CodeIgniter is an Open Source
Web Application Framework that helps
you write kick-ass PHP programs

Code Igniter

Guia do Usuário

Sumário

Diagramado por: edson jr - edsonthadeu@hotmail.com

Code Igniter URLs

Por default, URLs no Code Igniter são projetadas para serem amigáveis aos motores de busca e ao ser humano. No lugar de usar a abordagem padrão "query string" que é sinônim de sistemas dinâmicos, Code Igniter usa a abordagem baseada em segmentos:

www.seu-site.com/noticias/artigos/meu_artigo

Note: Query string URLs podem ser opcionalmente habilitadas, como descrito abaixo.

Segmentos URI

Os segmentos na URL, de acordo com a abordagem Model-View-Controller, geralmente representam:

www.seu-site.com/classe/funcao/ID

O primeiro segmento representa a classe controller que será invocada.

O segundo segmento representa a classe função, ou método, que será chamada.

O terceiro, e qualquer outro subseqüente, representa o ID ou qualquer outra variável que serão passados com controller.

A Classe URI e o Assistente URL contêm funções que facilitarão o trabalho com seus dados URI. Além disto, suas URLs podem ser remapeadas usando a característica URI Routing, proporcionando mais flexibilidade.

Removendo o arquivo index.php

Por default, o arquivo index.php será incluso em suas URLs:

www.seu-site.com/index.php/noticias/artigos/meu_artigo

Você pode remover facilmente este arquivo usando um arquivo .htaccess com algumas regras simples. Aqui está um exemplo de tal arquivo, usando o método "negativo" no qual tudo é redirecionado exceto os itens especificados:

```
RewriteEngine on

RewriteCond $1 !^(index\.php|images|robots\.txt)

RewriteRule ^(.*)$ /index.php/$1 [L]
```

No exemplo acima, qualquer requisição HTTP, outra que aquela pelo index.php, imagens e robots.txt será tratada como uma requisição pelo arquivo index.php.

Adicionando um sufixo URL

No seu arquivo config/config.php você pode espeficicar um sufixo que será adicionado a todas as URLs geradas pelo Code Igniter. Por exemplo, se uma URL é assim:

www.seu-site.com/index.php/produtos/view/sapatos

Você pode opcionalmente adicionar um sufixo como .html, fazendo a página aparecer como de um certo tipo:

www.seu-site.com/index.php/produtos/view/sapatos.html

Habilitando Query Strings

Em alguns casos, você pode preferir usar URLs query strings:

index.php?c=produtos&m=view&id=345

Model-View-Controller

O Code Igniter é baseado no padrão de desenvolvimento Model-View-Controller. A MVC é uma abordagem de software que separa a lógica da aplicação de sua apresentação. Em prática, ela permite que suas páginas web contenham o mínimo de scripts já que a apresentação é separa do código PHP.

O **Model** representa suas estruturas de dados. Tipicamente, suas classes model irão conter funções que lhe ajudarão a buscar, inserir e substituir informações em sua bando de dados..

A **View** é a informação que será apresentada ao usuário. A View será normalmente uma página web, mas no Code Igniter, uma view pode ser também um fragmento de página como um cabeçalho ou rodapé. Ela pode ser também uma página RSS, ou qualquer outro tipo de "página".

O **Controller** serve como um intermediário entre o Model, a View, e qualquer outro recurso necessário para processar a requisição HTTP e gerar a página web.

O Code Igniter tem uma abordagem mais solta em relação ao MVC, já que Models não são obrigatórios. Se você não precisa desta facilidade, ou acha que manter models gerará mais complexidade que você quer, pode ignorá-las e construir sua aplicação apenas usando Controllers e Views. Code Igniter também proporciona que você incorpore seus próprios scripts, ou mesmo desenvolva bibliotecas para o sistema, lhe possibilitando trabalhar de um jeito que faça mais sentido para você.

Pontapé Inicial

Qualquer software demanda certo esforço em seu aprendizado. Nós fizemos nosso melhor para minimizar a curva de aprendizado mantendo o processo o mais apreciável possível.

O primeiro passo é instalar o Code Igniter e então ler todos os tópicos na seção Introdução deste Guia.

Em seguida, leia cada um dos Assuntos Gerais na ordem em que estão listados. Cada assunto se baseia no anterior e inclui exemplos de código que você é encorajado a experimentar.

Quanto compreender o básico, você está pronto a explorar as páginas das Referências de Classes e as Referências de Assistentes para aprender a utilizar as bibliotecas e arquivos assistentes nativos.

Fique a vontade para aproveitar nosso Fórum Comunitário se tiver dúvidas ou problemas e nosso Wiki para ver exemplos de código postados por outros usuários.

O Code Igniter opcionalmente suporta esta capacidade, que pode ser habilitada no arquivo application/config.php. Se você abrir seu arquivo de configuração verá estes itens:

$config['enable_query_strings'] = FALSE;

$config['controller_trigger'] = 'c';

$config['function_trigger'] = 'm';

Se você alterar "enable_query_strings" para TRUE, este recurso será ativado. Seus controles e funções serão então acessados usando palavras "gatilho" que você configurará para invocar seus controllers e métodos:

index.php?c=controller&m=metodo

Por favor, note: Se você está usando query strings, terá que construir suas próprias URLs, ao invés de utilizar os assistentes URL (e outros assistentes que geral URLs, como alguns dos assistentes de formulário) já que eles foram projetados para trabalhar com URLs segmentadas.

Controllers

Controllers são o coração da sua aplicação, pois determinam como as requisições HTTP devem ser tratadas.

O que é um Controler?

Um Controller é simplesmente um arquivo de classe que é nomeado de uma forma em que possa ser associado a uma URI.

Considere este URI:

www.seu-site.com/index.php/blog/

No exemplo acima, o Code Igniter tentará encontrar um controller chamado blog.php e o carregará.

Quando um nome de um controller bate com o primeiro segmento do URI, ele será carregado.

Vamos experimentar: Hello World!

Vamos criar um simples controller para que você possa vê-lo em ação. Usando um editor de texto, crie um arquivo chamado blog.php e escreva nele o seguinte código:

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

Então salve o arquivo no seu diretório application/controllers/.

Agora, visite o seu site usando uma URL parecida com esta:

www.seu-site.com/index.php/blog/

Se você fez certo, deverá ver o seguinte: Hello World!.

Note: Nomes de Classes devem começar com uma letra maiúscula. Em outras palavras, isto é válido:

```
<?php
class Blog extends Controller {
}
?>
```

E isto não é válido:

```
<?php
class blog extends Controller {
}
?>
```

E também sempre tenha certeza que seu controller estenda o controller pai para que possa herdar todas as funções deste.

Funções

No exemplo acima, o nome da função é index(). A função "index" é sempre carregada por default se o segundo segmento do URI estiver vazio. Outro jeito de mostrar sua mensagem "Hello World" poderia ser este:

www.seu-site.com/index.php/blog/index/

A segunda parte do segmento do URI determina qual função no controller é chamada.

Vamos tentar isto. Adicione uma nova função ao seu controller:

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

Carregue agora a seguinte URL para ver a função comment:

www.seu-site.com/index.php/blog/comments/

Você deverá ver sua nova mensagem.

Passando Segmentos UTI para suas Funções

Se seus URI contêm mais de 2 segmentos, eles serão passados à suas funções como parâmetros.

Por exemplo, digamos que você tenha um URI como este:

www.seu-site.com/index.php/produtos/sapatos/sandalias/123

Para sua função serão passados os segmentos URI 3 e 4 ("sandalias" e "123"):

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

Importante: Se você está usando o recurso URI Routing, os segmentos passados à suas funções serão aqueles previamente roteados (re-routed).

Definindo um Controller Default

Ao Code Igniter pode ser dito para carregar o controller default quando um URI não está presente, pois este será o caso quando apenas a URL raiz do seu site é requisitada. Para especificar um controller default, abra seu arquivo application/config/routes.php e altere esta variável:

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

Onde Blog é o nome do classe controller que você quer usar. Se você carregar agora seu index.php geral, sem especificar qualquer segmento URI, verá sua mensagem "Hello World".

Remapeando Chamadas de Funções

Como notado acima, o segundo segmento do URI tipicamente determina qual função no controller será chamada. Code Igniter permite que você sobreponha este comportamente através do uso da função _ remap():

```
function _remap()
{
  // Algum código aqui }
```

Importante: Se seu controller contém uma função chamada _remap(), ela sempre será chamada independente do seu URI contenha. Ela sobrepõe o comportamente normal no qual o URI determina qual função será chamada, permitindo que você defina suas próprias regras de roteamento de funções.

A chamada de função sobreposta (tipicamente o segundo segmento do URI) será passado como um parâmetro à função _remap():

```
function _remap($metodo)
{
  if ($metodo == 'algum_metodo')
  {
    $this->$metodo();
  }
  else
  {
    $this->metodo_default();
  }
}
```

Processando a Saída

O Code Igniter tem uma classe de saída que toma conta de enviar os dados renderizado finais ao web browser automaticamente. Mais informações sobre isto pode ser encontrado nas páginas Views e Classe Saída. Em alguns casos, no entanto, você pode preferir pós-processar os dados finalizdos e então enviar para o browser você mesmo. Code Igniter permite que você adicione uma função chamada _output() em seu controller que irá receber os dados de saída finalizados.

Importante: Se seu controller contém uma função chamada _output(), ela sempre será chamada pela classe Saída no lugar de dar echo diretamente nos dados finalizados. O primeiro parâmetro desta função conterá a saída finalizada.

Aqui está um exemplo:

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

Por favor, note que sua função _output() irá receber os dados em seu estado final. Benchmark e dados de uso de memória serão renderizados, arquivos de cache serão escritos (se o cache estiver habilitado) e cabeçalhos serão enviados (se você usar tal recurso) antes que sejam entregues à função _output(). Se você está usando este recurso, o temporizador de execução da página e as estatísticas de uso de memória poderão não ser perfeitamente precisos pois eles não serão considerarão qualquer processamento adicional que você fizer. Para uma forma alternativa de controlar a saída antes que qualquer processamento final é feito, por favor veja os métodos disponíveis na Classe Output.

Funções Privadas

Em alguns casos, você pode querer que certas funções não sejam acessas publicamente. Para fazer uma função privada, simplesmente adicione um underscore como prefixo do nome e ela não será servida via requisação URL. Por exemplo, se você tiver uma função como esta:

```
function _utility()
{
  // algum código
}
```

Tentar acessá-la via URL, da seguinte forma, não funcionará:

www.seu-site.com/index.php/blog/_utility/

Organizando seus Controllers em Subdiretórios

Se você está construindo uma aplicação grande, pode achar conveniente organizar seus controllers em sub-diretórios. Code Igniter permite que o faça.

Simplesmente crie subdiretórios em seu diretório application/controllers e coloque seuas classe controllers dentro deles.

Note: Utilizando-se este recurso, o primeiro segmento de seu URI precisará especificar o diretório. Por exemplo, digamos que você tenha um controller localizado aqui:

application/controllers/produtos/sapatos.php

Para chamar o controle acima, seu URI irá se parecer com isto:

www.seu-site.com/index.php/produtos/sapatos/123

Cada um de seus subdiretórios poderá conter um controller default que será chamado se a URL contiver apenas um subdiretório. Simplesmente nomeie seu controller default como espeficiado em arquivo application/config/routes.php.

Code Igniter também lhe permite remapear seus URIs usando seu recurso Roteamento URI.

Construtores de Classes

Se você pretende usar um construtor em algum de seus Controllers, você DEVE colocar a seguinte linha de código nele:

parent::Controller();

A razão para esta linha ser necessária é porque seu construtor local irá sobrepor aquele na classe controller pai, por isto precisamos chamá-la manualmente.

Se você não está familirizado com construtores (constructors), em PHP 4, um construtor é simplesmente uma função que tem o mesmo nome da classe:

```
<?php
class Blog extends Controller {
    function Blog()
    {       parent::Controller(); }
} ?>
```

Em PHP 5, construtores usam a seguinte sintaxe:

```
<?php
class Blog extends Controller {
    function __construct()
    {
        parent::Controller();
    }
}
?>
```

Construtores são úteis se você precisa definir alguns valores default, ou rodar um processo default enquanto sua classe é instanciada. Construtores não podem retornar um valor, mas podem executar algum trabalho default.

Nomes Reservedos de Funções

Já que suas classes controller irão estender o controller principal da aplicação, você precisa tomar cuidado em não nomeá-las de forma idêntica àquelas usadas pela principal, caso contrário suas funções locais irão sobrescrevê-las. Abaixo, uma lista de nomes reservados. Não nomeie suas funções como uma destas:

Controller

CI_Base

_ci_initialize

_ci_scaffolding

Se estiver rodando PHP 4, ainda há alguns nomes adicionais. Estes se aplicam SOMENTE se você estiver rodando PHP 4.

CI_Loader
config
database
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

É isto!

Resumidamente, isto é tudo que precisa saber sobre controllers.

Views

Uma view é simplesmente uma página ou um fragmento de página, como um cabeçalho, um rodapé, uma coluna lateral etc. Aliás, views podem também ser embutidas dentro de outras views se você precisar.

Views nunca são chamadas diretamente, elas precisam ser carregadas por um controller. Lembre-se que num framework MVC, o Controller atua como um guarda de trânsito, sendo responsável por controlar o tráfego de dados e chamar a view quando preciso. Se você ainda não leu a pgáina Controllers, o faça antes de continuar lendo isto.

Usando o controller de exemplo que você criou na página controller, vamos adicionar uma view a ele.

Criando uma View

Usando seu editor de texto, crie um arquivo chamado blogview.php, e coloque isto nele:

```
<html>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/general/views.html
by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:55 GMT -->
<head>
<title>Meu Blog</title>
</head>
<body>
    <h1>Bem-vindo ao meu Blog!</h1>
</body>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/general/views.html
by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:55 GMT -->
</html>
```

Salve o arquivo em seu diretório application/views/.

Carregando uma View

Para carregar um arquivo view específico, você deve usar a seguinte função:

$this->load->view('nome');

Onde nome é o nome de seu arquivo view. Note: A extensão .php não precisa ser especificada a menos que você use outra coisa além dela.

Agora, abra o arquivo controller blog.php que você fez anteriormente e substitua o comando echo com a função de carregamento da view:

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

Se visitar seu site usando aquela mesma URL de antes, deverá ver sua nova view. A URL era similar a esta:

www.seu-site.com/index.php/blog/

Armazenando Views dentro de subdiretórios

Seus arquivos view podem também ser armazenados dentro de subdiretórios se você assim quiser organizá-los. Para isto, será preciso incluir o nome do diretório no carregamento da view. Exemplo:

$this->load->view('nome-do-diretorio/nome-do-arquivo');

Adicionando Dados Dinâmicos em uma View

Dados são passados do controller para a view através de uma array ou de um object no segundo parâmetro fa função de carregamento da view. Aqui vai um exemplo utilizando uma array:

```
$data = array(
          'title' => 'Meu Título',
          'heading' => 'Meu cabeçalho'
          'message' => 'Minha Mensagem'
       );
$this->load->view('blogview', $data);
```

E aqui está um exemplo utilizando um objeto:

```
$data = new ClasseQualquer();
$this->load->view('blogview', $data);
```

Note: Se você usar um objeto, as variáveis da classe serão transformadas em elementos de uma array.

Vamos experimentar em nosso arquivo controller. Abra-o e digite isto:

```
<?php
class Blog extends Controller {
	function index()
	{
		$data['title'] = "Meu título real";
		$data['heading'] = "Meu cabeçalho real";
		$this->load->view('blogview', $data);
	}
}
?>
```

Agora, abra seu arquivo view e altere o texto para variáveis que correspondem as chaves (keys) da array em seus dados:

```
<html>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/general/views.html
by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:55 GMT -->
<head>
<title><?=$title;?></title>
</head>
<body>
	<h1><?=$heading;?></h1>
</body>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/general/views.html
by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:55 GMT -->
</html>
```

Carrega a página usando a URL costumeira e você deverá ver as variáveis substituídas.

Note: Você notará que no exemplo acima estamos usando a sintaxe alternativa do PHP. Se você não está familiarizado com ela, leia isto.

