforge()
```

Una vez inicializada tendrá acceso a las funciones usando el objeto *$this->dbforge*:

```
$this->dbforge->alguna_funcion()
```

$this->dbforge->create_database('nombre_base_de_datos')

Permite crear una base de datos especificada en el primer parámetro. Devuelve TRUE/FALSE basado en el éxito o fallo:

```
if ($this->dbforge->create_database('mi_base_de_datos'))
{
    echo 'Base de datos creada!';
}
```

$this->dbforge->drop_database('nombre_base_de_datos')

Permite borrar una base de datos especificada en el primer parámetro. Devuelve TRUE/FALSE basado en el éxito o fallo:

```
if ($this->dbforge->drop_database('mi_base_de_datos'))
{
    echo 'Base de datos eliminada!';
}
```

Creando y Eliminando Tablas

Existen varias cosas que se pueden querer hacer al crear tablas. Agregar campos, agregar claves, cambiar columnas. CodeIgniter provee un mecanismo para ello.

Agregando Campos

Los campos son creados a través de un arreglo asociativo. Dentro del arreglo se debe incluir la clave 'type' que establece el tipo de dato del campo. Por ejemplo, INT, VARCHAR, TEXT, etc. Muchos tipos de datos (por ejemplo VARCHAR) también requieren una clave 'constraint'.

```
$campos = array(
    'usuarios' => array(
        'type' => 'varchar',
        'constraint' => '100',
        ),
    );

// será traducido como "usarios VARCHAR(100)" cuando el campo sea agregado.
```

Adicionalmente, el siguiente clave/valor puede ser usado:

- unsigned/true : para generar "UNSIGNED" en la definición del campo
- default/value : para generar un valor por defecto en la definición del campo
- null/true : para generar "NULL" en la definición del campo. Sin ellos, el campo será "NOT NULL" por defecto
- auto_increment/true : genera una bandera auto_increment al campo. Note que el tipo de campo debe ser entero

```
$campos = array(
    'blog_id' => array(
        'type' => 'INT',
        'constraint' => 5,
        'unsigned' => TRUE,
        'auto_increment' => TRUE
        ),
    'blog_titulo' => array(
        'type' => 'VARCHAR',
        'constraint' => '100',
        ),
    'blog_autor' => array(
        'type' =>'VARCHAR',
        'constraint' => '100',
        'default' => 'King of Town',
        ),
    'blog_descripcion' => array(
        'type' => 'TEXT',
        'null' => TRUE,
                ),
        );
);
```


Luego de que estos campos fueron definidos, pueden ser agregados usando *$this->dbforge->add_field($campos);* seguido por una llamada a la función *create_table()*.

$this->dbforge->add_field()

La función agregar campos aceptará el arreglo anterior.

Pasando cadenas como campos

Si se sabe exactamente como quiere que sea creado un campo, se puede pasar una cadena a las definiciones de campo con add_field()

```
$this->dbforge->add_field("etiqueta varchar(100) NOT NULL DEFAULT 'etiqueta
por defecto'");
```

Nota: Múltiples llamadas a *add_field()* son acumuladas.

Creando un campo id

Hay una excepción para crear campos id. Un campo de tipo id automaticamente será asignado como INT(9) auto_incrementing Primary Key.

```
$this->dbforge->add_field('id');
// resulta id INT(9) NOT NULL AUTO_INCREMENT
```

Agregando Claves

En general, se desea que una tabla tenga Claves. Esto es logrado con *$this->dbforge->add_key('campo')*. Un opcional segundo parámetro establecido como TRUE lo hará clave primaria. Note que *add_key()* debe ser seguido por una llamada a *create_table()*.

```
$this->dbforge->add_key('blog_id', TRUE);
// resulta PRIMARY KEY (blog_id)

$this->dbforge->add_key('blog_nombre');
// resulta KEY (blog_nombre)
```

Creando una tabla

Luego de que los campos y las claves fueron declarados, se puede crear una tabla con

```
$this->dbforge->create_table('nombre_tabla');
// resulta CREATE TABLE table_name
```

Un segundo parámetro establecido como TRUE agrega la cláusula "IF NOT EXISTS" a la definición

```
$this->dbforge->create_table('nombre_tabla', TRUE);
// resulta CREATE TABLE IF NOT EXISTS nombre_tabla
```

Borrando una tabla

Ejecuta un sql DROP TABLE

$this->dbforge->drop_table('nombre_tabla');
// resulta DROP TABLE IF EXISTS table_name

Modificando Tablas

$this->dbforge->add_column()

La función add_column() es usada para modificar una tabla existente. Acepta el mismo arreglo que arriba, y puede ser usada para un ilimitado número de campos adicionales.

```
$campos = array(
        'preferencias' => array('type' => 'TEXT')
);
$this->dbforge->add_column('sitios', $campos);

// resulta ALTER TABLE sitios ADD preferencias TEXT
```


$this->dbforge->drop_column()

Usado para remover una columna de una tabla.

```
$this->dbforge->drop_column('nombre_tabla', 'columna_a_borrar');
```

$this->dbforge->modify_column()

El uso de esta función es idéntico a add_column(), excepto que altera una columna existente, en vez de agregar una nueva. Para utilizarla se debe agregar la clave "name" al arreglo de definición del campo.

```
$campos = array(
        'nombre_viejo' => array(
                        'name' => 'nombre_nuevo',
                        'type' => 'TEXT',
                ),
);
$this->dbforge->modify_column('sites', $campos);

// gives ALTER TABLE sites CHANGE nombre_viejo nombre_nuevo TEXT
```

Clase de Utilidades de Base de Datos

La Clase de Utilidades de Base de Datos contiene funciones que le ayudan a manejar su base de datos.

Inicializando la Clase de Utilidades

Importante: Para poder inicializar la clase de Utilidades, debe estar corriendo el driver de la base de datos, ya que la clase de utilidades depende de él.

Se carga la Clase de Utilidades así:

```
$this->load->dbutil()
```

Una vez inicializada, se tiene acceso a las funciones usando el objeto *$this->dbutil*:

```
$this->dbutil->alguna_funcion()
```

$this->dbutil->list_databases()

Devuelve un arreglo de los nombres de bases de datos:

```
$dbs = $this->dbutil->list_databases();

foreach($dbs as $db)
{
     echo $db;
}
```

$this->dbutil->optimize_table('nombre_tabla');

Nota: Esta característica está sólo disponible para las bases de datos MySQL/MySQLi.

Permite optimizar una tabla usando el nombre de la tabla especificado en el primer parámetro. Devuelve TRUE/FALSE basado en el éxito o fallo:

```
if ($this->dbutil->optimize_table('nombre_tabla'))
{
     echo 'Éxito!';
}
```

Nota: Not all database platforms support table optimization.

$this->dbutil->repair_table('nombre_tabla');

Nota: Esta característica está sólo disponible para las bases de datos MySQL/MySQLi.

Permite reparar una tabla usando el nombre de la tabla especificado en el primer parámetro. Devuelve TRUE/FALSE basado en el éxito o fallo:

```
if ($this->dbutil->repair_table('nombre_tabla'))
{
     echo 'Éxito!';
}
```

Nota: Not all database platforms support table optimization.

$this->dbutil->optimize_database();

Nota: Esta característica está sólo disponible para las bases de datos MySQL/MySQLi.

Permite optimizar la base de datos a la que actualmente está conectada la clase. Devuelve un arreglo que contiene los mensajes de estado o FALSE si falla:

```
$resultado = $this->dbutil->optimize_database();

if ($resultado !== FALSE)
{
     print_r($resultado);
}
```

Nota: Not all database platforms support table optimization.

$this->dbutil->csv_from_result($db_resultado)

Permite generar un archivo CSV de un resultado de consulta. El primer parámetro de la función debe contener el objeto resultado de la consulta. Ejemplo:

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

$query = $this->db->query("SELECT * FROM mitabla");

echo $this->dbutil->csv_from_result($query);
```

El segundo y tercer parámetro permite establecer el caracter delimitador y el caracter de nueva línea. Por defecto se utiliza tabulaciones como delimitadores y "\n" como nuevas líneas. Ejemplo:

```
$delimitador = ",";
$nuevalinea = "\r\n";

echo $this->dbutil->csv_from_result($consulta, $delimitador, $nuevalinea);
```

Importante: Esta función NO escribirá el archivo. Simplemente crea el la disposición del CSV. Si necesita escribir el archivo use el Asistente de Archivos.

$this->dbutil->xml_from_result($db_resultado)

Permite generar un archivo XML de un resultado de consulta. El primer parámetro espera un objeto de resulta de consulta, el segundo puede contener un arreglo opcional de parámetros de configuración. Ejemplo:

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

$consulta = $this->db->query("SELECT * FROM mitabla");

$configuracion = array (
        'root'   => 'root',
        'element' => 'element',
        'newline' => "\n",
        'tab'   => "\t"
        );

echo $this->dbutil->xml_from_result($consulta, $configuracion);
```

Importante: Esta función NO escribirá el archivo. Simplemente crea el la disposición del XML. Si necesita escribir el archivo use el Asistente de Archivos.

$this->dbutil->backup()

Permite respaldar toda la base de datos o tablas individuales. La copia de seguridad puede ser comprimida con formato Zip o Gzip.

> **Nota:** Esta característica sólo está disponible en las bases de datos MySQL.

Nota: Dado el limitado tiempo de ejecución y memoria disponible para PHP, respaldar bases de datos muy grandes puede no ser posible. Si su base de datos es muy grande, puede necesitar respaldarla directamente desde el servidor SQL a través de la línea de comando, o que su administrador de servidor lo haga por usted si no tiene privilegios suficientes.

Ejemplo de Uso

```
// Carga la clase de utilidades de base de datos
$this->load->dbutil();

// Crea una copia de seguridad de toda la base de datos y la asigna a una
variable
$copia_de_seguridad =& $this->dbutil->backup();

// Carga el asistente de archivos y escribe el archivo en su servidor
```


```
$this->load->helper('file');
write_file('/ruta/a/copia_de_seguridad.gz', $copia_de_seguridad);

// Carga el asistente de descarga y envía el archivo a su escritorio
$this->load->helper('download');
force_download('copia_de_seguridad.gz', $copia_de_seguridad);
```

Estableciendo Preferencias de Copia de Seguridad

Las preferencias de copia de seguridad son establecidas enviando un arreglo de valores como primer parámetro de la función "backup". Ejemplo:

```
$prefs = array(
    'tables'=> array('tabla1', 'tabla2'),  // Arreglo de tablas para
respaldar.
    'ignore'=> array(), // Lista de tablas para omitir en la copia de
seguridad
    'format'=> 'txt', // gzip, zip, txt
    'filename'=>'mybackup.sql', //Nombre de archivo-NECESARIO SOLO CON
ARCHIVOS ZIP
    'add_drop' => TRUE, //Agregar o no la sentencia DROP TABLE al archivo de
respaldo
    'add_insert'  => TRUE, // Agregar o no datos de INSERT al archivo de
respaldo
    'newline'  => "\n" // Caracter de nueva línea usado en el archivo de
respaldo
    );

$this->dbutil->backup($prefs);
```

Descripción de Preferencias de Copia de Seguridad

Preferencia	Valor por Defecto	Opciones	Descripción
tables	arreglo vacío	Ninguna	Un arreglo de las tablas que desee respaldar. Si se deja vacío, todas las tablas serán exportadas.
ignore	arreglo vacío	Ninguna	Un arreglo de tablas que desee que la rutina de respaldo ignore.
format	gzip	gzip, zip, txt	El formato del archivo a exportar.
filename	fecha/hora actual	None	El nombre del archivo de respaldo. El nombre es necesario sólo si usa compresión zip.
add_drop	TRUE	TRUE/FALSE	Agregar o no la sentencia DROP TABLE en el archivo SQL exportado.
add_insert	TRUE	TRUE/FALSE	Agregar o no las sentencias de INSERT en el archivo SQL exportado.
newline	"\n"	"\n", "\r", "\r\n"	Tipo de nueva línea usado en el archivo SQL exportado.

Clase Email

La robusta clase Email de CodeIgniter soporta las siguientes características:

- Múltiple Protocols: Mail, Sendmail, y SMTP
- Varios Destinatarios
- CC y BCCs
- HTML o email de texto plano
- Adjuntos
- Ajuste de Linea(Word wrapping)
- Prioridades
- BCC Modo Batch, permitiendo grandes listas de correo electrónico que se divida en pequeños lotes BCC.
- Herramientas de depuración de Email

Enviando Correo Electrónico

Enviar un correo electrónico no es simple, pero puedes configurarlo 'al vuelo' o establecer tus preferencias en un archivo de configuración.

Aquí está un ejemplo básico demostrando como debes enviar un email. Nota: Este ejemplo asume que estas enviando un email desde uno de tus controladores.

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

$this->email->from('tu_direccion@tu_sitio.com', 'Tu nombre');
$this->email->to('alguien@ejemplo.com');
$this->email->cc('otro@otro-ejemplo.com');
$this->email->bcc('ellos@su-ejemplo.com');

$this->email->subject('Correo de Prueba');
$this->email->message('Probando la clase email');

$this->email->send();

echo $this->email->print_debugger();
```

Estableciendo Preferencias de Correo Electrónico

Hay 17 diferentes preferencias disponibles para adaptar la forma en la que se envían sus correos electrónicos. Puedes establecer cualquiera de ellos manualmente como se describe aquí, o automáticamente por las preferencias establecidas en tu archivo de configuración, descrita mas abajo:

Las preferencias son establecidas por el paso de valores de un arreglo de preferencias a la funcion *initialize(inicialización)* de email. Aqui tenemos un ejemplo de como debes establecer algunas preferencias:

```
$config['protocol'] = 'sendmail';
$config['mailpath'] = '/usr/sbin/sendmail';
$config['charset'] = 'iso-8859-1';
$config['wordwrap'] = TRUE;

$this->email->initialize($config);
```

Nota: La mayoría de las preferencias tienen un valor por defecto que será usado si no los estableces

Estableciendo preferencias de Email (Correo Electrónico) en un archivo de configuración

Si prefieres no establecer preferencias usando el método anterior, puedes ubicarlos en un archivo de configuración. Simplemente debes crear un nuevo archivo llamado *email.php*, agrega el arreglo *$config* en ese archivo. Luego guárdalo en *config/email.php* y el sera usado automáticamente. Tu NO necesitas usar la función *$this->email->initialize()* si guardas las preferencias en un archivo de configuración.

Email Preferences

La siguiente es una lista de todas las preferencias que pueden ser establecidas cuando se envia un e-mail.

Preferencia	Valor por defecto	Opciones	Descripción
useragent	CodeIgniter	None	El "user agent".
protocol	mail	mail, sendmail, or smtp	El protocolo en envio de email.
mailpath	/usr/sbin/sendmail	None	La ruta al Sendmail.
smtp_host	No Default	None	Dirección del servidor SMTP.
smtp_user	No Default	None	Usuario SMTP.
smtp_pass	No Default	None	Clave SMTP.
smtp_port	25	None	Puerto SMTP.
smtp_timeout	5	None	SMTP Timeout (en segundos).
wordwrap	TRUE	TRUE or FALSE (boolean)	Activar ajuste de linea.

wrapchars	76		Cantidad de caracteres para ajustar.
mailtype	text	text or html	Tipo de correo. Si envias un correo HTML debes enviarla como una pagina web completa. Asegurese de que no tiene links relativos o rutas de imagenes relativas, de otra forma no funcionaran.
charset	utf-8		Juego de Caracteres(utf-8, iso-8859-1, etc.).
validate	FALSE	TRUE or FALSE (boolean)	Si se validara la direccion del correo.
priority	3	1, 2, 3, 4, 5	Prioridad del email. 1 = Alta. 5 = Baja. 3 = Normal.
newline	\n	"\r\n" or "\n"	Caracter de nueva linea. (Usa "\r\n" para cumplir con la RFC 822).
bcc_batch_mode	FALSE	TRUE or FALSE (boolean)	Activa el Modo Batch BCC.
bcc_batch_size	200	None	Numero de emails en cada BCC batch

Referencia de funciones

$this->email->from()

Establece la dirección de email y el nombre de la persona que envía el email:

```
$this->email->from('tu@tu-sitio.com', 'Tu Nombre');
```

$this->email->reply_to()

Establece la direccion de respuesta. Si la información no es proveida, se usara de la funcion "from". Ejemplo::

```
$this->email->reply_to('tu@tu-sitio.com', 'Tu Nombre');
```

$this->email->to()

Establece la(s) direccion(es) de email del destinatario(s). Puede ser una direccion simple, una lista separada por comas o un arreglo:

```
$this->email->to('alguien@ejemplo.com');
```

```
$this->email->to('uno@ejemplo.com, dos@ejemplo.com, tres@ejemplo.com');
```

```
$list = array('uno@ejemplo.com', 'dos@ejemplo.com', 'tres@ejemplo.com');
$this->email->to($list);
```

$this->email->cc()

Estable las direcciones CC (con copia). Igual al "to", puede ser una dirección simple, o una lista separada por comas, o un arreglo.

$this->email->bcc()

Establece las direcciones BCC (con copia oculta). Es igual a la forma "to", puede ser una direccion simple, una lista delimitada por comas, o un array.

$this->email->subject()

Establece el asunto del email:

```
$this->email->subject('Este es el asunto');
```

$this->email->message()

Establece el cuerpo del mensaje:

```
$this->email->message('Este es mi mensaje');
```

$this->email->set_alt_message()

Establece un cuerpo de mensaje alternativo:

```
$this->email->set_alt_message('Este es el mensaje alternativo');
```

Este es una cadena de mensaje opcional que puede ser usado si envias un email con formato HTML. El te permite especificar un mensaje alternativo sin formato HTML el cual es adherido a la cadena de la cabecera. Para la gente que no acepta email con formato HTML. Si no estableces un mensaje alternativo, CodeIgniter extraera el mensaje de tu email HTML y quitara las etiquetas.

$this->email->clear()

Inicializa todas las variables de email a un estado vacio. Esta función esta hecha para ser usada en caso de que se use la funcion enviar email en un ciclo repetitivo (loop), permitiendo que los datos sean restablecidos entre los ciclos.

```
foreach ($lista as $nombre => $direccion
{
   $this->email->clear();
   $this->email->to($direccion);
   $this->email->from('tu@tu-sitio.com');
   $this->email->subject('Aqui tu informacion '.$nombre);
   $this->email->message('Hola '.$nombre.' Aqui esta la info que pediste.');
   $this->email->send();
}
```

Si estableces el parametro a TRUE cualquier adjunto sera limpiado también:

```
$this->email->clear(TRUE);
```

$this->email->send()

La función de envío de emails. Retorna el booleano TRUE o FALSE basado en el exito o el fracaso, permitiendo su uso condicionalmente:

```
if ( ! $this->email->send())
{
    // Generar error
}
```

$this->email->attach()

Te permite enviar un adjunto. Coloque el path/nombre en el primer parametro. Nota: Use un path al archivo, no una URL. Para multiples adjuntos use la función varias veces. Por ejemplo:

```
$this->email->attach('/path/to/photo1.jpg');
$this->email->attach('/path/to/photo2.jpg');
$this->email->attach('/path/to/photo3.jpg');

$this->email->send();
```

$this->email->print_debugger()

Retorna una cadena conteniendo cualquier mensaje del servidor, la cabecera del email, y el mensaje. Muy util para debugging.

Overriding Word Wrapping

Si tienes el ajuste de palabras (word wrapping) activado (recomendado para cumplir con la RFC 822) y tienes un enlace muy largo en tu email, este puede ser ajustado tambien, causando que se vuelva in-clickeable por la persona que lo reciba. CodeIgniter te permite manualmente sobre-escribir el (word wrapping) ajuste de palabra en una parte de tu mensaje como este:

```
El texto de tu email que
sera ajustado normalmente.

{unwrap}http://www.ejemplo.com/
un_largo_enlace_que_deberia_no_ser_ajustado.html{/unwrap}

Mas texto que luego deberia
ser ajustado normalmente.
```

Ubique el item que quieres que no sea ajustado entre :{unwrap} {/unwrap}

Clase de Encriptación

La clase de encriptacion provee dos formas de encriptacion. Usa un esquema que pre-compila el mensaje usando una randomly hashed bitwise XOR esquema de codificación, el cual es entonces encriptado usando la libreria Mcrypt. Si Mcrypt no esta disponible en el servidor el mensaje codificado todavia proveera aun un razonable grado de seguridad para sesiones encriptadas u otros de proposito "ligero". Si Mcrypt está disponible, efectivamente tendrás una cadena de mensaje doblemente encriptada, la cual provee un alto grado de seguridad.

Estableciendo tu Clave

Una Clave es una pieza de información que controla el proceso criptográfico y permite que una cadena encriptada sea decodificada. De hecho, la clave que elijas proveera el unico medio para decodificar el dato que has encriptado con esa clave, así no solamente debes elejir cuidadosamente la clave, nunca debes cambiarla si tiene la intención de usarala para para mantener los datos persistentes.

Y ni que decir de que debes ser cuidadoso de donde guardas la clave, pues si alguien obtiene acceso a la clave, las informaciones seran facilmente decodificadas. Si tu servidor no esta completamente bajo tu control es imposible asegurar la clave de seguridad, asi que deberás pensar cuidadosamente antes de usaala en algo que requiere alta seguridad, como almacenar numeros de tarjetas de credito.

Para tomar la maxima ventaja del algoritmo de encriptación, tu clave deberá ser de 32 caracteres de longitud (128 bits). La clave debe ser aleatoriamente una cadena que tu puedes concoct, con numeros y letras tanto mayusculas como minusculas. Tu claveno deberá ser una cadena de texto simple. De manera a que sea lo mas seguara criptograficamente hablando se necesita que la clave sea lo mas aleatoria posible.

Tu clave tampoco puede ser almacenada en tu application/config/config.php, o puedes diseñar tu propio mecanismo de almacenamiento y pasarla dinamicamente cuando codificas/decodificas.

Para guardar tu clave en tu application/config/config.php, abre el archivo y establece:

```
$config['encryption_key'] = "TU CLAVE";
```


Longitud del Mensaje

Es importante para usted saber que los mensajes codificados generan aproximadamente 2.6 veces la longitud del mensaje original Por ejemplo, si encriptas la cadena "mi super dato secret", cuya longitud es de 21 caracteres en longitud, terminaras con una cadena que es de aproximadamente 55 caracteres (decimos aproximadamente debido a que la cadena codificada incrementa en agrupaciones de 64 bit, por tanto no es necesariamente linear). Manteniendo esta información en mente cuando seleccione el mecanismo de almacenamietno. Los cookies, por ejemplo pueden mantener solamente 4K de información.

Inicializando la clase

Igual que en las otras clases en CodeIgniter, la clase Encryptation es inicializada en tu controlador usando la funcion: *$this->load->library*

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

Una vez cargada, el objeto Encrypt estara disponible usando: *$this->encrypt*

$this->encrypt->encode()

Realizando el cifrado de datos y lo devuelve como una cadena. Ejemplo:

```
$msg = 'Mi mensaje secreto';
$encrypted_string = $this->encrypt->encode($msg);
```

Opcionalmente puedes pasar tu clave de encriptación en el segundo parámetro si no quiere usar la clave que esta en tu archivo de configuración:

```
$msg = 'Mi mensaje secreto';
$key = 'clave-super-secreta';
$encrypted_string = $this->encrypt->encode($msg, $key);
```

$this->encrypt->decode()

Desencripta un mensaje encriptado. Ejemplo:

```
$encrypted_string = 'APANtByIGI1BpVXZTJgcsAG8GZl8pdwwa84';
$plaintext_string = $this->encrypt->decode($encrypted_string);
```

$this->encrypt->set_cipher();

Te permite establecer un cifrado Mcrypt. Por defeco usa MCRYPT_RIJNDAEL_256. Ejemplo:

```
$this->encrypt->set_cipher(MCRYPT_BLOWFISH);
```

Por favor visite php.net para tener un alista de cifradores disponibles.

Si prefieres probar manualmente si su servidor soporta Mcrypt puedes usar:

```
echo ( ! function_exists('mcrypt_encrypt')) ? 'Nop' : 'Sip';
```

$this->encrypt->set_mode();

Te permite establecer un modo MCRYPT. Por defecto usa MCRYPT_MODE_ECB. Ejemplo:

```
$this->encrypt->set_mode(MCRYPT_MODE_CFB);
```

Por favor visite php.net para tener un alista de modos disponibles.

$this->encrypt->sha1();

La funcion de codificacion SHA1. Provee una cadena y la retorna un hash de 160 bit de una sola forma. Nota: SHA1, es como el MD5 es no-decodificable. Ejemplo:

```
$hash = $this->encrypt->sha1('alguna cadena'');
```

Muchas instalaciones de PHP tienen soporte para SHA1 por defecto, por tanto lo unico que necesitas para codificar es usar la funcion nativa:

```
$hash = sha1('Alguna Cadena');
```

Si tu servidor no soporta SHA1 puedes usar la funcion proveida.

Clase de Carga de Archivos (Upload)

La Clase de Carga de Archivo de CodeIgniter le permite subir archivos. Puede establecer varias preferencias, restringir el tipo y tamaño de los archivos.

The Process

Subir un archivo involucra el siguiente proceso general:

- Un formulario de carga es mostrado, permitiendole al usuario seleccionar un archivo y subirlo.
- Cuando el formulario es enviado, el archivo es subido a la destinación que especifique.
- En el camino, el archivo es validado para estar seguro que es permitido basado en las preferencias que haya establecido.
- Una vez cargado, al usuario se le mostrará un mensaje de éxito.

Para demostrar este proceso aquí hay un pequeño tutorial. Luego encontrará información de referencia.

Creando el Formulario de Carga

Usando un editor de texto, crear un formulario llamado formulario_carga.php. En él, ubique este código y guardelo en su applications/views/ folder:

```
<html>
<head>
<title>Formulario de Carga de archivos</title>
</head>
<body>
<?php echo $error;?>
<?php echo form_open_multipart('upload/do_upload');?>
<input type="file" name="userfile" size="20" />
<br /><br />
<input type="submit" value="upload" />
</form>
</body>
</html>
```

Notará que estamos usando el asistente de formulario para crear la etiqueta de apertura de formulario. La carga de archivos requiere un formulario "multipart", así que el asistente crea la sintaxis apropiada para usted. También notará que hay un variable

$error. Esto es para que podamos mostrar mensajes de error en el caso que el usuario haga algo mal.

La página de éxito

Usando un editor de texto, debes crear un formulario llamado upload_success.php. En el, coloque este código y guárdelo dentro de su carpeta applications/views/:

```
<html>
<head>
<title>Formulario de carga de archivos</title>
</head>
<body>
<h3>Su archivo fue exitosamente subido!</h3>
<ul>
<?php foreach($upload_data as $item => $value):?>
<li><?php echo $item;?>: <?php echo $value;?></li>
<?php endforeach; ?>
</ul>
<p><?php echo anchor('upload', 'Subir otro archivo!'); ?></p>
</body>
</html>
```

El controlador

Usando un editor de texto, debe crear un controlador llamado upload.php. En el, coloque el siguiente código y guárdelo en su carpeta applications/controllers/:

```
<?php
class Upload extends Controller {

    function Upload()
    {
        parent::Controller();
        $this->load->helper(array('form', 'url'));
    }

    function index()
    {
        $this->load->view('upload_form', array('error' => ' ' ));
    }

    function do_upload()
    {
        $config['upload_path'] = './uploads/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size']   = '100';
        $config['max_width']  = '1024';
        $config['max_height']  = '768';

        $this->load->library('upload', $config);
```


```
        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);
        }
    }
}
?>
```

La carpeta Upload

Ud. necesitará una carpeta de destino para sus imágines subidas. Cree una carpeta en la raiz de su instalación CodeIgniter llamado uploads y establezca los permisos de archivo a 777.

Pruébelo!

Para probar su propio formulario, visite su sitio usando una URL similar a esta:

```
www.your-site.com/index.php/upload/
```

You should see an upload form. Try uploading an image file (either a jpg, gif, or png). If the path in your controller is correct it should work.

Reference Guide

Initializing the Upload Class

Like most other classes in CodeIgniter, the Upload class is initialized in your controller using the *$this->load->library* function:

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

Once the Upload class is loaded, the object will be available using: *$this->upload*

Setting Preferences

Similar to other libraries, you'll control what is allowed to be upload based on your preferences. In the controller you built above you set the following preferences:

```
$config['upload_path'] = './uploads/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size'] = '100';
$config['max_width'] = '1024';
$config['max_height'] = '768';

$this->load->library('upload', $config);

// Alternately you can set preferences by calling the initialize function.
Useful if you auto-load the class:
$this->upload->initialize($config);
```

The above preferences should be fairly self-explanatory. Below is a table describing all available preferences.

Preferences

The following preferences are available. The default value indicates what will be used if you do not specify that preference.

Preferencia	Valor por Defecto	Opciones	Descripción
upload_path	None	None	The path to the folder where the upload should be placed. The folder must be writable and the path can be absolute or relative.
allowed_types	None	None	The mime types corresponding to the types of files you allow to be uploaded. Usually the file extension can be used as the mime type. Separate multiple types with a pipe.
overwrite	FALSE	TRUE/FALSE (boolean)	If set to true, if a file with the same name as the one you are uploading exists, it will be overwritten. If set to false, a number will be appended to the filename if another with the same name exists.
max_size	0	None	The maximum size (in kilobytes) that the file can be. Set to zero for no limit. Note: Most PHP installations have their own limit, as specified in the php.ini file. Usually 2 MB (or 2048 KB) by default.
max_width	0	None	The maximum width (in pixels) that the file can be. Set to zero for no limit.
max_height	0	None	The maximum height (in pixels) that the file can be. Set to zero for no limit.

encrypt_name	FALSE	TRUE/FALSE (boolean)	If set to TRUE the file name will be converted to a random encrypted string. This can be useful if you would like the file saved with a name that can not be discerned by the person uploading it.
remove_spaces	TRUE	TRUE/FALSE (boolean)	If set to TRUE, any spaces in the file name will be converted to underscores. This is recommended.

Setting preferences in a config file

If you prefer not to set preferences using the above method, you can instead put them into a config file. Simply create a new file called the *upload.php*, add the *$config* array in that file. Then save the file in: *config/upload.php* and it will be used automatically. You will NOT need to use the *$this->upload->initialize* function if you save your preferences in a config file.

Function Reference

The following functions are available

```
$this->upload->do_upload()
```

Performs the upload based on the preferences you've set. Note: By default the upload routine expects the file to come from a form field called *userfile*, and the form must be a "multipart type:

```
<form method="post" action="some_action" enctype="multipart/form-data" />
```

If you would like to set your own field name simply pass its value to the *do_upload* function:

```
$field_name = "some_field_name";
$this->upload->do_upload($field_name)
```

$this->upload->display_errors()

Retrieves any error messages if the *do_upload()* function returned false. The function does not echo automatically, it returns the data so you can assign it however you need.

Formatting Errors

By default the above function wraps any errors within <p> tags. You can set your own delimiters like this:

```
$this->upload->display_errors('<p>', '</p>');
```

$this->upload->data()

This is a helper function that returns an array containing all of the data related to the file you uploaded. Here is the array prototype:

```
Array
(
    [file_name]    => mypic.jpg
    [file_type]    => image/jpeg
    [file_path]    => /path/to/your/upload/
    [full_path]    => /path/to/your/upload/jpg.jpg
    [raw_name] => mypic
    [orig_name]    => mypic.jpg
    [file_ext] => .jpg
    [file_size]    => 22.2
    [is_image] => 1
    [image_width]  => 800
    [image_height] => 600
    [image_type]   => jpeg
    [image_size_str] => width="800" height="200"
)
```

Explanation

Here is an explanation of the above array items.

Item	Description
file_name	The name of the file that was uploaded including the file extension.
file_type	The file's Mime type
file_path	The absolute server path to the file
full_path	The absolute server path including the file name
raw_name	The file name without the extension
orig_name	The original file name. This is only useful if you use the encrypted name option.
file_ext	The file extension with period
file_size	The file size in kilobytes

is_image	Whether the file is an image or not. 1 = image. 0 = not.
image_width	Image width.
image_heigth	Image height
image_type	Image type. Typically the file extension without the period.
image_size_str	A string containing the width and height. Useful to put into an image tag.

Clase FTP

La Clase FTP de CodeIgniter permite que los archivos sean transferidos a un servidor remoto. Los archivos remotos pueden también ser movidos, renombrados, y borrados. La Clase FTP también incluye una función de "espejado" que permite que permite a todo un directorio local ser recreado en forma remota a través de FTP.

> **Nota:** Los protocolos FTP SFTP y SSL no están soportados, solo FTP estándar.

Inicializando la Clase

Al igual que la mayoría de las otras clases en CodeIgniter, la clase FTP se inicializa en sus controlador usando la función *$this->load->library*:

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

Una vez cargado, el objeto FTP estará disponible utilizando: *$this->ftp*

Ejemplos de Uso

En este ejemplo, una conexión se abre con el servidor FTP, y un archivo local es leído y cargado en modo ASCII.Los permisos de archivo se establecen a 755. Nota: La configuración de permisos requiere PHP 5.

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

$config['hostname'] = 'ftp.your-site.com';
```

```
$config['username'] = 'your-username';
$config['password'] = 'your-password';
$config['debug'] = TRUE;

$this->ftp->connect($config);

$this->ftp->upload('/local/path/to/myfile.html', '/public_html/myfile.html',
'ascii', 0775);

$this->ftp->close();
```

En este ejemplo se recupera una lista de archivos desde el servidor.

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

$config['hostname'] = 'ftp.your-site.com';
$config['username'] = 'your-username';
$config['password'] = 'your-password';
$config['debug'] = TRUE;

$this->ftp->connect($config);

$list = $this->ftp->list_files('/public_html/');

print_r($list);

$this->ftp->close();
```

En este ejemplo, se refleja un directorio local en el servidor.

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

$config['hostname'] = 'ftp.your-site.com';
$config['username'] = 'your-username';
$config['password'] = 'your-password';
$config['debug'] = TRUE;

$this->ftp->connect($config);

$this->ftp->mirror('/path/to/myfolder/', '/public_html/myfolder/');

$this->ftp->close();
```


$this->ftp->connect()

Se conecta y registra en el servidor FTP. Las preferencias de conexión se establecen pasando un arreglo a la función, o almacenándolas en un archivo de configuración.

Aquí hay un ejemplo que muestra cómo establecer las preferencias manualmente:

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

$config['hostname'] = 'ftp.your-site.com';
$config['username'] = 'your-username';
$config['password'] = 'your-password';
$config['port']    = 21;
$config['passive']  = FALSE;
$config['debug']  = TRUE;

$this->ftp->connect($config);
```

Establacer las Preferencias de FTP en un Archivo de Configuracion

Si Ud. prefiere, puede almacenar sus preferencias de FTP en un archivo de configuración. Simplemente crea un nuevo archivo llamado *ftp.php*, agrega el arreglo *$config* en ese archivo. A continuación guarde el archivo en *config/ftp.php* y se usará automáticamente.

Opciones de Conexión Disponibles:

- **hostname** - el nombre del host FTP. Por lo general, algo como: *ftp.example.com*
- **username** - el nombre de usuario FTP.
- **password** - la contraseña FTP.
- **port** - el número de puerto. Establecido a *21* por defecto.
- **debug** - TRUE/FALSE (boolean). Ya sea para permitir o no la depuración para mostrar mensajes de error.
- **passive** - TRUE/FALSE (boolean). Ya sea para usar o no el modo pasivo. Pasivo se establece automáticamente por defecto.

$this->ftp->upload()

Carga un archivo a su servidor. Usted debe proporcionar la ruta local y la ruta remota, opcionalmente se puede establecer el modo y los permisos. Ejemplo:

```
$this->ftp->upload('/local/path/to/myfile.html', '/public_html/myfile.html',
'ascii', 0775);
```

Las opciones de Modo son: `ascii`, `binary`, and `auto` (el valor por defecto). Si `auto` es usado, será el modo base en la extensión de archivos del archivo fuente.

Los permisos están disponibles si está corriendo PHP 5 y se pueden pasar como un valor `octal` en el cuarto parámetro.

$this->ftp->rename()

Permite renombrar un archivo. Suministre el nombre/ruta del archivo fuente y el nombre/ruta del nuevo archivo.

```
// Renombra green.html a blue.html
$this->ftp->rename('/public_html/foo/green.html', '/public_html/foo/
blue.html');
```

$this->ftp->move()

Le permite mover un archivo. Suministre la ruta fuente y la de destino:

```
// Mueve blog.html desde "joe" a "fred"
$this->ftp->move('/public_html/joe/blog.html', '/public_html/fred/blog.html');
```

Nota: si el nombre del archivo destino es diferente el archivo será renombrado.

$this->ftp->delete_file()

Le permite borrar un archivo. Suministre la ruta fuente con el nombre del archivo.

```
$this->ftp->delete_file('/public_html/joe/blog.html');
```

$this->ftp->delete_dir()

Le permite borrar un directorio y todo lo que contiene. Suministre la ruta fuente al directorio con una barra diagonal.

Importante Tenga MUCHO CUIDADO con esta función. Ésta borrará recursivamente **todo** en la ruta suministrada, incluidas las sub-carpetas y todos los archivos. Hacer uso de la misma estando absolutamente seguro de que la ruta es correcta. Trate de usar la función list_files() en primer lugar para verificar que su ruta es correcta.

```
$this->ftp->delete_dir('/public_html/path/to/folder/');
```

$this->ftp->list_files()

Le permite obtener una lista de archivos de su servidor retornados como un *arreglo*. Usted debe proporcionar la ruta al directorio deseado.

```
$list = $this->ftp->list_files('/public_html/');

print_r($list);
```

$this->ftp->mirror()

Lee recursivamente una carpeta local y todo lo que contiene (incluidos las sub-carpetas) y crea un espejo a través de FTP basado en él. Cualquiera que sea la estructura de directorios de la ruta del archivo original será recreado en el servidor. Ud. debe suministrar una ruta fuente y una ruta destino:

```
$this->ftp->mirror('/path/to/myfolder/', '/public_html/myfolder/');
```

$this->ftp->mkdir()

Le permite crear un directorio en su servidor. Suministre la ruta final en el nombre de la carpetaque desea crear, con una barra diagonal. Los permisos pueden ser establecidos pasando un valor `octal` en el segundo parámetro (si está ejecutando PHP 5).

```
// Crea una carpeta llamada "bar"
$this->ftp->mkdir('/public_html/foo/bar/', 0777);
```

$this->ftp->chmod()

Le permite establecer permisos de archivo. Suministre la ruta del archivo o carpeta que desea modificar los permisos en:

```
// Chmod "bar" a 777
$this->ftp->chmod('/public_html/foo/bar/', 0777);
```

$this->ftp->close();

Cierra la conexión con el servidor. Es recomendable que utilice esto cuando haya terminado de cargar.

Clase de Tabla HTML

La Clase de Tabla provee funciones que le permite autogenerar una tabla HTML desde arreglos o juegos de resultados de base de datos.

Inicializando la Clase

Como la mayoría de las otras clases en CodeIgniter, la clase de Tabla es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez cargada, el objeto de la librería de Tabla estará disponible usando *$this->table*

Ejemplos

Aquí hay un ejemplo mostrando como puede crear una tabla usando arreglos multidimensionales. Note que el primer índice del arreglo será el encabezado de la tabla (o puede establecer sus propios encabezazdos usando la función *set_heading()* descrita en la referencia de funciones debajo).

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

$data = array(
    array('Nombre', 'Color', 'Tamaño'),
    array('Fred', 'Azul', 'Pequeño'),
    array('Mary', 'Rojo', 'Grande'),
    array('John', 'Verde', 'Mediano')
    );

echo $this->table->generate($data);
```

Aquí hay un ejemplo de una tabla creada desde un resultado de una consulta a la base de datos. La clase de tabla generará los encabezados basados en el nombre de la tabla (o puede establecer sus propios encabezados usando la función *set_heading()* descrita en la referencia de funciones debajo).

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

$consulta = $this->db->query("SELECT * FROM my_table");

echo $this->table->generate($consulta);
```

Aquí hay un ejemplo mostrando como puede crear una tabla usando parámetros discretos:

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

$this->table->set_heading('Nombre', 'Color', 'Tamaño');

$this->table->add_row('Fred', 'Azul', 'Pequeño');
$this->table->add_row('Mary', 'Rojo', 'Grande');
$this->table->add_row('John', 'Verde', 'Mediano');

echo $this->table->generate();
```

Aquí está el mismo ejemplo, excepto que en vez de parámetros individuales, se usan arreglos:

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

$this->table->set_heading(array('Nombre', 'Color', 'Tamaño'));

$this->table->add_row(array('Fred', 'Azul', 'Pequeño'));
$this->table->add_row(array('Mary', 'Rojo', 'Grande'));
$this->table->add_row(array('John', 'Verde', 'Mediano'));

echo $this->table->generate();
```

La Clase de Tabla le permite establecer una plantilla de tabla con la cual puede especificar el diseño. Aquí hay un prototipo de plantilla:

```
$plantilla = 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>'
   );

$this->table->set_template($plantilla);
```

Nota: Notará que hay dos juegos de bloques de "fila" en la plantilla. Esto le permite crear alternativamente colores de fila o diseños de elementos que alternen en cada iteración de los datos de fila.

NO requiere que envíe un template completo. Si sólo necesita cambiar partes del diseño puede simplemente enviar esos elementos. En este ejemplo, sólo la etiqueta de apertura de tabla es cambiada:

```
$plantilla = array ( 'table_open'  => '<table border="1" cellpadding="2"
cellspacing="1" class="mytable">' );

$this->table->set_template($plantilla);
```


Referencia de funciones

$this->table->generate()

Devuelve una cadena que contiene la tabla generada. Acepta un parámetro opcional que puede ser un arreglo o un objeto de resultado de base de datos.

$this->table->set_caption()

Permite agregar un "caption" a la tabla.

```
$this->table->set_caption('Colores');
```

$this->table->set_heading()

Permite establecer el encabezado de la tabla. Puede enviar un arreglo o parámetros discretos:

```
$this->table->set_heading('Nombre', 'Color', 'Tamaño'); $this->table-
>set_heading(array('Nombre', 'Color', 'Tamaño'));
```

$this->table->add_row()

Le permite agregar una fila a su tabla. Puede enviar un arreglo o parámetros discretos:

```
$this->table->add_row('Azul', 'Rojo', 'Verde'); $this->table-
>add_row(array('Azul', 'Rojo', 'Verde'));
```

$this->table->make_columns()

Esta función toma un arreglo unidimensional como entrada y crea un arreglo multidimensional con una profundidad igual al número de columnas deseadas. Esto permite a un simple arreglo con varios elementos ser mostrado en una tabla que tiene una cantidad fija de columnas. Considere este ejemplo:

```
$lista = array('uno', 'dos', 'tres', 'cuatro', 'cinco', 'seis', 'siete',
ocho', 'nueve', 'diez', 'once', 'doce');

$nueva_lista = $this->table->make_columns($lista, 3);

$this->table->generate($nueva_lista);

// Genera una tabla con este prototipo

<table border="0" cellpadding="4" cellspacing="0">
<tr>
<td>uno</td><td>dos</td><td>tres</td>
</tr><tr>
<td>cuatro</td><td>cinco</td><td>seis</td>
</tr><tr>
<td>siete</td><td>ocho</td><td>nueve</td>
</tr><tr>
<td>diez</td><td>once</td><td>doce</td></tr>
</table>
```

$this->table->set_template()

Le permite establece su plantilla. Puede enviar una plantilla total o parcial.

```
$plantilla = array ( 'table_open'  => '<table border="1" cellpadding="2"
cellspacing="1" class="mytable">' );

$this->table->set_template($plantilla);
```

$this->table->set_empty()

Le permite establecer un valor por defecto para usar en cualquier celda de la tabla que esté vacía. Puede, por ejemplo, establecer un espacio sin quiebre (non-breaking space):

$this->table->set_empty(" ");

$this->table->clear()

Le permite limpiar el encabezado y filas de datos. Si necesita mostrar múltiples tablas con diferentes datos debería llamar a esta función después de que cada tabla ha sido generada para vaciar la información de la tabla previa. Ejemplo:

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

$this->table->set_heading('Nombre', 'Color', 'Tamaño');
```


```
$this->table->add_row('Fred', 'Azul', 'Pequeño');
$this->table->add_row('Mary', 'Rojo', 'Grande');
$this->table->add_row('John', 'Verde', 'Mediano');

echo $this->table->generate();

$this->table->clear();

$this->table->set_heading('Nombre', 'Día', 'Envío');
$this->table->add_row('Fred', 'Miércoles', 'Expreso');
$this->table->add_row('Mary', 'Lunes', 'Aéreo');
$this->table->add_row('John', 'Sábado', 'De la noche a la mañana');

echo $this->table->generate();
```

Clase de Manipulación de Imagen

La clase de Manipulación de Imagen de CodeIgniter le permite realizar las siguientes acciones:

- Redimención de Imagen
- Creación de Thumbnail
- Recorte de Imagen
- Rotación de Imagen
- Marca de Agua en Imagen

Todas las tres librerías de imágenes principales son soportadas: GD/GD2, NetPBM, e ImageMagick

> **Nota:** Marca de Agua sólo está disponible usando la librería GD/GD2. Además, aunque las otras librerías son soportadas, GD es requerida para que el programa calcule las propiedades de la imagen. El procesamiento de la imagen, sin embargo, será realizado con la librería que haya especificado.

Inicializando la Clase

Como la mayoría de las otras clases en CodeIgniter, la clase de imagen es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez que la librería es cargada estará lista para el uso. El objeto de la librería de imagen que usará para llamar todas las funciones es: *$this->image_lib*

Procesando una Imagen

Sin importar el tipo de procesamiento que quiera realizar (redimencionar, recortar, rotar o marca de agua), el proceso general es idéntico. Establecerá algunas preferencias correspondientes a la acción que tenga intenciones de realizar, entonces llamará a una de las cuatro funciones de procesamiento disponibles. Por ejemplo, para crear un thumbnail de una imagen hará esto:

```
$config['image_library'] = 'GD';
$config['source_image'] = '/ruta/a/imagen/mifoto.jpg';
$config['create_thumb'] = TRUE;
$config['maintain_ratio'] = TRUE;
$config['width'] = 75;
$config['height'] = 50;

$this->load->library('image_lib', $config);

$this->image_lib->resize();
```

El código anterior le dice a la función *image_resize* que busque por una imagen llamada *mifoto.jpg* ubicada en la carpeta *source_image*, entonces cree un thumbnail que sea de 75 X 50 pixeles usando la *image_library* GD2. Ya que la opción *maintain_ratio* está habilitada, el thumb será tan cercano al *width* y *height* de destino como sea posible mientras preserve la proporción del aspecto original. El thumbnail será llamado *mifoto_thumb.jpg*

> **Nota:** Para que la clase de imagen pueda hacer algún procesamiento, la carpeta contenedora de los archivos de imagen debe tener permisos de 777.

Funciones de Procesamiento

Hay cuatro funciones de procesamiento:

- $this->image_lib->resize()
- $this->image_lib->crop()
- $this->image_lib->rotate()
- $this->image_lib->watermark()
- $this->image_lib->clear()

Estas funciones devuelve un buleano TRUE si funcionan y FALSE si fallan. Si fallan puede recuperar el mensaje de error usando esta función:

```
echo $this->image_lib->display_errors();
```

Una buena costumbre es usar la función de procesamiento condicionalmente, mostrando un error cuando falla, así:

```
if ( ! $this->image_lib->resize())
{
    echo $this->image_lib->display_errors();
}
```

Nota: Puede opcionalmente especificar el formato HTML a ser aplicado a los errores, enviando las etiquetas de apertura/clausura en la función, así:

```
$this->image_lib->display_errors('<p>', '</p>');
```

Preferencias

Las 14 preferencias disponibles son descritas debajo permitiendole utilizar el procesamiento de imagenes de forma tal que se adecue a sus necesidades.

Note que no todas las preferencias están disponibles para cada función. Por ejemplo, las preferencias de hachas x/y sólo están disponibles para recortar imagen. De la misma manera, Las preferencias de ancho y alto no tienen efecto al recortar. La columna "disponibilidad" indica que función soporta una preferencia determinada.

Leyendas Disponibles:

- *R* - Redimensionar Imagen
- *C* - Recortar Imagen
- *X* - Rotar Imagen
- *W* - Marca de Agua

Preferencia	Valor por Defecto	Opciones	Descripción
image_library	GD2	GD, GD2, ImageMagick, NetPBM	Establece la librería de imagen a ser usada. R, C, X, W
library_path	Ninguna	Ninguna	Establece la ruta del servidor a su librería ImageMagick o NetPBM. Si usa una de estas librerías debe suministrar la ruta. R, C, X
source_image	Ninguna	Ninguna	Establece el nombre/ruta de la imagen de origen. La ruta debe ser una ruta relativa o absoluta del servidor, no una URL. R, C, S, W
dynamic_output	FALSE	TRUE/FALSE (buleano)	Determina si el nuevo archivo de imagen debe ser escrito a disco o generado dinámicamente. Nota: Si elije la configuración dinámica, sólo una imagen puede ser mostrada a la vez, y no puede ser posicionada en la página. Simplemente muestra la imagen dinámicamente a su explorador, junto con los encabezados de imagen. R, C, X, W
quality	90%	1 - 100%	Establece la calidad de la imagen. Mayor calidad implica mayor tamaño del archivo. R, C, X, W
new_image	Ninguna	Ninguna	Establece el nombre/ruta de destino de la imagen. Usará esta preferencia cuando cree una copia de una imagen. La ruta debe ser una ruta relativa o absoluta del servidor, no una URL. R, C, X, W
width	Ninguna	Ninguna	Establece el ancho que quiere establecerle a la imagen. R, C
height	Ninguna	Ninguna	Establece el alto que quiere establecerle a la imagen. R, C
create_thumb	FALSE	TRUE/FALSE (boolean)	Le dice a la función de procesamiento de imagen que cree un thumb. R
thumb_marker	_thumb	Ninguna	Especifica el indicador de thumbnail. Será insertado justo antes de la extensión del archivom así que mifoto.jpg será mifoto_thumb.jpg R
maintain_ratio	TRUE	TRUE/FALSE (boolean)	Especifica si mantener la proporción del aspecto original cuando redimensiona o usar valores duros. R
master_dim	auto	auto, width, height	Especifica que usar como la hacha maestra cuando redimensiona o crea thumbs. Por ejemplo, digamos que quiere redimensionar una imagen a 100 X 75 pixeles. Si su imagen de origen no permite redimensionamiento perfecto a esas dimensiones, esta configuración determina que hacha debe ser usada como valor duro. "auto" establece el hacha automáticamente basado en si la imagen es más alta que ancha o viceversa. R

rotation_angle	Ninguna	90, 180, 270, vrt, hor	Especifica el ángulo de rotación cuando se rotan imagenes. Note que PHP rota contra las agujas del reloj, así que una rotación de 90 grados a la derecha debe ser especificada como 270. X
x_axis	Ninguna	Ninguna	Establece la cordenada X en pixeles para recortar una imagen. Por ejemplo, una configuración de 30 cortará una imagen 30 pixeles desde la izquierda. C
y_axis	Ninguna	Ninguna	Establece la cordenada X en pixeles para recortar una imagen. Por ejemplo, una configuración de 30 cortará una imagen 30 pixeles desde arriba. C

Estableciendo preferencias en un archivo de configuración

Si prefiere no establecer preferencias usando los métodos anteriores, puede en su lugar ponerlos en un archivo de configuración. Simplemente cree un nuevo archivo llamado *image_lib.php* y el arreglo *$config* en ese archivo. Entonces guarde el archivo en: *config/image_lib.php* y será usado automáticamente. NO necesitará usar la función *$this->image_lib->initialize* si guarda sus preferencias en un archivo de configuración.