Criando Loops

A array de dados que você passa a seus arquivos view não é limitada a simples variáveis. Você pode passar arrays multidimensionais, que podem se repetir em loop para gerar múltiplas linhas. Por exemplo, se você puxar alguma informação de seu banco de dados, ela geralmente virá na forma de uma array multidimensional.

Aqui vai um simples exemplo. Adicione isto em seu controller:

```
<?php
class Blog extends Controller {
	function index()
	{
		$data['afazeres'] = array('Limpar a Casa', 'Ligar pra a mamãe', 'Jogar uma pelada');
		$data['title'] = "Meu título real";
		$data['heading'] = "Meu cabeçalho real";
		$this->load->view('blogview', $data);
	}
}
?>
```

Abra agora seu arquivo view e crie um loop:

```
<html>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/general/views.html
by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:55 GMT -->
<head>
<title><?=$title;?></title>
</head>
<body>
<h1><?=$heading;?></h1>
<h3>Meus Afazeres</h3>
<ul>
<?php foreach($afazeres as $item):?>
<li><?=$item;?></li>
<?php endforeach;?>
</ul>
</body>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/general/views.html
by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:55 GMT -->
</html>
```

Models

Models estão opcionalmente disponíveis para aqueles que quiserem usar a abordagem MVC mais tradicional.

O que é um Model?

Models são classes PHP projetadas para trabalhar a informação em seu banco de dados. Por exemplo, digamos que você use o Code Igniter para gerenciar um blog. Você poderia ter uma classe model contendo funções para inserir, atualizar e buscar seus dados do banco. Mais abaixo, um exemplo de como se parece uma classe Model:

```
class Blogmodel extends Model {

  var $title   = '';

  var $content = '';

  var $date    = '';

  function Blogmodel()

  {

    // Chama o construtor do Model

    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']));

    }

}
```

Note: As funções do exemplo acima usam as funções Active Record para banco de dados.

Anatomia de um Model

Classes Model são armazenadas em seu diretórios application/models/. Elas podem ser aninhadas em subdiretórios se você preferir.

O protótipo básico para uma classe Model é este:

```
class Model_name extends Model {

    function Model_name()

    {

        parent::Model();

    }

}
```

Onde Model_name é o nome da sua classe. Nomes de Classes devem ter inicial maiúscula. Tenha certeza que sua classe estende a classe Model básica.

O nome do arquivo será o nome da classe todo em minúsculas. Por exemplo, se sua classe é essa:

```
class User_model extends Model {

    function User_model()

    {

        parent::Model();

    }

}
```

Então seu arquivo será:

application/models/user_model.php

Carregando um Model

Seus models serão tipicamente carregados e chamados dentro das funções de seus controller. Para carregar um model, você usará a seguinte função:

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

Se seu model estiver em um subdiretório, inclua o caminha relativo para sua diretório. Po exemplo, se você tem um model localizado em application/models/blog/queries.php você o carregará usando:

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

Uma vez carregado, você acessará as funções de seu model utilizando um objeto como o mesmo classe que havia criado:

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

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

Se você quiser atribuire seu model a um objeto de nome diferente, pode especificar isto através do segundo parâmetro da função de carregamento:

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

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

Abaixo um exemplo de um controller, que carrega um model e então serve uma view:

```
class Blog_controller extends Controller {

  function blog()
  {
    $this->load->model('Blog');
    $data['query'] = $this->Blog->get_last_ten_entries();
    $this->load->view('blog', $data);
  }
}
```

Conectando-se a seu Bando de Dados

Quanto um model é carregado, ele NÂO se conecta automaticamente ao seu banco. As seguintes opções de conexão estão disponíveis:

Você pode conectar usando os métodos padrões de bancos de dados, descritos aqui, ou através de sua classe Controller class ou de sua classe Model.

Você pode mandar a função de carregamento do model para se autoconectar passando TRUE (booleano) através do terceiro parâmetro, como definido em arquivo de configuração do banco de dados:

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

Você pode passar as configuração de conexão manualmente, também através do terceiro parâmetro:

```
$config['hostname'] = "localhost";

$config['username'] = "myusername";

$config['password'] = "mypassword";

$config['database'] = "mydatabase";

$config['dbdriver'] = "mysql";

$config['dbprefix'] = "";

$config['pconnect'] = FALSE;

$config['db_debug'] = TRUE;

$config['active_r'] = TRUE;

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

Funções Assistentes (Helpers)

Assistentes, como o nome sugere, lhe ajudam em suas tarefas. Cada arquivo assistente é simplesmente uma comjunto de funções de uma categoria particular. Há os Assistentes de URL, que lhe ajudam a criar links, os Assisntes de Formulário que lhe ajudam a criar os elementos de seu form, os Assistentes de Texto que geram várias rotinas de formatação de texto, os Assistentes de Cookie que criam e lêem cookies, os Assistentes de Arquivo que lhe ajudam a trabalhar com arquivos etc.

Ao contrário de outros sistemas no Code Igniter, os Assistentes não foram escritos num formato Orientado a Objetos. Eles são simples, procedurais funções. Cada função assistente executa uma específica tarefa, sem qualquer dependência com outras funções.

O Code Igniter não carrega os arquivos assistentes por default, por isto o primeiro passo quando for utilizar um Assistente é carregá-lo. Uma vez carregado, ele se fica disponível globalmente em seu controller e views.

Assistentes são tipicamente armazenados em seu diretório system/helpers. Como alternativa, você pode criar um diretório chamado helpers dentro do seu diretório application e armazená-los lá. O Code Igniter irá procurar primeiro em seu diretório system/application/helpers. Caso ele não exista, ou o assistente referenciado não estiver lá, CI irá procurar em seu diretório global system/helpers.

Carregando um Assistente

Carregar um arquivo assistente é bem simples utilizando-se a função:

$this->load->helper('name');

Onde name é o nome do arquivo do assistente, sem a extensão .php ou a parte "helper" aparecendo.

Por exemplo, para carregar o Assistente de URL, que é chamado url_helper.php, você deveria fazer isto:

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

Um assistente pode ser carregado em qualquer lugar dentro de suas funções do controller (ou até dentro de seus arquivos View, embora esta não seja uma boa prática), desde que você o carregue antes de usar. Você pode carregar seus assistentes em seu construtor controller para que eles se tornem disponíveis automaticamente a qualquer função, ou você pode apenas carregá-los diretamente na função que o requer.

Note: A função de carregamento de um Assistente não retorna um valor, por isso não tente atribuí-la a uma variável. Então só a use como mostrado.

Carregando Múltipos Assistentes

Se você precisa carregar mais de um assistente pode especificá-los numa array, como esta:

$this->load->helper(array('assistente1', 'assistente2', 'assistente3'));

Carregando Assistentes Automaticamente

Se você descobrir que precisa de um assistente global em sua aplicação, pode dizer ao Code Igniter para carregá-lo automaticamente durante a inicialização do sistema. Isto é feito adicionando-se ao arquivo application/config/autoload.php, o assistente ao array autoload.

Usando um Assistente

Carregado o arquivo assistente que contém a função que você pretende usar, você deverá chamá-lo da mesma forma que chamaria uma função PHP.

Por exemplo, para criar um link utilizando a função anchor() em uma de suas views, faça o seguinte:

<?=anchor('blog/comments', 'Clique Aqui');?>

Onde "Clique aqui" é o texto do link e "blog/comments" é o URI para o controller/função que que será linkado.

E agora?

Na listagem de Conteúdos você irá encontrar uma lista de todos os Assistentes disponíveis. Navegue por todos para descubrir o que eles fazem.

Plugins

Plugins trabalham de forma praticamente idêntica aos Assistentes. A principal diferença é que um plugin geralmente provê uma simples função, enquanto o assistente é geralmente uma coleção de funções. Assistentes são também considerados uma parte do núcleo do sistema; A idéia é que os plugins sejam criados e compartilhados por nossa comunidade..

Plugins deveriam ser salvos em seu diretório system/plugins mas você pode criar um diretório chamado plugins dentro do seu application e guardá-los lá. Code Igniter irá primeiro procurá-los em seu diretório system/application/plugins. Se o diretório não existir ou se o plugin não for localizado lá, o CI procurará então em seu diretório global system/plugins.

Carregando um Plugin

Carregar um plugin é bem simples. Utilize a seguinte função:

$this->load->plugin('name');

Onde name é o nome do arquivo do plugin, sem a extensão .php ou "plugin".

Por exemplo, para carregar o plugin Captcha, cujo nome é captcha_pi.php, você deverá fazer isto:

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

Um plugin pode ser carregando de qualquer lugar dentro das funções de seu controller (ou até mesmo dentro de suas View files, mas esta não é boa prática), desde que você os carregando antes de usá-los. Você pode carregar seus plugins em seu controller construstor para que fiquem disponíveis automaticamente para qualquer função, ou você pode carregá-los dentro da função que precisa deles.

Note: A função de carregamento de Plugins acima não retorna um valor, então não tente atribuí-la à uma variável. Então só a use como mostrado.

Carregando Múltiplos Plugins

Se você precisa carregar mais de um plugin pode especificá-los numa array, como esta:

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

Carregando Plugins Automaticamente

Se você descobrir que precisa de um plugin global em sua aplicação, pode dizer ao Code Igniter para carregá-lo automaticamente durante a inicialização do sistema. Isto é feito adicionando-se ao arquivo application/config/autoload.php, o plugin ao array autoload.

Usando um Plugin

Carregado o arquivo Plugin, você deverá chamá-lo da mesma forma que chamaria uma função PHP.

Utilizando as Bibliotecas do Code Igniter

Todas as bibliotecas disponíveis estão localizadas em seu diretório system/libraries. Na maioria dos casos, usar uma dessas classes envolve inicializá-la dentro de umcontroller utilizando a seguinte função:

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

Onde classe é a classe que você deseja chamar. Por exemplo, para carregar a classe de validação, você deverá fazer o seguinte:

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

Uma vez inicializada, você pode usá-la como indicado na página correspondente deste Guia.

Criando suas próprias Bibliotecas

Por favor, leia a seção do Guia do Usuário que discute como criar suas próprias bibliotecas.

Criando Bibliotecas

Quando utilizamos o termo "Bibliotecas", normalmente estamos nos referindo as clases localizadas no diretório libraries e descritas na seção Referências de Classes deste guia. Neste caso, porém, iremos mostrar como você pode criar suas próprias bibliotecas dentro de seu diretório application/libraries a fim de manter uma separação entre seus recursos locais e os recursos globais do framework.

Como bônus, o Code Igniter permite que suas bilbliotecas estender classes nativas se você simplismente precisar adicionar alguma funcionalidade para uma biblioteca existente. Você ainda pode sobrescrever as bibliotecas nativas pelas suas em seu diretório application/libraries.

Sumário:

Você pode criar bibliotecas inteiramente novas.

Você pode estender bibliotecas nativas.

Você pode sobrescrever bibliotecas nativas.

O texto abaixo detalha estes três conceitos.

Note: As classes de Banco de Dados e as classes Controller principais não podem ser estendidas ou sobrescritas pelas classes que você criar. Mas todas as outras podem.

Armazenamento

Sua biblioteca de classes deve ser colocada em seu diretório application/libraries, pois este será o lugar onde o Code Igniter irá procurá-las quando foram inicializadas.

Convenções de Nomenclatura

Nomes de arquivo devem ter sua inicial maiúscula. Por exemplo: Minhaclasse.php

Declarações de Classes devem ter sua inicial maiúscula. Por exemplo: class Minhaclasse

Nomes de classes e nomes de arquivos devem ser iguais.

O arquivo de Classe

Classes devem ter este protótipo básico (Note: Estamos utilizando o nome Algumaclasse puramente como exemplo):

```
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');

class Algumaclasse {

   function some_function()

   {

   }

}

?>
```

Utilizando sua classe

De dentro de qualquer função Controller você pode inicializar suas classes utilizando o padrão:

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

Onde Algumaclasse é o nome do arquivo, sem a extensão ".php". Você pode dar submit no nome do arquivo como inicial maiúscula ou minúscula. O Code Igniter não se importa.

Uma vez carregada, você pode acessar sua classe utilizando a versão em caixa baixa:

$this->Algumaclasse->some_function(); // Instâncias de Objeto serão sempre em caixa baixa

Passando parâmetros ao inicializar sua classe

Em sua função de carregamento da biblioteca, você pode passar dados dinamicamente, através do segundo parâmetro, para seu construtor de classes:

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

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

Se você usar este recurso deve obrigatoriamente configurar seu construtor para esperar o dado:

```
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');

class Algumaclasse {

   function Algumaclasse($params)

   {

      // Faça alguma coisa com o $params

   }

}

?>
```

Você pode também passar parâmetros armazenados no arquivo de configuração. Simplesmente crie um arquivo config com nome idêntico ao do nome de arquivo da classe e salve-o em seu diretórioapplication/config/. Note que se você passar dinamicamente parâmetros, como descrito acima, a opção do arqivo config não estará disponível.

Utilizando os Recursos do Code Igniter dentro de sua Biblioteca

Para acessar os recursos nativos do Code Igniter dentro de sua biblioteca use a função get_instance(). Esta função retorna o super objeto Code Igniter.

Normalmente dentro de suas funções controller você poderá chamar qualquer função disponível no CI usando o construtor $this:

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

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

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

etc.
```

$this, porém, apenas trabalha diretamente dentro de seus controllers, models ou views. Se você quiser usar as classes do Code Igniter dentro de suas próprias funções personalizadas, pode fazer da sequinte forma:

Primeiro, atribua o objeto Code Igniter a uma variável:

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

Uma vez feito isto, você irá usar esta variável no lugar de $this:

```
$CI =& get_instance();

$CI->load->helper('url');

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

$CI->config->item('base_url');

etc.
```

Note: Você irá notar que a função get_instance() acima está sendo passada por referência:

$CI =& get_instance();

Isto é muito importante. Atribuir por referência permite que você use o objeto Code Igniter original, ao invés de criar uma cópia dele.

Sobrescrevendo Bibliotecas Nativas por suas Próprias Versões

Simplesmente nomear seus aquivos de classe de forma idêntica à uma biblioteca nativa irá forçar o Code Igniter a usar aquele no lugar do nativo. Para usar este recurso você deve nomear o arquivo e a declaração da classe com o mesmo nome da biblioteca nativa. Por exemplo, para sobrescrever a biblioteca nativa Email crie um arquivo chamado application/libraries/Email.php, e declare sua variável como:

```
class CI_Email {

}
```

Note que a maioria das classes nativas contém o prefixo CI_.

Para carregar sua biblioteca use a função padrão:

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

Note: Até este momento as classes de Banco de Dados não podem ser sobrescritas pelas versões que você criar.

Estendendo Bibliotecas Nativas

Se tudo o que precisa é adicionar uma certa funcionalidade à uma biblioteca existente - talvez uma ou duas funções - então é desnecessário substituir a biblioteca inteira com sua versão. Neste caso, é melhor simplismente estender a classe. Estender uma classe é praticamente igual a substituí-la, com algumas exceções:

A declaração da classe deve estender a classe pai.

O novo nome de seu arquivo da classe devem conter o prefixo MY_ (este item é configurável. Veja abaixo.):

Por exemplo, para estender a classe Email nativa, você precisa criar um arquivo chamado application/libraries/MY_Email.php, e declarar sua classe com:

```
class MY_Email extends CI_Email {

}
```

Note: Se você precdisar usar um construtor em sua classe, tenha certeza de estender o construtor pai:

```
class MY_Email extends CI_Email {

  function My_Email()

  {

    parent::CI_Email();

  }

}
```

Carregando sua Subclasse

Para carregar sua subclasse você irá usar a sintaze padrão normalmente utilizada. NÂO INCLUA seu prefixo. Por exemplo, para carregar o exemplo acima, que estende a classe Email, use:

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

Uma vez carregada, você irá usar a variável classe da mesma forma que usaria para a classe que irá estender. No caso da classe email, todas as chamadas irão usar:

$this->email->alguma_função();

Configurando seu próprio Prefixo

Para configurar seu próprio prefixo de subclasse, abra seu arquivo application/config/config.php e procure este item:

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

Por favor, note que todas as bibliotecas nativas do Code Igniter têm o prefixo CI_ então NÂO USE isto como seu prefix.

Criando Classes do Núcleo do Sistema

Toda vez que o Code Igniter roda, há várias classes básicas que são inicializadas automaticamente como parte do número do framework. É possível, porém, trocar qualquer uma destas classes básicas como suas próprias versões ou mesmo estender as versões de núcleo.

A maioria dos usuários não irá precisar de nada disto, mas a opção de substituir ou estender classes existem para aqueles que podem querer alterar significativamente o número do Code Igniter.

Note: Mexer com uma classe de núcleo do sistema tem muitas implicações, por isso tenha certeza que você saiba o que está fazendo antes de tentá-lo.

Lista de Classes do Sistema

Abaixo vai a lista dos aruqivos do núcleo do sistema que são chamados toda vez que o Code Igniter roda:

Benchmark

Input

Config

Hooks

Router

URI

Language

Loader

Controller

Output

Substituindo Classes de Núcleo

Para usar suas próprias classes de sistema no lugar das padrões, simplesmente coloque suas versão dentro de seu diretório local application/libraries:

application/libraries/algumaclasse.php

Se este diretório não existir, você pode criá-lo.

Qualquer arquivo com o mesmo nome de um item da lista acima será escolhido no lugar daquele que normalmente seria usado.

Por favor, note que sua classe precisa usar o prefixo CI. Por exemplo, se seu arquivo chama-se Input.php, o nome da classe será:

```
class CI_Input {

}
```

Estendendo uma Classe de Núcleo

Se tudo o que precisa é adicionar uma certa funcionalidade à uma biblioteca existente - talvez uma ou duas funções - então é desnecessário substituir a biblioteca inteira com sua versão. Neste caso, é melhor simplismente estender a classe. Estender uma classe é praticamente igual a substituí-la, com algumas exceções:

A declaração da classe deve estender a classe pai.

O novo nome de seu arquivo da classe devem conter o prefixo MY_ (este item é configurável. Veja abaixo.):

Por exemplo, para estender a classe Input nativa, você precisa criar um arquivo chamado application/libraries/MY_Input.php, e declarar sua classe com:

```
class MY_Input extends CI_Input {

}
```

Note: Se você precdisar usar um construtor em sua classe, tenha certeza de estender o construtor pai:

```
class MY_Input extends CI_Input {
  function My_Input()
  {
    parent::CI_Input();
  }
}
```

Dica: Quaisquer funções em sua classe que tiverem o mesmo nome das funções na classe pai serão usadas no lugar as nativas (isto é conhecido como "método do sobrecarregamento"). Isto permite que você altere substancialmente o núcleo do Code Igniter.

Configurando seu próprio Prefixo

Para configurar seu próprio prefixo de subclasse, abra seu arquivo application/config/config.php e procure por este item:

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

Por favor, note que todas as bibliotecas nativas do Code Igniter têm o prefixo CI_ então NÂO USE isto como seu prefixo.

Ganchos - Estendendo o Núcleo do Framework

O recurso de Ganchos do Code Igniter provê uma maneira de acessar e modificar os processos internos do framework sem hackear os arquivos base. Quando o Code Igniter roda, ele segue um específico processo de execução, diagramado na página do Fluxograma da Aplicação. Haverá momentos, no entanto, onde você queira que alguma ação seja executada em um estágio particular do processo de execução. Por exemplo, você pode querer rodar um script logo antes do seu controller ser carregado, o logo depois, ou você pode querer disparar um de seus próprios scripts em algum outro lugar.

Habilitando os Ganchos

O recurso de ganchos pode ser habilitado/desabilitado globalmente, configurando-se o seguinte item no arquivo application/config/config.php:

$config['enable_hooks'] = TRUE;

Definindo um Gancho

Ganchos são definidos no arquivo application/config/hooks.php. Cada um é especificado como uma array com esse protótipo:

```
$hook['pre_controller'] = array(
                'class'    => 'MinhaClasse',
                'function' => 'MinhaClasse',
                'filename' => 'MinhaClasse.php',
                'filepath' => 'MinhaClasse',
                'params'   => array('cerveja', 'vinho', 'salgadinhos')
                );
```

Notas:

O índice da array corresponde ao nome do particular ponto de gancho que você queira usar. No exemplo acima, o ponto de gancho é pre_controller. Uma lista de pontos de ganchos aparece abaixo. Os seguintes itens devem ser definidos em sua array de gancho associada:

class O nome da classe que você deseja chamar. Se desejar usar uma função procedural no lugar de uma classe, deixe este item em branco.

function A função que deseja chamar.

filename O nome do arquivo contendo sua classe/função.

filepath O nome do diretório contendo seu script. Note: Seu script deve estar localizado num diretório DENTRO do seu diretório application, portanto o caminho do arquivp é relativo a este diretório. Por exemplo, se seu script está localizado em application/hooks, simplesmente use hooks como seu caminho de arquivo. Se seu script está localizado em application/hooks/utilities use hooks/utilities como seu caminho de arquivo. Sem a barra invertida.

params Quaisquer parâmetros que deseje passar ao script. Este item é opcional

Múltiplas chamadas para o mesmo Gancho

Se você quer usar o mesmo ponto de gancho em mais de um script, simplemente faça sua array multi-dimensional, como esta:

```
$hook['pre_controller'][] = array(
                'class'    => 'MinhaClasse',
                'function' => 'MinhaClasse',
                'filename' => 'MinhaClasse.php',
                'filepath' => 'MinhaClasse',
                'params'   => array('cerveja', 'vinho', 'salgadinhos')
                );
$hook['pre_controller'][] = array(
                'class'    => 'MinhaOutraClasse',
                'function' => 'MinhaOutraFuncao',
                'filename' => 'MinhaOutraClasse.php',
                'filepath' => 'ganchos',
                'params'   => array('vermelho', 'amarelo', 'azul')
                );
```

Note os colchetes após cada indíce da array:

$hook['pre_controller'][]

Isto permite que você use o mesmo ponto de gancho com múltiplos scripts. A ordem que você definir em sua array será a ordem de execução.

Pontos de Gancho

Abaixo está uma lista dos pontos disponíveis.

pre_system

Chamado bem cedo durante a execução do sistema. Apenas as classes benchmark e hooks estarão carregadas nesta hora. Nem roteamento ou outros processos rodaram.

pre_controller

Chamado imediatamente antes de qualquer um de seus controllers serem chamados. Todas as classes básicas, roteamento, e checagem de segurança rodaram.

post_controller_constructor

Chamado imediatamente após seu controller ser instanciado, mas antes de qualquer chamadas de método rodarem.

post_controller

Chamado imediatamente após seu controller ser totalmente executado.

display_override

Sobrepôe a função _display(), usada para enviar a página finalizada ao web browser no final da execução do sistema. Isto permite que você use sua própria metodologia de display. Note que os dados finalizados estão disponíveis ao se chamar $this->output->get_output()

cache_override

Permite que você chame sua própria função no lugar da _display_cache() na classe output. Isto possibilita que você use seu próprio mecanismo de cache display.

scaffolding_override

Permite uma requisição scaffolding para disparar seu próprio script.

post_system

Chamado após a página renderizada final ser enviada ao browser, no final da execução do sistema após os dados finalizados serem enviados ao browser.

Recursos de Autocarregamento

O Code Igniter vem com um recurso de "Auto-load" que permite que bibliotecas, assistentes e plugins sejam inicializados automaticamente toda vez que o sistema rodar. Se você precisa de certos recursos globais em sua aplicação, deve considerar carregá-los automaticamente para sua conveniência.

Os itens abaixo podem ser carregados automaticamente:

Classes núcleo encontradas no diretório "libraries"

Arquivos de Assistentes encontrados no diretório "helpers"

Plugins encontrados no diretório "plugins"

Arquivos de configuração personalizados encontrados no diretório "config"

Para carregar recuros automaticamente, abra o arquivo application/config/autoload.php e adicione o item que deseja carregar à array autoload. Você encontrará instruções no arquivo correspondente a cada tipo de item.

Note: Não inclusa a extensão (.php) quando estiver adicionando os itens à array autoload.

Scaffolding

O recurso Scaffolding do Code Igniter provê uma maneira rápida e muito conveniente de adicionar, editar ou apagar informações de seu banco de dados durante o desenvolvimento.

Muito Importante: O Scaffolding foi planejado para uso apenas durante o desenvolvimento. Ele provê mínima segurança, apenas através de uma palavra "mágica", portanto qualquer pessoa que tenha acesso ao seu site pode potencialmente editar ou apagar suas informações. Se você utilizar este recurso, tenha certeza de desabilitá-lo imediatamente ao término do desenvolvimento do site. NÂO DEIXE-O habilitado em um site no ar. E por favor, configure uma palavra secreta antes de usá-lo.

Porque você usaria Scaffolding?

Um típico cenário: Você criou uma nova tabela no banco durante o desenvolvimento e gostaria de um jeito rápido de inserir alguma informação nela para testar. Sem o scaffolding suas escolhas são ou dar alguns inserts via linha de comando ou usar uma gerenciador como o phpMyAdmin. Com o recurso de scaffolding do Code Igniter, você pode rapidamente adicionar alguma informação usando uma interface no browser. E quando terminar de usar a informação, pode facilmente apagá-la.

Configurando uma Palavra Secreta

Antes de habilitar o scaffolding, por favor, crie uma palavra secreta. Esta, quando encontrar em sua URL, iniciará a interface scaffolding, então por favor, escolha algo tão obscuro que ninguém será capaz de advinhar..

Para configurar uma palavra secreta, abra seu arquivo application/config/routes.php e procure por este item:

$route['scaffolding_trigger'] = '';

Ao encontrá-lo, adicione sua distinta palavra.

Note: A palavra do scaffolding não pode iniciar com um underscore.

Habilitando o Scaffolding

Note: As informações nesta página assume que você já sabe como os controllers funcionam, e que você tem algum deles funcionando e disponível. Também se assume que você configurou o Code Igniter para se autoconectar a seu banco de dados. Se não, as informações aqui não serão muito relevantes, por isso você é encorajado a executar os dois itens acima primeiro. Por último, assume-se que você o que é um construtor de classe. Se não, leia a última seção da página controllers.

Para habilitar o scaffolding você precisa inicializá-lo em seu construtor como mostrado abaixo:

```
<?php
class Blog extends Controller {
    function Blog()
    {
        parent::Controller();
        $this->load->scaffolding('table_name');
    }
}
?>
```

Onde table_name é o nome da sua tabela (tabela, e não da base de dados) com a qual você deseja trabalhar.

Uma vez inicializado o scaffolding, você o acessrá com este protótipo de URL:

www.seu-site.com/index.php/classe/palavra_secreta/

Por exemplo, utilizando um controller chamado Blog, e abracadabra sendo a palavra secreta, você acessaria o scaffolding desta forma:

www.seu-site.com/index.php/blog/abracadabra/

A interface do scaffolding deverá ser autoexplicativa. Você pode adicionar, editar ou apagar registros.

Nota Final:

O recurso de scaffolding somente irá funcionar com tabelas que contém uma chave primária, por ser esta a informação necessária para executar as várias funções do banco de dados.

Roteamento URI

Tipicamente há uma relação um-a-um entre a string URL e seu correspondente controller classe/método. Os segmentos em um URI normalmente seguem este padrão:

www.seu-site.com/classe/funcao/id/

Em alguns momentos, porém, você pode querer remapear estar relação para que uma classe/função diferente possa ser chamada no lugar daquela correspondente à URL.

Por exemplo, vamos dizer que você queria suas URLs tenham este protótipo:

www.seu-site.com/produto/1/

www.seu-site.com/produto/2/

www.seu-site.com/produto/3/

www.seu-site.com/produto/4/

Normalmente o segundo segmento da URL está reservado para o nome da função, mas no exemplo acima há um ID de produto no lugar. Para resolver isto, o Code Igniter permite que você remapeie o handler URI.

Configurando suas próprias regras de roteamento

Regras de roteamento são definidas em seu arquivo application/config/routes.php. Nele, você verá uma array chamada $route que permite que você especifique seus próprios critérios. As rotas podem tanto ser especificadas usando caracteres coringa (wildcards) como Expressões Regulares.

Caractere Coringa - Wildcards

Uma típica rota coringa se parece com esta:

$route['produto/:num'] = "catalogo/produto_busca";

Numa rota, a chace da array contém o URI a ser igualado, enquanto que o valor da arry contém o destino para qual ele deverá ser encaminhado, a nova rota. No exemplo acima, se a palavra literal "produto" for encontrada no primeiro segmento da URL, e um número for encontrado no segundo segmento, a classe "catalogo" e o método "produto_busca" serão utilizados no lugar.

Você pode igualar valores literais ou então usar dois tipos de coringas:

:num

:qualquer

:num irá igualar um segmento contendo apenas números.

:qualquer irá igualar um segmento contendo qualquer caracter.

Note: As rotas serão executadas na ordem em que forem definidas. Rotas de nível mais alto sempre terão precedência às de nível mais baixo.

Exemplos

Aqui vão alguns exemplo de roteamento:

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

Qualquer URL contendo a palavra "jornauls" no primeiro segmento irá ser remapeada à classe "blogs".

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

Qualquer URL contendo o segmento blog/joe será remapeada à classe "blogs" e ao método "users". O ID será 'setado' como "34".

$route['produto/:qualquer'] = "catalogo/produto_busca";

Qualquer URL contendo "produto" como primeiro segmento, e nada no segundo, será remapeada à classe "catalogo" e ao método "produto_busca".

Importante: Não use barras invertidas ou normais.

Expressões Regulares

Se você preferir, pode usar expressões regulares para definir suas regras de rotas. Qualquer expressão regular válida é permitida, assim como back-references.

Note: Se você usar back-references, deve usar a sintaxe com o sinal de dolar ($) no lugar da de dupla barra invertida (//).

Uma típica rota RegEx se parece com algo assim:

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

No exemplo acima, um URI similar a produtos/camisas/123 iria chamar a classe controller camisas e a função id_123.

Você pode também misturar coringas com expressões regulares.

Rotas Reservadas

Há duas rotas reservadas:

$route['default_controller'] = 'bem-vindo';

Esta rota indica qual classe controle deverá ser chamada se o URI não contiver dados, o que seria o caso quando alguém carrega a URL raiz. No exemplo cima, a classe "bem-vindo" seria carregada. Você é encorajado a sempre ter uma rota default, do contrário uma página 404 irá aparecer como default.

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

Esta rota permite que você configure a palavra secreta, que quando presente na URL, dispara o recurso de scaffolding. Por favor, leia a página Scaffolding para detalhes.

Tratamento de Erros

O Code Igniter permite que você construa relatórios de erros em suas aplicações utilizando as funções descritas abaixo. Além disso, ele tem uma classe de log de eros que permite que as mensagens de erro e debug sejam salvas como arquivos texto.

Note: Por default, O Code Igniter mostra todos os erros PHP. Você poderia desejar alterar este comportamente uma vez que seu desenvolvimento estiver completo. Descubrirá que a função error_reporting(), localizada no topo de seu aquivo index.php principal. Desabilitar relatórios de erros NÂO irá evitar que os arquivos de log sejam escritos se houver algum erro.

Ao contrário da maioria dos sistemas no Code Igniter, as funções de erro são simples interfaces procedurais que estão disponíveis globalmente através de sua aplicação. Esta abordagem permite que mensagens de erro sejam disparadas sem que você se preocupe com o escopo da classe/função.

As seguintes funções permitem que você gere erros:

show_error('messagem')

Esta função irá mostrar a mensagem de erro disponível à ela utilizando o seguinte template:

application/errors/error_general.php

show_404('pagina')

Esta função irá mostrar a mensagem de erro 404 disponível à ela utilizando o seguinte template:

application/errors/error_404.php

A função expera que a string passada à ela seja o caminho do arquivo para a página que não foi encontrada. Note que o Code Igniter automaticamente mostra mensagem 404 se os controllers não são encontrados.

log_message('nivel', 'messagem')

Esta função permite que você escreva mensagens em seus arquivos de log. Você deve fornecer um destes 3 'níveis' no primeiro parâmetro, indicando qual é o tipo de mensagem (debug, erro, info), e com a própria mensagem como segundo parâmetro. Exemplo:

```
if ($qualquer_variavel == "")
{
   log_message('error', 'Qualquer variável não contém um valor.');
}
else
{
   log_message('debug', 'Qualquer variável foi atribuída corretamente');
}
log_message('info', 'O propósito de qualquer variável é fornecer algum valor.');
```

Há três tipos de mensagens:

Mensagens de Erro. Estas não são na verdade erros, como os erros de PHP ou de usuário.