```
$this->image_lib->resize()
```

La función de redimensión de imagen le permite redimensionar la imagen original, crear una copia (con o sin redimensión), o crear una imagen thumbnail.

Por propósitos prácticos no hay diferencia entre crear una copia y crear un thumbnail excepto que un thumb tendrá el marcador thumbnail como parte de su nombre (por ejemplo, mifoto_thumb.jpg).

Todas las preferencias listadas en la tabla anterior están disponibles por esta función excepto estas tres: rotation, x_axis, and y_axis.

Creando un Thumbnail

La función de redimensión creará un archivo thumbnail (y preserva el original) si establece esta preferencia como TRUE:

```
$config['create_thumb'] = TRUE;
```

Esta preferencia sólo determina un thumbnail es creado o no.

Creando una Copia

La función de redimensión creará una copia del archivo de imagen (y preservará el original) su establece una ruta y/o un nuevo nombre de archivo usando esta preferencia:

```
$config['new_image'] = '/ruta/a/nueva_imagen.jpg';
```

Nota acerca de esta preferencia:

- Si sólo el nombre de la nueva imagen es especificado será ubicada en la misma carpeta que el original
- Si sólo la ruta es especificada, la nueva imagen será ubicada en el destino con el mismo nombre que el original.
- Si ambos, la ruta y nombre de la imagen, son especificados será ubicado en su propia destinación y con el nuevo nombre dado.

Redimensionando la Imagen Original

Si ninguno de las dos preferencias listadas arriba (create_thumb, y new_image) son usadas, la función de redimensión usará la imagen original como destino de procesamiento.

$this->image_lib->crop()

La función de recortar funciona casi idénticamente a la función de redimensión excepto que requiere que establezca preferencias para las hachas X e Y (en pixeles) especificando donde cortar, así:

```
$config['x_axis'] = '100';
$config['y_axis'] = '40';
```

Todas las preferencias listadas en la tabla anterior están disponibles excepto estas: rotation, width, height, create_thumb, new_image.

Aquí hay un ejemplo mostrando como puede recortar una imagen:

```
$config['image_library'] = 'imagemagick';
$config['library_path'] = '/usr/X11R6/bin/';
$config['source_image'] = '/ruta/a/imagen/mifoto.jpg';
$config['x_axis'] = '100';
$config['y_axis'] = '60';
```


```
$this->image_lib->initialize($config);

if ( ! $this->image_lib->crop())
{
    echo $this->image_lib->display_errors();
}
```

Nota: Sin una interface visual es difícil recortar imagenes, así que esta función no es muy útil a menos que tenga intenciones de construir dicha interface. Esto es exactamente lo que hicimos para el módulo de galería de fotos en ExpressionEngine, el CMS que desarrollamos. Agregamos una Interface de Usuario que permite que el area a recortar sea seleccionada.

$this->image_lib->rotate()

La función de rotación de imagen requiere que el ángulo de rotación sea establecido a través de esta preferencia:

```
$config['rotation_angle'] = '90';
```

Hay 5 opciones de rotación:

1. 90 - rota 90 grados contra las agujas del reloj.
2. 180 - rota 180 grados contra las agujas del reloj.
3. 270 - rota 270 grados contra las agujas del reloj..
4. hor - Voltea la imagen horizontalmente.
5. vrt - Voltea la imagen verticalmente.

Aquí hay un ejemplo mostrando como puede rotar una imagen:

```
$config['image_library'] = 'netpbm';
$config['library_path'] = '/usr/bin/';
$config['source_image'] = '/ruta/a/imagen/mifoto.jpg';
$config['rotation_angle'] = 'hor';

$this->image_lib->initialize($config);

if ( ! $this->image_lib->rotate())
{
    echo $this->image_lib->display_errors();
}
```

$this->image_lib->clear()

La función "clear" restablece todos los valores por defecto usados cuando se procesa una imagen. Usted podría querer llamarla si está procesando imágenes en un ciclo.

```
$this->image_lib->clear();
```

Marca de Agua

La característica Marca de Agua requiere la librería GD/GD2.

Dos tipos de Marca de Agua

Hay dos tipos de marca de agua que puede usar:

Texto: El mensaje de marca de agua será generado usando texto, ya sea con una fuente True Type que especifique, o usando la salida de texto nativo que la librería GD soporta. Si usa la versión True Type, su instalación de GD debe ser compilada con soporte True Type (la mayoría lo son, pero no todos).

Revestimiento: El mensaje de marca de agua será generado al revestir la imagen (usualmente un PNG o GIF transparent) conteniendo su marca de agua sobre la imagen de origen.

Marca de agua en una imagen

Al igual que con la otra funciones el procesamiento general para marca de agua involucra establecer las preferencias correspondientes a la acción que intente realizar, entonces llamar a la función "watermark". Aquí hay un ejemplo:

```
$config['source_image'] = '/ruta/a/imagen/mifoto.jpg';
$config['wm_text'] = 'Copyright 2006 - John Doe';
$config['wm_type'] = 'text';
$config['wm_font_path'] = './system/fonts/texb.ttf';
$config['wm_font_size'] = '16';
$config['wm_font_color'] = 'ffffff';
$config['wm_vrt_alignment'] = 'bottom';
$config['wm_hor_alignment'] = 'center';
$config['wm_padding'] = '20';
$this->image_lib->initialize($config);
$this->image_lib->watermark();
```


El ejemplo anterior usará una fuente de 16 píxeles para crear el texto "Copyright 2006 - John Doe". La marca de agua será posicionada abajo/al centro de la imagen, 20 píxeles desde el fondo de la imagen.

> **Nota:** Para que la clase de imagen pueda hacer cualquier procesamiento, el archivo de imagen debe tener permisos de 777.

Preferencias de Marca de Agua

Esta tabla muestra las preferencias que están disponibles para ambos tipos de marcas de agua (texto o revestimiento)

Preferencia	Valor por Defecto	Opciones	Descripción
wm_type	text	type, overlay	Establece el tipo de marca de agua que debe ser usado.
source_image	Ninguna	Ninguna	Establece el nombre/ruta de la imagen de origen. La ruta debe ser relativa o una ruta absoluta del servidor, no una URL.
dynamic_output	FALSE	TRUE/FALSE (boolean)	Determina si el nuevo archivo de imagen debe ser escrito a disco o generado dinámicamente. Nota: Si elije la configuración dinámica, sólo una imagen puede ser mostrada a la vez, y no puede ser posicionada en la página. Simplemente muestra la imagen dinámicamente a su explorador, junto con los encabezados de imagen.
quality	90%	1 - 100%	Establece la calidad de la imagen. Mientras más alta sea la calidad, mayor será el tamaño del archivo.
padding	Ninguna	Un número	La cantidad de espaciado, establecido en píxeles, que serán aplicados a la marca de agua para establecer la distancia desde el borde de sus imágenes.
wm_vrt_alignment	bottom	top, middle, bottom	Establece el alineamiento vertical para la marca de agua de la imagen.
wm_hor_alignment	center	left, center, right	Establece el alineamiento horizontal para la marca de agua de la imagen.
wm_vrt_offset	Ninguna	Ninguna	You may specify a horizontal offset (in pixels) to apply to the watermark position. The offset normally moves the watermark to the right, except if you have your alignment set to "right" then your offset value will move the watermark toward the left of the image.
wm_hor_offset	Ninguna	Ninguna	You may specify a horizontal offset (in pixels) to apply to the watermark position. The offset normally moves the watermark down, except if you have your alignment set to "bottom" then your offset value will move the watermark toward the top of the image.

Text Preferences

This table shown the preferences that are available for the text type of watermarking.

Preferencia	Valor por Defecto	Opciones	Descripción
wm_text	Ninguna	Ninguna	The text you would like shown as the watermark. Typically this will be a copyright notice.
wm_font_path	Ninguna	Ninguna	The server path to the True Type Font you would like to use. If you do not use this option, the native GD font will be used.
wm_font_size	16	Ninguna	The size of the text. Note: If you are not using the True Type option above, the number is set using a range of 1 - 5. Otherwise, you can use any valid pixel size for the font you're using.
wm_font_color	ffffff	Ninguna	The font color, specified in hex. Note, you must use the full 6 character hex value (ie, 993300), rather than the three character abbreviated version (ie fff).
wm_shadow_color	Ninguna	Ninguna	The color of the drop shadow, specified in hex. If you leave this blank a drop shadow will not be used. Note, you must use the full 6 character hex value (ie, 993300), rather than the three character abbreviated version (ie fff).
wm_shadow_distance		3	Ninguna The distance (in pixels) from the font that the drop shadow should appear.

Overlay Preferences

This table shown the preferences that are available for the overlay type of watermarking.

Preferencia	Valor por Defecto	Opciones	Descripción
wm_overlay_path	Ninguna	Ninguna	The server path to the image you wish to use as your watermark. Required only if you are using the overlay method.
wm_opacity	50	1 - 100	Image opacity. You may specify the opacity (i.e. transparency) of your watermark image. This allows the watermark to be faint and not completely obscure the details from the original image behind it. A 50% opacity is typical.
wm_x_transp	4	A number	If your watermark image is a PNG or GIF image, you may specify a color on the image to be "transparent". This setting (along with the next) will allow you to specify that color. This works by specifying the "X" and "Y" coordinate

			pixel (measured from the upper left) within the image that corresponds to a pixel representative of the color you want to be transparent.
wm_y_transp	4	A number	Along with the previous setting, this allows you to specify the coordinate to a pixel representative of the color you want to be transparent.

Clase de Entrada (Input)

La Clase de Entrada sirve para dos propósitos:

1. Pre-procesa datos de entrada global por seguridad.
2. Provee alguna funciones auxiliares para recuperar datos de entradas y pre-procesarlos.

Nota: Esta clase es inicializada automáticamente por el sistema, así que no hay necesidad de que lo haga manualmente.

Filtros de Seguridad

La función de filtro de seguridad es llamada automáticamente cuando un nuevo controlador es invocado. Hace lo siguiente:

- Destruye todo el arreglo global GET. Ya que CodeIgniter no utiliza cadenas GET, no hay razón para permitirla.
- Destruye todas las variables globales en el caso de que register_globals esté habilitado.
- Filtra las claves de POST/COOKIE, permitiendo sólo caracteres alfanuméricos (y algunos pocos más).
- Provee filtro XSS (Cross-site Scripting Hacks). Esto puede ser habilitado globalmente, o a pedido.
- Estandariza las caracteres de nueva línea a \n

Filtro XSS

CodeIgniter viene con un filtro de prevención de Ataques Cross Site Scripting el cual puede ser corrido automáticmente para filtrar todos los datos POST y COOKIE que se

encuentren, o puede correrlo en una base por ítem. Por defécto **no** corre globalmente ya que requiere un poco de sobrecarga de procesamiento, y además puede no ser necesario en todos los casos.

El filtro XSS busca técnicas comunes usadas para activar Javascript u otros tipos de código que intenten asaltar cookies o hacer otras cosas maliciosas. Si cualquier cosa no permitida es encontrada es mostrada de forma seguro convirtiendo los datos en caracteres de entidad.

Nota: Esta función sólo debe ser usada para manejarse con datos enviadas. No es algo que quiera usar para un procesamiento general de rutina ya que requiere una buena cantidad de sobrecarga de procesamiento.

Para filtrar datos a través del filtro XSS use esta función:

$this->input->xss_clean()

Aquí hay un ejemplo de uso:

```
$datos = $this->input->xss_clean($datos);
```

Si quiere que el filtro se ejecute automáticamente cada vez que encuentr datos POST o COOKIE puede habilitarlo abriendo su archivo `application/config/config.php` y estableciendo esto:

```
$config['global_xss_filtering'] = TRUE;
```

Nota: Si usa la clase de validación de formularios, esta ofrece la opción de filtro XSS también.

Usando Datos POST, COOKIE, or SERVER

CodeIgniter viene con tres funciones asistentes que le permiter recuperar ítems POST, COOKIE or SERVER. La principal ventaja de usar las funciones provistas en vez de traer el ítem directamente ($_POST['algo']) es que la función verificará si el ítem existe y devuelve false (buleano) si no. Esto le permite convenientemente usar datos sin tener que probar si un ítem existe primero. En otras palabras, normalmente puede hacer algo así:

```
if ( ! isset($_POST['algo']))
{
    $algo = FALSE;
}
else
{
    $algo = $_POST['algo'];
}
```

Con las funciones de CodeIgniter puede simplemente hacer esto:

$algo = $this->input->post('algo');

Las tres funciones son:

- $this->input->post()
- $this->input->cookie()
- $this->input->server()

$this->input->post()

El primer parámetro contendrá el nombre del ítem POST que está buscando:

```
$this->input->post('algun_dato');
```

La función devuelve FALSE (buleano) si el ítem que intenta recuperar no existe.

El segundo opcional parámetro le permite correr los datos a través del filtro XSS. Es habilitada al establecer el segundo parámetro como un buleano TRUE;

```
$this->input->post('algun_dato', TRUE);
```

$this->input->cookie()

Esta función es idéntica a la función post, sólo que recupera datos de cookie:

```
$this->input->cookie('algun_dato', TRUE);
```

$this->input->server()

Esta función es idéntica a las funciones anteriores, sólo que recupera datos del servidor:

```
$this->input->server('algun_dato');
```

$this->input->ip_address()

Devuelve la dirección de IP del usuario actual. Si la dirección IP no es válida, la función devolverá un IP de: 0.0.0.0

```
echo $this->input->ip_address();
```

$this->input->valid_ip(*$ip*)

Toma una dirección de IP como entrada y devuelve TRUE o FALSE (buleano) si es válida o no. Nota: La función $this->input->ip_address() de arriba valida el IP automáticamente.

```
if ( ! $this->input->valid_ip($ip))
{
     echo 'Not Valid';
}
else
{
     echo 'Valid';
}
```

$this->input->user_agent()

Devuelve el agente del usuario (explorador web) en uso por el usuario actual. Devuelve FALSE si no está disponible.

```
echo $this->input->user_agent();
```


Clase de Carga

El cargador, como el nombre sugiere, es usado para cargar elementos. Estos elementos pueden ser librerías (clases), Archivos de Vista, Asistentes, Plugins, o sus propios archivos.

> **Nota:** Esta clase es inicializada automáticmente por el sistema, así que no necesita hacerlo manualmente.

Las siguientes funciones están disponibles en esta clase:

$this->load->library('*nombre_de_clase*')

Esta función es usada para cargar clases de núcelo. Donde *nombre_de_clase* es el nombre de la clase que quiere cargar. Nota: Usamos los términos "clase" y "librería" intercambiablemente.

Por ejemplo, si quiere enviar un email con CodeIgniter, el primer paso es cargar la clase de email dentro de su controlador:

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

Una vez cargado, la librería estará lista para ser usada, usando `$this->email->alguna_funcion()`. Cada librería es descripta en detalle en su propia página, así que por favor lea la información acerca de cada una que desee usar.

Los parámetros pueden ser pasados a la librería a través de un arreglo en el segundo parámetro.

$this->load->view('*nombre_archivo*', $data, true/false)

Esta función es usada para cargar sus archivos Vista. Si no hay leído la sección Vistas de la guía del usuario, es recomendable que lo haga ya que muestra como esta función es típicamente usada.

El primer parámetro es requerido. Es el nombre del archivo vista que quiere cargar. Nota: La extensión el archivo .php no necesita ser especificada a menos que use alguna distinta que `.php`.

El segundo parámetro **opcional** puede tomar un arreglo asociativo o un objeto como entrada, el cual corre a través de la función de PHP extract para convertirlo en variables que puedan ser usadas en su archivo de vistas. De nuevo, lea la página Vistas para aprender como esto puede ser útil.

El tercer parámetro **opcional** le permite cambiar el comportamiento de la función para que devuelva los datos como una cadena en vez de enviarlo a su explorador. Esto puede ser útil si quiere procesar los datos de alguna manera. Si establece el parámetro como `true` (buleano) devolverá datos. El comportamiento por defecto es `false`, el cual lo envía a su explorador. Recuerde asignarlo a una variable si quiere que los datos sean devueltos:

```
$cadena = $this->load->view('miarchivo', '', true);
```

$this->load->database('*opciones*', true/false)

Esta función le permite cargar su clase de base de datos. Los dos parámetros son **opcionales**. Por favor lea la sección base de datos para más información.

$this->load->scaffolding('*nombre_tabla*')

Esta función le permite habilitar scaffolding. Por favor vea la sección de scaffolding para más información.

```
$this->load->vars($arreglo)
```

Esta función toma un arreglo asociativo como entrada y genera variables usando la función de PHP extract. Esta función produce el mismo resultado que usar el segundo parámetro de la función *$this->load->view()* anterior. La razón por la que puede querer usar esta función independientemente es si quiere establecer algunas variables globales en el constructor de su controlador y tenerlas disponibles en cualquier archivo de vista desde cualquier función. Puede tener múltiples llamadas a esta función. Los datos son guardados y mezclados en un sólo arreglo para convertir en variables

$this->load->helper('*nombre_archivo*')

Esta función carga archivos asistentes, donde *nombre_archivo* es el nombre del archivo, sin la extensión `_helper.php`.

$this->load->plugin('*nombre_archivo*')

Esta función carga archivos plugin, donde *nombre_archivo* es el nombre del archivo sin la extensión `_plugin.php`.

$this->load->file('*directorio/archivo*', true/false)

Esta es una función de carga de archivos genérica. Suministre la ruta y el nombre en el primer parámetro y abrirá y leerá el archivo. Por defecto, los datos son enviados a su explorador, como un archivo Vista, pero si establece el segundo parámetro como `true` (buleano) devolverá los datos como una cadena.

$this->load->lang('*nombre_archivo*')

Esta función es un alias de la función de carga de lenguaje: $this->lang->load()

$this->load->config('*nombre_archivo*')

Esta función es un alias de la función de carga de archivos de configuración: $this->config->load()

Clase Idioma

La Clase Idioma proporciona funciones para recuperar los archivos de idioma y las líneas de texto a los efectos de la internacionalización.

En su carpeta "system" de CodeIgniter encontrará una llamada *language* que contendrá conjuntos de archivos de idiomas. Usted puede crear su propio archivo de idiomas cuando sea necesario, a fin de mostrar mensajes de error y otros mensajes en otros idiomas.

Los archivos de idioma estan comúnmente almacenados en su directorio *system/language*. Alternativamente puede crear una carpeta llamada `language` dentro de su carpeta `application` y almacenarlos en ella. CodeIgniter verá primero en su directorio *system/application/language*. Si el directorio no existe o el idioma especificado no está ubicado allí, CI mirará en su carpeta global *system/language*.

Nota: Cada idioma debe ser almacenado en su propia carpeta. Por ejemplo, los archivos en Inglés estan localizados en: *system/language/english*

Creando Archivos de Idioma

Los archivos de Idioma deben ser nombrados con `_lang.php` como extensión del archivo. Por ejemplo, supongamos que quiere crear un archivo que contiene los mensajes de error. Ud. puede nombrarlo asi: `error_lang.php`

En el archivo asignará cada linea de texto a un arreglo llamado *$lang* con este prototipo:

```
$lang['language_key'] = "The actual message to be shown";
```

Nota: Es una buena práctica utilizar un prefijo común para todos los mensajes de un determinado archivo para evitar colisiones con elementos llamados en forma similar en otros archivos. Por ejemplo, si está creando mensajes de error puede poner un prefijo en ellos con *error_*

```
$lang['error_email_missing'] = "You must submit an email address";
$lang['error_url_missing'] = "You must submit a URL";
$lang['error_username_missing'] = "You must submit a username";
```

Cargando un Archivo de Idioma

Con el fin de obtener una línea de un archivo particular Ud. debe cargar el archivo primero. La Carga de un archivo de idioma se hace con el siguiente código:

```
$this->lang->load('filename', 'language');
```

Donde `filename` es el nombre del archivo que desea cargar (sin la extensión del archivo), y *language* es el idioma establecido que la contenga (por ejemplo, english). Si el segundo parámetro falta, el idioma por defecto establecido en su archivo `application/config/config.php` será usado.

Obtención de una línea de texto

Una vez que su archivo de idioma deseado carga, usted puede acceder a cualquier línea de texto utilizando esta función:

```
$this->lang->line('language_key');
```

Donde `language_key` Es el arreglo clave correspondiente a la línea que desea mostrar.

Nota: Esta función simplemente retorna la linea. No la imprime por usted.

Auto-cargando Idiomas

Si Ud. encuentra que necesita un idioma particular globalmente en toda su aplicación, puede decirle a CodeIgniter que lo auto-cargue durante la inicialización del sistema auto-load. Esto se hace abriendo el archivo application/config/autoload.php y agregando el/los lenguaje/s al arreglo de autocarga.

Clase de Sálida

La clase de Sálida es una pequeña clase con una función principal: Mandar la página web finalizada al explorador. También es responsable de cachear sus páginas web, si usa esta característica.

> **Nota:** Esta clase es inicializada automáticamente por el sistema, así que no hay necesidad de hacerlo manualmente.

Bajo circunstancias normales, no notará la clase de Sálida ya que funciona transparentemente sin su intervención. Por ejemplo, cuando use la clase de Carga para cargar un archivo de vista, es automáticamente pasado a la clase de Sálida, la cual será llamada por CodeIgniter al final de la ejecución del sistema. Es posible, sin embargo, intervenir manualmente con la sálida si necesita, usando alguna de las dos funciones siguientes:

$this->output->set_output();

Le permite establecer la cadena de sálida final. Ejemplo de uso:

```
$this->output->set_output($data);
```

Importante: Si establece su sálida manualmente, debe ser la última cosa que haga en la función desde donde llama. Por ejemplo, si construye una página en una de las funciones de controlador, no establezca la sálida hasta el fin.

$this->output->get_output();

Le permite recuperar manualmente cualqueir sálida que haya sido enviada para almacenar en la clase de sálida. Ejemplo de uso:

$string = $this->output->get_output();

Note que los datos sólo serán recuperable desde esta función si han sido previamente enviados a la clase de sálida por una de la funciones de CodeIgniter como *$this->load->view()*.

$this->output->set_header();

Le permite establecer encabezados de servidor, los que la clase de sálida enviara cuando se envíen los datos finales. Ejemplo:

```
$this->output->set_header("HTTP/1.0 200 OK");
$this->output->set_header("HTTP/1.1 200 OK");
$this->output->set_header('Last-Modified: '.gmdate('D, d M Y H:i:s', $last_update).' GMT');
$this->output->set_header("Cache-Control: no-store, no-cache, must-revalidate");
$this->output->set_header("Cache-Control: post-check=0, pre-check=0", false);
$this->output->set_header("Pragma: no-cache");
```

$this->output->enable_profiler();

Le permite habilitar/deshabilitar el Perfilador, el cual mostrará datos de puntos de referencia y otros datos al final de sus páginas para propósitos de depuración y optimización.

Para habilitar el perfilador ubique la siguiente función en cualquier lugar dentro de sus funciones de Controlador:

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

Cuando lo habilite, un reporte será generado e insertado al final de sus páginas.

Para deshabilitar el perfilador usará:

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

Clase Paginación

La Clase Paginación de CodeIgniter es muy fácil de usar, y es 100% personalizable, ya sea dinámicamente o a través de las preferencias almacenadas.

Si no está familiarizado con el término "paginación", se refiere a los vínculos que le permite navegar de una página a otra, como este:

```
« First  < 1 2 3 4 5 >  Last »
```

Ejemplo

Este es un ejemplo sencillo que muestra cómo crear una de paginación en sus funciones del controller:

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

$config['base_url'] = 'http://www.your-site.com/index.php/test/page/';
$config['total_rows'] = '200';
$config['per_page'] = '20';

$this->pagination->initialize($config);

echo $this->pagination->create_links();
```

Notas:

El arreglo *$config* contiene sus variables de configuración. estas son pasadas a la función *$this->pagination->initialize* como se muestra arriba. A pesar de que hay una

veintena de items que usted puede configurar, el mínimo que necesita son los tres mostrados. Aquí hay una descripción de lo que representan esos items:

- **base_url** Esta es la URL completa a la clase/función del controlador que contiene su número de páginas. En el ejemplo anterior, ésta apunta a un controlador denominado "Test" y a una función llamada "page". Tenga en cuenta que Ud. puede modificar el trazado de su URI si necesita una estructura diferente.
- **total_rows** Este número representa la cantidad total de filas en el conjunto de resultados que está creando para el número de páginas. Normalmente, este será el número total de filas que retorno su consulta a la base de datos.
- **per_page** El número de elementos que tiene intención de mostrar por página. En el ejemplo anterior, Ud. desea mostrar 20 elementos por página.

The *create_links()* función que retorna un arreglor vacío cuando no hay paginación para mostrar.

Establecer preferencias en un archivo de configuración

Si prefiere no definir las preferencias utilizando el método anterior, puede ponerlas en un archivo de configuración. Basta con crear un nuevo archivo llamado *pagination.php*, y agregar el arreglo *$config* en ese archivo. A continuación guarde el archivo en: *config/pagination.php* y será utilizado automáticamente. Ud. No necesita usar la función *$this->pagination->initialize* si guarda sus preferencias en un archivo de configuración.

Personalizando la Paginación

La siguiente es una lista de todas las preferencias que se pueden pasar a la función de inicialización para ajustar la pantalla.

$config['uri_segment'] = 3;

La función determina automáticamente cual segmento de su URI contiene el número de página. Si usted necesita algo diferente se puede especificar.

$config['num_links'] = 2;

El número de links en "digitos" que Ud. desea antes y después del número de página seleccionado. Por ejemplo, el número 2 indica que habrá dos dígitos en ambos lados, como en el ejemplo de enlaces en la parte superior de esta página.

Agregando Marcas de Cierre

Si desea rodear toda la paginación con algunas marcas lo puede hacer con estas dos preferencias:

$config['full_tag_open'] = ' ';

La etiqueta de apertura situada en la parte izquierda de todo el resultado.

$config['full_tag_close'] = ' ';

La etiqueta de cierre situada en la parte derecha de todo el resultado.

Personalizando el enlace "Primero"

$config['first_link'] = 'First';

El texto que le gustaría que se muestre en el "primer" enlace de la izquierda.

$config['first_tag_open'] = ' ';

La etiqueta de apertura para el enlace "first".

$config['first_tag_close'] = ' ';

La etiqueta de cierre para el enlace "first".

Personalizando el enlace "Último"

$config['last_link'] = 'Last';

El texto que le gustaría que se muestre en el "último" enlace de la derecha.

$config['last_tag_open'] = ' ';

La etiqueta de apertura para el enlace "last".

$config['last_tag_close'] = ' ';

La etiqueta de cierre para el enlace "last".

Personalizando el enlace "Siguiente"

$config['next_link'] = '>';

El texto que le gustaría que se muestre en el enlace de página "siguiente".

$config['next_tag_open'] = ' ';

La etiqueta de apertura para el enlace "next".

$config['next_tag_close'] = ' ';

La etiqueta de cierre para el enlace "next".

Personalizando el enlace "Anterior"

$config['prev_link'] = '<';

El texto que le gustaría que se muestre en el enlace de página "anterior".

$config['prev_tag_open'] = ' ';

La etiqueta de apertura para el enlace "previous".

$config['prev_tag_close'] = ' ';

La etiqueta de cierre para el enlace "previous".

Personalizando el enlace "Página Actual"

$config['cur_tag_open'] = '';

La etiqueta de apertura para el enlace "current".