Mensagens de Debug. São mensagens que ajudam no debugging. Por exemplo, se uma classe foi inicializada, você pode registrar isto como uma informação de debugging.

Mensagens Informacionais. Estas são as demais baixa prioridade, simplismente dão informações sobre algum processo. O Code Igniter não gera nativamente qualquer mensagem informacional mas você pode querê-las em sua aplicação.

Note: Para que o arquivo de log seja escrito, o diretório 'logs' deve estar com permissão de escrita. E você deve, ainda, configurar o que deve ser registrado. Você pode, por exemplo, querer que apenas as mensagens de erro sejam registradas e não os outros dois tipos. Se você configurá-las para zero, o registro será desabilitado.

Cache de Página

O Code Igniter permite que você cacheie suas páginas para atingir máxima performance. Mesmo o Code Igniter sendo bem rápido, a quantidade de informações dinâmicas que você mostra em suas páginas será igualada diretamente aos recursos do seu servidor, memória e ciclos de processamento utilizados, o que afetam a velocidade com que suas páginas são carregadas. Cacheando suas páginas, uma vez que elas são salvas em seu estado totalmente renderizado, você pode obter uma performance próxima a de uma página estática.

Como o cache funciona?

O cache pode ser habilitado página-a-página e você pode configurar a duração do tempo que a página deve permanecer cacheada antes de ser atualizada. Quando uma página é carregada pela primeira vez, o arquivo de cache será escrito em seu diretório system/cache. Nos carregamentos subsequentes, o arquivo de cache será buscado e enviado ao browser que o requisitou. Se ele houver expirado, será apagado e atualizado antes de ser enviado ao browser.

Note: a tag de Benchmark não é cacheada, por isso você ainda poderá ver sua velocidade de carregamento mesmo com o cache habilitado.

Habilitando o Cache

Para habilitá-lo, coloque a seguinte tag em qualquer uma de suas funções controller:

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

Onde n é o número de minutos que você deseja que a página permaneça cacheada entre as atualizações.

A tag acima pode ser colocada em qualquer lugar dentro da função. Ela não é afetada pela ordem em que aparece, por iso coloque-o onde for mais lógico para você. Uma vez que a tag esteja no lugar, suas páginas passarão a ser cacheadas.

Note: Antes dos arquivos de cache poderem ser escritos, você deve configurar as permissões de escrita em seu diretório system/cache. (666 é geralmente apropriado).

Apagando Caches

Se você não mais deseja cachear um arquivo, pode remover a tag de cache e ele não mais será atualizado quando expirar. Note: Remover a tag não irá apagar o cache imediatamente. Ele irá expirar normalmente. Se você precisa removê-lo antes disto, precisará apagá-lo manualmente de seu diretório.

Gerenciando suas Aplicações

Por default, assume-se que você pretende utilizar o Code Igniter para gerenciar somente uma aplicação, a qual você constrói em seu diretório system/application/. É possível, no entando, ter múltiplas aplicações que compartilham uma única instalação do CI, ou mesmo renomear ou realocar seu diretório application.

Renomeando o Diretório de Aplicação

Se você desejar renomar seu diretório applicaiton pode fazê-lo, desde que abra seu arquivo index.php e altere seu nome utilizando a variável $application_folder:

$application_folder = "nome-da-aplicacao";

Realocando seu Diretorio de Aplicação

É possível mover seu diretório application para um outro lugar em seu servidor ao invés do diretório system. Para isto, abra seu arquivo index.php principal e configure um caminho completo do servidor na variável $application_folder.

$application_folder = "/caminho/para/sua/aplicacao";

Rodando Múltipas Aplicações com uma só Instalação do Code Igniter

Se você quiser compartilhar uma instação do CI comum para gerenciar várias aplicações diferentes, simplesmente coloque todos os diretórios que estão localizados dentro de seu diretório application para dentro de seu próprio subdiretório.

Por exemplo, digamos que você queira criar duas aplicações, "foo" e "bar". Você iria estruturar seu diretório de aplicação desta forma:

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/

Selecionar uma aplicação particular para uso, requer que você abra seu arquivo index.php principal e configure a variável $application_folder. Por exemplo, para selecionar a aplicação "foo" você deverá fazer isto:

$application_folder = "application/foo";

Note: Cada uma das suas aplicações precisará ter seu próprio arquivo index.php que chamará a aplicação desejada. O arquivo index.php pode ser renomeado para qualquer coisa que você quiser.

Sintaxe PHP Alternativa para arquivos de Visualização (View files)

Se você não utilizar o template engine do Code Igniter, estará utilizando puro PHP em seus arquivos de visualização. Para minimizar o código PHP nestes arquivos, e tornar mais fácil identificar os blocos de código, é recomendado que você use a sintaxe alternativa do PHP para estruturas de controle e expressões echo curtas. Se você não está familiarizado com esta sintaxe, ela permite que você elimine as chaves no código, e também os comandos "echo".

Suporte automático a Short Tag

Nota: Se descobrir que a sintaxe descrita nesta página não funciona em seu servidor, deve ser porque as "short tags" estão desabilitadas em seu arquivo PHP ini. O Code Igniter irá, opcionalmente, reescrever as short tags on-the-fly, permitindo que você use esta sintaxe mesmo que seu servidor não a suporte. Esta funcionalidade pode ser habilitada em seu arquivo config/config.php.

Por favor note que, usando funncionalidade, erros PHP que possam acontecer em seus arquivos View, não terão sua mensagem e número de linha precisamente mostrados. Ao invés disto, todos os erros serão mostrados como erros eval().

Echos Alternativos

Normalmente para dar echo ou print numa váriavel, você faria assim:

<?php echo $variable; ?>

Com a sintaxe alternativa, pode fazer desta maneira:

<?=$variable?>

Estruturas de Controles Alternativas

Estruturas de controle, como if, for, foreach, e while também podem ser escritas de um jeito simplificado. Aqui vai um exemplo usando foreach:

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

Note que não há chaves. Ao invés disto, a chave do final foi trocada por um endforeach. Cada estrutura listada acima tem uma sintaxe de fechamento similar: endif, endfor, endforeach, e endwhile

Note também que no lugar de usar um ponto e vírgula para cada estrutura (exceta a última), há o sinal de dois pontos. Isto é importante!

Outro exemplo, usando if/elseif/else. Note os dois pontos:

```
<?php if ($username == 'sally'): ?>
  <h3>Oi Sally</h3>
<?php elseif ($username == 'joe'): ?>
  <h3>Oi Joe</h3>
<?php else: ?>
  <h3>Oi usuário desconhecido</h3>
<?php endif; ?>
```

Segurança

Esta página descreve algumas "boas práticas" acerca da segurança na web e detalha as funcionalidades internas de segurança do Code Igniter.

Segurança URI

O Code Igniter é basicamente restritivo acerca dos caracteres aceitos em suas strings URI, de maneira a ajudar a minimizar a possibilidade que dados maliciosos possam ser passados para sua aplicação. URIs podem conter apenas o seguinte:

Texto Alfanumérico

Til: ~

Ponto-final: .

Dois pontos: :

Underscore: _

Hífen: -

GET, POST, e COOKIES

Dados via GET estão simplesmente desabilitados pelo Code Igniter uma vez que o sistema utiliza segmentos URI no lugar das tradicionais query strings URI (a menos que você habilite a opção de usá-las no seu arquivo de configuração). A array global GET não é setada pela classe Input durante a inicialização do sistema.

Register_globals

Durante a inicialização do sistemas, todas as variáveis globais não estão setadas, exceto as encontradas nas arrays $_POST e $_COOKIE. A rotina de configuração (set/unset) é efetivamente a mesma à register_ globals = off.

magic_quotes_runtime

A diretiva magic_quotes_runtime está desligada durante a inicialização do sistema, por isso você não precisa remover as barras quando buscar dados do banco.

Boas Práticas

Antes de aceitar qualquer dado em sua aplicação, sejam dados POST vindos de um submit de um formulário, COOKIES, dados URI, dados XML_RPC, ou mesmo dados vindos da array SERVER, você é encorajados a praticar esta abordagem de 3 passos:

Filtrar os dados supondos estarem infectados ou corrompidos.

Validar os dados para assegurar que estão em conformidade aos corretos tipo, extensão, tamanho etc. (algumas vezes esta passo pode substituir o primeiro)

Dê um Escape nos dados antes de submetê-los ao banco.

O Code Igniter provê as seguintes funções para ajudá-lo neste processo:

Filtragem XSS

O Code Igniter vem com um filtro Cross Site Scripting. Este filtro procura por técnicas comumente usadas para embutir código Javascript malicioso em seus dados, ou outros tipos de códigos que tentam sequestrar cookies ou causar outros danos. O filtro XSS é descrito aqui.

Valide os dados

O Code Igniter tem uma Classe de Validação que ajuda você a validar, filtrar e preparar seus dados.

Dê um Escape nos dados antes de submetê-los ao banco

Nunca insira informação em seu banco sem antes convertê-la via escape. Por favor, veja a seção que discorre sobre queries para mais informações.

Referência de Classes

Classe Benchmarking

O Code Igniter tem uma classe de Benchmarking que está sempre ativa, permitindo verificar a diferença de tempo entre dois pontos marcados para serem calculados.

Nota: Esta classe é inicializada automaticamente pelo sistema, então não é necessário fazer isto manualmente.

O benchmark sempre é iniciado no momento em que o framework é carregado, e terminado pela classe de saída (output), assim que é enviada a visualização para o navegador, permitindo uma eficiente verificação e exibição de tempo de execução de todo o sistema.

Conteúdo

Usando a classe de Benchmark

Marcas de Benchmark na classe de Profile

Exibindo o Tempo Total de Execução

Exibindo o Consumo de Memória

Usando a classe de Benchmark

A classe de Benchmark pode ser utilizada com seus controllers, views, ou seus Models. O processo de uso é:

Marcar um ponto inicial

Marcar um ponto final

Chamar a função de "tempo decorrido" (elapsed_time) para visualizar os resultados

Segue aqui um exemplo real de uso:

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

// Algum codigo e executado aqui

$this->benchmark->mark('code_end');

echo $this->benchmark->elapsed_time('code_start', 'code_end');
```

Nota: As palavras "code_start" e "code_end" podem ser quaisquer palavras. Eles são apenas palavras para referenciar as duas marcas. Voce pode usar quaisquer palavras, opu ainda usar multiplas marcas. Veja este exemplo:

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

// Algum codigo aqui

$this->benchmark->mark('gato');

// Mais codigo aqui

$this->benchmark->mark('rato');

echo $this->benchmark->elapsed_time('cao', 'gato');

echo $this->benchmark->elapsed_time('gato', 'rato');

echo $this->benchmark->elapsed_time('cao', 'rato');
```

Marcas de Benchmark na classe de Profile

Se você quer que os dados do benchmark estejam disponíveis na classe Profiler, suas marcas devem ser deinidas em par, e cada marca precisa ter o sufixo _start e _end. Cada par de marcas precisam ser nomeadas de forma identica. Por exemplo:

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

// Algum codigo aqui...

$this->benchmark->mark('marca_end');

$this->benchmark->mark('outra_marca_start');

// Mais codigo aqui...

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

Para mais informações leia a página de Profiler.

Exibindo o Tempo Total de Execução

Se você pretende exibir o tempo total do moment em que o Code Igniter iniciou até o momento em que a saída é enviada ao navegador, coloque este código no template da sua view:

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

Você pode perceber que esta é a mesma função utilizada nos exemplos acima para calcular o tempo entre duas marcas, exceto pelo fato de que você não passou nenhum parametro. Quando os parametros são enviados, o Code Igniter não para de calcular até o momento que a saída é enviada ao navegador. Não importa onde você chame a função, o tempo continuará a rodar até o final.

Uma maneira alternativa de exibir o tempo decorrido nos seus arquivos de visualização é usando esta pseudo-variável, caso você preferir não usar o PHP:

{elapsed_time}

Nota: Se você quiser verificar qualquer coisa em seus controllers, você precisará definir suas próprias marcas de início e fim.

Exibindo o Consumo de Memória

Se sua instalação do PHP foi configurada com o parametro --enable-memory-limit, você pode exibir a quantidade de memória consumida pelo sistema usando o seguinte código em seus arquivos de visualização:

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

Nota: Esta função só pode ser usada em arquivos de visualização. O consumo será referente ao consumo total de memória usado por toda a aplicação.

Uma maneira alternativa de exibir o tempo decorrido nos seus arquivos de visualização é usando esta pseudo-variável, caso você preferir não usar o PHP:

{memory_usage}

Classe Calendar

A classe Calendário permite a criação dinâmica de calendários. Os calendários podem ser formatados através do uso de um template, lhe dando liberdade total de cada aspecto do design do mesmo. Além disso, você pode passar dados para as células do seu calendário.

Inicializando a Classe

Assim como a maioria das classes no Code Igniter, a classe de Calendário é inicializada em seu controller usando o método $this->load->library():

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

Uma vez carregado, o objeto Calendar estará disponível utilizando: $this->calendar

Exibindo um Calendário

Segue um simples exemplo de como exibir um calendário:

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

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

O código acima irá gerar o calendário para o mês e ano corrente, baseado nas configurações do seu servidor. Para exibir um calendário de uma data específica (ano e mês), você pode passar esta informação através do método de geração do calendário:

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

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

O código acima irá gerar o calendário exibindo o mês de Junho de 2006. O primeiro parâmetro especifica o ano, o segundo especifica o mês.

Passando dados para as Células do Calendário

Para adicionar informações às células do calendário, é necessário criar um array associativo onde as chaves correspondem aos dias que você quer popular e os valores do array contém as informações. O array é passado como terceiro parametro do método de geração do calendário. Veja este exemplo:

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

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

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

No exemplo acima, os dias 3, 7, 13 e 26 receberão links apontando para as URLs passadas.

Nota: Por padrão assume-se que seu array conterá links. Na seção que explica os templates do calendário logo abaixo, você verá como customizar como as informações passadas para suas células serão tratadas, assim você poderá passar diferentes tipos de informações.

Definindo Preferências de Exibição

Existem sete preferências que você pode usar para controlar diversos aspectos do calendário. Preferências são definidas passando um array de preferências como segundo parâmetro do método de carregamento. Veja o exemplo:

```
$prefs = array (
        'start_day'   => 'saturday',
        'month_type'  => 'long',
        'day_type'    => 'short'
       );
$this->load->library('calendar', $prefs);
echo $this->calendar->generate();
```

O código acima iniciaria o calendário no sábado, exibe o nome do mês completo no cabeçalho e os nomes reduzidos dos dias. Mais informações sobre preferências mais abaixo.

Preferências	Padrão	Opções	Descrição
template	None	None	Uma string contendo o template do seu calendário. Veja a seção de templates abaixo.
local_time	time()	None	Unix timestamp correspondente ao horário atual.
start_day	sunday	Qualquer dia da semana (sunday, monday, tuesday, etc.)	Define o dia da semana que o calendário deve iniciar.
month_type	long	long, short	Determina qual versão do nome do mês deve ser exibido no cabeçalho. long = January, short = Jan.
day_type	abr	long, short, abr	Determina qual versão dos nomes dos dias da semana usar no cabeçalho das colunas. long = Sunday, short = Sun, abr = Su.
show_next_prev	FALSE	TRUE/FALSE (booleano)	Determina onde exibis os links permitindo navegar nos meses. Veja mais informações sobre isso abaixo.
next_prev_url	None	A URL	Define o caminho base usado em seus links de próximo/anterior do calendário.

Exibindo Links de Mês Anterior/Próximo

Para permitir que seu calendário possa navegar entre os dias e meses através de links de próximo/anterior, você deve configurar o calendário de forma parecida ao exemplo abaixo:

```
$this->load->library('calendar');
$prefs = array (
        'show_next_prev'  => TRUE,
        'next_prev_url'   => 'http://www.seu-site.com/index.php/calendar/show/'
      );
echo $this->calendar->generate($this->uri->segment(3), $this->uri->segment(4), $prefs);
```

Algumas considerações sobre o exemplo acima:

Você precisa definir "show_next_prev" como TRUE.