$config['cur_tag_close'] = '';

La etiqueta de cierre para el enlace "current".

Personalizando el enlace "Dígito"

$config['num_tag_open'] = ' ';

La etiqueta de apertura para el enlace "digit".

$config['num_tag_close'] = ' ';

La etiqueta de cierre para el enlace "digit".

Clase de Sesión

La clase de Sesión le permite mantener el "estado" de un usuario y seguir su actividad mientras navegue su sitio. La clase de Sesión guarda información de sesión para cada usuario como datos serializados (y opcionalmente encriptados) en una cookie. También guarda los datos de sesión en una tabla de base de datos para seguridad agregada, ya que permite qle el ID de la sesión en la cookie del usuario sea contrastada contra la guardada. Por defecto sólo la cookie es guardada. Si elije usar la opción de base de datos, necesitará crear la table de sesión como se indica debajo.

> **Nota:** La clase de Sesión **no** utiliza las sesiones nativas de PHP. Genera sus propios datos de sesión, ofreciendo más flexibilidad para los desarrolladores.

Inicializando una Sesión

Las sesiones típicamente correrán globalmente con cada carga de la página, así que la clase de sesión debe ser o inizializada en el constructor de su controlador, o puede ser auto-cargada por el sistema. Para la mayor parte, la clase de sesión correrá desatendida en el transfondo, así que simplemente inicializar la clase causará leer, crear y actualizar sesiones.

Para inicializar la clase de Sesión manualmente en el constructor del controlador, use la función *$this->load->library*:

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

Una vez cargada, el objeto de la librería de Sesión estará disponible usando: *$this->session*

¿Cómo funcionan las sesiones?

Cuando una página es cargada, la clase de sesión verificará si datos de sesión válidos existen en la cookie de sesión del cliente. Si los datos de sesión **no** existen (o si ha expirado) una nueva sesión será creada y guardada en la cookie. Si una sesión existe, su información será actualizada y la cookie será actualizada. Con cada actualización, el session_id (identificador de sesión) será regenerado.

Es importante para usted entender que una vez inicializada, la clase de Sesión corre automáticamente. No hay nada que necesite hacer para que el comportamiento anterior ocurre. Puede, como verá luego, trabajar con datos de sesión o incluso agregar sus propios datos a la sesión del usuario, pero el proceso de lectura, escritura y actualización de sesión es automático.

Que son los Datos de Sesión?

Una *sesión*, hasta donde le concierne a CodeIgniter, es simplemente un arreglo conteniendo la siguiente información:

- El identificador de sesión único (esto es estadísticamente una cadena aleatoria con muy fuerte entropia, una encriptación con MD5 por portabilidad, y es regenerado (por defecto) cada cinco minutos)
- La dirección IP del usuario
- Los datos del Agente del Usuario (los primeros 50 caracteres de la cadena de datos del explorador)
- La marca de tiempo de la "última actividad".

Los datos anteriores son guardados en una cookie como un arreglo serializado con este prototipo:

```
[array]
(
    'session_id'   => encriptación aleatoria,
    'ip_address'   => 'cadena - dirección IP del usuario',
    'user_agent'   => 'cadena - datos del agente del usuario',
    'last_activity' => marca de tiempo
)
```

Si tiene la opción de encriptación habilitada, el arreglo será encriptado antes de ser guardado en la cookie, haciendo los datos altamente seguros e impermeables de ser leídos o alterados por alguien. Más información acerca de la encriptación puede ser encontrada aquí, aunque la clase de Sesión se ocupará de la inicialización y encriptación de datos automáticamente.

Nota: Las cookies de sesión son sólo actualizadas cada cinco minutos por defecto para reducir la carga de procesamiento. Si recarga repetidamente una página, notará que el tiempo de "last activity" sólo se actualizará si cinco munitos o más han pasado desde la última vez que la cookie fue escrita. Este tiempo es configurable cambiando la línea de $config['time_to_update'] en su archivo system/config/config.php .

Recuperando Datos de Sesión

Cualquier pieza de información desde el arreglo de sesión está disponible usando la siguiente función:

```
$this->session->userdata('item');
```

Donde `item` es la clave del arreglo correspondiente al item que desea traer. Por ejemplo, para traer el ID de sesión hará lo siguiente:

```
$session_id = $this->session->userdata('session_id');
```

Nota: La función devuelve FALSE (booleano) si el item al que intenta acceder no existe.

Agregando Datos de Sesión Especiales

Un aspecto útil del arreglo de sesión es que puede agregar sus propios datos a él y ellos serán guardados en la cookie del usuario. ¿Por qué querría hacer esto? Aquí hay un ejemplo:

Digamos que un usuario en particular ingresó a su sitio. Una vez autenticado, puede agregar su nombre de usuario y dirección de email a la cookie de sesión, haciendo ese dato globalmente disponible sin tener que correr una consulta de base de datos cuando lo necesite.

Para agregar sus datos al arreglo de sesión debe pasar un arreglo que contenga sus nuevos datos a esta función:

```
$this->session->set_userdata($arreglo);
```

Donde `$arreglo` es un arreglo asociativo conteniendo sus nuevos datos. Aquí hay un ejemplo:

```
$nuevosdatos = array(
        'nombre_de_usuario'  => 'johndoe',
        'email'=> 'johndoe@algun-sitio.com',
        'ingresado' => TRUE
    );

$this->session->set_userdata($nuevosdatos);
```

Si quiere agregar un dato a la vez, set_userdata() también soporta esta sintaxis.

```
$this->session->set_userdata('algun_nombre', 'algun_valor');
```

Nota: Las cookies pueden guardar sólo 4KB de datos, así que sea cuidadoso de no exceder la capacidad. La El proceso de encriptación en particular produce una cadena de datos más largo que la original, así que sea cuidadoso siguiente cuanto datos están guardando.

Removiendo Datos de Sesión

Simplemente como set_userdata() puede ser usado para agregar información en la sesión, unset_userdata() puede ser usado para remover, al pasar la clase de sesión. Por ejemplo, si quiere remover 'algun_nombre' de la información de sesión:

```
$this->session->unset_userdata('algun_nombre');
```

Esta función puede también puede recibir un arreglo asociativo de items para borrar.

```
$arreglo_items = array('nombre_de_usuario' => '', 'email' => '');

$this->session->unset_userdata($arreglo_items);
```

Flashdata

CodeIgniter soporta "flashdata", o datos de sesión que sólo estarán disponibles para el próximo pedido al servidor. Esto puede ser muy útil, y son típicamente usados para informar o mensajes de estado (por ejemplo: "registro 2 borrado").

Nota: Las variables flash son prefijadas con "flash_" así que evite esos prefijos en sus propios nombres de sesión.

Para agregar flashdata:

```
$this->session->set_flashdata('item', 'valor');
```

También puede pasar un arreglo a set_flashdata(), de la misma manera que con set_userdata().

Para leer una variable flashdata:

```
$this->session->flashdata('item');
```

Si encuentra que necesita preservar una variable flashdata durante un pedido adicional, puede hacerlo usando la función keep_flashdata().

```
$this->session->keep_flashdata('item');
```

Guardando Datos de Sesión en la Base de Datos

Mientras el arreglo de datos de sesión guardada en la cookie del usuario contenga un identificador de sesión, a menos que guarde los datos de sesión en una base de datos, no hay forma de validarlo. Para algunas aplicaciones que requieren poco o nada de seguridad, la validación del ID de sesión puede no ser necesaria, pero si la aplicación requiere seguridad, la validación es obligatoria.

Cuando los datos de sesión están disponibles en una base de datos, cada vez que una sesión válida es encontrada en la cookie del usuario, una consulta a la base de datos es hecha para compararla. Si el identificador de sesión no coincide, la sesión es destruida. Los identificadores de sesión nunca pueden ser actualizados, sólo pueden ser generados cuando se crea una nueva sesión.

Para guardar sesiones, debe crear una tabla de base de datos primero. Aquí está el prototipo básico (para MySQL) requerido por la clase de sesión:

```
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,
PRIMARY KEY (session_id)
);
```

Nota: Por defecto, la tabla es llamada *ci_sessions*, pero puede nombrarla como desee siempre y cuando actualice el archivo `application/config/config.php` para que contenga el nombre que ha elegido. Una vez que haya creado su tabla de base de datos puede habilitar la opción de base de datos en su archivo config.php como sigue:

```
$config['sess_use_database'] = TRUE;
```

Una vez habilitada, la clase de Sesión guardará datos de sesión en la base de datos.

Esté seguro que haya especificado el nombre de la tabla en su archivo de configuración también:

```
$config['sess_table_name'] = 'ci_sessions";
```

Nota: La clase de Sesión tiene un recolector de basura incluido que limpia las sesiones expiradas para que usted no necesito escribir su propia rutina para hacerlo.

Destruir una sesión

Para limpiar la sesión actual:

```
$this->session->sess_destroy();
```

Nota: esta función debería ser la última que se llama, e incluso las variables de Flash serán destruidas. Si usted sólo desea que algunos items sean destruidos (y no todos), use unset_userdata().

Preferencias de Sesión

Encontrará las siguientes preferencias relacionadas a la Sesión en su archivo `application/config/config.php`:

Preferencia	Valor por Defecto	Opciones	Descripción
sess_cookie_name	ci_session	Ninguna	El nombre que quiere que la cookie de sesión sea guardada.
sess_expiration	7200	Ninugna	El número de segundos que quiere que dure la sesión. El valor por defecto es de 2 horas (7200 segundos). Si quiere una sesión que no expire establezca el valor a cero: 0
sess_encrypt_cookie		FALSE	TRUE/FALSE (buleano) Si encriptar o no los datos de sesión.
sess_use_database	FALSE	TRUE/FALSE (buleano)	Si guardar los datos de sesión a una base de datos. Debe crear la tabla antes de habilitar esta opción.

sess_table_name	ci_sessions	Cualquier nombre de tabla válido en SQL	El nombre de la tabla de sesión en la base de datos.
sess_time_to_update	300	Tiempo en segundos	Estas opciones controlan cuan a menudo la clase de sesión se regenerará y creará un nuevo identificador de sesión.
sess_match_ip	FALSE	TRUE/FALSE (buleano)	Si coincide la dirección de IP del usuario cuando se lean los datos de sesión. Note que algunos proveedores de servicio de Internet (ISPs) cambian dinámicamente el IP, así que si quiere una sesión que no expire deberá establecerlo en FALSE.
sess_match_useragent	TRUE	TRUE/FALSE (buleano)	Si coincide el Agente del Usuario cuando se leen los datos de sesión.

Clase de Vínculos de referencia (Trackback)

La Clase de Vínculos de referencia (trackback) provee funciones que le permiten enviar y recibir datos de vínculos de referencia.

Si no está familiarizado con los Vínculos de referencias encontrará más información en http://en.wikipedia.org/wiki/Trackback

Inicializando la Clase

Como la mayoría de las otras clases en CodeIgniter, la clase de Vínculos de referencia es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez cargada, el objeto de la librería de Vínculos de referencia estará disponible usando: *$this->trackback*

Enviando Vínculos de Referencias

Un vínculo de referencia puede ser enviado desde cualquiera de sus funciones de sus controladores usando un código similar a este ejemplo:

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

$tb_data = array(
        'ping_url'  => 'http://ejemplo.com/trackback/456',
        'url'  => 'http://www.mi-ejemplo.com/blog/entrada/123',
        'title'=> 'El Título de Mi Entrada',
        'excerpt'  => 'El contenido de la entrada.',
        'blog_name' => 'El Nombre de mi Blog',
        'charset'  => 'utf-8'
        );

if ( ! $this->trackback->send($tb_data))
{
    echo $this->trackback->display_errors();
}
else
{
    echo 'Vínculo de referencia enviado!';
}
```

Descripción del arreglo de datos:

- **ping_url** - La URL del sitio a la que está enviando el vínculo de referencia. Puede enviar vínculos de referencias a múltiples URLs separando cada URL con una coma.
- **url** - La URL de SU sitio donde la entrada del weblog puede ser vista.
- **title** - El título de su entrada weblog.
- **excerpt** - El contenido de su entrada weblog. Nota: la clase de vínculos de referencia automáticamente enviará sólo los primeros 500 caracteres de su entrada. También le quitará todas el HTML.
- **blog_name** - El nombre de su weblog.
- **charset** - La codificación de caracteres en que su weblog está escrito. Si se omite, será usado UTF-8.

La función de envío de vínculos de referencia devolverá TRUE/FALSE (buleano) en éxito o fallo. Si falla, puede recuperar el mensaje de error usando:

```
$this->trackback->display_errors();
```

Recibiendo Vínculos de Referencias

Antes de que pueda recibir vínculos de referencias debe crear un weblog. Si no tiene una bitácora aún no tiene sentido continuar.

Recibir vínculos de referencias es un poco más complicado que enviarlos, sólo porque necesitará una tabla de base de datos dónde almacenarlos, y necesitará validar los datos entrantes del vínculo de referencia. Se alienta a implementar un minucioso proceso de validación para resguardarse en contra del spam y los datos duplicados. Puede también querer limitar el número de vínculos de referencias que permite desde una dirección IP particular dentro de un rango de tiempo para resguardarse del spam. El proceso de recepción de vínculos de referencia es bastante simple; la validación es lo que toma la mayoría del esfuerzo.

Su URL de Ping

Para aceptar vínculos de referencias debe mostrar una URL de vínculos de referencia próxima a cada una de sus entradas weblog. Esta será la URL que la gente usará para enviar sus vínculos de referencias (nos referiremos a ella como su "URL de Ping").

Su URL de Ping debe aputar a un controlador donde su código de recepción de vínculos de referencia es ubicado, y la URL debe contener el número de ID para cada entrada en particular, así cuando el vínculos de referencia es recibido será capaz de asociarlo con una entrada en particular.

Por ejemplo, si su clase controlador es llamada *Trackback*, y la función de recepción es llamada *recibir*, su URL de Ping será algo así:

```
http://www.su-sitio.com/index.php/trackback/receive/entrada_id
```

Donde `entrada_id` representa el número de ID individual para cada una de sus entradas.

Creando una Tabla de Vínculos de Referencia

Antes de poder recibir sus vínculos de referencias debe crear una tabla donde guardarlos. Aquí hay un prototipo de tal tabla:

```
CREATE TABLE trackbacks (
 tb_id int(10) unsigned NOT NULL auto_increment,
 entry_id int(10) unsigned NOT NULL default 0,
 url varchar(200) NOT NULL,
 title varchar(100) NOT NULL,
 excerpt text NOT NULL,
 blog_name varchar(100) NOT NULL,
 tb_date int(10) NOT NULL,
 ip_address varchar(16) NOT NULL,
 PRIMARY KEY (tb_id),
 KEY (entry_id)
);
```

La especificación de vínculos de referencia sólo requiere cuatro piezas de información para enviar en un vínculo de referencia (url, título, contenido, nombre_blog), pero para hacer los datos más utiles hemos agregado unos campos más en la tabla anterior (fecha, dirección IP, etc.).

Procesando un Vínculo de Referencia

Aquí hay un ejemplo mostrando como recibirá y procesará un vínculo de referencia. El siguiente código está hecho para usar dentro de la función controlador donde espera recibir vínculos de referencias.

```
$this->load->library('trackback');
$this->load->database();

if ($this->uri->segment(3) == FALSE)
{
    $this->trackback->send_error("No se pudo determinar el ID de la entrada");
}

if ( ! $this->trackback->receive())
{
    $this->trackback->send_error("El Vínculo de Referencia no contenía datos
válidos");
}

$data = array(
        'tb_id'=> '',
        'entrada_id'   => $this->uri->segment(3),
        'url'      => $this->trackback->data('url'),
        'titulo'   => $this->trackback->data('title'),
        'texto'=> $this->trackback->data('excerpt'),
        'nombre_blog'  => $this->trackback->data('blog_name'),
        'tb_fecha' => time(),
        'direccion_ip' => $this->input->ip_address()
        );

$sql = $this->db->insert_string('trackbacks', $data);
$this->db->query($sql);

$this->trackback->send_success();
```

Notas:

El número de ID de la entrada es esperado como el tercer segmento de su URL. Esto se basa en la URI de ejemplo que dimos anteriormente:

```
http://www.su-sitio.com/index.php/trackback/receive/entrada_id
```


Note que entrada_id está en el tercer segmento de la URI, el cual puede ser recuperado usando:

```
$this->uri->segment(3);
```

En sus el código de recepción de vínculos de referencia previo, si el tercer segmento no se encuentra, responderemos un error. Sin un ID de entrada válido no hay razón para continuar.

La función *$this->trackback->receive()* es simplemente una función de validación que verifica los datos entrantes y se asegura que contiene las cuatros piezas de datos que son requeridas (url, titulo, texto, nombre_blog). Devuelve TRUE en éxito y FALSE si falla. Si falla, devolveremos un mensaje de error.

Los datos de vínculos de referencia entrantes pueden ser recuperados usando esta función:

```
$this->trackback->data('item')
```

Donde *item* representa uno de estas cuatro piezas de información: url, title (título), excerpt (texto), o blog_name (nombre del blog)

Si los datos de vínculos de referencia son recibidos correctamente, devolverá un mensaje de éxito usando:

```
$this->trackback->send_success();
```

Nota: El código anterior no contiene validación de datos, la cual está alentado a agregar.

Clase de Sintaxis de Plantilla (Template parser)

La Clase de Sintaxis de Plantilla le permite usar pseudo variables dentro de sus archivos de vista. Pueden ser simples variables o pares de etiquetas de variables. Si nunca ha usado un motor de plantillas, las pseudo variables se ven así:

```
<html>
<head>
<title>{blog_titulo}</title>
</head>
<body>

<h3>{blog_encabezado}</h3>

{blog_entradas}
<h5>{titulo}</h5>
<p>{cuerpo}</p>
{/blog_entradas}
</body>
</html>
```

Estas variables no son verdaderas variables de PHP, sino representaciones en texto plano que le permiten eliminar PHP de sus plantillas (archivos de vista).

> **Nota:** CodeIgniter **no** requiere que use esta clase ya que usar puro PHP en sus páginas de vista las hace correr un poco más rápido. Sin embargo, algunos desarrolladores prefieren usar un motor te plantillas si trabajar con diseñadores que pueden sentir consfusión trabajando con PHP.

También Note: La Clase de Sintaxis de Plantillas **no** es una solución completa a las sintaxis de plantillas. Lo hemos mantenido muy delgada para mantener al máximo el desempeño.

Inicializando la Clase

Como la mayoría de las otras clases en CodeIgniter, la clase de Sintaxis es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez cargada, el objeto de la librería de Sintaxis estará disponible usando: *$this->parser*

Las siguientes funciones están disponibles en esta librería:

$this->parser->parse()

Esta función acepta un nombre de plantilla y un arreglo de datos como entrada, y genera una versión en consecuencia. Ejemplo:

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

$data = array(
    'blog_titulo' => 'El Título de mi Blog',
    'blog_heading' => 'El Encabezado de mi Blog'
    );

$this->parser->parse('blog_plantilla', $data);
```

El primer contiene el nombre del archivo de vista (en este ejemplo sería llamado blog_plantilla.php), y el segundo parámentro contiene un arreglo asociativo de datos para ser reemplazados en la plantilla. En el ejemplo anterior, la plantilla contendría dos variabels: {blog_titulo} y {blog_heading}

No hay necesidad de "echo" o de hacer algo con los datos devueltos por *$this->parser->parse()*. es automáticamente pasado a la clase de salida para ser enviado al explorador. Sin embargo, si quiere que los datos sean devueltos en vez de enviados a la clase de salida puede pasar TRUE (buleano) al tercer parámetro:

```
$string = $this->parser->parse('blog_plantilla', $data, TRUE);
```

Pares de variables

En el ejemplo anterior el código permite variables simples ser reemplazadas. ¿Qué pasa si quiere un bloque entero de variables sean repetidos, con cada iteración conteniendo nuevos valores? Considere el ejemplo de la plantilla que mostramos al principio de la página:

```
<html>
    <head>
        <title>{blog_titulo}</title>
    </head>
    <body>
        <h3>{blog_heading}</h3>
        {blog_entradas}
        <h5>{titulo}</h5>
```

```
        <p>{cuerpo}</p>
        {/blog_entradas}
    </body>
</html>
```

En el código anterior notará un par de variables: `{blog_entradas}` datos... `{/blog_entradas}`. En un caso como este, el pedazo entero de datos entre este par se repetirá múltiples veces, correspondiendo al número de filas en un resultado.

La sintaxis de pares de variables es hecho usando el idéntico código mostrado arriba para sintaxis de variables simples, a excepción que agregará un arreglo multi-dimensional correspondiente a sus datos de pares de variables. Considere este ejemplo:

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

$data = array(
    'blog_titulo'   => 'El Título de mi Blog',
    'blog_heading' => 'El Encabezado de mi Blog',
    'blog_entradas' => array(
           array('titulo' => 'Titulo 1', 'cuerpo' => 'Cuerpo 1'),
           array('titulo' => 'Titulo 2', 'cuerpo' => 'Cuerpo 2'),
           array('titulo' => 'Titulo 3', 'cuerpo' => 'Cuerpo 3'),
           array('titulo' => 'Titulo 4', 'cuerpo' => 'Cuerpo 4'),
           array('titulo' => 'Titulo 5', 'cuerpo' => 'Cuerpo 5')
           )
    );

$this->parser->parse('blog_plantilla', $data);
```

Si sus "pares" de datos vienen de un resultado de base de datos, el cual ya es un arreglo multidimensional, puede simplemente usar la función result de base de datos:

```
$query = $this->db->query("SELECT * FROM blog");

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

$data = array(
    'blog_titulo'   => 'El Título de mi Blog',
    'blog_heading' => 'El Encabezado de mi Blog',
    'blog_entradas' => $query->result_array()
    );

$this->parser->parse('blog_plantilla', $data);
```


Clase de Prueba de Unidad (Unit Testing)

La Prueba de Unidad es una aproximación al desarrollo de software en la cual las pruebas son escritas para cada función en su aplicación. Si no está familiarizado con el concepto puede hacer un pequeño googleo en el tema.

La clase de Prueba de Unidad de CodeIgniter es bastante simple, constando de una función de evaluación y dos funciones de resultado. No se intenta ser una prueba completa sino un simple mecanismo para evaluar su código para determinar si está produciendo el tipo de dato y resultado correcto.

Inicializando la Clase

Como la mayoría de las otras clases en n CodeIgniter, la clase de Prueba de Unidad es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez cargada, el objeto de Prueba de Unidad estará disponible usando: *$this->unit*

Corriendo Pruebas

Correr una prueba involucra suministrar una prueba y un resultado esperado a la siguiente función:

$this->unit->run(*prueba*, *resultado esperado*, '*nombre de prueba*');

Donde *prueba* es el resultado del código que desea probar, *resultado esperado* es el tipo de dato que espera, y *nombre de prueba* es un nombre opcional que puede darle a su prueba. Ejemplo:

```
$prueba = 1 + 1;

$resultado_esperado = 2;

$nombre_prueba = 'Adds one plus one';

$this->unit->run($prueba, $resultado_esperado, $nombre_prueba);
```

El resultado esperado que suministre puede ser o un valor literal, o un tipo de dato a comparar. Aquí hay un ejemplo literal:

```
$this->unit->run('Foo', 'Foo');
```

Aquí hay un ejemplo de tipo de dato a comparar:

```
$this->unit->run('Foo', 'is_string');
```

Nota el uso de "is_string" en el segundo parámetro? Esto le dice a la función que evalúe si su prueba está produciendo una cadena como resultado. Aquí hay una lista de tipos de comparación posibles:

- is_string
- is_bool
- is_true
- is_false
- is_int
- is_numeric
- is_float
- is_double
- is_array
- is_null

Generando Reportes

Puede o mostrar los resultados después de cada prueba, o puede correr varias pruebas y generar un reporte al final. Para mostrar un reporte directamente simplemente imprima o devuelva la función *run*:

```
echo $this->unit->run($prueba, $resultado_esperado);
```

Para correr un reporte completo de todas las pruebas, use esto:

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


El reporte tendrá formato en una tabla HTML para ver. Si prefiere datos crudos puede recuperar un arreglo usando:

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

Modo estricto

Por defecto la clase de prueba de unidad evalúa coincidencias literales sin tipo. Considere este ejemplo:

```
$this->unit->run(1, TRUE);
```

La prueba está evaluando un entero, pero el resultado esperado es un buleano. PHP, sin embargo, debido a su simpleza de tipo de datos evaluará el código como TRUE usando una prueba de igualdad normal:

```
if (1 == TRUE) echo 'Esto evalúa como true';
```

Si prefiere, puede poner la clase de prueba de unidad en modo estricto, que comparará los tipos de datos así también como el valor:

```
if (1 === TRUE) echo 'Esto evalúa como FALSE';
```

Para habilitar el modo estricto use esto:

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

Habilitando/Deshabilitando Pruebas de Unidad

Si quiere dejar algunas pruebas en el lugar en su código, pero no ejecutarlas excepto que lo necesite puede deshabilitar la prueba de unidad usando :

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

Creando una Plantilla

Si desea que sus resultados de prueba tengan un formato diferente del por defecto puede establecer su propia plantillas. Aquí hay una ejemplo de una plantilla simple. Note las requeridas pseudo-variables:

```
$cadena = '
<table border="0" cellpadding="4" cellspacing="1">
    {rows}
        <tr>
        <td>{item}</td>
        <td>{result}</td>
        </tr>
    {/rows}
</table>';

$this->unit->set_template($cadena);
```

Nota: Su plantilla debe ser declarada **antes** de correr el proceso de prueba de unidad.

Clase de URI

La Clase de URI provee funciones que le ayudarán a obtener información desde cadenas URI. Si usa ruteo URI, puede también obtener información de segmentos redirigidos.

Nota: Esta clase es inicializada automáticamente por el sistema, así que no necesita hacerlo manualmente.

$this->uri->segment(*n*)

Permite recuperar un segmento específico. Donde *n* es el número de segmento que desea recuperar. Los segmentos están numerados de izquierda a derecha. Por ejemplo, si su URL completa es esta:

```
http://www.your-site.com/index.php/noticias/local/ciudad/subio_el_crimen
```


Los números de segmentos serán estos:

1. noticias
2. local
3. ciudad
4. subio_el_crimen

Por defecto, la función devuelve (buleano) si el segmento no existe. Existe un opcional segundo parámetro que le permite establecer su propio valor por defecto si el segmento no se encuentra. Por ejemplo, esto le dirá a la función que devuelva el número cero en caso de fallo:

```
$producto_id = $this->uri->segment(3, 0);
```

Ayuda a evitar tener que escribir líneas de código así:

```
if ($this->uri->segment(3) === FALSE) {
    $product_id = 0;
} else {
    $product_id = $this->uri->segment(3);
}
```

$this->uri->rsegment(*n*)

Esta función es idéntica a la previa, excepto que le permite recuperar un segmento específico de su redirigida URI en el caso de que esté usando la característica de Ruteo de URI de CodeIgniter.

$this->uri->slash_segment(*n*)

Esta función es casi idéntica a *$this->uri->segment()*, excepto que agrega una barra al principio y/o final basado en el segundo parámetro. Si el parámetro no es usado, una barra es agregada al final. Ejemplos:

```
$this->uri->slash_segment(3);
$this->uri->slash_segment(3, 'leading');
$this->uri->slash_segment(3, 'both');
```

Retorna:

1. segment/
2. /segment
3. /segment/

$this->uri->slash_rsegment(*n*)

Esta función es idéntica a la previa, excepto que le permite recuperar un segmento específico de su redirigida URI en el caso de que esté usando la característica de Ruteo de URI de CodeIgniter.

$this->uri->uri_to_assoc(*n*)

Esta función le permite transformar segmentos URI en un arreglo asociativo de pares de claves/valores. Considere esta URI:

```
index.php/usuario/busqueda/nombre/jose/ubicacion/UK/sexo/masculino
```

Usando esta función puede transformar la URI en un arreglo asociativo con este prototipo:

```
[array]
(
    'nombre' => 'jose'
    'ubicacion' => 'UK'
    'sexo' => 'masculino'
)
```

El primer parámetro de la función establece un inicio. Por defecto es establecido en 3 ya que sus URI normalmente contendrán un controlador/función en el primer y segundo segmento. Ejemplo:

```
$arreglo = $this->uri->uri_to_assoc(3);

echo $arreglo['nombre'];
```

El segundo parámetro le permite establecer nombres de clave por defecto, así el arreglo devuelto por la función siempre contendrá los índices esperados, incluso si no están en la URI. Ejemplo:

```
$por_defecto = array('nombre', 'genero', 'ubicacion', 'tipo', 'orden');

$arreglo = $this->uri->uri_to_assoc(3, $por_defecto);
```

Si la URI no contiene un valor por defecto, un índice del arreglo será establecido con ese nombre, con el valor de FALSE.

Por último, si un valor correspondiente no es encontrado para una clave dada (si hay un número impar de segmentos de URI) el valor será establecido en FALSE (boolean).

$this->uri->ruri_to_assoc(*n*)

Esta función es idéntica a la previa, excepto que le permite recuperar un segmento específico de su redirigida URI en el caso de que esté usando la característica de Ruteo de URI de CodeIgniter.

```
$this->uri->assoc_to_uri()
```

Toma un arreglo asociativo como entrada y genera una cadena URI de él. La claves del arreglo serán incluidos en la cadena. Ejemplo:

```
$arreglo = array('producto' => 'zapatos', 'tamano' => 'largo', 'color' =>
'rojo');

$str = $this->uri->assoc_to_uri($arreglo);

// Produce: producto/zapatos/tamano/largo/color/rojo
```

$this->uri->uri_string()

Devuelve una cadena con la URI completa. Por ejemplo, si su URL completa es.

```
http://www.su-sitio.com/index.php/noticias/local/345
```

La función devolverá esto:

```
/noticias/local/345
```

$this->uri->ruri_string(*n*)

Esta función es idéntica a la previa, excepto que le permite recuperar un segmento específico de su redirigida URI en el caso de que esté usando la característica de Ruteo de URI de CodeIgniter.

```
$this->uri->total_segments()
```

Devuelve el número de segmentos totales.

```
$this->uri->total_rsegments()
```

Esta función es idéntica a la previa, excepto que le permite recuperar un segmento específico de su redirigida URI en el caso de que esté usando la característica de Ruteo de URI de CodeIgniter.

```
$this->uri->segment_array()
```

Devuelve un arreglo que contiene los segmentos URI. Por ejemplo:

```
$segs = $this->uri->segment_array();

foreach ($segs as $segmento)
{
    echo $segmento;
    echo '<br />';
}
```

$this->uri->rsegment_array(*n*)

Esta función es idéntica a la previa, excepto que le permite recuperar un segmento específico de su redirigida URI en el caso de que esté usando la característica de Ruteo de URI de CodeIgniter.

Clase de Agente del Usuario

La Clase de Agente del Usuario le provee funciones para ayudar a identificar información acerca del explorador, dispositivo móvil o robot visitando su sitio. Además puede obtener información de referencia así como el lenguaje e información sobre los juegos de caracteres soportados.

Inicializando la Clase

Como la mayoría de las otras clases en CodeIgniter, la clase de Agente del Usuario es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez carada, el objeto estará disponible usando: *$this->agent*

Definiciones de Agente del Usuario

La definición de nombres del agente del usuario está ubicada en un archivo de configuración que se encuentra en: *application/config/user_agents.php*. Puede agregar items al variado arreglo de agentes de usuario si lo necesita.

Ejemplo

Cuando la clase de Agente del Usuario es inicializada, intentará determinar si el agente del usuario navegando su sitio es un navegador web, un dispositivo móvil, o un robot. También juntará información sobre la plataforma, si esta está disponible.

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

if ($this->agent->is_browser())
{
    $agent = $this->agent->browser().' '.$this->agent->version();
}
elseif ($this->agent->is_robot())
{
    $agent = $this->agent->robot();
}
elseif ($this->agent->is_mobile())
{
    $agent = $this->agent->mobile();
}
```

```
else
{
    $agent = 'Unidentified User Agent';
}

echo $agent;

echo $this->agent->platform(); // Información de plataforma (Windows, Linux,
Mac, etc.)
```

Referencia de funciones

$this->agent->is_browser()

Devuelve TRUE/FALSE (boolean) si el agente del usuario es un navegador web conocido.

$this->agent->is_mobile()

Devuelve TRUE/FALSE (boolean) si el agente del usuario es un dispositivo móvil conocido.

$this->agent->is_robot()

Devuelve TRUE/FALSE (boolean) si el agente del usuario es un robot conocido.

Nota: La librería del agente del usuario sólo contiene las definiciones de robots más comunes. No es una lista de bots completa. Hay cientos de ellos por lo que buscar por cada uno no sería muy eficiente. Si encuentra que algún bot que comúnmente visita su sitio falta en la lista, puede agregarlo a su archivo *application/config/user_agents.php*.

$this->agent->is_referral()

Devuelve TRUE/FALSE (boolean) si el agente del usuario fue referido desde otro sitio.

$this->agent->browser()

Devuelve una cadena que contiene el nombre del navegador web viendo su sitio.

$this->agent->version()

Devuelve una cadena que contiene el número de versión del navegador web viendo su sitio.

$this->agent->mobile()

Devuelve una cadena que contiene el nombre del dispositivo móvil viendo su sitio.

$this->agent->robot()

Devuelve una cadena que contiene el nombre del robot viendo su sitio.

$this->agent->platform()

Devuelve una cadena que contiene la plataforma viendo su sitio (Linux, Windows, OS X, etc.).

$this->agent->referrer()

El referente, si el agente del usuario fue referido de otro sitio. Típicamente probará de esta esta forma:

```
if ($this->agent->is_referral())
{
    echo $this->agent->referrer();
}
```

$this->agent->agent_string()

Devuelve una cadena que contiene la cadena completa del agente del usuario. Típicamente será algo así:

```
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.0.4) Gecko/20060613
Camino/1.0.2
```

$this->agent->accept_lang()

Le permite determinar si el agente del usuario acepta un lenguaje en particular. Ejemplo:

```
if ($this->agent->accept_lang('en'))
{
    echo 'Acepta Inglés!';
}
```

Nota: Esta función no es muy confiable típicamente ya que algunos exploradores no proveen información de lenguaje, e incluso entre los que lo hacen, no siempre es la forma más precisa.

$this->agent->accept_charset()

Le permite determinar si el agente del usuario acepta un juego de caracteres en particular. Ejemplo:

```
if ($this->agent->accept_charset('utf-8'))
{
    echo 'Su explorador soporta UTF-8!';
}
```

Nota: Esta función no es muy confiable típicamente ya que algunos exploradores no proveen información de lenguaje, e incluso entre los que lo hacen, no siempre es la forma más precisa.

Clase de Validación de Formularios

Antes de explicar la aproximación de CodeIgniter a la validación de datos, describamos el escenario ideal:

1. Un formulario es mostrado.
2. Se llena y se envía.
3. Si envió algo inválido, o quizás se olvidó de un item requerido, el formulario es vuelto a mostrar conteniendo sus datos, junto con un mensaje de error que describa el problema.
4. Este proceso continúa hasta haber envíado un formulario válido.

En la recepción final, el script debe:

1. Chequear datos requeridos.
2. Verificar que los datos sean de tipo correcto, y cumplan el criterio correcto. (Por ejemplo si se envia un nombre de usuario debe ser validado para contener sólo caracteres permitidos. Debe tener un largo mínimo, y no exceder un largo máximo. El nombre de usuario no puede ser el mismo de un usuario existente, o quizás una palabra reservada. Etc.)
3. Limpiar los datos por seguridad.
4. Pre-formatear los datos si es necesario (Necesita limpiar los espacios? Encodear en HTML? Etc.)
5. Preparar los datos para insertar en la base de datos.

Aunque no hay nada complejo acerca del proceso, usualmente requiere una significativa cantidad de código, y para mostrar mensajes de errores, varias estructuras de control que usualmente se ubican en el formulario HTML. La validación de formulario, aunque es simple de crear, es generalmente muy poco prolija y tediosa de implementar.

CodeIgniter provee una comprensiva validación que verdaderamente minimiza la cantidad de código que escribirá. También remueve todas las estructuras de control de su formulario HTML, permitiéndole ser limpio y libre de código.

Visión General

Para implementar la clase de validación de CodeIgniter necesitará tres cosas:

1. Un archivo Vista que contenga el formulario.

2. Un archivo de Vista que contenga un mensaje de "éxito" para ser mostrado cuando el envío es exitoso.

3. Una función controlador que recibe y procesa los datos enviados.

Creemos estas tres cosas, usando un formulario de inscripción de miembros como ejemplo.

EL Formulario

Usando un editor de texto, cree un formulario llamado *miformulario.php*. En él, ubique este código y guárdelo en su carpeta `applications/views/`:

```
<html>
<head>
<title>My Form</title>
</head>
<body>
<?=$this->validation->error_string; ?>
<?=form_open('form'); ?>
<h5>Username</h5>
<input type="text" name="username" value="" size="50" />
<h5>Password</h5>
<input type="text" name="password" value="" size="50" />
<h5>Password Confirm</h5>
<input type="text" name="passconf" value="" size="50" />
<h5>Email Address</h5>
<input type="text" name="email" value="" size="50" />
<div><input type="submit" value="Submit" /></div>
</form>
</body>
</html>
```

Página de Éxito

Usando un editor de texto, cree un formulario llamado *formulario_exito.php*. En él, ubique este código y guárdelo en la carpeta `applications/views/`:

```
<html>
<head>
<title>My Form</title>
</head>
<body>
<h3>Your form was successfully submitted!</h3>
<p><?php echo anchor('form', 'Try it again!'); ?></p>
</body>
</html>
```


El Controlador

Usando un editor de texto, cree un controlador llamado *formulario.php*. En él, ubique este código y guárdelo en su carpeta `applications/controllers/`:

```
<?php
class Form extends Controller {
    function index()
    {
        $this->load->helper(array('form', 'url'));

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

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

Inténtalo!

Para probar su formulario, visite su sitio usando una URL similar a esta:

```
www.su-sitio.com/index.php/formulario/
```

Si usted envía el formulario debería simplemente ver el formulario recargado. Eso es porque ninguna regla de validación ha sido establecida aún, lo que haremos en un momento.

Explicación

Notará varias cosas acerca de las páginas anteriores:

El *formulario* (miformulario.php) es un formulario web estándar, con un par de excepciones:

1. Usa el *asistente de formulario* para crear la apertura del formulario. Técnicamente, esto no es necesario. Puede crear el formulario usando HTML estándar. Sin embargo, el beneficio de usar este asistente es que genera la URL de acción por usted, basado en la URL en su archivo de configuración. Esto hace que su aplicación sea más portable y flexible en caso de que sus URLs cambien.

2. Al comienzo del formulario notará la siguiente variable:

```
<?=$this->validation->error_string; ?>
```

Esta variable mostrará cualquier mensaje de error enviado por el validador. Si no hay mensajes, no devolverá nada.

El *controlador* (formulario.php) tiene una función: *index()*. Esta función inicializa la clase de validación, carga el *asistente de formulario* y *asistente de URL* usado por sus archivos de vista. También `corre` la rutina de validación. Basado en si la validación fue exitosa presenta el formulario o la página de éxito.

Como no ha dicho a la clase de validación que valide nada, devuelve "falso" (boolean false) por defecto. La función `run()` sólo devuelve "verdadero" si ha aplicado éxitosamente sus reglas sin que ninguna halla fallado.

Estableciendo Reglas de Validación

CodeIgniter le permite establecer tantas reglas de validación como necesita para un campo dado, encascándolos en orden, e incluso le permite preparar y pre-procesar los datos de los campos al mismo tiempo. Veámoslo en acción, lo explicaremos luego.

En su *controlador* (formulario.php), agrega este código justo debajo de la función de inicialización de validación:

```
$reglas['usuario'] = "required";
$reglas['contrasena'] = "required";
$reglas['conf_contrasena'] = "required";
$reglas['email'] = "required";

$this->validation->set_rules($reglas);
```

Su controlador ahora debería verse así:

```
<?php
class Form extends Controller {
    function index()
    {
        $this->load->helper(array('form', 'url'));

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

        $rules['username']= "required";
        $rules['password']= "required";
        $rules['passconf']= "required";
        $rules['email']       = "required";
```


```
        $this->validation->set_rules($rules);

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

Ahora envíe el formulario con los campos en blanco y debería ver el mensaje de error. Si envía el formulario con todos los campos rellenados, verá la página de éxito.

Nota: Los campos del formulario no están aún rellenados con los datos cuando hay un error. Empezaremos con eso en poco, una vez que hayamos terminado de explicar las reglas de validación.

Cambiando los Delimitadores de Error

Por defecto, el sistema agrega una etiqueta de párrafo (<p>) alrededor de cada mensaje de error mostrado. Puede fácilmente cambiar estos delimitadores con este código, ubicado en su controlador:

```
$this->validation->set_error_delimiters('<div class="error">', '</div>');
```

En este ejemplo, hemos cambiado para usar etiquetas div.

Reglas en Cascada

CodeIgniter le permite encadenar múltiples reglas con pipes(|). Intentémoslo. Cambie su arreglo de reglas así:

```
$reglas['usuario'] = "required|min_length[5]|max_length[12]";
$reglas['contrasena'] = "required|matches[conf_contrasena]";
$reglas['conf_contrasena'] = "required";
$reglas['email'] = "required|valid_email";
```

El código anterior requiere que:

1. El campo usuario debe ser no más corto de 5 caracteres y no más de 12.
2. El campo contraseña debe coincidir con el campo de confirmación de contraseña.
3. El campo email debe contener una dirección de email válida.

Inténtelo!

Nota: Hay numerosas reglas disponibles las cuales pueden ser leídas en la referencia de validación.

Preparando Datos

Además de las funciones de validación como las usadas anteriormente, puede preparar los datos de varias formas. Por ejemplo, puede establecer reglas así:

```
$reglas['usuario'] = "trim|required|min_length[5]|max_length[12]|xss_clean";
$reglas['contrasena'] = "trim|required|matches[conf_contrasena]|md5";
$reglas['conf_contrasena'] = "trim|required";
$reglas['email'] = "trim|required|valid_email";
```

En el anterior ejemplo, estamos "limpiando" los campos, convirtiendo la contraseña a MD5, y corriendo al usuario a través de la función "xss_clean", la que remueve datos maliciosos.

Cualquier función nativa de PHP que acepte un parámetro puede ser usada como regla, como *htmlspecialchars*, *trim*, *MD5*, etc.

Nota: Generalmente querrá usar las funciones de preparación **después** de las reglas de validación, así si hay un error, los datos originales serán mostrados en el formulario.

Funciones de retorno (callback): Sus propias Funciones de Validación

El sistema de validación soporta llamadas de retorno a sus propias funciones de validación. Esto permite extender la clase de validación para concretar sus necesidades. Por ejemplo, si desea correr una consulta de base de datos para ver si el usuario está eligiendo un nombre de usuario único, puede crear una función callback que lo haga. Creemos un ejemplo simple.

En su controlador, cambie la regla "usuario" a esto:

```
$reglas['usuario'] = "callback_usuario_check";
```

Luego agregue una nueva función llamada *usuario_check* a su controlador. Así es como su controlador debería verse:

```
<?php
class Form extends Controller {
    function index()
    {
        $this->load->helper(array('form', 'url'));

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

        $rules['username']= "callback_username_check";
        $rules['password']= "required";
        $rules['passconf']= "required";
        $rules['email']       = "required";

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

    function username_check($str)
    {
        if ($str == 'test')
        {
            $this->validation->set_message('username_check', 'The %s field can
not be the word "test"');
            return FALSE;
        }
        else
        {
            return TRUE;
        }
    }

}
?>
```

Recargue su formulario y envíelo con la palabra "test" como el usuario. Puede ver que el dato del campo del formulario fue pasado a su función de retorno para su procesamiento.

Para invocar una llamada de retorno sólo ponga el nombre de la función en una regla, con "callback_" como prefijo de la regla.

El mensaje de error fue establecido usando la función *$this->validation->set_message*. Simplemente recuerde que la clave del mensaje (el primer parámetro) debe coincidir con su nombre de función.

> **Nota:** Puede aplicar su propio mensaje de error a cualquier regla, sólo al escablecer el mensaje, similarmente. Por ejemplo, para cambiar el mensaje para la regla "required" haría esto:

```
$this->validation->set_message('required', 'Su mensaje personal aquí');
```

Re-populando el formulario

Hasta aquí sólo hemos tratado con errores. Es tiempo de repopular los campos del formulario con los datos enviados. Esto es hecho de manera similar a sus reglas. Agregue este código a su controlador, justo debajo de sus reglas:

```
$fields['usuario'] = 'Usuario';
$fields['contrasena'] = 'Contraseña';
$fields['conf_contrasena'] = 'Confirmación de Contraseña';
$fields['email'] = 'Dirección de Email';

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

Las claves del arreglo son los nombre de los campos del formulario, el valor representa el nombre completo que quiere mostrar en el mensaje de error.

La función índice del controlador debería verse así ahora:

```
function index()
{
    $this->load->helper(array('form', 'url'));
    $this->load->library('validation');
    $rules['username']= "required";
    $rules['password']= "required";
    $rules['passconf']= "required";
    $rules['email']       = "required";

    $this->validation->set_rules($rules);

    $fields['username']   = 'Username';
    $fields['password']   = 'Password';
    $fields['passconf']   = 'Password Confirmation';
```


```
    $fields['email']  = 'Email Address';

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

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

Ahora abra su archivo de vista *miformulario.php* y actualice el valor en cada campo para que tenga un atributo del objeto correspondiente a su nombre:

```
<html>
    <head>
        <title>My Form</title>
    </head>
    <body>

        <?php echo $this->validation->error_string; ?>

        <?php echo form_open('form'); ?>

        <h5>Username</h5>
        <input type="text" name="username" value="<?php echo $this->validation-
>username;?>" size="50" />

        <h5>Password</h5>
        <input type="text" name="password" value="<?php echo $this->validation-
>password;?>" size="50" />

        <h5>Password Confirm</h5>
        <input type="text" name="passconf" value="<?php echo $this->validation-
>passconf;?>" size="50" />

        <h5>Email Address</h5>
        <input type="text" name="email" value="<?php echo $this->validation-
>email;?>" size="50" />

        <div><input type="submit" value="Submit" /></div>

        </form>

    </body>
</html>
```

Ahora recargue su página y envíe el formulario de manera que dispare un error. Sus campos de formulario deben ser populados y los mensajes de error contendrán un nombre de campo más relevante.

Si prefiere mostrar un mensaje de error próximo a cada campo del formulario, en vez de como una lista, puede cambiar su formulario para que se vea así:

```
<h5>Username</h5>
<?php echo $this->validation->username_error; ?>
<input type="text" name="username" value="<?php echo $this->validation-
>username;?>" size="50" />

<h5>Password</h5>
<?php echo $this->validation->password_error; ?>
<input type="text" name="password" value="<?php echo $this->validation-
>password;?>" size="50" />

<h5>Password Confirm</h5>
<?php echo $this->validation->passconf_error; ?>
<input type="text" name="passconf" value="<?php echo $this->validation-
>passconf;?>" size="50" />

<h5>Email Address</h5>
<?php echo $this->validation->email_error; ?>
<input type="text" name="email" value="<?php echo $this->validation->email;?>"
size="50" />
```

Si no hay errores, nada será mostrado. Si hay un error, el mensaje aparecerá, encerrado en los delimitadores que haya establecido (las etiquetas <p> por defecto).

Nota: Para mostrar errores de esta forma debe recordar establecer sus campos usando la función
$this->validation->set_fields descripta anteriormente. Los errores serán transformados en variable, que hace "_error" después del nombre de campo. Por ejemplo, el error de "usuario" estará disponible en:
$this->validation->usuario_error.

La siguiente es una lista de todas las reglas nativas que están disponibles para usar:

Regla	Parámetro	Descripción	Ejemplo
required	No	Devuelve FALSE si el elemento del formulario es vacío.	
matches	Sí	Devuelve FALSE si el elemento del formulario no coincide con el del parámetro.	matches[item_formulario]
min_length	Sí	Devuelve FALSE si el elemento del formulario es más corto que el valor del parámetro.	min_length[6]
max_length	Sí	Devuelve FALSE si el elemento del formulario es más largo que el valor del parámetro.	max_length[12]
exact_length	Sí	Devuelve FALSE si el largo del elemento del formulario no es exactamente el del valor del parámetro.	exact_length[8]
alpha	No	Devuelve FALSE si el elemento del formulario contiene algo distinto que caracteres del alfabeto.	
alpha_numeric	No	Devuelve FALSE si el elemento del formulario contiene algo distinto que caracteres alfanuméricos.	
alpha_dash	No	Devuelve FALSE si el elemento del formulario contiene algo que no sean caracteres alfanuméricos, guiones bajos o guiones.	
numeric	No	Devuelve FALSE si el elemento del formulario contiene algo distinto a caracteres numéricos.	
integer	No	Devuelve FALSE si el elemento del formulario contiene algo distinto a un entero.	
valid_email	No	Devuelve FALSE si el elemento del formulario no contiene una dirección de email válida.	
valid_ip	No	Devuelve FALSE si la IP suministrada no es válida.	
valid_base64	No	Devuelve FALSE si la cadena suministrada contiene algún caracter distinto a los caracteres válidos de Base64.	

Nota: Estas reglas pueden ser también llamadas como funciones discretas. Por ejemplo:

```
$this->validation->required($cadena);
```

Nota: Puede usar también cualquier función de PHP que permita un parámetro.

Prepping Reference

La siguiente es una lista de todas las funciones de preparación que están disponibles para usar:

Nombre	Parámetro	Descripción
xss_clean	No	Corre el dato a través de la función de filtro XSS, descripta enla página de la Clase de Entrada.
prep_for_form	No	Conviernte caracteres especiales para que los datos HTML puedan ser mostrados en un campo de formulario sin romperlo.
prep_url	No	Agrega "http://" a la URLs si no lo tienen.
strip_image_tags	No	Quitan las etiquetas de imagen de HTML dejando la URL plata.
encode_php_tags	No	Convierte las etiquetas PHP a entidades.

Nota: Puede también usar cualquier función nativa de PHP que permita un parámetro, como trim, htmlspecialchars, urldecode, etc.

Estableciendo Mensajes de Error Especiales

Todos los mensajes ed error nativos están ubicados en el siguiente archivo de lenguaje: *language/english/validation_lang.php*

Para establecer sus propios mensajes puede o editar ese archivo, o utilizar la siguiente función:

```
$this->validation->set_message('regla', 'Mensaje de Error');
```

Donde *regla* corresponde al nombre de la regla en perticular, y *Mensaje de Error* es el texto que quiere mostrar.

Tratando con Menú desplegable y casillas de verificación (radio y checkbox)

Si usa un menú desplegable, casillas de verificación (radio y checkbox), querrá el estado de esos items que sea retenido en el caso de un error. La clase de Validación tiene tres funciones que ayudan a esto:

set_select()

Permite mostrar el item del menú que fue seleccionado. El primer parámetro debe contener el nombre del menú de selección, el segundo parámetro debe contener el valor de cada item. Ejemplo:

```
<select name="miselect">
<option value="uno" <?= $this->validation->set_select('miselect', 'uno'); ?>
>Uno</option>
<option value="dos" <?= $this->validation->set_select('miselect', 'dos'); ?>
>Dos</option>
<option value="tres" <?= $this->validation->set_select('miselect', 'tres'); ?>
>Tres</option>
</select>
```

set_checkbox()

Permite mostrar una casilla de verificación (checkbox) en el estado que fue enviada. El primer parámetro debe contener el nombre de la casilla, el segundo parámetro debe contener su valor. Ejemplo:

```
<input type="checkbox" name="micheck" value="1" <?= $this->validation-
>set_checkbox('micheck', '1'); ?> />
```

set_radio()

Permite mostrar una casilla de verificación (checkbox) en el estado que fue enviada. El primer parámetro debe contener el nombre de la casilla, el segundo parámetro debe contener su valor. Ejemplo:

```
<input type="radio" name="miradio" value="1" <?= $this->validation-
>set_radio('miradio', '1'); ?> />
```

Clases de XML-RPC y Servidor XML-RPC

Las clases de XML-RPC de CodeIgniter le permiten envíar peticiones a otro sevidor, o establecer su propio servidor XML-RPC para recibir peticiones.

Qué es XML-RPC?

De manera simple, es una forma de dos computadores para comunicarse a través de internet usando XML. Una computadora, a la que llamaremos *client*, envía un **pedido** XML-RPC a otra computadora, la cual llamaremos el *servidor*. Una vez que el servidor recibe y procesa la petición envíara una **respusta** al cliente.