Você precisa colocar a URL para o controller que contém o calendário na opção "next_prev_url".

Você precisa passar o "ano" e "mês" para o método de geração do calendário através dos segmentos de URI onde eles aparecem (Nota: A classe Calendário adiciona o ano/mês automaticamente à URL base que você passar.).

Criando Templates de Calendários

Criando templates de calendários você terá controle total sobre o design do mesmo. Cada componente do seu calendário será definido em pares de pseudo-variáveis, conforme explicado abaixo:

```
$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();
```

Classe Config

A classe Config provê meios de recuperar suas preferências de configuração. Estas preferências podem vir do arquivo de configuração padrão (application/config/config.php) ou de seus próprios arquivos de configuração.

Nota: Esta class é inicializada automaticamente pelo sistema e não é necessário carregá-la manualmente.

Anatomia de um Arquivo de Configuração

Por padrão, o Code Igniter tem um arquivo de configuração primário, localizado em application/config/config.php. Se você abrir este arquivo usando seu editor de texto, você verá que os itens de configuração são armazenados em um array chamado $config.

Você pode adicionar seus próprios itens de configuração a este arquivo, ou, se preferir manter suas configurações separadas, simplesmente crie seu próprio arquivo e salve o mesmo no diretório config.

Nota: Se você criar seus próprios arquivos de configuração, use o mesmo formato do primário, armazenando seus itens em um array chamado $config. O Code Igniter irá gerenciar estes arquivos de forma inteligente para que não hajam conflitos até mesmo se o array possuir o mesmo nome (desde que um indice deste array não tenha o mesmo nome que outro).

Carregando um Arquivo de Configuração

Nota: O Code Igniter carrega automaticamente o arquivo primário (application/config/config.php), então você só precisa carregar seus próprios arquivos de configuração.

Existem duas maneiras de se carregar um arquivo de configuração:

Carregamento Manual

Para carregar um de seus próprios arquivos de configuração você usará a seguinte função em seus controllers:

$this->config->load('nomedoarquivo');

Onde nomedoarquivo é o nome do seu arquivo de configuração, sem a extensão .php.

Se você precisa carregar vários arquivos de configuração, eles serão agregados em um único array de configuração. Colisão de nomes podem ocorrer, porém, se você tiver nomeado índices do array identicos em diferentes arquivos. Para evitar colisões, você pode passar o segundo parametro como TRUE e cada arquivo de configuração será armazenado em um índice de array correspondente ao nome do arquivo de

configuração. Veja o exemplo:

// Armazenado em um array com esta estrutura: $this->config['blog_settings'] = $config

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

Veja a seção chamada Recuperando Itens de Configuração abaixo para aprender como recuperar itens de configuração desta forma.

O terceiro parametro permite que você suprima erros caso o arquivo de configuração não exista:

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

Carregamento Automático (Auto-loading)

Se você precisar de uma configuração global em particular, você pode configurar o sistema para carregá-la automaticamente. Para isso, abra o arquivo autoload.php, localizado em application/config/autoload.php, e adicione seu arquivo de configuracao como indicado no arquivo.

Recuperando Itens de Configuração

Para recuperar um item do seu arquivo de configuração, utilize o seguinte método:

$this->config->item('nome do item');

Onde nome do item é o índice do array $config que você deseja recuperar. Por exemplo, para recupera a linguagem escolhida, você usaria::

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

O método retorna FALSE (booleano) se o item que você passou não existir.

Se você está usando o segundo parametro do método $this->config->load() para definir seus itens de configuração em um índice específico, você pode recuperar esta informação especificando o nome do índice como segundo parametro do método $this->config->item(). Por exemplo:

// Carrega o arquivo de configuracao chamado blog_settings.php e define para o índice chamado "blog_settings"

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

// Recupera o item de configuração chamado site_name do array blog_settings

$site_name = $this->config->item('site_name', 'blog_settings');

// Modo alternativo de especificar o mesmo item:

$blog_config = $this->config->item('blog_settings');

$site_name = $blog_config['site_name'];

Definindo um Item de Configuração

Se você quiser definir um item de configuração dinamicamente, ou alterar um já existente, pode usar:

$this->config->set_item('nome_item', 'valor_item');

Onde name_item é o índice do array $config que você quer alterar, e value_item é o seu valor.

Funções Assistentes (Helper)

A classe Config tem os seguintes métodos assistentes::

$this->config->site_url();

Esta função recupera a URL para o seu site, com o valor que você tiver especificado no arquivo de

configuração.

$this->config->system_url();

Esta função retorna a URL para o seu diretório de sistema (system folder).

Classe Email

A classe de Email do Code Igniter suporta as seguintes funcionalidades:

Protocolos Multiplos: Mail, Sendmail e SMTP

Recipientes Multiplos

CC e BCCs

Emails em HTML ou texto puro (plain text)

Arquivos anexos

Quebra de palavras

Prioridades

BCC Batch Mode, permitindo que grandes lotes de email sejam quebradas em pequenos lotes BCC.

Ferramentas de Debug de Email

Enviando Emails

Enviar email não é apenas simples, mas você pode configurar isso em tempo de execução ou definir suas preferências em arquivos de configuração.

Veja uma demonstração básica de como enviar um email. Nota: Este exemplo assume que você está enviando o email de um de seus controllers.

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

$this->email->from('voce@seu-site.com', 'Seu Nome');

$this->email->to('alguem@algum-site.com');

$this->email->cc('outro@outro-site.com');

$this->email->bcc('fulano@qualquer-site.com');

$this->email->subject('Teste de Email');

$this->email->message('Testando a classe de email.');

$this->email->send();

echo $this->email->print_debugger();
```

Definindo as Preferencias de Email

Existem 17 preferências disponíveis para definir como suas mensagens serão enviadas. Você pode defini-las, tanto manualmente como descrito aqui, quanto automaticamente usando seu arquivo de configuração, como descrito abaixo:

Estas preferências são definidas passando um array para a função de inicialização do email. Segue um exemplo de como você poderia definir algumas preferências:

```
$config['protocol'] = 'sendmail';

$config['mailpath'] = '/usr/sbin/sendmail';

$config['charset'] = 'iso-8859-1';

$config['wordwrap'] = TRUE;

$this->email->initialize($config);
```

Nota: A maior parte das preferências possuem valores padrão que serão utilizados caso você não os defina.

Definindo as Preferências de Email em Arquivo de Configuração

Se você preferir não definir as preferências usando o método acima, você pode colocá-las em um arquivo de configuração. Crie um novo arquivo chamado email.php, adicione o array $config neste arquivo. Então, salve no diretório config/email.php da sua aplicação e ele será carregado automaticamente. Você NÃO precisará usar o método $this->email->initialize() se salvar suas preferências no Arquivo de Configuração.

Preferencias de Email

Abaixo segue a lista de todas as preferências que podem ser definidas para o envio de um email.

Preferências	Padrão	Opções	Descrição
useragent	Code Igniter	None	O "user agent".
protocol	mail	mail, sendmail, ou smtp	Protocolo de envio de email.
mailpath	/usr/sbin/ sendmail	None	Caminho do Sendmail no servidor.
smtp_host	No Default	None	Endereço do Servidor SMTP.
smtp_user	No Default	None	Usuário SMTP.
smtp_pass	No Default	None	Senha SMTP.
smtp_port	25	None	Porta SMTP.
smtp_timeout	5	None	Timeout SMTP (em segundos).
wordwrap	TRUE	TRUE ou FALSE	Liga a quebra de palavra.
wrapchars	76		Número de caracteres para quebrar.
mailtype	text	text or html	Tipo de email. Se enviar em HTML, é necessário enviar uma página web completa. Certifique-se de que não utilizou links ou imagens com caminho relativo, pois se não elas não vão funcionar.
charset	utf-8		Tipo de codificação de caracter (utf-8, iso-8859-1, etc.).
validate	FALSE	TRUE ou FALSE	Se deve validar o endereço de email.
priority	3	1, 2, 3, 4, 5	Prioridade do email. 1 = maior. 5 = menor. 3 = normal.
newline	\n	"\r\n" or "\n"	Caracter de quebra de linha. (Use "\r\n" para compatibilizar com o RFC 822).
bcc_batch mode	FALSE	TRUE or FALSE (boolean)	Liga BCC Batch Mode (modo de lote).
bcc_batch_size	200	None	Número de emails em cada lote BCC.

Referências de Funções de Email

$this->email->from()

Define o endereço de email e o nome da pessoa que envia o email:

$this->email->from('voce@seu-site.com', 'Seu Nome');

$this->email->reply_to()

Define o endereço de resposta (reply-to). Se ocultar esta informação, a informação do "from" será utilizada. Exemplo:

$this->email->reply_to('voce@seu-site.com', 'Seu Nome');

$this->email->to()

Define o(s) endereço(s) de e-mail de destino. Pode ser um único email, uma lista separada por vírgulas ou um array:

$this->email->to('alguem@site.com');

$this->email->to('alguem@site.com, outro@outro-site.com, fulano@site.com');

$list = array('alguem@site.com, outro@outro-site.com, fulano@site.com');

$this->email->to($list);

$this->email->cc()

Define o(s) endereço(s) de email para CC. Funciona como o "to", podendo ser um único email, lista separada por vírgulas ou um array.

$this->email->bcc()

Define o(s) endereço(s) de email para BCC. Funciona como o "to", podendo ser um único email, lista separada por vírgulas ou um array.

$this->email->subject()

Define o Assunto do email:

$this->email->subject('Este é o assunto');

$this->email->message()

Define a mensagem do email:

$this->email->message('Esta é a mensagem');

$this->email->set_alt_message()

Define a mensagem alternativa do email:

$this->email->set_alt_message('Esta é a mensagem alternativa');

Este é uma mensagem opcional que pode ser usada se você enviar email em HTML. Ela permite você especificar uma mensagem alternativa sem formatação HTML, que é adicionada ao cabeçalho do email para pessoas que não aceitam emails HTML. Se você não definir esta mensagem, o Code Igniter vai extraí-la do seu email removendo as tags HTML do mesmo.

$this->email->clear()

Reseta todas as variaveis de email. Esta função deve ser usada se você precisar rodar a função de envio de email em um loop, resetando as informações entre os ciclos.

```
foreach ($list as $name => $address)
{
   $this->email->clear();
   $this->email->to($address);
   $this->email->from('voce@seu-site.com');
   $this->email->subject('Seu nome '.$name);
   $this->email->message('Ola '.$name.' Aqui esta a informacao que voce precisa.');
   $this->email->send();
}
```

Se você passar TRUE como parametro, todos os anexos também serão resetados:

$this->email->clear(TRUE);

$this->email->send()

Função de envio de email. Retorna o booleano TRUE ou FALSE baseado no sucesso ou falha, permitindo

```
if ( ! $this->email->send())
{
   // Gerar Erro
}
```

$this->email->attach()

Permite que você envie um anexo. Passe o caminho e nome do arquivo como primeiro parametro. Nota: Use o caminho do arquivo, e não sua URL. Para anexar vários arquivos, use a função várias vezes. Por exemplo:

```
$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 uma string contendo qualquer mensagem de servidor, cabeçalho de email e a própria mensagem. Útil para debug.

Evitando Quebra de Palavra

Se você deixou a quebra de palavras ligado (o que é recomendado para seguir o RFC 822) e você tem um link no corpo do email, ele pode ser quebrado também, fazendo com que o mesmo não seja "clicável" pela

pessoa que recebeu o email. O Code Igniter permite que você evite a quebra de palavra em determinadas partes da sua mensagem, dessa forma:

The text of your email that

gets wrapped normally.

{unwrap}http://www.some-site.com/a_long_link_that_should_not_be_wrapped.html{/unwrap}

More text that will be

wrapped normally.

Coloque o trecho que você não quer quebrar entre: {unwrap} {/unwrap}

Classe Criptografia

A Classe Criptografia provê a criptografia de dados em duas vias. Ela usa um esquema que pré-compila a mensagem usando um processo de codificação binária XOR via um algoritmo de dispersão randômico (randomly hashed bitwise XOR encoding scheme), que é então criptografado usando a biblioteca Mcrypt. Se a Mcrypt não estiver disponível no seu servidor, a mensagem codificada ainda manterá um razoável grau de segurança para sessões criptografadas ou outros propósitos "leves" como esse. Se a Mcrypt estiver disponível, você terá, efetivamente, uma mensagem string bicriptografada, que deverá prover um alto grau de segurança.

Configurando sua Chave

Uma chave é um pedaço de informação que controle o processo de criptografia e permite que uma string criptografada seja decodificada. De fato, a chave que você escolher fornecerá as únicas maneiras de decodificar os dados que foram criptografados com essa chave, por isso não apenas você deve escolhê-la com cuidado, como também nunca a alterar, se sua intenção for usá-la para dados persistentes.

Nem é preciso dizer que você deve guardar sua chave com cuidado. Se alguém obtier acesso a ela, seus dados poderão ser facilmente decodificados. Se seu servidor não estiver sob seu total cntrole, é impossível garantir a segurança da chave, por isso, pense duas vezes antes de usá-la para qualquer coisa que requeira alta segurança, como números de cartão de crédito.

Para obter o máximo do algoritmo de criptografia, sua chave deverá ter o comprimento 32 caracteres (128 bits). A chave deverá ser a mais doida que você conseguir bolar, com números e letras maiúsculas e minúsculas. Sua chave não deve ser uma simples string. Para ser criptograficamente segura, ela precisa ser a mais randômica possível.

Sua chave pode ser armazenada em seu aquivo application/config/config.php, ou você pode projetar seu próprio sistema de armazenamento, e passar dinamicamente a chave quando estiver codificando/decodificando.

Para salvar sua chave para seu arquivo application/config/config.php, abrá-o e configure:

$config['encryption_key'] = "SUA CHAVE";

Comprimento da Menssagem

É importante saber que as mensagens codificadas, geradas pela função de criptografia, são aproximadamente 2.6 vezes mais longas que a mensagem original. Por exemplo, se você criptogravar a string "meu dado super secreto", que tem 21 caracteres, terminará com uma string codificada que terá mais ou menos 55 caracteres (dizemos "mais ou menos" pois o comprimento da string codificada é incrementada em clusters de 64 bits, e por isso não é exatamente linear). Lembre-se desta informação quando selecionar seu mecanismo de armazenamento. Cookies, por exemplo, podem apenas conter 4K de informação.

Inicializando a Classe

Como a maioria das classes no Code Igniter, a classe de Criptografia é inicializada em seu controller usando a função $this->load->library:

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

Uma vez carregada, o objeto estará disponível usando: $this->encrypt

$this->encrypt->encode()

Gerar a criptografia do dado e retorná-o como uma string. Exemplo:

```
$msg = 'Minha mensagem secreta';

$encrypted_string = $this->encrypt->encode($msg);
```

Você pode, opcionalmente, passar sua chave de criptografia via o segundo parâmetro se não desejar usar a que está em seu arquivo de configuração;:

```
$msg = 'Minha mensagem secreta';

$key = 'chave-super-secreta';

$encrypted_string = $this->encrypt->encode($msg, $key);
```

$this->encrypt->decode()

Descriptografa a string codificada. Exemplo:

```
$encrypted_string = 'APANtByIGI1BpVXZTJgcsAG8GZl8pdwwa84';

$textopuro_string = $this->encrypt->decode($encrypted_string);
```

$this->encrypt->set_cipher();

Permite que você configure uma cifra Mcrypt. Por default, é usado MCRYPT_RIJNDAEL_256. Exemplo:

$this->encrypt->set_cipher('MCRYPT_BLOWFISH');

Por favor, visite o php.net para uma lista das cifras disponíveis.

Se você desejar testar manualmente se seu servidor suporta a Mcrypt, você pode usar:

echo (! function_exists('mcrypt_encrypt')) ? 'Nao' : 'Sim';

$this->encrypt->set_mode();

Permite configurar um mod Mcrypt. Por default, é usado MCRYPT_MODE_ECB. Exemplo:

$this->encrypt->set_mode('MCRYPT_MODE_CFB');

Por favor, visite o php.net para uma lista dos modos disponíveis.