Por ejemplo, usando la API MetaWeblog, un Cliente XML-RPC (usualmente un a herramienta de publicación de escritorio) enviará una petición a un Servidor XML-RPC corriendo en su sitio. Esta petición puede ser una nueva entrada de weblog siendo envíada para publicar, o puede ser un pedidod para una editar una entrada existente. Cuando el Servidor XML-RPC recibe esta petición la examinará para determinar que clase/método debe llamar para procesar el pedido. Una vez procesado, el servidor entonces enviara un mensaje de respuesta.

Para especificaciones detalladas, puede visitar el sitio http://www.xmlrpc.com/

Inicializando la clase

Como la mayoría de las otras clases en CodeIgniter, las clases XML-RPC y XML-RPCS son inicializadas en su controlador usando la función *$this->load->library*:

Para cargar la clase XML-RPC usará:

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

Una vez cargada, el objeto de la librería xml-rpc estará disponible usando: *$this->xmlrpc*

Para cargar la clase de Servidor XML-RPC usará:

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


Una vez cargado, el objeto de la librería xml-rpcs estará disponible usando: *$this->xmlrpcs*

> **Nota:** Cuando use la clase XML-RPC Server debe cargar AMBAS clases de XML-RPC y XML-RPC Server.

Mandando un Pedido XML-RPC

Para envíar un pedido a un servidor XML-RPC debe especificar la siguiente información:

- La URL del servidor
- El método del servidor que desea llamar
- Los datos de la *petición* (explicados debajo).

Aquí hay un ejemplo básico que envía un simple Weblogs.com ping al Ping-o-Matic (http://pingomatic.com/)

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

$this->xmlrpc->server('http://rpc.pingomatic.com/', 80);
$this->xmlrpc->method('weblogUpdates.ping');

$pedido = array('Mi Fotoblog', 'http://www.mi-site.com/photoblog/');
$this->xmlrpc->request($pedido);

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

Explicación

El código anterior usa la clase XML-RPC, establece la URL del servidor y el método a ser llamado (weblogUpdates.ping). El pedido (en este caso, el título y la URL del sitio) es ubicado en el arreglo para transportación y compilado usando la función request(). Por último, el pedido completo es enviado. Si el método *send_request()* devuelve falso, mostraremos el mensaje de error envíado desde el Servidor XML-RPC.

Anatomía de una Petición

Una *petición* XML-RPC es simplemente los datos que son enviados al servidor XML-RPC. Cada pieza de datos en un pedido es referido como un *parámetro de petición*. El

ejemplo anterior tiene dos parámetros: La URL y título de su sitio. Cuando el servidor XML-RPC recibe su petición, buscará los parámetros que requiera.

Los parámetros de petición deben ser ubicados en un arreglo para transportar, y cada parámetro puede ser uno de siete tipos de datos (cadenas, números, fechas, etc.). Si sus parámetros son algo distinto de cadenas debe incluir el tipo de dato en el arreglo de la petición.

Aquí hay un ejemplo de un simple arreglo con tres parámetros:

```
$peticion = array('John', 'Doe', 'www.algun-sitio.com');
$this->xmlrpc->request($peticion);
```

Si usa tipos de datos que no son cadenas, o si tiene diferentes tipos de datos, debe ubicar cada parámetro en su propio arreglo, con el tipo de dato en la segunda posición:

```
$peticion = array (
       array('John', 'string'),
       array('Doe', 'string'),
       array(FALSE, 'boolean'),
       array(12345, 'int')
       );
$this->xmlrpc->request($peticion); La sección Tipos de Datos de abajo tiene
una lista completa de tipos de datos.
```

Creando un Servidor XML-RPC

Un Servidor XML-RPC actúa como un policía de tráfico de cosas, esperando por peticiones entrantes y redirigiendolas a la función apropiada para procesamiento.

Crear su propio servidor XML-RPC implica inicializar la clase Servidor XML-RPC en su controlador, donde espera las peticiones entrantes que aparezcan, y luego estableciendo un arreglo con instrucciones de mapeo para que las peticiones entrantes puedan ser enviadas a la clase/método apropiada por procesamiento.

Aquí hay un ejemplo para ilustrar:

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

$config['functions']['nuevo_post']  = array('function' =>
'Mi_blog.nueva_entrada');
$config['functions']['actualizar_post'] = array('function' =>
'Mi_blog.actualizar_entrada');
$this->xmlrpcs->initialize($config);
$this->xmlrpcs->serve();
```


El ejemplo anterior contiene un arreglo especificando dos métodos de peticiones que el Servidor permite. Los métodos permitidos están en el lado izquierdo del arreglo. Cuando alguno de esos dos son recibidos, ellos serán enviados a la clase y método en la derecha.

En otras palabras, si un Cliente XML-RPC envía una petición al método *nuevo_post*, su servidor cargará la clase *Mi_blog* y llamará a la función *nueva_entrada*. Si la petición es para el método *actualizar_post*, su servidor cargará la clase *Mi_blog* y llamará a la función *update_entry*.

Los nombres de la función en el ejemplo aneterior son arbitrarios. Puede decidir como deben llamarse en su servidor, o si usa APIs estandarizadas, como Blogger o MetaWeblog API, usará el nombre de las funciones.

Procesando Peticiones al Servidor

Cuando el Servidor XML-RPC recibe una petición y carga la clase/método para procesar, pasará un objeto al método que contiene los datos enviados por el cliente.

Usando el ejemplo anterior, si el método *nuevo_post* es pedido, el servidor esperará que exista una clase con este prototipo:

```
class Mi_blog extends Controller {

    function nuevo_post($peticion)
    {

    }
}
```

La variable *$peticion* es un objeto compilado por el Servidor, que contiene los datos enviados por el Cliente XML-RPC. Usando este objeto tendrá acceso a los *parámetros de la petición* permitiéndole procesar el pedido. Cuando termine enviará una *Respuesta* de vuelta al Cliente.

Debajo hay un ejemplo del mundo real, usando la API de Blogger. Uno de los métodos en la API de Blogger es *getUserInfo()*. Usando este método, un Cliente XML-RPC puede enviar un usuario y contraseña, y en respuesta el Servidor envía información acerca del usuario en particular (nickname, user ID, dirección de email, etc.). Aquí es como la función de procesamiento podría ser:

```
class Mi_blog extends Controller {

    function getUserInfo($peticion)
    {
        $usuario = 'smitty';
        $contrasena = 'secretsmittypass';

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

        $parametros = $peticion->output_parameters();

        if ($parametros['1'] != $usuario AND $parametros['2'] != $contrasena)
        {
    return $this->xmlrpc->send_error_message('100', 'Acceso Invalido');
        }

        $respuesta = array(array('nickname'  => array('Smitty','string'),
               'userid'   => array('99','string'),
               'url'  => array('http://susitio.com','string'),
               'email'=> array('jsmith@susitio.com','string'),
               'apellido'  => array('Smith','string'),
               'nombre' => array('John','string')
               ),
        'struct');

        return $this->xmlrpc->send_response($respuesta);
    }
}
```

Notas:

La función *output_parameters()* devuelve e un arreglo indexado correspondiente a los parámetros de petición enviados por el cliente. En el ejemplo anterior, los parámetros de salida serían el usuario y la contraseña.

Si el usuario y la contraseña envíados por el cleinte no son válidos, un mensaje de error es devuelto usando *send_error_message()*.

Si la operación es exitosa, el cliente recibirá una respuesta con un arreglo conteniendo la información del usuario.

Dándole formato a una respuesta

Similar a una *Petición*, las *Respuestas* deben ser formateadas como un arreglo. Sin embargo, a diferencía de las peticiones, una respuesta es un arreglo **que contiene un sólo item**. El item puede ser un arreglo con varios arreglos adicionales, pero sólo puede ser un índice de arreglo principal. En otras palabras, el prototipo básico es este:

```
$respuesta = array('Datos respuesta', 'array');
```


Las respuestas, sin embargo, usualmente contienen múltiples piezas de información. Para cumplir con esto, debemos poner la respuesta en su propio arreglo, así el arreglo primario continúa conteniendo una sola pieza de datos. Aquí hay un ejemplo mostrando como puede ser logrado:

```
$respuesta = array (
       array(
       'nombre' => array('John', 'string'),
       'apellido' => array('Doe', 'string'),
       'miembro_id' => array(123435, 'int'),
       'lista_para_hacer' => array(array('limpiar la casa', 'llamar a mamá',
'regar las plantas'), 'array'),
       ),
       'struct'
       );
```

Note que el arreglo anterior es formateado como un *struct*. Este es el más común tipo de datos para respuestas.

Como con Peticiones, una respuesta puede ser uno de los siete tipos de datos listados en la sección Tipos de Datos.

Enviando una Respuesta de Error

Si necesita enviar al cliente una respuesta de error usará lo siguiente:

```
return $this->xmlrpc->send_error_message('123', 'Los datos pedidos no están
disponibles');
```

El primer parámetro es el número de error, mientras qeu el segundo parámetro es el mensaje de error.

Creando su Propio Cliente y Servidor

Para ayudar a entender todo lo que hemos cubierto hasta aquí, creemos un par de controladores que actúen como Servidores y Clientes XML-RPC. Usará el Cliente para enviar una petición al Servidor y recibir una respuesta.

El cliente

Usando un editor de texto, cree un controlador llamado *xmlrpc_client.php*. En él, ubique este código y guárdelo en su carpeta `applications/controllers/`:

```
<?php

class Xmlrpc_client extends Controller {

    function index()
    {
        $this->load->helper('url');
        $server_url = site_url('xmlrpc_server');

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

        $this->xmlrpc->server($server_url, 80);
        $this->xmlrpc->method('Greetings');

        $request = array('How is it going?');
        $this->xmlrpc->request($request);

        if ( ! $this->xmlrpc->send_request())
        {
            echo $this->xmlrpc->display_error();
        }
        else
        {
            echo '<pre>';
            print_r($this->xmlrpc->display_response());
            echo '</pre>';
        }
    }
}
?>
```

Nota: En el código anterior estamos usando un "asistente de url". Puede encontrar más información en la página de Funciones Asistentes.

El Servidor

Usando un editor de texto, cree un controlador llamado *xmlrpc_server.php*. En él, ubique este código y guárdelo en su carpeta `applications/controllers/`:

```
<?php

class Xmlrpc_server extends Controller {

    function index()
    {
        $this->load->library('xmlrpc');
        $this->load->library('xmlrpcs');

        $config['functions']['Greetings'] = array('function' =>
'Xmlrpc_server.process');

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


    function process($request)
    {
        $parameters = $request->output_parameters();

        $response = array(
                        array(
                               'you_said'  => $parameters['0'],
                               'i_respond' => 'Not bad at all.'),
                        'struct');

        return $this->xmlrpc->send_response($response);
    }
}
?>
```

Pruebelo!

Ahora visite su sitio usando una URL similar a esta:

```
www.su-sitio.com/index.php/xmlrpc_client/
```

Ahora verá el mensaje que envió al servidor, y la respuesta de vuelta a él.

El cliente que creo envía un mensaje ("Cómo estás?") al servidor, junto con la petición para el método "Saludos". El Servidor recibe la petición y lo asigna a la función "process", dónde una respuesta es devuelta.

Referencia de funciones XML-RPC

$this->xmlrpc->server()

Establece la URL y el número de puerta del servidor al que se le enviara el pedido:

```
$this->xmlrpc->server('http://www.aveces.com/pings.php', 80);
```

$this->xmlrpc->timeout()

Establece el período de espera (en segundos) después de cuando la petición será cancelada:

$this->xmlrpc->timeout(6);

$this->xmlrpc->method()

Establece el método que será pedido al servidor XML-RPC:

```
$this->xmlrpc->method('metodo');
```

Donde *método* es el nombre del método.

$this->xmlrpc->request()

Toma un arreglo de datos y construye la petición que se enviará al servidor XML-RPC:

```
$peticion = array(array('Mi Fotoblog', 'string'), 'http://www.susitio.com/
fotoblog/');
$this->xmlrpc->request($peticion);
```

$this->xmlrpc->send_request()

La función que envía la petición. Devuelve el booleano TRUE o FALSE basado en el éxito o fallo, permitiendo el uso condicionalmente.

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

Habilita la depuración, que mostrará una variedad de información y datos de errores útiles durante el desarrollo.

$this->xmlrpc->display_error()

Devuelve un mensaje de error como cadena si su petición falló por alguna razón.

```
echo $this->xmlrpc->display_error();
```

$this->xmlrpc->display_response()

Devuelve la respuesta desde el servidor remoto una vez que el pedido es recibido. La respuesta será típicamente un arreglo asociativo.

```
$this->xmlrpc->display_response();
```

$this->xmlrpc->send_error_message()

Esta función le permite enviar un mensaje de error desde un servidor al cliente. El primer parámetro es el número de error, mientras que el segundo parámetro es el mensaje de error.

```
return $this->xmlrpc->send_error_message('123', 'Los datos pedidos no están
disponibles');
```

$this->xmlrpc->send_response()

Le permite enviar la respuesta desde su servidor al cliente. Le permite enviar la respuesta desde su servidor al cliente. Un arreglo de datos válidos debe ser enviado a este método.

```
$respuesta = array(
        array(
        'flerror' => array(FALSE, 'boolean'),
        'mensaje' => "Gracias por el ping!")
            )
        'struct');
return $this->xmlrpc->send_response($respuesta);
```

Tipos de Datos

De acuerdo a las especificaciones de XML-RPC hay siete tipos de valores que pueden ser envíados a través de XML-RPC:

- *int* o *i4*
- *boolean*
- *string*
- *double*
- *dateTime.iso8601*
- *base64*
- *struct* (contiene un arreglo de valores)
- *array* (contiene un arreglo de valores)

Clase de Codificación Zip

La Clase de Codificación Zip de CodeIgniter le permite crear archivos Zip. Los archivos pueden ser descargados a su computadora o guardados en un directorio.

Initializando la Clase

Como la mayoría de las otras clases en CodeIgniter, La clase Zip es inicializada en su controlador usando la función *$this->load->library*:

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

Una vez cargada, el objeto de la librería Zip estará disponible usando: *$this->zip*

Ejemplos de Uso

Este ejemplo demuestra como comprimir un archivo, guardarlo en una carpeta en su servidor, y descargarlo a su computadora.

```
$nombre = 'misdatos1.txt';
$datos = 'Una Cadena de Datos!';

$this->zip->add_data($nombre, $datos);
```


```
// Escribe el archivo zip en una carpeta en su servidor. Lo nombra
"mi_copia_de_seguridad.zip"
$this->zip->archive('/ruta/al/directorio/mi_copia_de_seguridad.zip');

// Descarga el archivo a su escritorio. Lo nombra "mi_copia_de_seguridad.zip"
$this->zip->download('mi_copia_de_seguridad.zip');
```

Referencia de funciones

$this->zip->add_data()

Le permite agregar datos al archivo Zip. El primer parámetro debe contener el nombre que quisiera darle al archivo, el segundo parámetro debe contener los datos del archivo como una cadena:

```
$nombre = 'mi_bio.txt';
$datos = 'Yo nací en un ascencor...';

$this->zip->add_data($nombre, $datos);
```

Puede realizar múltiples llamadas a esta función para agregar varios archivos a su archivo. Ejemplo:

```
$nombre = 'misdatos1.txt';
$datos = 'Una Cadena de Datos!';
$this->zip->add_data($nombre, $datos);

$nombre = 'misdatos.txt';
$datos = 'Otra Cadena de Datos!';
$this->zip->add_data($nombre, $datos);
```

O puede pasar múltiples archivos usando como un arreglo:

```
$datos = array(
       'misdatos1.txt' => 'Una Cadena de Datos!',
       'misdatos2.txt' => 'Otra Cadena de Datos!'
   );

$this->zip->add_data($datos);

$this->zip->download('mi_copia_de_seguridad.zip');
```

Si quiere que sus datos comprimidos se organicen en sub-carpetas, incluya la ruta como parte del archivo:

```
$nombre = 'personal/mi_bio.txt';
$datos = 'Yo nací en un ascensor...';

$this->zip->add_data($nombre, $datos);
```

El ejemplo anterior ubicará *mi_bio.txt* dentro de la carpeta llamada `personal`.

$this->zip->add_dir()

Le permite agregar un directorio. Usualmente esta función no es necesaria ya que puede ubicar sus datos en carpetas cuando use *$this->zip->add_data()*, pero si quiere crear una carpeta vacía puede hacerlo.Ejemplo:

```
$this->zip->add_dir('micarpeta'); // Crea una carpeta llamada "micarpeta"
```

$this->zip->read_file()

Le permite comprimir un archivo que ya existe en algún lugar en su servidor. Suministra una ruta al archivo y la clase zip lo leerá y lo agregará al archivo:

```
$ruta = '/ruta/a/foto.jpg';

$this->zip->read_file($ruta);

// Descarga el archivo a su escritorio. Lo nombra "mi_copia_de_seguridad.zip"
$this->zip->download('mi_copia_de_seguridad.zip');
```

Si quiere que su archivo Zip mantenga la estructura de directorio donde se encuentra el archivo, pase TRUE (booleano) en el segundo parámetro. Ejemplo:

```
$ruta = '/ruta/a/foto.jpg';

$this->zip->read_file($ruta, TRUE);

// Descarga el archivo a su escritorio. Lo nombra "mi_copia_de_seguridad.zip"
$this->zip->download('mi_copia_de_seguridad.zip');
```

En el ejemplo anterior, *foto.jpg* será ubicado dentro de dos carpetas: `path/to/`

$this->zip->read_dir()

Le permite comprimir una carpeta (y su contenido) que ya existe en algún lugar en su servidor. Suministra la ruta al directorio y la clase zip lo leerá recursivamente y lo recreará como archivo Zip. Todos los archivos contenidos dentro de la ruta suministrada serán codificados, así también cualquier subcarpeta contenida en él. Ejemplo:

```
$ruta = '/ruta/a/su/directorio/';

$this->zip->read_dir($ruta);

// Descarga el archivo a su escritorio. Lo nombra "mi_copia_de_seguridad.zip"
$this->zip->download('mi_copia_de_seguridad.zip');
```

$this->zip->archive()

Escribe el archivo codificado Zip a un directorio en su servidor. Suministra una ruta de servidor válida finalizando en el nombre del archivo. Asegurese que el directorio es escriturable (666 o 777 usualmente está OK). Ejemplo:

```
$this->zip->archive('/ruta/a/carpeta/miarchivo.zip'); // Crea un archivo
llamado miarchivo.zip
```

$this->zip->download()

Causa que el archivo Zip sea descargado a su computadora. La función debe Causes the Zip file to be downloaded to your server. A la función debe pasarle el nombre que quiere que el zip sea llamado. Ejemplo:

```
$this->zip->download('cosas_nuevas.zip'); // El archivo será llamado
"cosas_nuevas.zip"
```

Nota: No muestre ningún dato en el controlador en el cual llama a esta función ya que envía varios encabezados que causa que la descarga ocurra y el archivo sea tratado como binario.

$this->zip->get_zip()

Devuelve los datos comprimidos del archivo Zip. Generalmente no necesitará esta función a menos que quiera hacer algo único con los datos. Ejemplo:

```
$nombre = 'mi_bio.txt';
$datos = 'Nací en un ascensor...';

$this->zip->add_data($nombre, $datos);

$zip_file = $this->zip->get_zip();
```

$this->zip->clear_data()

La clase de Zip guarda todos sus datos así no necesita recompilar el archivo Zip por cada llamada que use anteriormente. Si, sin embargo, necesita crear múltiples Zip, cada uno con diferentes datos, puede limpiar el caché entre llamadas. Ejemplo:

```
$nombre = 'mi_bio.txt';
$datos = 'Nací en un ascensor...';

$this->zip->add_data($nombre, $datos);
$zip_file = $this->zip->get_zip();

$this->zip->clear_data();

$nombre = 'foto.jpg';
$this->zip->read_file("/ruta/a/photo.jpg"); // Lee el contenido del archivo


$this->zip->download('miphotos.zip');
```


Referencia de Asistentes

Helpers

Esta página está en blanco de forma intencional

Asistente de Arreglo

El archivo Asistente de Arreglo contiene funciones que asisten al trabajo con arreglos.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

element()

Le permite traer un item de un arreglo. La función prueba si el índice del arreglo existe y si tiene un valor. Si un valor existe, es devuelto. Si el valor no existe, devuelve FALSE o lo que haya especificado como valor por defecto a través del tercer parámetro. Ejemplo:

```
$arreglo = array('color' => 'rojo', 'forma' => 'redondo', 'tamaño' => '');

// devuelve "rojo"
echo element('color', $arreglo);

// devuelve NULL
echo element('tamaño', $arreglo, NULL);
```

random_element()

Toma un arreglo como entrada y devuelve un elemento aleatorio de él. Ejemplo de uso:

```
$citas = array(
    "I find that the harder I work, the more luck I seem to have. - Thomas
Jefferson",
    "Don't stay in bed, unless you can make money in bed. - George Burns",
    "We didn't lose the game; we just ran out of time. - Vince Lombardi",
    "If everything seems under control, you're not going fast enough. - Mario
Andretti",
    "Reality is merely an illusion, albeit a very persistent one. - Albert
Einstein",
    "Chance favors the prepared mind - Louis Pasteur"
    );

echo random_element($citas);
```

Compatibility Helper

The Compatibility Helper file contains PHP 4 implementations of some PHP 5 only native PHP functions and constants. This can be useful if you'd like to take advantage of some of these native function but your application may end up running on a PHP 4 server. In these cases, it may be advantageous to Auto-load the Compatibility Helper so you do not have to load it in each controller.

> Note: There are a few compatibility functions that are in CodeIgniter's native Compat.php file. You may use those functions without loading this helper. The functions are split between that file and this Helper so that only functions required by the framework are included by default. This way, whether or not you load the additional functions in this Helper remains your choice.

Loading this Helper

This helper is loaded using the following code:

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

Available Constants

The following constants are available:

PHP_EOL

The newline character for the server's current OS, e.g. on Windows systems "\r\n", on *nix "\n".

Available Functions

The following functions are available (see linked PHP documentation for documentation):

file_put_contents() - The fourth parameter, $context, is not supported.

fputcsv()

http_build_query()

str_ireplace() - The fourth parameter, $count, is not supported, as PHP 4 would make it become required.

stripos()

Asistente de Cookie

El archivo Asistente de Cookie contiene funciones que asisten al trabajo con cookies.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

set_cookie()

Establece una cookie conteniendo el valor especificado. Hay dos formas de pasar información a esta función para que la cookie pueda ser establecida: Método de Arreglo, y Parámetros Discretos:

Método de Arreglo

Usando este método, un arreglo asociativo es pasado al primer parámetro:

```
$cookie = array(
        'name' => 'El nombre de la Cookie',
        'value'  => 'El Valor',
        'expire' => '86500',
        'domain' => '.un-dominio.com',
        'path' => '/',
        'prefix' => 'miprefijo_',
    );

set_cookie($cookie);
```

Notas:

Sólo el nombre y valor son requieridos.

La expiración es establecida en **segundos**, que serán agregados al horario actual. No incluya el horario, sino sólo el número de segundos desde *ahora* que desea que la cookie sea valida. Si la expiración es establecida como cero, la cookie sólo durará tanto como el explorador esté abierto.

Para borrar una cookie establezcala con expiración vacía.

Para cookies a lo ancho del sitio independientemente de como el sitio es requerido, agregue su URL al **dominop** empezando con punto, como esto: .tu-dominio.com

La ruta es usualmente no necesaria ya que la función establece una ruta a la raíz.

El prefijo es sólo necesario para evitar colisioens de nombre con otros nombres de cookies idénticos para su servidor.

Parámetros Discretos

Si prefieres, puede establecer una cookie pasando datos usando parámetros individuales:

```
set_cookie($nombre, $valor, $expiracion, $dominio, $ruta, $prefijo);
```


get_cookie()

Le permite traer una cookie. El primer parámetro contendrá el nombre de la cookie que está buscando:

```
get_cookie('alguna_cookie');
```

Esta función devuelve FALSE (booleano) si el item que está intentando recuperar no existe.

El segundo parámetro opcional le permite correr los datos a través de un filtro XSS. Está hablitado al establecer el segundo parámetro como el booleano TRUE;

```
get_cookie('alguna_cookie', TRUE);
```

delete_cookie()

Le permite eliminar una cookie. A menos que haya establecido una ruta especial u otros valores, sólo el nombre de la cookie es requerido:

```
delete_cookie("name");
```

Esta función es de otra forma idéntica a *set_cookie()*, excepto que no tiene los parámetros del valor y la expiración. Puede enviar un arreglo de valores en el primer parámetro o establecerlos de forma discreta.

```
delete_cookie($nombre, $dominio, $ruta, $prefijo)
```

Asistente de Fecha

El archivo Asistente de Fecha contiene fucniones que ayudan a trabajar con fechas.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

now()

Devuelve el valor actual como una marca de tiempo de Unix, referenciando al horario local de su servidor o GMT, basado en el "tiempo de referencia" establecido en su archivo de configuración. Si no intenta establecer un tiempo de referencia maestro a GMT (lo cuál tipicamente hará si corre un sistio que le permite a cada usuario establecer su propia configuración de zona horaria) no hay beneficio de usar esta función sobre la función time() de PHP.

mdate()

Esta función es idéntica a la función de PHP date(), excepto que le permite usar estilo de código de fecha de MySQL, conde cada letra de código es precedida con un signo de porcentaje: %Y %m %d etc.

El beneficio de usar fechas de esta forma es que no tiene que preocuparse acerca de escapar cualquier caracter que no sea código de fecha, cómo normalmente tiene que hacer con la función date(). Ejemplo:

```
$cadenafecha = "Año: %Y Mes: %m Día: %d - %h:%i %a";
$tiempo = time();

echo mdate($cadenafecha, $tiempo);
```

Si una marca de tiempo no es incluida en el segundo parámetro, el horario actual será usado.

standard_date()

Le permite generar cadenas de fecha en una de los varios formatos estándares. Ejemplo:

```
$formato = 'DATE_RFC822';
$tiempo = time();

echo standard_date($formato, $tiempo);
```

El primer parámetro debe contener el formato, el segundo parámetro debe contener una fecha como marca de tiempo de Unix.

Formatos soportados:

- DATE_ATOM
- DATE_COOKIE
- DATE_ISO8601
- DATE_RFC822
- DATE_RFC850
- DATE_RFC1036
- DATE_RFC1123
- DATE_RFC2822
- DATE_RSS
- DATE_W3C

local_to_gmt()

Toma una marca de tiempo de Unix como entrada y lo devuelve como GMT. Ejemplo:

```
$ahora = time();

$gmt = local_to_gmt($ahora);
```

gmt_to_local()

Tome una marca de tiempo de Unix (referenciado a GMT) como entrada, y lo convierte en una marca de tiempo localizado basado en la zona horaria y tiempo de ahorro de energía envíado. Ejemplo:

```
$marca_de_tiempo = '1140153693';
$zonadetiempo = 'UM8';
$ahorro_de_energia = TRUE;

echo gmt_to_local($marca_de_tiempo, $zonadetiempo, $ahorro_de_energia);
```

Nota: Para una lista de zonas horarias vea la referencia al final de esta página.

mysql_to_unix()

Toma una marca de tiempo como entrada y lo devuelve como Unix. Ejemplo:

```
$mysql = '20061124092345';

$unix = mysql_to_unix($mysql);
```

unix_to_human()

Toma una marca de tiempo Unix como entrada y la devuelve en formato leíble por humanos con este prototipo:

```
AAAA-MM-DD HH:MM:SS AM/PM
```

Esto puede ser útil si necesita mostrar una fecha en un campo de formulario para envío.

Al tiempo se le puede dar formato con o sin segundos, y puede ser establecido para el formato europeo o estadounidanse. Si sólo la marca de tiempo es enviada, devolverá el tiempo sin los segundos con formato de EE.UU. Ejemplos:

```
$ahora = time();
echo unix_to_human($ahora); // tiempo de EE.UU., sin segundos
echo unix_to_human($ahora, TRUE, 'us'); // tiempo de EE.UU., con segundos
echo unix_to_human($ahora, TRUE, 'eu'); // Tiempo de Europa con segundos
```


human_to_unix()

Lo opuesto a la función anterior. Toma un tiempo "humano" como entrada y lo devuelve como Unix. Esta función es útil si acepto que fechas con formato "humano" dates sean enviados a través de un formulario. Devuelve FALSE (booleano) si la cadena de fecha pasada no tiene el formato como se indica arriba. Ejemplo:

```
$ahora = time();

$humano = unix_to_human($ahora);

$unix = human_to_unix($humano);
```

timespan()

Da formato a una marca de tiempo unix para que aparezca de una forma similar a esta:

```
1 Año, 10 Meses, 2 Semanas, 5 Días, 10 Horas, 16 Minutos
```

El primer parámetro debe contener una marca de tiempo Unix. El segundo debe contener una marca de tiempo que sea mayor a la primera. Si el segundo es vacío, el horario actual será usado. El propósito más común para esta función es mostrar cuando tiempo pasó de un punto en el tiempo pasado a ahora. Ejemplo:

```
$fecha_de_envio = '1079621429';
$ahora = time();

echo timespan($fecha_de_envio, $ahora);
```

Nota: El texto generado por esta función es encontrado en el siguiente archivo de lenguaje: language/<your_lang>/date_lang.php

days_in_month()

Devuelve el número de días de un mes/año dado. Toma años bisiestos en cuenta. Ejemplo:

```
echo days_in_month(06, 2005);
```

Si el segundo parámetro es vacío, el año actual será usado.

timezones()

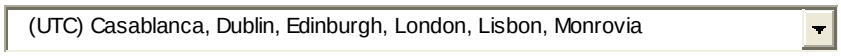
Toma una zona horaria de referencia (por una lista de zonas horarias válidas, vea la "Referencia de Zona Horaria" debajo) y devuelve un número de horas desde UTC.

```
echo timezones('UM5');
```

Esta función es útil cuando se usa con timezone_menu().

timezone_menu()

Genera un menú desplegable de zonas horarias como este:



El menú es útil si corre un sitio con membresia en el cual los usuarios pueden establecer su valor de zona horaria local.

El primer parámetro permite establecer el estado "seleccionado" del menu. Por ejemplo, para establecer horario del Pacífico como predeterminado haría esto:

```
echo timezone_menu('UM8');
```

Por favor vea la referencia de zona horaria debajo para ver los valores de este menú.

El segundo parámetro permite establecer el atributo class de CSS del menú.

Nota: El texto contenido en el menú es encontrado en el siguiente archivo de lenguaje: language/<your_lang>/date_lang.php

Referencia de Zona Horaria

La siguiente tabla indica cada zona horaria y su ubicación.

Time Zone	Location
UM12	(UTC - 12:00) Enitwetok, Kwajalien
UM11	(UTC - 11:00) Nome, Midway Island, Samoa
UM10	(UTC - 10:00) Hawaii
UM9	(UTC - 9:00) Alaska
UM8	(UTC - 8:00) Pacific Time
UM7	(UTC - 7:00) Mountain Time
UM6	(UTC - 6:00) Central Time, Mexico City
UM5	(UTC - 5:00) Eastern Time, Bogota, Lima, Quito
UM4	(UTC - 4:00) Atlantic Time, Caracas, La Paz
UM25	(UTC - 3:30) Newfoundland
UM3	(UTC - 3:00) Brazil, Buenos Aires, Georgetown, Falkland Is.
UM2	(UTC - 2:00) Mid-Atlantic, Ascention Is., St Helena
UM1	(UTC - 1:00) Azores, Cape Verde Islands
UTC	(UTC) Casablanca, Dublin, Edinburgh, London, Lisbon, Monrovia
UP1	(UTC + 1:00) Berlin, Brussels, Copenhagen, Madrid, Paris, Rome
UP2	(UTC + 2:00) Kaliningrad, South Africa, Warsaw
UP3	(UTC + 3:00) Baghdad, Riyadh, Moscow, Nairobi
UP25	(UTC + 3:30) Tehran
UP4	(UTC + 4:00) Adu Dhabi, Baku, Muscat, Tbilisi
UP35	(UTC + 4:30) Kabul
UP5	(UTC + 5:00) Islamabad, Karachi, Tashkent
UP45	(UTC + 5:30) Bombay, Calcutta, Madras, New Delhi
UP6	(UTC + 6:00) Almaty, Colomba, Dhakra
UP7	(UTC + 7:00) Bangkok, Hanoi, Jakarta
UP8	(UTC + 8:00) Beijing, Hong Kong, Perth, Singapore, Taipei
UP9	(UTC + 9:00) Osaka, Sapporo, Seoul, Tokyo, Yakutsk
UP85	(UTC + 9:30) Adelaide, Darwin
UP10	(UTC + 10:00) Melbourne, Papua New Guinea, Sydney, Vladivostok
UP11	(UTC + 11:00) Magadan, New Caledonia, Solomon Islands
UP12	(UTC + 12:00) Auckland, Wellington, Fiji, Marshall Island

Asistente de Directorio

El archivo Asistente de Directorio contiene funciones que asisten al rtabajo con directorios.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

directory_map('*directorio de origen*')

Esta función lee la ruta de directorio especificada en el primer parámetro y construye un arreglo de él y todos los archivos contenidos. Ejemplo:

```
$mapa = directory_map('./midirectorio/');
```

Nota: Las rutas son casi siempre relativas a tu archivo principal index.php.

Las subcarpetas contenidas dentro del directorio serán mapeadas también. Si desea hacer un mapa de sólo el nivel de directorio superior establezca el segundo parámetro a *true* (booleano):

```
$mapa = directory_map('./midirectorio/', TRUE);
```

Cada nombre de carpeta será un índice del arreglo, mientras los archivos que contenga tendrás índices numéricos. Aquí hay un ejemplo de un arreglo típico:

```
Array
(
    [libraries] => Array
    (
    [0] => benchmark.html
    [1] => config.html
    [database] => Array
    (
    [0] => active_record.html
    [1] => binds.html
    [2] => configuration.html
    [3] => connecting.html
    [4] => examples.html
    [5] => fields.html
    [6] => index.html
    [7] => queries.html
        )
    [2] => email.html
    [3] => file_uploading.html
    [4] => image_lib.html
    [5] => input.html
    [6] => language.html
    [7] => loader.html
    [8] => pagination.html
    [9] => uri.html
)
```

Asistente de Descarga

El Asistente de Descarga le permite descargar datos a su computadora.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

force_download('*nombre de archivo*', '*datos*')

Genera cabeceras de servidor que fuerzan que los datos sean descargados a su computadora. Útil con archivos de descarga. El primer parámetro es el **nombre que**

quiere que el archivo descargado posea, el segundo parámetro son los datos de archivo. Ejemplo:

```
$datos = 'Aquí hay algo de texto!';
$nombre = 'mitexto.txt';

force_download($nombre, $datos);
```

Si quiere descargar un archivo existente de su servidor necesitará leer el archivo dentro de una cadena:

```
$datos = file_get_contents("/path/to/photo.jpg"); // Leer el contenido del
archivo
$nombre = 'mifoto.jpg';

force_download($nombre, $datos);
```

Asistente de Email

El Asistente de Email provee algunas funciones de asistencia para trabajar con Email. Para una solución más robusta de emails, vea la Clase de Email de CodeIgniter.

Cargando este Asistente

Este asistente se carga usando el siguiente código:

$this->load->helper('email');

Las siguientes funciones están disponibles:

valid_email('*email*')

Verifica si un email tiene un formato correcto de email. Note que en realidad no prueba que el email reciba el correo, simplemente que es una dirección validamente formada.

Devuelve TRUE/FALSE

```
$this->load->helper(email);

if (valid_email('email@algunsitio.com'))
{
    echo 'email es valido';
}
else
{
    echo 'email no es valido';
}
```

send_email('*recipients*', '*asunto*', '*mensaje*')

Envía un email usando la función nativa de PHP mail(). Para una solución más robusta, vea la Clase de Email de CodeIgniter.

Asistente de Archivo

El Asistente de Archivo contiene funciones que asisten al trabajo con archivos.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las funciones siguientes están disponibles:

read_file('*ruta*')

Devuelve los datos contenidos en el archivo especificado en la ruta. Ejemplo:

```
$cadena = read_file('./ruta/al/archivo.php');
```

La ruta puede ser relativa o absoluta del servidor. Devuelve FALSE (booleano) en fallo.

Nota: La ruta es relativa al archivo principal del sitio index.php, NO de tus archivos controladores o vistas. CodeIgniter usa un controlador frontal, así que las rutas siemrpe son relativas al índice del sitio principal.

Si tu servidor corre con la restricción open_basedir esta función puede no funcionar si está tratando de acceder a un archivo superior al script llamado.

write_file('*ruta*', $datos)

Escribe los datos al archivo especificado en la ruta. Si el archivo no existe, la función lo creará. Ejemplo:

```
$datos = 'Algunos datos de archivo';

if ( ! write_file('./ruta/al/archivo.php', $datos))
{
     echo 'No se pudo escribir el archivo';
}
else
{
     echo 'Archivo escrito!';
}
```

Puede opcionalmente establecer el modo de escritura a través del tercer parámetro:

```
write_file('./ruta/al/archivo.php', $datos, 'r+');
```

El modo por defecto es wb. Por favor vea la guía del usuario de PHP para opciones de modo.

Nota: Para que esta función escriba datos en un archivo sus permisos deben ser tales que sea escriturable (666, 777, etc.). Si el archivo no existe todavía, el directorio que lo contiene debe ser escriturable.

Nota: La ruta es relativa al archivo principal del sitio index.php, NO de tus archivos controladores o vistas. CodeIgniter usa un controlador frontal, así que las rutas siemrpe son relativas al índice del sitio principal.

delete_files('*ruta*')

Borra TODOS los archivos contenidos en la ruta suuministrada. Ejemplo:

```
delete_files('./ruta/a/directorio/');
```

So el segundo parámetro es establecido como `true`, cualquier directorio contenido dentro de la raíz de la ruta será eliminado también. Ejemplo:

```
delete_files('./ruta/a/directorio/', TRUE);
```

Nota: Los archivos deben ser escriturables o del sistema para poder ser borrados.

get_filenames('*ruta/a/directorio/*')

Toma una ruta del servidor como entrada y devuelve un arreglo que contiene el nombre de toso los archivos contenidos en él. La ruta puede ser opcionalmente agregada a los nombres de los archivos estableciendo el segundo parámetro como TRUE.

Asistente de Formulario

El archivo Asistente de Formulario contiene funciones que asisten al trabajo con formularios.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

form_open()

Crea una etiqueta de formulario con la URL base **construida desde su archivo de configuración**. Puede opcionalmente agregar atributos de formularios y campos de entrada ocultos.

El principal beneficio de usar esta etiqueta en vez de hardcodear su propio HTML es que permite que su sitio sea más portable en caso de que sus URLs cambien alguna vez.

Aquí hay un ejemplo simple:

```
echo form_open('email/enviar');
```

El ejemplo de arriba crea un formulario que apunta a su URL base más los segmentos "email/enviar", como esto:

```
<form method="post" action="http:/www.tu-sitio.com/index.php/email/enviar" />
```

Agregando atributos

Los atributos pueden ser agregados pasando un arreglo asociativo como segundo parámetro, así:

```
$atributos = array('class' => 'email', 'id' => 'miformulario');

echo form_open('email/enviar', $atributos);
```

El ejemplo previo crea un formulario similar a esto:

```
<form method="post" action="http:/www.tu-sitio.com/index.php/email/enviar"
 class="email"  id="miformulario" />
```

Agregando Campos de Entrada Ocultos

Campos ocultos pueden ser agregados pasando un arreglo asociativo como tercer parámetro, así:

```
$oculto = array('username' => 'Jose', 'miembro_id' => '234');

echo form_open('email/enviar', '', $oculto);
```

El ejemplo previo crearía un formulario similar a este:

```
<form method="post" action="http:/www.your-site.com/index.php/email/enviar">
<input type="hidden" name="username" value="Jose" />
<input type="hidden" name="miembro_id" value="234" />
```

form_open_multipart()

Esta función es absolutamente identica a la etiqueta *form_open()* de arriba, excepto que agrega el atributo multipart, el cual es necesario si desea usar el formulario para subir archivos.

form_hidden()

Le permite generar campos de entrada ocultos. Puede enviar un nombre/valor para crear un campo:

```
form_hidden('username', 'johndoe');

// Produciría:

<input type="hidden" name="username" value="johnodoe" />
```

O puede envíar un arreglo asociativo para crear mútiples campos:

```
$datos = array(
    'nombre'  => 'John Doe',
    'email' => 'john@example.com',
    'url'  => 'http://www.example.com'
    );

echo form_hidden($datos);

// Produciría:

<input type="hidden" name="nombre" value="John Doe" />
<input type="hidden" name="email" value="john@example.com" />
<input type="hidden" name="url" value="http://www.example.com" />
```

form_input()

Le permite generar un campo de texto de entrada estándar. Puede pasar minimamente el nombre del campo y el valor en el primer y segundo parámetro:

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

O puede pasar un arreglo asociativo que contenga cualquier dato que quiera que su formulario contenga:

```
$datos = array(
    'name'     => 'username',
    'id'       => 'username',
    'value'    => 'johndoe',
    'maxlength'=> '100',
    'size'     => '50',
    'style'    => 'width:50%',
    );

echo form_input($datos);

// Produciría:

<input type="text" name="username" id="username" value="johndoe"
maxlength="100" size="50" style="width:50%" />
```

Si desearía que su formulario contenga algún dato adicional, como JavaScript, puede pasar una cadena en el tercer parámetro:

```
$js = 'onClick="alguna_funcion()"';

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

form_password()

Esta función es idéntica en todo aspecto a la función *form_input()* de arriba, excepto que establece el tipo "password".

form_upload()

Esta función es idéntica en todo aspecto a la función *form_input()* de arriba excepto que establece el tipo "file", permitiendo ser usado para subir archivos.

form_textarea()

Esta función es idéntica en todo aspecto a la función *form_input()* de arriba excepto que genera un tipo "textarea". Nota: En vez de los atributos "maxlength" y "size" en el ejemplo anterior, especificará "rows" y "cols".

form_dropdown()

Le permite crear un campo de menú desplegable estándar. El primer parámetro contendrá el nombre del campo, el segundo parámetro contendrá un arreglo asociativo de opciones, y el tercer parámetro contendrá el valor que desea sea preseleccionado. También puede pasar un arreglo a través del tercer parámetro, y CodeIgniter creará un select mútiple para tí. Ejemplo:

```
$opciones = array(
        'small'  => 'Remera Pequeña',
        'med'  => 'Remera Mediana',
        'large'   => 'Remera Grande',
        'xlarge' => 'Remera Extra Grande',
        );

$remeras_a_la_venta = array('small', 'large');

echo form_dropdown('remeras', $opciones, 'large');

// Producirá:

<select name="shirts">
<option value="small">Remera Pequeña</option>
<option value="med">Remera Mediana</option>
<option value="large" selected="selected">Remera Grande</option>
<option value="xlarge">Remera Extra Grande</option>
</select>

echo form_dropdown('remeras', $opciones, $remeras_a_la_venta);

// Producirá:

<select name="shirts" multiple="multiple">
<option value="small" selected="selected">Remera Pequeña</option>
<option value="med">Remera Mediana</option>
<option value="large" selected="selected">Remera Grande</option>
<option value="xlarge">Remera Extra Grande</option>
</select>
```

Si desea que el <select> de apertura contenga datos adicionales, como JavaScript, puede pasar una cadena en el cuarto parámetro:

```
$js = 'onChange="alguna_funcion()"';

echo form_dropdown('remeras', $opciones, 'large', $js);
```

form_fieldset()

Le permite generar campos fieldset/legend.

```
echo form_fieldset('Información de Dirección');
echo "<p>Contenido del Fieldset</p>\n";
echo form_fieldset_close();

// Produce
<fieldset id="address_info">
<legend>Información de Dirección</legend>
<p>Contenido del Fieldset</p>
</fieldset>
```

Similarmente a otras funciones, puede enviar un arreglo asociativo en el segundo parámetro si prefiere establecer atributos adicionales.

```
$atributos = array('id' => 'address_info', 'class' => 'address_info');
echo form_fieldset('Información de Dirección', $atributos);
echo "<p>Contenido del Fieldset</p>\n";
echo form_fieldset_close();

// Produce
<fieldset id="address_info" class="address_info">
<legend>Address Information</legend>
<p>form content here</p>
</fieldset>
```

form_fieldset_close()

Produce una etiqueta </fieldset> de cierre. La única ventaja de usar esta función es que permite pasar datos que serán agregados debajo de la etiqueta. Por ejemplo:

```
$cadena = "</div></div>";

echo fieldset_close($cadena);

// Produce:
</fieldset>
</div></div>
```


form_checkbox()

Le permite generar campos de casillas de verificación (checkbox). Ejemplo simple:

```
echo form_checkbox('newsletter', 'aceptar', TRUE);

// Produce:

<input type="checkbox" name="newsletter" value="aceptar" checked="checked" />
```

El tercer parámetro contiene un booleano TRUE/FALSE para determinar si la casilla debe estar marcada o no.

Similarmente a las otras funciones en este asistente, también puede pasar un arreglo de atributos a la función:

```
$datos = array(
    'name'     => 'newsletter',
    'id'       => 'newsletter',
    'value'=> 'aceptar',
    'checked'  => TRUE,
    'style' => 'margin:10px',
    );

echo form_checkbox($datos);

// Produce:

<input type="checkbox" name="newsletter" id="newsletter" value="aceptar"
checked="checked" style="margin:10px" />
```

Como con otras funciones, si quiere que la etiqueta contenga datos adicionales, como JavaScript, puede pasar una cadena en el cuarto parámetro:

```
$js = 'onClick="alguna_funcion()"';

echo form_checkbox('newsletter', 'aceptar', TRUE, $js)
```

form_radio()

Esta función es idéntica a la función *form_checkbox()* en todo aspecto excepto que establece el tipo "radio".

form_submit()

Le permite generar un botón de envío estándar. Ejemplo simple:

```
echo form_submit('mienvio', 'Enviar mensaje!');

// Produce:

<input type="submit" name="mienvio" value="Enviar mensaje!" />
```

Similar a otras funciones, puede envíar un arreglo asociativo en el primer parámetro si prefiere establecer sus propios atributos. El tercer parámetro le deja establecer datos adicionales a su formulario, como JavaScript.

form_label()

Le permite generar una etiqueta <label>. Ejemplo simple:

```
echo form_label('Cual es tu nombre', 'username');

// Produce:
<label id="username">Cual es tu nombre</label>
```

Similar a las otras funciones, puede enviar un arreglo asociativo en el tercer parámetro si prefiere establecer atributos adicionales.

```
$atributos = array(
'class' => 'miclase',
'style' => 'color: #000;',
);
echo form_label('Cual es tu nombre', 'username', $atributos);

// Produce:
<label id="username" class="miclase" style="color: #000;">Cual es tu nombre</
label>
```

form_reset()

Le permite generar un botón estándar de restablecer. El uso es idéntico a *form_submit()*.

form_close()

Produce una etiqueta de cierre </form>. La única ventaja de usar esta función es que le permite pasar datos que serán agregador debajo de la etiqueta. Por ejemplo:

```
$cadena = "</div></div>";

echo form_close($cadena);

// Produce:

</form>
</div></div>
```

form_prep()

Permite usar HTML y caracteres como comillas dentro de elementos de formulario en forma segura, sin romperlo. Considere este ejemplo:

```
$cadena = 'Aquí hay una cadena que contiene texto entre "comillas".';

<input type="text" name="myform" value="$cadena" />
```

Ya que la cadena anterior contiene un juego de comillas hará que el formulario se rompa. La función form_prep conviente HTML para que pueda ser usado con seguridad:

```
<input type="text" name="myform" value="<?php echo form_prep($cadena); ?>" />
```

Nota: Su ysa cualquiera de las funciones del asistente de formularios listados en esta página, los valores de formularios serán preparados automáticamente, así que no hay necesidad de llamara esta función. Úsela sólo si crea sus propios elementos.

Asistente de HTML

El archivo Asistente de HTML contiene funciones que asisten en el trabajo con HTML.

Cargando estos Asistentes

Este Asistente se carga usando el siguiente código:

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

Las siguientes funciones estan disponibles:

br()

Genera etiquetas de salto de línea (
) basado en el número que se le presente. Por ejemplo:

```
echo br(3);
```

Lo de arriba produce:

heading()

Le permite crear etiquetas HTML <h1>. El primer parámetro contendrá el dato, el segundo el tamaño del encabezado. Por ejemplo:

```
echo heading('Welcome!', 3);
```

Lo de arriba produce: <h3>Welcome!</h3>

img()

Le permite crear etiquetas HTML <img />. El primer parámetro contiene la fuente de la imágen. No hay datos como segundo parámetro. Por ejemplo:

```
echo img('images/picture.jpg');
// Da <img src="http://site.com/images/picture.jpg" />
```


Existe un segundo parámetro opcional que es un valor TRUE / FALSE que especifica si el src debería haber especificado la página de $config['index_page'] añadida a la dirección que crea. Presumiblemente, este sería si se utiliza un controlador como medio de comunicación.

```
echo img('images/picture.jpg', TRUE);
// Da <img src="http://site.com/index.php/images/picture.jpg" />
```

Además, un arreglo asociativo puede ser pasado a la función img () para un control completo de todos los atributos y valores.

```
$image_properties = array(
        'src' => 'images/picture.jpg',
        'alt' => 'Yo, demostrando como comer 4 de pizza de pizza a la vez',
        'class' => 'post_images',
        'width' => '200',
        'height' => '200',
        'title' => 'Esa fue una noche pesada',
        'rel' => 'lightbox',
);

img($image_properties);
// <img src="http://site.com/index.php/images/picture.jpg" alt="Yo,
demostrando como comer 4 de pizza de pizza a la vez" class="post_images"
width="200" height="200" title="Esa fue una noche pesada" rel="lightbox" />
```

link_tag()

Le permite crear etiquetas HTML <link />. Esto es útil para los enlaces de hojas de estilo, así como para otros enlaces. Los parámetros son href, como opcional rel, type, title, media e index_page. index_page es un valor TRUE/FALSE que especifica si el href debería haber sido especificado por la página de $config['index_page'] agregado a la dirección que creo. echo link_tag('css/mystyles.css');
// Da <link href="http://site.com/css/mystyles.css" rel="stylesheet" type="text/css" />

Otros ejemplos:

```
echo link_tag('favicon.ico', 'shortcut icon', 'image/ico');
// <link href="http://site.com/favicon.ico" rel="shortcut icon" type="image/
ico" />

echo link('feed', 'alternate', 'application/rss+xml', 'My RSS Feed');
// <link href="http://site.com/feed" rel="alternate" type="application/
rss+xml" title="My RSS Feed" />
```

Además, un arreglo asociativo puede ser pasado a la función img () para un control completo de todos los atributos y valores.

```
$link = array(
       'href' => 'css/printer.css',
       'rel' => 'stylesheet',
       'type' => 'text/css',
       'media' => 'print'
);

echo link_tag($link);
// <link href="http://site.com/css/printer.css" rel="stylesheet" type="text/
css" media="print" />
```

nbs()

Genera los espacios no-rompibles ( ) basados en el número que presente. Por ejemplo:

```
echo nbs(3);
```

Lo de arriba produce:

ol() and ul()

Le permite generar listas HTML ordenadas o desordenadas de arreglos simplres o multi-dimensionales. Por ejemplo:

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

$list = array(
    'red',
    'blue',
    'green',
    'yellow'
    );

$attributes = array(
           'class' => 'boldlist',
           'id'   => 'mylist'
           );

echo ul($list, $attributes);
```

El código anterior produciría esto:

```
<ul class="boldlist" id="mylist">
  <li>red</li>
  <li>blue</li>
  <li>green</li>
  <li>yellow</li>
</ul>
```

Este es un ejemplo mas complejo, usando un arreglo multi-dimensional:

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

$list = array(
    'colors' => array(
                'red',
                'blue',
                'green'
            ),
    'shapes' => array(
                'round',
                'square',
                'circles' => array(
                    'ellipse',
                    'oval',
                    'sphere'
                    )
            ),
    'moods'=> array(
                'happy',
                'upset' => array(
                    'defeated' => array(
                        'dejected',
                        'disheartened',
                        'depressed'
                        ),
                    'annoyed',
                    'cross',
                    'angry'
                )
            )
    );


echo ul($list);
```

El código anterior producirá esto:

```
<ul class="boldlist" id="mylist">
  <li>colors
    <ul>
    <li>red</li>
    <li>blue</li>
    <li>green</li>
    </ul>
  </li>
```

```
  <li>shapes
    <ul>
    <li>round</li>
    <li>suare</li>
    <li>circles
        <ul>
        <li>elipse</li>
        <li>oval</li>
        <li>sphere</li>
        </ul>
    </li>
    </ul>
  </li>
  <li>moods
    <ul>
    <li>happy</li>
    <li>upset
        <ul>
        <li>defeated
    <ul>
    <li>dejected</li>
    <li>disheartened</li>
    <li>depressed</li>
    </ul>
        </li>
        <li>annoyed</li>
        <li>cross</li>
        <li>angry</li>
        </ul>
    </li>
    </ul>
  </li>
</ul>
```

Asistente de Inflexión

El archivo Asistente de Inflexión contiene funciones que le permiten cambiar palbras a plural, singular, camel case, etc. (en inglés)

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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


Las siguientes funciones están disponibles:

singular()

Cambia una palabra plural a singular. Ejemplo:

```
$palabra = "perros";
echo singular($palabra); // Returns "perro"
```

plural()

Cambia una palabra singular a plural. Ejemplo:

```
$palabra = "perro";
echo plural($palabra); // Returns "perros"
```

Para forzar una palabra para terminar con "es" use un segundo argumento "true".

```
$palabra = "pass";
echo plural($palabra, TRUE); // Returns "passes"
```

camelize()

Cambia una cadena de palabras separada por espacios o guiones bajos a camel case. Ejemplo:

```
$palabra = "my_dog_spot";
echo camelize($palabra); // Returns "myDogSpot"
```

underscore()

Toma múltiples palabras separadas por espacios y le agrega guiones bajos. Ejemplo:

```
$palabra = "my dog spot";
echo underscore($palabra); // Returns "my_dog_spot"
```

humanize()

Toma múltiples palabras separadas por guiones bajos y le agrega espacios entre ellos. Cada palabra es capitalizada. Ejemplo:

```
$palabra = "my_dog_spot";
echo humanize($palabra); // Returns "My Dog Spot"
```

Asistente de Ruta

El archivo Asistente de Ruta contiene funciones que le permiten trabajar con rutas de archivos en el servidor.