$this->encrypt->sha1();

É a função de codificação SHA1. Forneça uma string e ela irá retornar um hash de 160 bit de via única. Nota: SHA1, assim como o MD5, não é decodificável. Exemplo:

$hash = $this->encrypt->sha1('Alguma string');

Várias instalações PHP têm suporte à SHA1 por default. Por isso, se tudo que você precisa fazer é codificar o hash, é mais simples usar a função nativa:

$hash = sha1('Alguma string');

Se seu servidor não suportar o SHA1, pode usar a função fornecida aqui.

Classe Upload de Arquivo

Esta classe permite que o upload de arquivos. Você pode configurar várias preferências, restringindo o tipo e tamanho dos arquivos.

O Processo

Subir um arquivo envolve o processo geral:

Um formulário de upload é exibido, permitndo ao usuário selecionar um arquivo para enviar.

Quando o form é submetido, o arquivo é subido ao destino que você especificar.

Durante o trajeto, o arquivo é validado para ter certeza que é permitido, baseado em suas preferências.

Uma vez enviado o arquivo, ao usuário será exibida uma mensagem de sucesso.

Para demonstrar este processo, abaixo vai um breve tutorial. Logo após, você irá encontrar informações de referência.

Criando o Formulário de Upload

Usando um editor de texto, crie um formulário chamado upload_form.php. Nele, coloque este código e o salve na pasta applications/views/:

```
<html>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/file_uploading.html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:58 GMT -->

<head>

<title>Formulário de Upload</title>

</head>

<body>

<?=$error;?>

<?=form_open_multipart('upload/do_upload'); ?>

<input type="file" name="userfile" size="20" />

<br /><br />

<input type="submit" value="upload" />

</form>

</body>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/file_uploading.html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:58 GMT -->

</html>
```

Você irá notar que estamos usando um assistente de formulário para criar a tag form de abertura. Uploads de arquivo requerem um formulário multiparte, por isso o assistente criará a sintaxe específica para você. Note também que temos uma variável $error. Com ela podemos mostrar mensagens de erro no caso do usuário fazer alguma coisa errada.

A Página de Sucesso

Usando um editor de texto, crie um formulário chamado upload_success.php. Nele, coloque este código e o salve na pasta applications/views/:

```
<html>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/file_uploading.html by
HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:58 GMT -->
<head>
<title>Formulário de Upload</title>
</head>
<body>
<h3>Seu arquivo foi enviado com sucesso!</h3>
<ul>
<?php foreach($upload_data as $item => $value):?>
<li><?=$item;?>: <?=$value;?></li>
<?php endforeach; ?>
</ul>
<p><?=anchor('upload', 'Deseja enviar outro?'); ?></p>
</body>
<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/file_uploading.html by
HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:58 GMT -->
</html>
```

O Controller

Usando um editor de texto, crie um controller chamado upload.php. Nele, coloque este código e o salve em sua pasta 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);
                }
        }
}
?>
```

A Pasta Upload

Você irá precisar de uma pasta de destino para os arquivos enviados. Crie uma pasta na raiz do diretório de instalação do Code Igniter chamada uploads e configure sua permissão de arquivo para 777.

Experimente!

Para testar seu formulário, visite seu site usando uma URL semelhante a esta:

www.seu-site.com/index.php/upload/

Você deverá ver o formulário. Experimente enviar uma imagem (no formato jpg, gif, ou png). Se o caminho em seu controller estiver correto, deverá funcionar.

Guia de Referência

Inicializando a Classe Upload

Como a maioria das classes no Code Igniter, a classe Upload é inicializada em seu controller usando a função $this->load->library:

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

Uma vez carregada, o objeto estará disponível usando: $this->upload

Configurando as Preferências

De forma similar às outras bibliotecas, você controlará o que é permitido enviar através de suas preferências. No controller que você construiu acima, você deve configurar o seguinte:

```
$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);
// Alternativamente você pode configurar as preferências chamando a função initialize. É útil se
você auto-carregar a classe:
$this->upload->initialize($config);
```

As preferências acima devem ser quase auto-explicativas. Abaixo vai uma tabela descrevendo todas as preferências disponíveis.

Preferências

As seguintes preferências estão disponíveis. O valor default indica o que será caso você não especifique nada ali.

Preferências	Valor % nbsp; Default	Opções	Descrição
upload_path	Nenhum	Nenhum	O caminho para o diretório onde o arquivo deve ser colocaod. O diretório deve ter permissão de escrita e o caminho pode ser absoluto ou relativo.
allowed_types	Nenhum	Nenhum	Os mime types correspondentes aos tipos de arquivo que você permite serem enviados. Geralente a extesão do arquivo pode ser usada como mime type. Separe-os com uma barra vertical.
overwrite	FALSE	True/False	Se TRUE e se um arquivo com mesmo nome já existir, ele será sobrescrito. Se FALSE, um número será adicionado ao nome do arquivo para diferenciá-lo.

max_size	0	Nenhum	O tamanho máximo (em kilobytes) que um arquivo pode ter. Coloque zero para ilimitado. Nota: A maioria das instalações PHP têm seu próprio limite, como especificado em seu arquivo php.ini. Geralmente são 2MB (ou 2048 KB).
max_width	0	Nenhum	A largura máxima (em pixels) do arquivo. Coloque zero para ilimitada.
max_height	0	Nenhum	A altura máxima (em pixels) do arquivo. Coloque zero para ilimitada.
encrypt_name	FALSE	True/False	Se TRUE, o nome do arquivo será convertido em uma string randomicamente criptografada. Pode ser útil se você quiser que arquivo seja salvo com um nome initeligível para a pessoa que o está enviando.
remove_spaces	TRUE	True/False	Se TRUE, qualquer espaço no nome do arquivo será convertido em underscores. Esta é recomendada.

Configurando preferências no arquivo config

Caso preferir não configurar as preferências do jeito mostrado acima, pode colocá-las num arquivo de configuração. Simplesmente crie um novo arquivo chamado upload.php e adicione a array $config. Salve então o arquivo em: config/upload.php e ele será usado automaticamente. Você NÃO precisará usar a função $this->upload->initialize se salvar suas preferências neste arquivo de configuração.

Referência para Funcões

As seguintes funções estão disponíveis

$this->upload->do_upload()

Executa o upload baseado nas preferências que você configurou. Nota: Por default, a rotina de upload espera que o arquivo venha a partir de um campo chamado userfile no formulário, e o formulário deve ser do tipo "multiparte":

<form method="post" action="some_action" enctype="multipart/form-data" />

Se você quiser configurar seu próprio nome para o campo, simplesmente passe seu valor para a função do_upload:

```
$field_name = "nome_do_seu_campo";

$this->upload->do_upload($field_name)
```

$this->upload->display_errors()

Captura qualquer mensagem de erro se a função do_upload() retornar FALSE. A função não dá echo automaticamente, ela retorna o dado para que então você faça o que desejar com ele.

Erros de Formatação

Por default, a função acima envolve qualquer erro com tags . Mas você pode configurar seu próprio delimitador assim:

$this->upload->display_errors(' ', ' ');

$this->upload->data()

Esta é uma função assistente que retorna uma array contendo todos os dados relacionados com o arquivo enviado. Aqui está um protótipo dela:

```
Array
(
   [file_name]   => minhaimagem.jpg
   [file_type]   => image/jpeg
   [file_path]   => /caminho/para/seu/arquivo/
   [full_path]   => /caminho/para/seu/arquivo.jpg
   [raw_name]    => minhaimagem
   [orig_name]   => minhaimagem.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"
)
```

Explicação

Esta é a explicação para os itens da array acima.

Item	Descrição
file_name	O nome do arquivo que foi enviado incluidno sua extensão.
file_type	O mime type do arquivo.
file_path	O caminho absoluto do arquivo no servidor.
full_path	O caminho absoluto do arquivo no servidor incluindo seu nome.
raw_name	O nome do arquivo sem sua extensão.
orig_name	O nome original do arquivo. Para o caso de você tê-lo encriptado via preferências.
file_ext	A extensão do arquivo com ponto final.
file_size	O tamanho do arquivo em kilobytes.
is_image	Quando o arquivo for imagem ou não. 1 = imagem. 0 = não.
image_width	Largura da imagem.
image_heigth	Altura da imagem
image_type	Tipo da imagem. Tipicamente o nome de sua extensão sem o ponto final.
image_size_str	Uma string contendo a largura e altura. Útil para colocar nas propriedades da imagem, como numa tag.

Classe FTP

Esta classe permite que arquivos sejam transferidos para um servidor remoto. Estes arquivos remotos, podem também ser movidos, renomeados e apagados. A classe FTP ainda inclui uma função de "espelhamento", que permite que um diretório local seja 100% recriado, remotamente, via FTP.

Nota: Os protocolos SFTP e SSL FTP não são suportados, apenas o FTP padrão.

Inicializando a Classe

Como a maioria das classes no Code Igniter, a classe FTP é inicializada em seu controller usando a função $this->load->library:

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

Uma vez carregada, o objeto estará disponível usando: $this->ftp

Exemplos de Uso

Neste exemplo, uma conexão é aberta com o servidor FTP e um arquivo local é lido e enviado em modo ASCII. As permissões de arquivo são configuradas em 755. Nota: Esta configuração de permissões requer o PHP 5.

```
$this->load->library('ftp');
$config['hostname'] = 'ftp.seu-site.com';
$config['username'] = 'seu-usuario';
$config['password'] = 'sua-senha';
$config['debug'] = TRUE;
$this->ftp->connect($config);
$this->ftp->upload('/local/path/to/meuarquivo.html', '/public_html/meuarquivo.html', 'ascii',
0775);
$this->ftp->close();
```

Neste exemplo, uma lista de arquivos é buscada no servidor.

```
$this->load->library('ftp');
$config['hostname'] = 'ftp.seu-site.com';
$config['username'] = 'seu-usuario';
$config['password'] = 'sua-senha';
$config['debug'] = TRUE;
$this->ftp->connect($config);
$list = $this->ftp->list_files('/public_html/');
print_r($list);
$this->ftp->close();
```

Neste exemplo, um diretório local é espelhado no servidor.

```
$this->load->library('ftp');
$config['hostname'] = 'ftp.seu-site.com';
$config['username'] = 'seu-usuario';
$config['password'] = 'sua-senha';
$config['debug'] = TRUE;
$this->ftp->connect($config);
$this->ftp->mirror('/path/to/myfolder/', '/public_html/meudiretorio/');
$this->ftp->close();
```

Referência da Função

$this->ftp->connect()

Conecte e logue-se em seu servidor FTP. As preferências de conexão são configuradas passando uma array para a função, ou então você pode armazená-las em um arquivo de configuração.

Aqui vai um exemplo monstrando como configurar suas preferências manualmente:

```
$this->load->library('ftp');
$config['hostname'] = 'ftp.seu-site.com';
$config['username'] = 'seu-usuario';
$config['password'] = 'sua-senha';
$config['port']    = 21;
$config['passive']  = FALSE;
$config['debug']    = TRUE;
$this->ftp->connect($config);
```

Configurando as preferências FTP em um arquivo

Se preferir, você pode armazenar suas preferências de FTP num arquivo config. simplesmente, crie um novo arquivo chamado ftp.php, e adicionar a array $config neste arquivo. Salve-o em config/ftp.php e ele será usado automaticamente.

Opções de conexão disponíveis:

hostname - O FTP hostname. Geralmente algo como: ftp.algum-site.com

username - O Usuário FTP.

password - A senha FTP.

port - O número da porta. É configurado como 21 por default.

debug - TRUE/FALSE (boolean). Habilitar ou não o debugging para mostrar mensagens de erro.

passive - TRUE/FALSE (boolean). Usar ou não o modo passivo. Ele é TRUE por default.

$this->ftp->upload()

Envia um arquivo ao seu servidor. Você deve fornecer o caminho local e o remoto, e também pode, opcionalmente, configurar o modo e permissões. Exemplo:

$this->ftp->upload('/local/caminho/para/meuarquivo.html', '/public_html/meuarquivo.html', 'ascii', 0775);

As opções de modo são: ascii, binário, e auto (o default). Se auto for usado, ele irá se basear na extensão do arquivo enviado.

As permissões estão disponíveis se você estiver rodando o PHP 5 e podem ser passadas como um valor octal no quarto parâmetro.

$this->ftp->rename()

Permite renomear um arquivo. Forneça o arquivo e caminho iniciais e os finais.

// Renomeia verde.html para azul.html

$this->ftp->rename('/public_html/teste/verde.html', '/public_html/teste/azul.html');

$this->ftp->move()

Lhe permite mover um arquivo. Forneça o caminho inicial e o final:

// Moverá o blog.html de "joe" para "fred"

$this->ftp->move('/public_html/joe/blog.html', '/public_html/fred/blog.html');

Nota: se o nome do arquivo de destino for diferente do inicial, ele será renomeado.

$this->ftp->delete_file()

Lhe permite apagar um arquivo. Forneça o nome e caminho do arquivo a ser apagado.

$this->ftp->delete_file('/public_html/joe/blog.html');

$this->ftp->delete_dir()

Permite apagar um diretório e todo o seu conteúdo. Forneça o caminho para o diretório seguido de uma barra inclinada.

Importante Tenha MUITO cuidado com esta função. Ela irá apagar recursivamente TUDO dentro do caminho fornecido, incluindo subdiretóris e todos os arquivos. Tenha certeza absoluta que seu caminho está correto. Experimente usar a função list_files() antes, para verificar se seu caminho está correto.

$this->ftp->delete_dir('/public_html/caminho/para/diretorio/');

$this->ftp->list_files()

Permite que você busque uma lista de arquivos no servidor e a retorne como uma array. Você deve fornecer o caminho para o diretório desejado.

$list = $this->ftp->list_files('/public_html/');

print_r($list);

$this->ftp->mirror()

Lê, recursivamente, um diretório local e tudo que ele contém (incluindo subdiretórios) e creia um espelhamento via FTP baseado nisto. Qualquer que seja estrutura de diretório, ela será recriada no servidor. Você deve fornecer o caminho inicial e o final:

$this->ftp->mirror('/caminho/para/meudiretorio/', '/public_html/meudiretorio/');

$this->ftp->mkdir()

Permite criar um diretório em seu servidor. Forneça o caminho final no diretório em quer deseja criar, seguido de uma barra invertida. As permissões podem ser passadas via um valor octal no segundo parâmetro (se você estiver usando o PHP 5).

// Cria um diretório chamado "buteco"

$this->ftp->mkdir('/public_html/buteco/', 0777);

$this->ftp->chmod()

Permite que você configure as permissões de arquivo. Forneça o caminho para o arquivo ou diretório:

// Chmod "buteco" to 777

$this->ftp->chmod('/public_html/buteco/', 0777);

$this->ftp->close();

Fecha a conexão com o servidor. É recomendado que você use isto ao terminar de enviar os arquivos.

Classe Tabela HTML

A Classe Tabela provê funções para que você gere tabelas HTML automaticamente a parti de arrays ou de conjutos de resultados do banco de dados.

Inicializando a Classe

Como a maioria das classes no Code Igniter, a classe Tabela é inicializada em seu controller usando a função $this->load->library:

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

Uma vez carregada, o objeto estará disponível usando: $this->table

Exemplos

Este exemplo mostra como você pode criar uma tabela a partir de uma array multidimensional. Note que o primeiro índice da array se tornará o cabeçalho da tabela (ou você pode configurar seu próprio cabeçalho usando a função set_heading() descrita na referência da função abaixo).

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

$data = array(
        array('Nome', 'Cor', 'Tamanho'),
        array('Fred', 'Azul', 'Pequeno'),
        array('Mary', 'Vermelho', 'Grande'),
        array('John', 'Verde', 'Médio')
        );

echo $this->table->generate($data);
```

Aqui vai um exemplo de uma tabela criada a partir de um resultado de uma query no banco. A classe Tabela irá gerar automaticamnete o cabeçalho baseada no nome das tabelas (lembre-se que você pode confitgurar eu próprio cabeçalho usando a função set_heading() descrita na referência da função abaixo).

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

$query = $this->db->query("SELECT * FROM minha_tabela");

echo $this->table->generate($query);
```

Aqui vai um exemplo mostrando como você poderia criar uma tabela usando parâmetros discretos:

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

$this->table->set_heading('Nome', 'Cor', 'Tamanho');

$this->table->add_row('Fred', 'Azul', 'Pequeno');

$this->table->add_row('Mary', 'Vermelho', 'Grande');

$this->table->add_row('John', 'Verde', 'Médio');

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

Aqui vai o mesmo exemplo acima, exceto que no lugar de parâmetros individuais, foram usadas arrays:

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

$this->table->set_heading(array('Nome', 'Cor', 'Tamanho'));

$this->table->add_row(array('Fred', 'Azul', 'Pequeno'));

$this->table->add_row(array('Mary', 'Vermelho', 'Grande'));

$this->table->add_row(array('John', 'Verde', 'Médio'));

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

Alterando o visual da sua tabela

A classe Tabela permite que você use um template com o qual você pode especificar o layout de sua tabela. Aqui vai um protótipo do template:

```
$tmpl = array (
            'table_open'       => '<table border="0" cellpadding="4" cellspacing="0">',
            'heading_row_start'  => '<tr>',
            'heading_row_end'    => '</tr>',
            'heading_cell_start' => '<th>',
            'heading_cell_end'   => '</th>',
            'row_start'        => '<tr>',
            'row_end'          => '</tr>',
            'cell_start'       => '<td>',
            'cell_end'         => '</td>',
            'row_alt_start'     => '<tr>',
            'row_alt_end'       => '</tr>',
            'cell_alt_start'    => '<td>',
            'cell_alt_end'      => '</td>',
            'table_close'      => '</table>'
          );
$this->table->set_template($tmpl);
```

Nota: Você irá notar que há 2 conjuntos de blocos de "linha" (rows) no template. Isto lhe permite criar linhas com cores alternadas ou projetar elementos que se alteram em cada interação ocm os dados da linha.

Você NÃO é obrigado a submeter um template completo. Se apenas precisa alterar partes do layout, pode simplismente submeter estes elementos. Neste exemplo, apenas a tag de abertura da tabela foi alterada:

```
$tmpl = array ( 'table_open'  => '<table border="1" cellpadding="2" cellspacing="1"
class="minhatabela">' );

$this->table->set_template($tmpl);
```

Referência da Função

$this->table->generate()

Retorna uma string contendo a tabela gerada. Aceita parâmetros opcionais os quais podem ser uma array ou um objeto vindo do banco.

$this->table->set_caption()

Lhe permite adicinar uma legenda para a tabela.

$this->table->set_caption('Cores');

$this->table->set_heading()

Lhe permite configurar o cabeçalho da tabela. Você pode submeter uma array ou parâmetros discretos:

$this->table->set_heading('Nome', 'Cor', 'Tamanho');

$this->table->set_heading(array('Nome', 'Cor', 'Tamanho'));

$this->table->add_row()

Lhe permite adicionar uma linha à tabela. Você pode submeter uma array ou parâmetros discretos:

$this->table->add_row('Azul', 'Vermelho', 'Verde');

$this->table->add_row(array('Azul', 'Vermelho', 'Verde'));

$this->table->make_columns()

Esta função pega uma array unidimensional como entrada e cria uma multidimensional com uma profundidade (depth) igual ao número de colunas desejadas. Isto permite que uma única array, com vários elementos, seja mostrada numa tabela que tem um número fixo de colunas. Considere este exemplo:

```
$lista = array('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten', 'eleven', 'twelve');

$nova_lista = $this->table->make_columns($lista, 3);

$this->table->generate($nova_lista)

// Gera uma tabela com este protótipo

<table border="0" cellpadding="4" cellspacing="0">

<tr>

<td>one</td><td>two</td><td>three</td>

</tr><tr>
```

```
<td>four</td><td>five</td><td>six</td>
</tr><tr>
<td>seven</td><td>eight</td><td>nine</td>
</tr><tr>
<td>ten</td><td>eleven</td><td>twelve</td></tr>
</table>
```

$this->table->set_template()

Lhe permite configurar seu template. Você pode submeter um template completo ou parcial.

```
$tmpl = array ( 'table_open'  => '<table border="1" cellpadding="2" cellspacing="1"
class="minhatabela">' );
$this->table->set_template($tmpl);
```

$this->table->set_empty()

Lhe permite configurar um valor default para usar em qualquer célula de tabela que é vaza. Você pode, por exemplo, usar :

$this->table->set_empty(" ");

$this->table->clear()

Lhe permite limpar o cabeçalho ou dado de uma linha da tabela. Se precisar mostrar múltiplas tabelas com diferentes dados, você deve chamar esta função depois que cada tabela for gerada, para limpar a informação da tabela anterior. Exemplo:

```
$this->load->library('table');
$this->table->set_heading('Nome', 'Cor', 'Tamanho');
$this->table->add_row('Fred', 'Azul', 'Pequeno');
$this->table->add_row('Mary', 'Vermelho', 'Grande');
$this->table->add_row('John', 'Verde', 'Médio');
echo $this->table->generate();
$this->table->clear();
$this->table->set_heading('Nome', 'Dia', 'Tipo de Entrega');
$this->table->add_row('Fred', 'Quarta-feira', 'Expressa');
$this->table->add_row('Mary', 'Segunda-feira', 'Por avião');
$this->table->add_row('John', 'Sábado', 'De noite para o dia');
echo $this->table->generate();
```

Classe Manipulação de Imagem

Esta classe do Code Igniter lhe permite executar as seguintes ações:

Redimensionar uma imagem

Criar uma miniatura da imagem

Cortar uma imagem

Rotacionar uma imagem

Aplicar uma imagem como marca-d'água

Todas essas principais bibliotecas de imagens são suportadas: GD/GD2, NetPBM, e ImageMagick

Nota: A Marca-d'água está apenas disponível na biblioteca GD/GD2. Além disto, mesmo que as outras bibliotecas sejam suportadas, a GD é exigida para que o script calcule as propriedades da imagem. O processamento da imagem, no entando, será executado pela biblioteca que você especificar.

Inicializando a Classe

Como a maioria das classes no Code Igniter, a classe Manipulação de Imagem é inicializada em seu controller usando a função $this->load_library:

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

Uma vez carregada, estará pronta para usar. O objeto desta biblioteca que você usará para chamar todas as funções é: $this->image_lib

Processando uma Imagem

Não importa qual o tipo de processamento você queria executar (redimensionar, cortar, rotacionar ou aplicar marca-d'água), o processo geral é idêntico. Você configurará algumas preferências correspndentes à ação desejada, e então chamará uma das quatro funções disponíveis. Por exemplo, para criar uma miniatura da imagem, use isto:

```
$config['image_library'] = 'GD';

$config['source_image'] = '/caminho/para/imagens/minha_imagem.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();
```

O código acima diz à função image_resize procurar uma imagem chamada minha_imagem.jpg localizada no diretório especificado, e então criar uma miniatur que terá 75x50 pixels, usando a bilioteca image_ library GD2. Uma vez que a opção maintain_ratio está habilitada, a miniatura será terá as mais próximas largura e altura possíveis enquanto preservará a proporção original. A miniatura será chamda minha_ imagem_thumb.jpg

Note: Para que a classe tenha permissão para executar qualquer processo, o diretório contendo os arquivos das imagens deve ter permissão 777.

Funções de Processamento

Há quatro funções de processamento disponíveis:

$this->image_lib->resize()

$this->image_lib->crop()

$this->image_lib->rotate()

$this->image_lib->watermark()

Essas funções retornam TRUE booleano se tudo der certo e FALSE se algo falhar. Se houver falha, você pode recuperar a mensagem de erro usando esta função:

echo $this->image_lib->display_errors();

Uma boa prática é usar a função de processamento condicionalmente, mostrando um erro se houver falha, como desta maneira:

```
if ( ! $this->image_lib->resize())

{

   echo $this->image_lib->display_errors();

}
```

Nota: Você pode opcionalmente especificar qual formatação do HTML será aplicada aos erros, submentendo as tags de abertura/fechamento na função, como desta maneira:

$this->image_lib->display_errors(' ', ' ');

Preferências

As 14 preferências disponíveis descriatas abaixo permitem que você ajuste o processamento de imagens às suas necessidades.

Note que nem todas as preferências estão disponíveis para todas as funções. Por exemplo, as preferências dos eixos x/y estão apenas disponíveis para cortar imagens. Da mesma forma, as preferências de largura e altura não tem efeito quando estiver cortando uma imagem. A coluna "disponibilidade" indica quais funções suportam uma dada preferência.

Legenda de Disponibilidade:

R - Redimensionar a Imagem

C - Cortar a Imagem

X - Rotacionar a Imagem

W - Aplicar marca-d'água (W = watermark)

Preferência	Valor default	Opções	Descrição	Disponibilidade
image_library	GD2	GD, GD2, ImageMagick, NetPBM	Configurar qual biblioteca deve ser usada.	R, C, X, W
library_path	None	None	Configura o caminho, no servidor, para as bibliotecas ImageMagick ou NetPBM. Se usar uma dessas bibliotecas você deve fornecer o caminho.	R, C, X
source_image	None	None	Configura o caminho/nome da imagem original. O caminho do servidor deve ser absoluto ou relatico e não uma URL.	R, C, S, W
dynamic_ output	FALSE	TRUE/FALSE (boolean)	Determina quando uma nova imagem deve ser escrita no disco ou gerada dinamicamente. Nota: Se você escolher a opção dinâmica, apenas uma imagem pode ser mostrada de cada vez, e ela não pode ser posicionada na página. Esta preferência simplismente dá saída automaticamente da imagem bruta (raw) no seu browser, junto com os cabeçalhos da imagem.	R, C, X, W
quality	90%	1 - 100%	Configurar a qualidade da imagem. Quanto maior a qualidade, maior será o arquivo.	R, C, X, W
new_image	Nenhum	Nenhum	Configura o nome/caminho de destino da imagem. Você usará esta preferência quando criar uma cópia de uma imagem. O caminho do servidor deve ser absoluto ou relativo e não uma URL.	R
width	Nenhum	Nenhum	Configura a largura da imagem final.	R
height	Nenhum	Nenhum	Cria uma miniatura da imagem.	R
create_thumb	FALSE	TRUE/FALSE (boolean)	Especifica o sufixo da miniatura. Ele será inserido antes da extensão do aquivo, portanto, minhaimagem.jpg virará minhaimagem_thumb.jpg	R
thumb_ marker	_thumb	Nenhum	Especifica se deve manter a proporção da imagem original ou então usar valores forçados.	R
maintain_ratio	TRUE	TRUE/FALSE (boolean)	Especifica o que usar como eixo mestre ao redimensionar uma imagem ou criar miniaturas. Por exemplo, digamos que você quer redimensionar uma imagem para 100 x 75 pixels. Se a imagem original não permitir o redimensionamento ideal para essas dimensões, esta preferência determina qual eixo deverá ser usado como valor forçado. "auto" escolhe o eixo automaticamente baseado na condição se a imagem é mais alta do que larga, ou vice-versa.	R
master_dim	auto	auto, width, height	Especifica o ângulo de rotação ao rotacionar imagens. Note que o PHP rotcionar em sentido anti-horário, portanto uma rotação de 90 graus para a direita deve ser especificada como 270 graus.	R
rotation_ angle	Nenhum	90, 180, 270, vrt, hor	Configura a coordenada X, em pixels, para o corte da imagem. Por exemplo, um valor igual a 30 irá cortar a imagem 30 pixels vindo da esquerda.	X
x_axis	Nenhum	Nenhum	Configura a coordenada X, em pixels, para o corte da imagem. Por exemplo, um valor igual a 30 irá cortar a imagem 30 pixels vindo da esquerda.	C
y_axis	Nenhum	Nenhum	Configura a coordenada Y, em pixels, para o corte da imagem. Por exemplo, um valor igual a 30 irá cortar a imagem 30 pixels vindo de cima.	C

Configurando as preferências em um arquivo

Se desejar não configurar as preferências do jeito mostrado acima, pode colocá=las num arquivo de configuração. simplesmente crie um arquivo chamado image_lib.php, e adicionar a array $config a ele. Salve-o em: config/image_lib.php e ele será usado automaticamente. Você NÃO precisará usar a função $this->image_lib->initialize se salvar suas preferências em um arquivo.

$this->image_lib->resize()

A função de redimensionamento de imagnes lhe permite alterar o tamanho da imagem original, criar uma cópia (com ou sem redimensionamento) ou criar uma miniatura.

Na prática, não há diferença entre criar uma cópia ou criar uma miniatura, exceto pelo fato que a minatura irá conter o sufixo em seu nome (ex.:, minhaimagem_thumb.jpg).

Todas as preferências listas na tabela acima estão disponíveis para essa função, exceto estas três: rotation, x_axis, and y_axis.

Criando uma Miniatura

A função resizing irá um arquivo da miniatura (e preservar a imagem original) se você configurar as preferências para TRUE:

$config['create_thumb'] = TRUE;

Esta preferência determina quando uma miniatura é criada ou não.

Criando uma Cópia

A função resizing, irá criar uma cópia do arquivo ogirinal (e irá preservá-lo) se você configurar o caminho e/ou um novo nome de arquivo usando esta preferência:

$config['new_image'] = '/caminho/para/nova_imagem.jpg';

Notas acerca desta preferência:

Se apenas o nome da nova imagem é especificado, ela será colocada no mesmo diretório que a imagem original

Se apenas o caminho for especificado, a nova imagem será colocada no diretório de destino com o mesmo nome da imagem original.

Se ambos caminho e nome da imagem foram especificados, ela irá ser colocada em seu próprio diretório de destino e lhe será dado o novo nome.

Redimensionando a Imagem Original

Se nenhuma das duas preferências listadas acima (create_thumb, e new_image) for usada, a função resizing irá se voltar à imagem original no processamento.

$this->image_lib->crop()

A função cropping trabalhar praticamente de forma idêntica à redising, exceto por requerer que você configure as preferências para os eixos X e Y (em pixels), especificando onde cortar, como desta maneira:

```
$config['x_axis'] = '100';

$config['x_axis'] = '40';
```

Todas as preferências listas na tabela acima estão disponíveis para esta função, exceto estas: rotation, width, height, create_thumb, new_image.

Aqui vai um exemplo mostrando como você poderia cortar uma imagem:

```
$config['image_library'] = 'imagemagick';
$config['library_path'] = '/usr/X11R6/bin/';
$config['source_image'] = '/caminho/para/minhaimagem.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: Sem uma interface visual, é difícil cortar imagens, por isto esta função não é muito útil a menos que você queria contruir tal interface. E foi exatamente isto que fizemos para o módulo galeria de fotos do ExpressionEngine, o CMS que desenvolvemos. Adicionamos uma Javascript UI que permite que as áreas de corte sejam selecionadas.

$this->image_lib->rotate()

A função de rotação requer que o ângulo de rotação seja configurado via sua preferência:

$config['rotation_angle'] = '90';

Há 5 opções de rotação:

90 - rotaciona 90 graus em sentido anti-horário.

180 - rotaciona 180 graus em sentido anti-horário.

270 - rotaciona 270 graus em sentido anti-horário.

hor - gira a imagem horizontalmente.

vrt - gira a imagem verticalmente.

Aqui vai um exemplo mostrando como você poderia rotacionar uma imagem:

```
$config['image_library'] = 'netpbm';
$config['library_path'] = '/usr/bin/';
$config['source_image'] = '/caminho/para/minhaimagem.jpg';
$config['rotation_angle'] = 'hor';
$this->image_lib->initialize($config);
if ( ! $this->image_lib->rotate())
{
    echo $this->image_lib->display_errors();
}
```

Marca-d'água

A funcionalidade de Marca-d'água requer a biblioteca GD/GD2.

Dois tipos de Marca-d'água

Há dois tipos de marca-d'água que você pode usar:

Texto: A mensagem de marca-d'água será gerada usando texto, usando tanto uma fonte True Type que você especificar ou então usando a saída nativa de texto que a biblioteca GD suporta. Se você usar a versão True Type, sua instalação da GD deve ser estar compilada como suporte a tais fontes (é mais raro, mas há uma chance).

Overlay: A marca-d'água será gerada sobrepondo uma imagem (geralmente um PNG ou GIF transparente) contendo a informação desejada sobre a imagem original.