Cargando estos Asistentes

Este Asistente se carga utilizando el siguiente código:

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

Las siguientes funciones están disponibles:

set_realpath()

Comprueba si existe la ruta. Esta función devolverá una ruta del servidor sin enlaces simbólicos o con una estructura de directorios relativa. Un segundo argumento opcional causará un error, que se activará si la ruta no pueden ser resuelta.

```
$directory = './../../etc/passwd';
echo set_realpath($directory);
// retorna "/etc/passwd"

$non_existent_directory = './../../path/not/found';
echo set_realpath($non_existent_directory, TRUE);
// retorna "/path/not/found"

echo set_realpath($non_existent_directory, FALSE);
// retorna un error, como la ruta no se puede resolver
```


Asistente de Seguridad

El archivo Asistente de Seguridad contiene funciones relacionadas con seguridad.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

xss_clean()

Provee un filtrado contra ataques XSS (Cross Site Script). Esta función es una alias a la de la clase de Entrada (input). Más info puede ser encontrada allí.

dohash()

Permite crear encriptaciones de una vía SHA1 o MD5 aptas para encriptar contraseñas. Creará SHA1 por defecto. Ejemplo:

```
$cadena = dohash($cadena); // SHA1

$cadena = dohash($cadena, 'md5'); // MD5
```

strip_image_tags()

Esta es una función de seguridad que eliminará las etiquetas de imagen de una cadena. Deja la URL de la imagen como texto plano.

```
$cadena = strip_image_tags($cadena);
```

encode_php_tags()

Esta es una función de seguridad que convierte etiquetas PHP a entidades. Nota: si usa la función de filtro XSS, realizará esto automáticamente.

```
$cadena = encode_php_tags($cadena);
```

Asistente de Smiley

El archivo Asistente de Smiley contiene funciones que le permiten manejar smileys (emoticons).

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Visión General

El asistente de Smiley tiene un "traductor" que toma smileys en texto plano, como *:-)* y lo transforma en una imagen de representación, como 

También le deja mostrar un juego de las imágenes smiley que cuando sean cliqueadas serán insertadas en un campo de formulario. Por ejemplo, si tiene un blog que le permite comentarios de usuarios, puede mostrar la smileys próxima al formulario de comentario. Los usuarios pueden cliquear el smiley deseado y con la ayuda de algo de JavaScript será ubicado dentro del campo de formulario.

Tutorial de Smileys Cliqueables

Aquí hay un ejemplo demostrando como puede crear un juego de smileys cliqueablos próximos a un campo de formulario. Este ejemplo requiere que primero baje e instale las imagenes smileys, y luego cree un controler y una Vista como se describe.

Importante: Antes de empezar, por favor descargue las imágenes smiley (http://codeigniter.com/download_files/smileys.zip) y póngalas en un lugar accesible públicamente en su servidor. Este asistente también asume que tiene el arreglo de reemplazo de smiley ubicado en *application/config/smileys.php*

El Controlador

En su carpeta *application/controllers/*, cree un archivo llamado `smileys.php` y ubique el código siguiente en él.

Importante: Cambie la URL en la función *get_clickable_smileys()* de abajo para que apunte a su carpeta *smiley*.

Notará que en adición al asistente de smiley usaremos la Clase de Tabla.

```
<?php

class Smileys extends Controller {

    function Smileys()
    {
        parent::Controller();
    }

    function index()
    {
        $this->load->helper('smiley');
        $this->load->library('table');

        $image_array = get_clickable_smileys('http://www.your-site.com/images/
smileys/');

        $col_array = $this->table->make_columns($image_array, 8);

        $data['smiley_table'] = $this->table->generate($col_array);

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

}
?>
```

En su carpeta *application/views/*, cree un archivo llamado `smiley_view.php` y ubique este código en él:

```
<html>
<head>
<title>Smileys</title>

<?php echo js_insert_smiley('blog', 'comments'); ?>
```

```
</head>
<body>

<form name="blog">
<textarea name="comments" cols="40" rows="4"></textarea>
</form>

<p>Click to insert a smiley!</p>

<?php echo $smiley_table; ?>

</body>
</html>
```

Cuando haya creado el controlador y la vista de arriba, carguelo visitando *http://www.su-sitio.com/index.php/smileys/*

Referencia de funciones

get_clickable_smileys()

Devuelve un arreglo que contiene sus imágenes smiley envueltas en un hipervínculo cliqueable. Debe suministrar la URL a su carpeta smiley a través del primer parámetro:

```
$arreglo_imagen = get_clickable_smileys("http://www.su-sitio.com/imagenes/
smileys/");
```

js_insert_smiley()

Genera el JavaScript que permite que las imagenes sean cliqueables e instertadas en un campo de formulario. El primer parámetro debe contener el nombre de su formulario, el segundo parámetro debe contener el nombre del código de formulario. Esta función está diseñada para ser ubicada en el area <head> de su página web.

```
<?php echo js_insert_smiley('blog', 'comentarios'); ?>
```


parse_smileys()

Toma una cadena de texto como entrada y reemplaza cualquier smiley de texto plano contenida en la imagen equivalente. El primer parámetro debe contener su cadena, el segundo debe contener la URL a la carpeta de smiley:

```
$cadena = 'Aquí hay algunos simileys: :-) ;-)'; $cadena = parse_smileys($str,
"http://www.su-sitio.com/imagenes/smileys/"); echo $cadena;
```

Asistente de Cadena

El archivo Asistente de Cadena contiene funciones que asistiran al trabajar con cadenas.

Cargando este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

random_string()

Genera una cadena aleatoria basada en el tipo y largo especificado. Útil para crear contraseñas o claves (hashes) aleatorias.

El primer parámetro especifica el tipo de cadena, el segundo el largo. Las siguientes opciones están disponibles:

- **alnum**: Cadenas alfanuméricas con caracteres en mayúscula y minúscula.
- **numeric**: Cadena numérica.
- **nozero**: Cadena numérica sin ceros.
- **unique**: Encriptada con MD5 y uniqid(). Nota: El parámetro de longitud no está disponible para este tipo. Devuelve una cadena de caracteres de largo.

Ejemplo de uso:

```
echo random_string('alnum', 16);
```

alternator()

Permite dos o más items alternar entre ellos cuando se recorre una iteración. Ejemplo:

```
for ($i = 0; $i < 10; $i++)
{
    echo alternator('cadena uno', 'cadena dos');
}
```

Puede agregar tantos parámetros como quiera, y con cada iteración el próximo item será devuelto.

```
for ($i = 0; $i < 10; $i++)
{
    echo alternator('uno', 'dos', 'tres', 'cuatro', 'cinco');
}
```

Nota: Para usar múltiples llamadas separadas a esta función simplemente llame a la función sin argumentos para reinicializarla.

repeater()

Genera una copia repetida de los datos que envía. Ejemplo:

```
$cadena = "\n";
echo repeater($cadena, 30);
```

Lo anterior generará 30 veces el carácter "nueva línea".

reduce_double_slashes()

Convierte las dobles barras de una cadena a una sola barra, excepto que se encuentran en http://. Ejemplo:

```
$cadena = "http://www.ejemplo.com//index.php";
echo reduce_double_slashes($cadena); // resulta en "http://www.ejemplo.com/
index.php"
```


trim_slashes()

```
Remueve cualquier barra al inicio o al final de una cadena. Ejemplo:

$cadena = "/esto/aquello/lootro/";
echo trim_slashes($cadena); // results in esto/aquello/lootro
```

reduce_multiples()

Reduce múltiples instancias de un caracter particular ocurriendo directamente uno después de otro. Ejemplo:

```
$cadena="Fred, Bill,, Joe, Jimmy";
$cadena=reduce_multiples($cadena,","); //resulta en "Fred, Bill, Joe, Jimmy"
```

La función acepta los siguientes parámetros:

```
reduce_multiples(cadena: texto en el que buscar, cadena: caracter a reducir,
booleano: remover el caracter del principio o el final de la cadena)
```

El primer parámetro contiene la cadena de la cual quiere reducir las multiplicidades. El segundo parámetro contiene el caracter que quiere reducir. El tercer parámetro es Falso por defecto. Si es verdadero removerá ocurrencias del caracter al principio y al final de la cadena. Ejemplo:

```
$cadena=",Fred, Bill,, Joe, Jimmy,";
$cadena=reduce_multiples($cadena,",",true); //resulta en "Fred, Bill, Joe,
Jimmy"
```

quotes_to_entities()

Convierte comillas simples y dobles en una cadena a la entidad HTML correspondiente. Ejemplo:

```
$cadena="Joe's \"dinner\"";
$cadena=quotes_to_entities($cadena); //resulta en "Joe&#39;s
&quot;dinner&quot;"
```

strip_quotes()

Remueve las comillas simples y dobles de una cadena. Ejemplo:

```
$cadena="Joe's \"dinner\"";
$cadena=strip_quotes($cadena); //results in "Joes dinner"
```

Asistente de Texto

El Asistente de Texto contiene funciones que asisten en el trabajo con el texto.

Cargando este Asistente

Este Asistente se carga usando el siguiente código:

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

Las siguientes funciones estan disponibles:

word_limiter()

Fragmenta una cadena al número de **palabras** especificadas. Por ejemplo:

```
$string = "He aquí una bonita cadena de texto que consta de doce palabras.";

$string = word_limiter($string, 4);

// Retorna: He aquí una bonita#8230;
```

El tercer parámetro es un sufijo opcional añadido a la cadena. Por defecto se añade una elipsis.

character_limiter()

Fragmenta una cadena al número de **caracteres** especificados. Mantiene la integridad de las palabras de manera que la cantidad de caracteres puede ser ligeramente, mas o menos, lo que especifique. Por ejemplo:

```
$string = "He aquí una bonita cadena de texto que consta de doce palabras.";

$string = character_limiter($string, 20);

// Retorna: He aquí una bonita cadena …
```

El tercer parámetro es un sufijo opcional añadido a la cadena. Por defecto se añade una elipsis.

ascii_to_entities()

Convierte valores ASCII entidades de caracteres, Incluyendo ASCII alto y MS Word caracteres que pueden causar problemas cuando se utilizan en una página web, de manera que puedan ser mostrado constantemente independientemente de la configuración del navegador o ser almacenados fiablemente en una base de datos. Existe cierta dependencia en su conjunto de caracteres soportados en su servidor, por lo que puede no ser 100% confiable en todos los casos, pero en su mayor parte se deben identificar correctamente los caracteres fuera del rango normal (como caracteres acentuados). Por ejemplo:

```
$string = ascii_to_entities($string);
```

entities_to_ascii()

Esta función hace lo contrario de la anterior; se convierte de vuelta las entidades de caracteres a ASCII.

word_censor()

Le permite censurar palabras dentro de una cadena de texto. El primer parámetro contendrá la cadena original. El segundo contendrá un arreglo de palabras inhabilitará. El tercer parámetro (opcional) puede contener un valor de reposición de

las palabras. Si no se especifica serán reemplazadas con signos numeral: ####. Por ejemplo:

```
$disallowed = array('darn', 'shucks', 'golly', 'phooey');

$string = word_censor($string, $disallowed, 'Beep!');
```

highlight_code()

Colorea una cadena de código (PHP, HTML, etc.). Por ejemplo:

```
$string = highlight_code($string);
```

Esya función usa la función de PHP highlight_string(), de modo que los colores usados son los especificados en su archivo php.ini.

highlight_phrase()

Destacará una frase dentro de una cadena de texto. El primer parámetro contendrá la cadena original, el segundo contendrá la frase que desea destacar. El tercer y cuarto parámetro contendrá las eqtiquetas HTML de apertura / cierre con la cual le gustaría envolver a la frase. Por ejemplo:

```
$string = "He aquí una bonita cadena de texto acerca de nada en particular.";

$string = highlight_phrase($string, "bonita cadena", '<span
style="color:#990000">', '</span>');
```

El texto anterior retorna:

He aquí una bonita cadena de texto acerca de nada en particular.

word_wrap()

Envuelve el texto a la cantidad de **caracteres** especificados mientras mantiene palabras completas. Por ejemplo:

```
$string = "Aquí hay una simple cadena de texto que nos ayudará a demostrar
esta función.";
```


```
echo word_wrap($string, 25);

// Produciría:

Aquí hay una simple cadena
de texto que nos ayudará
a demostrar esta
función
```

Asistente de Tipografía

El archivo Asistente de Tipografía contiene funciones que ayudan en su formato de texto en formas pertinentes semánticamente.

Cargando este Asistente

Este asistente se carga usando el siguiente código:

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

Las siguientes funciones están disponibles:

auto_typography()

Formatea el texto de modo que sea, semánticamente y tipográficamente, en HTML correcto. Toma una cadena como entrada y la retorna con el siguiente formato:

- Rodea a los párrafos con (Busca dobles saltos de línea para identificar los párrafos).
- Los saltos de lineas únicos se convierten a
, a excepción de las que aparecen dentro de las etiquetas <pre>.
- Los bloques de nivel de elementos, como la etiquetss , no son envueltas entre párrafos, pero su texto contenido si contiene párrafos.
- Las comillas con convertidas correctamente a entidades de comillas, a excepción de las que aparecen dentro de las etiquetas.
- Los Apostrofes son convertidos a entidades de apostrofe.

- Dobles guiones (ya sea como esto — o como esto—) son convertidos a guiones em—.
- Tres períodos consecutivos, ya sea anterior o posterior a una palabra se convierten a puntos suspensivos…
- Los doble espacios de las siguientes frases se convierten a espacios no-rompibles para imitar el doble espacio.

Ejemplo de uso:

```
$string = auto_typography($string);
```

Nota: Los formatos de tipografía pueden hacer uso intensivo del procesador, especialmente si se tiene una gran cantidad de contenido que se está formateado. Si decide utilizar esta función, puede considerar almacenar en caché sus páginas.

nl2br_except_pre()

Convierte nuevas líneas a etiquetas
 salvo que aparezcan entre etiquetas <pre>. Esta función es idéntica a la función nativa de PHP *nl2br()*, excepto que ignora las etiquetas <pre>.

Ejemplo de uso:

```
$string = nl2br_except_pre($string);
```

Asistente de URL

El archivo Asistente de URL contiene funciones que asiste al trabajo con URLs.

Cargando este Asistente

This helper is loaded using the following code:

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


Las siguientes funciones están disponibles:

site_url()

Devuelve una URL de tu sitio, como se especifica en el archivo de configuración. El archivo index.php (o lo que sea que haya establecido como la *index_page* de su sitio en el archivo de configuración) será agregado a la URL, así también como cualquier segmento URI pasado a la función.

Se recomienda usar esta función en cualquier momento que necesite generar una URL local para que sus páginas sean más portables en el caso de que su URL cambie.

Los segmentos pueden ser opcionalmente pasados a la función como una cadena o un arreglo. Aquí hay ejemplo con cadenas:

```
echo site_url("noticias/local/123");
```

El ejemplo anterior devolverá algo así: http://www.su-sitio.com/index.php/noticias/local/123

Aquí hay un ejemplo de segmentos pasados como un arreglo:

```
$segmentos = array('noticias', 'local', '123');

echo site_url($segments);
```

base_url()

Devuelve la URL base de su sitio, como se especifica en su archivo de configuración. Ejemplo:

```
echo base_url();
```

index_page()

Devuelve la página "index" de su sitio, como se especifica en su archivo de configuración. Ejemplo:

```
echo index_page();
```

anchor()

Crea un link HTML estándar basado en su URL de sitio local:

```
<a href="http://www.su-sitio.com">Cliquee aquí</a>
```

La etiqueta tiene tres parámetros opcionales:

```
anchor(segmentos de uri, texto, atributos)
```

El primer parámetro puede contener cualquier segmento que quiera agregarle a la URL. Como con la función *site_url()* previa, los segmentos pueden ser una cadena o un arreglo.

Nota: Si está construyendo hipervínculos que son internos a su aplicación no incluya la URL base (http://...). Esta será agregada automáticamente desde la información especificada en su archivo de configuración. Incluya sólo los segmentos URI que desee agregar a la URL.

El segundo segmento es el texto que quiere que diga el hipervínculo. Si lo deja vació, la URL será usada.

El tercer parámetro puede contener una lista de atributos que quiere agregar al hipervínculo. Los atributos puedem ser simples cadenas o arreglos asociativos.

Aquí hay algunos ejemplos:

```
echo anchor('noticias/local/123', 'Mis Noticias');
```

Would produce: Mis Noticias

```
echo anchor('noticias/local/123', 'Mis Noticias', array('title' => 'Las
mejores noticias!'));
```

Would produce: Mis Noticias

anchor_popup()

Casi idéntica a la función *anchor()* excepto que abre la URL en una ventana nueva. Puede especificar atributos de ventana de JavaScript en el tercer parámetro para controlar cómo la ventana es abierta. Si el tercer parámetro no está establecido simplemente abrirá una nueva ventana con su configuración de explorador. Aquí hay un ejemplo con atributos:

```
$atts = array(
    'width'=> '800',
    'height'   => '600',
    'scrollbars' => 'yes',
    'status'   => 'yes',
    'resizable'  => 'yes',
    'screenx'  => '0',
    'screeny'  => '0'
    );

echo anchor_popup('noticias/local/123', 'Cliqueame!', $atts);
```

Nota: los atributos anteriores son los predeterminados para la función así que sólo necesita establecer los que son diferentes de los que necesita. Si desea que la función use todos los predeterminados, simplemente pase un arreglo vacío en el tercer parámetro:

```
echo anchor_popup('noticias/local/123', 'Cliqueame!', array());
```

mailto()

Crea un hipervínculo estándar HTML. Ejemplo de uso:

```
echo mailto('yo@mi-sitio.com', 'Cliquee Aquí para Contactarme');
```

Como la función *anchor()* previa, puede establecer atributos usando el tercer parámetro.

safe_mailto()

Idéntica a la función anterior, excepto que escribe una versión ofuscada de la etiqueta mailto usando números ordinales escritos con JavaScript para ayudar para prevenir

que la dirección de email sea written with JavaScript to help prevent the email address from being recogido por un robot no deseado.

auto_link()

Automáticamente turna las URLs y direcciones de email contenidas en una cadena en hipervínculos. Ejemplo:

```
$cadena = auto_link($cadena);
```

El segundo parámetro determina si las URLs y los mails son convertidos o sólo uno de ellos. El comportamiento por defecto es ambos si el parámetro no está especificado

Convertir sólo URLs:

```
$cadena = auto_link($cadena, 'url');
```

Convertir sólo direcciones de Email:

```
$cadena = auto_link($cadena, 'email');
```

El tercer parámetro determina si los hipervínculos serán mostrados en una nueva ventana. El valor puede ser TRUE o FALSE (booleano):

```
$cadena = auto_link($cadena, 'both', TRUE);
```

url_title()

Toma una cadena como entrada y crea una cadena amigable para humanos. Esto es útil si, por ejemplo, tiene un blog en el cual desee usar como título de las entradas en la URL. Ejemplo:

```
$titulo = "Que hay de malo con 'CSS'?";

$titulo_url = url_title($titulo);

// Produce: que-hay-de-malo-con-css
```


El segundo parámetro determina el delimitador de palabra. Por defecto los guiones son usados. Las opciones son: *dash*, or *underscore*:

```
$titulo = "Que hay de malo con 'CSS'?";

$titulo_url = url_title($titulo, 'underscore');

// Produce: que_hay_de_malo_con_css
```

prep_url()

La función agregará `http://` en caso de que le falte a una URL. Pase una cadena URL a la función así:

```
$url = "www.ejemplo.com";

$url = prep_url($url);
```

redirect()

Hace un "redireccionamiento a través de cabeceras" a la URI especificada. Al igual que las otras funciones de este asistente, esta está designada para redireccionar a URL locales dentro de su sitio. **No** debe especificar la URL de su sitio, sino simplemente los segmentos URI al controlador al que quiere redireccionar. Esta función construirá la URL basada en los valores de sus archivos de configuración.

El sengudo parámetro le permite elegir entre el método "location" (por defecto) o el método "refresh". Location es más rápido, pero en servidores Windows puede ser un problema. Ejemplo:

```
if ($autorizado == FALSE)
{
    redirect('/entrar/formulario/', 'refresh');
}
// con redirección 301
redirect('/articulo/13', 'location', 301);
```

Nota: Para que esta función funcione debe ser usado antes que cualquier salida al explorador ya que utiliza encabezados del servidor.

Asistente de XML

El archivo Asistente de XML contiene funcioens que le asisten al trabajar con datos XML.

Cargar este Asistente

Este asistente es cargado usando el siguiente código:

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

Las siguientes funciones están disponibles:

xml_convert('*cadena*')

Toma una cadena como entrada y convierte los siguientes caracteres reservados de XML a entidades:

Ampersands: &
Los caracteres de menor y mayor: < >
Comillas simple y doble: ' "
Guiones: -

Esta función ignora ampersands (&) si son partes de un caracter de entidad. Ejemplo:

```
$cadena = xml_convert($cadena);
```


Recursos Adicionales

CLASS REFERENCE

benchmark->
- mark
- elapsed_time
- memory_usage

calendar->
- generate
- initialize

config->
- load
- item
- set_item
- site_url
- system_url

db-> (Query)
- query
- escape
- escape_str
- insert_string
- update_string

db-> (Results)
- result
- result_array
- row
- num_rows
- num_fields
- insert_id
- affected_rows
- versions

db-> (Active Records)
- get
- select
- from
- join
- where
- orwhere
- like
- orlike
- groupby
- having
- orderby
- limit
- insert
- set
- update
- delete

db-> (Field Data)
- field_names
- field_data

db-> (Table Data)
- tables
- table_exists

db-> (Custom Function Calls)
- call_function

email->
- from
- reply_to
- to
- cc
- bcc
- subject
- message
- alt_message
- clear
- send
- attach
- print_debugger

encrypt->
- encode
- decode
- set_cypher
- set_mode
- sha1

upload->
- do_upload
- display_errors
- data

image_lib->
- resize
- crop
- rotate
- watermark
- display_errors

input->
- xss_clean
- post
- cookie
- ip_address
- valid_ip
- user_agent

load->
- library
- database
- scaffolding
- view
- vars
- helper
- plugin
- file
- lang
- config

lang->
- load
- line

output->
- set_output
- get_output

pagination->
- initialize
- create_links

session->
- userdata
- set_userdata

trackback->
- send
- display_errors
- receive
- send_error
- send_success
- data

parser->
- parse

uri->
- segment
- slash_segment
- uri_to_assoc
- assoc_to_uri
- uri_string
- total_segments
- segment_array

validation->
- run
- set_rules
- required
- set_message
- set_fields
- set_error_delimiters

xmlrpc->
- server
- timeout
- method
- request
- send_request
- set_debug
- display_error
- display_response
- send_error_message
- send_response

xmlrpcs->
- initialize
- serve

RESERVED FUNCTION NAMES

- Controller
- CI_Base
- _ci_autoload
- _ci_autoloader
- _ci_assign_core
- _ci_initialize
- _ci_init_database
- _ci_init_scaffolding
- _ci_is_loaded
- _ci_load
- _ci_scaffolding
- _ci_set_view_path

Add'l Reserved PHP4 Only

- CI_Loader
- config
- database
- file
- helper
- helpers
- language
- library
- plugin
- plugins
- scaffolding
- script
- view
- vars

Directory Structure

- system
  - application
    - config
    - controllers
    - errors
    - models
    - scripts
    - views
  - cache
  - codeigniter
  - init
  - drivers
  - fonts
  - logs
  - helpers
  - language
  - libraries
  - plugins
  - scaffolding
- user_guide

HELPER REFERENCE

array
- random_element

cookie
- set_cookie

date
- now
- mdate
- local_to_gmt
- gmt_to_local
- mysql_to_unix
- unix_to_human
- human_to_unix
- timespan
- days_in_month
- timezone_menu

directory
- directory_map

file
- read_file
- write_file
- delete_files

form
- form_open
- form_open_multipart
- form_hidden
- form_input
- form_password
- form_upload
- form_textarea
- form_dropdown
- form_checkbox
- form_radio
- form_submit
- form_close
- form_prep

html
- heading
- nbs
- br

security
- xss_clean
- hash
- strip_image_tags
- encode_php_tags

string
- random_string
- alternator
- repeater

text
- word_limiter
- character_limiter
- ascii_to_entities
- entities_to_ascii
- word_censor
- highlight_code
- highlight_phrase
- word_wrap

typography
- auto_typography
- nl2br_except_pre

url
- site_url
- base_url
- index_page
- anchor
- anchor_popup
- mailto
- safe_mailto
- auto_link
- url_title
- prep_url
- redirect

xml
- xml_convert

Load Helpers with:
```
$this->load->helper('name');
```

Use Helpers just like any PHP function
```
<?=helper_name();?>
```

CodeIgniter Version 1.3 Quick Reference Chart

CodeIgniter™