Colocando uma marca-d'água numa imagem

Assim como as outras funções (resizing, cropping, e rotating) o processo geral para aplicar a marca-d'água envolve configurar as preferências correspondentes à ação que se deseja executar, e então chamar a função watermark. Aqui vai um exemplo:

```
$config['source_image'] = 'caminho/para/minhaimagem.jpg';

$config['wm_text'] = 'Copyright 2006 - Fulano de Tal';

$config['wm_type'] = 'text';

$config['wm_font_path'] = './system/fonts/texb.ttf';

$config['wm_font_size'] = '16';

$config['wm_text_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();
```

O exemplo acima usa uma fonte True Type em corpo 16 para criar o texto "Copyright 2006 - Fulano de Tal". A marca-d'água será posicionada centralizada no rodapé da imagem, 20 pixels acima da borda inferior.

Nota: Para que a classe tenha permissão para executar qualquer coisa, o arquivo da imagem deve ter permissão 777.

Preferências da Marca-d'água

Esta tabela mostra as preferências disponíveis para ambos os tipos de marca-d'água (texto ou overlay)

Preferências	Valor Default	Opções	Descrição
wm_type	text	type, overlay	Configura o tipo de marca-d'água a usar.
source_image	Nenhum	Nenhum	Configura o nome/caminho da imagem original. O caminho no servidor deve ser absoluto ou relativo e não uma URL.
dynamic_output	FALSE	TRUE/FALSE (boolean)	Determina quando uma nova imagem deve ser escrita no disco ou gerada dinamicamente. Nota: Se você escolher a opção dinâmica, apenas uma imagem pode ser mostrada de cada vez, e ela não pode ser posicionada na página. Esta preferência simplesmente dá saída automaticamente da imagem bruta (raw) no seu browser, junto com os cabeçalhos da imagem.
quality	90%	1 - 100%	Configurar a qualidade da imagem. Quanto maior a qualidade, maior será o arquivo.
padding	Nenhum	Um número	O montante de padding, em pixels, que será aplicado à marca-d'água para distanciá-la da borda da imagem.
wm_vrt_ alignment	bottom	top, middle, bottom	Configura o alinhamento vertical para a imagem de marca-d'água.
wm_hor_ alignment	center	left, center, right	Configura o alinhamento horizontal para a imagem de marca-d'água.
wm_vrt_offset	Nenhum	Nenhum	Você pode especificar um offset vertical (em pixels) para a posiçào da marca-d'água. O offset normalmente move a marca-d'água para a direita, exceto se vc tiver configurado o alinhamento como "right". Dessa forma o valor de offset moverá a marca-d'água na direção esquerda da imagem.
wm_hor_offset	Nenhum	Nenhum	Você pode especificar um offset horizontal (em pixels) para a posiçào da marca-d'água. O offset normalmente move a marca-d'água para baixo, exceto se vc tiver configurado o alinhamento como "bottom". Dessa forma o valor de offset moverá a marca-d'água para cima.

Preferências de Texto

Esta tabela mostra das preferências de texto disponíveis para o tipo de marca-d'água usando texto.

Preferências	Valor Default	Opções	Descrição
wm_text	Nenhum	Nenhum	O texto que você quer usar como marca-d'água. Tipicamente uma frase de copyright.
wm_font_path	Nenhum	Nenhum	O caminho no servidor para a fonte True Type que você gostaria de usar. O Code Igniter inclui uma fonte no diretório system/fonts. Se você não quiser usar esta opçào, a fonte nativa da GD será usada no lugar.
wm_font_size	16	Nenhum	O corpo do texto. Nota: Se você não estiver usando a opção da fonte True Type acima, o número deve estar na escala de 1 a 5. Do contrário, você pode usar qualquer valor válido em pixels para a fonte.
wm_font_ color	ffffff	Nenhum	A cor da fonte, especificada em hexa decimal. Note que você deve usar os 6 caracteres (ex.:, 993300), e não a versão abreviada de 3.
wm_shadow_ color	Nenhum	Nenhum	A cor da sombra, espeficida em hexa decimal. Se deixa este valor vazio, a sombra não será usada. Note que você deve usar os 6 caracteres (ex.:, 993300), e não a versão abreviada de 3.
wm_shadow_ distance	3	Nenhum	A distância (em pixels) da distância entre a fonte e a sombra.

Preferências de Overlay

Esta tabela mostra das preferências disponíveis para o tipo de marca-d'água usando overlay.

Preferências	Valor Default	Opções	Descrição
wm_overlay_path	Nenhum	Nenhum	O caminho no servidor da imagem que você deseja usar como marca-d'água. Obrigatória apenas para o método overlay.
wm_opacity	50	1 - 100	Opacidade (transparência) da imagem usada como marca-d'água. Permite que esta fique translúcida, mostrando os detalhes da imagem onde está aplicada. A opacidade de 50% é típica.
wm_x_transp	4	Um número	Se a sua imagem de marca-d'ágia for PNG ou GIF, você pode especificar uma cor na imagem que deverá ser transparente. Esta preferência (junto com a próxima) lhe permitirá especificar tal cor. Isto funciona especificando as coordenadas "X" e "Y" (medidas a partir do canto superior esquerdo) dentro da imagem que corresponde àquela cor que você quer tornar transparente.
wm_y_transp	4	Um número	Junto com a configuração anterior, permite que você especifique a coordenada para a representação em pixel daquela cor que quer transparente.

Classe Input

A classe Input serve dois propósitos:

É pré-processos globais de dados de entrada para a segurança.

Ele fornece algumas funções auxiliares para a coleta de dados de entrada e de pré-processamento-lo.

Nota: Esta categoria é inicializado automaticamente pelo sistema então não há necessidade de fazê-lo manualmente.

Filtragem de Segurança

A filtragem de segurança função é chamada automaticamente quando um novo controlador é invocado. Ele faz o seguinte:

Destrói a nível mundial GET array. Desde Código Igniter não utilizar GET cordas, não há motivo para deixá-la.

Destrói todas as variáveis globais em caso register_globals está ligada.

Filtros o POST / COOKIE array chaves, permitindo apenas alfa-numéricas (e alguns outros) caracteres.

Fornece XSS (Cross-site Scripting Hacks) filtragem. Isto pode ser ativado globalmente, ou mediante solicitação.

Standardizes newline caracteres para \ n

XSS Filtragem

Código Igniter vem com um Cross Site Scripting Hack prevenção filtro que pode ser executado automaticamente para filtrar todos os POST e COOKIE dados que são encontradas, ou você pode executá-lo em uma base por item. Por omissão não contrariem a nível mundial, uma vez que requer um pouco de processamento overhead, e uma vez que talvez você não precisa dele em todos os casos.

O filtro XSS olha para comumente utilizadas técnicas para acionar o Javascript ou outro tipo de código que a tentativa de roubar cookies maliciosos ou fazer outras coisas. Se alguma coisa está impedido encontrou-se tornado seguro convertendo os dados para entidades personagem.

Nota: Esta função só deve ser utilizado para lidar com os dados, mediante a apresentação. Não é algo que deve ser utilizado para processamento geral tempo de execução, uma vez que exige um número razoável de processamento de peso.

Para filtrar dados através do filtro XSS utilizar esta função:

$this->input->xss_clean() $ this-> input-> xss_clean ()

Aqui está um exemplo de utilização:

$data = $this->input->xss_clean($data);

$ dados = $ this-> input-> xss_clean ($ dados);

Se você quiser que o filtro a ser executado automaticamente cada vez que se depara POST ou COOKIE dados você pode habilitá-lo a abrir seu aplicativo / config / config.php neste arquivo e configuração:

$ config ['global_xss_filtering'] = true;

Nota: Se você utilizar o formulário de validação classe, dando-lhe a opção de filtragem XSS também.

Usando POST, COOKIE, ou servidor de dados

Código Igniter vem com três funções auxiliares que permitem que você busque POST, COOKIE ou servidor itens. A principal vantagem de utilizar as funções bastante desde então ir buscar um item directamente ($ _POST ['algo']) é que as funções irá verificar se o item é colocado e retornar falso (boolean) se não for.

Isso permite que você use dados convenientemente sem ter que testar se existe um item em primeiro lugar.

Em outras palavras, normalmente você pode fazer algo como isto:

```
if ( ! isset($_POST['something']))
{
   $something = FALSE;
}
else
{
   $something = $_POST['something'];
}
```

Com o Código da Igniter construído em funções que você pode simplesmente fazer isso:

$something = $this->input->post('something');

As três funções são as seguintes:

$this->input->post()

$this->input->cookie()

$this->input->server()

$this->input->post()

O primeiro parâmetro irá conter o nome do POST item que você está procurando:

$ this-> input-> post ('some_data');

A função retorna falso (boolean) se o item que você está tentando recuperar não existe.

O segundo parâmetro opcional permite que você execute os dados através do filtro XSS. Ele é ativado através da definição de parâmetros para a segunda boolean true;

$ this-> input-> post ('some_data', TRUE);

$ this-> input-> cookie ()

Esta função é idêntica à função post, só se recupera dados cookie:

$ this-> input-> cookie ('some_data', TRUE);

$ this-> input-> servidor ()

Esta função é idêntica à que anteriormente funções, apenas se recupera dados servidor:

$ this-> input-> servidor ('some_data');

$ this-> input-> ip_address ()

Retorna o endereço IP do usuário atual. Se o endereço IP não é válido, a função irá retornar de uma IP: 0.0.0.0

echo $ this-> input-> ip_address ();

$ this-> input-> valid_ip ($ ip)

Toma um endereço IP como entrada e retorna true ou false (boolean) se é válida ou não. Nota: O $ this-> input-> ip_address () acima valida a função IP automaticamente.

```
if ( ! valid_ip($ip))
{
    echo 'Not Valid';
}
else
{
    echo 'Valid';
}
```

$this->input->user_agent()

Retorna o usuário agent (navegador) a ser utilizado pelo usuário atual. Retorna falso se ele não está disponível.

echo $this->input->user_agent();

Classe Loader

Carregador, como o nome sugere, é usado para carregar elementos. Estes elementos podem ser bibliotecas (classes) Ver arquivos, Ajudantes, Plugins, ou seus próprios arquivos.

Nota: Esta categoria é inicializado automaticamente pelo sistema então não há necessidade de fazê-lo manualmente.

As seguintes funções estão disponíveis nesta categoria:

$ this-> load-> biblioteca ('class_name')

Esta função é usada para carregar o núcleo classes. Quando class_name é o nome da classe que você deseja carregar. Observação: Nós usamos a expressão "classe" e "biblioteca" intercambiavelmente.

Por exemplo, se você gostaria de enviar um email com o Código Igniter, o primeiro passo é a carga da classe dentro de sua controladora e-mail:

$ this-> load-> biblioteca ("email");

Uma vez carregado, a biblioteca estará pronta para uso, utilizando $ this-> email-> some_function (). Cada biblioteca é descrito em detalhes em sua própria página, então, por favor leia as informações sobre cada um que você gostaria de usar.

Parâmetros podem ser passadas para a biblioteca através de um array no segundo parâmetro.

$ this-> load-> vista ('file_name', $ data, true/false)

Esta função é usada para carregar o seu Exibir arquivos. Se você ainda não leu o Exibições seção do manual do utilizador é recomendado que você faça uma vez que ele mostra-lhe como é que esta função é normalmente usada.

O primeiro parâmetro é necessário. Trata-se de ver o nome do arquivo que você gostaria de carregar.

Nota: A. Php extensão não precisa de ser especificada, a menos que você use outra coisa então. Php.

O segundo parâmetro opcional pode ter um array associativo ou um objeto como entrada, o que ele pode ser executado através do PHP extrato função de converter a variáveis que podem ser utilizados na sua opinião arquivos. Mais uma vez, ler o Exibições página para saber como este pode ser útil.

O terceiro parâmetro opcional permite-lhe alterar o comportamento da função para que ele retorna dados como uma string ao invés de enviá-lo para o seu navegador. Isto pode ser útil se você quiser tratar os dados de alguma forma. Se você definir o parâmetro para true (boolean) que irá retornar dados. O comportamento padrão é falso, que envia a seu navegador. Lembre-se de associá-lo a uma variável se você wan os dados exibidos:

$ string = $ this-> load-> view ("myfile ","", true);

$ this-> load-> database ("options", true/false)

Essa função permite-lhe carregar o banco de dados classe. Os dois parâmetros são opcionais. Por favor, consulte a base de dados seção para obter mais informações.

$this->load->scaffolding('table_name')

Essa função permite ativar andaimes. Por favor, consulte a seção andaimes para mais info.

$ this-> load-> vars ($ array)

Esta função tem um array associativo como entrada e gera variáveis usando o PHP extrato função. Esta função produz os mesmos resultados que a segunda usando o parâmetro de US $ this-> load-> vista

() função acima. A razão pela qual você pode querer usar esta função é independente se você gostaria de definir algumas variáveis globais do construtor de seu controlador e vê-las se tornar disponível em qualquer vista arquivo carregado a partir de qualquer função. Você pode ter várias chamadas para esta função. Os dados em cache e obter fundidos em um array de conversão para variáveis.

$this->load->helper('file_name')

Esta função cargas ajudante de ficheiros, onde file_name é o nome do arquivo, sem o _helper.php extensão.

$ this-> load-> plugin ('file_name')

Essa função carrega plugins arquivos, onde file_name é o nome do arquivo, sem o _plugin.php extensão.

$ this-> load-> file ('filepath / filename', true/false)

Este é um genérico arquivo carregando função. Filepath a oferta e, em nome do primeiro parâmetro, ele irá abrir e ler o arquivo. Por defeito os dados são enviados para o navegador, apenas como um arquivo View, mas se você definir o segundo parâmetro para true (boolean) que irá retornar os dados em vez como uma string.

$ this-> load-> lang ('file_name')

Esta função é um apelido da língua carregamento função: $ this-> lang-> load ()

$ this-> load-> config ('file_name')

Esta função é um apelido de carregar o arquivo de configuração função: $ this-> config-> load ()

Classe Idioma

Cursos Idioma oferece funções para recuperar arquivos linguagem e as linhas de texto para fins de internacionalização.

Em seu Código Igniter sistema, você encontrará uma pasta chamada idioma contendo conjuntos de idiomas arquivos. Você pode criar sua própria língua arquivos que forem necessárias, a fim de exibir mensagens de erro e de outras línguas.

Língua arquivos são tipicamente armazenados em seu sistema / linguagem diretório. Alternativamente você pode criar uma pasta chamada linguagem dentro do seu pedido pasta e armazená-los ali. Código Igniter irá analisar em primeiro lugar em seu sistema / aplicação / linguagem diretório. Se o directório não existir ou o idioma especificado não é aí localizadas em vez CI irá procurar em seu sistema global / linguagem pasta.

Nota: Cada língua deve ser armazenado em sua própria pasta. Por exemplo, os arquivos estão localizados em Inglês: sistema / idioma / Inglês

Criando arquivos de Idioma

Arquivos de Idioma devem ser nomeados com _lang.php como a extensão do arquivo. Por exemplo, digamos que você deseja criar um arquivo contendo mensagens de erro. Você pode nome este: error_ lang.php

Dentro do arquivo que você irá atribuir cada linha de texto para um array chamado $ lang com este protótipo:

$ lang ['language_key'] = "A verdadeira mensagem a ser mostrada";

*Nota: É uma boa prática de usar um prefixo comum para todas as mensagens em um determinado arquivo para evitar colisões com similarmente chamado itens em outros arquivos. Por exemplo, se você está criando mensagens de erro que você pode lhes com prefixo Erro_*

```
$ lang [ 'error_email_missing'] = "Você deve enviar um e-mail";
$ lang [ 'error_url_missing'] = "Você deve apresentar uma URL";
$ lang [ 'error_username_missing'] = "Você tem de apresentar um nome de usuário";
```

Carregando uma linguagem arquivo

A fim de buscar uma linha a partir de um determinado arquivo que você deve carregar o arquivo primeiro. Carregando uma linguagem arquivo é feito com o seguinte código:

$ this-> lang-> load ('filename', 'linguagem');

Sempre que filename é o nome do arquivo que você deseja carregar (sem a extensão do arquivo), e que a língua é a língua que o contenham conjunto (isto é, Inglês). Se o segundo parâmetro está faltando, o idioma padrão estabelecido na sua candidatura / config / config.php arquivo será usado.

Buscando uma linha de texto

Uma vez que o seu idioma desejado arquivo é carregado você pode acessar qualquer linha de texto usando esta função:

$ this-> lang-> line ('language_key');

Onde está o arranjo language_key chave correspondente à linha que você deseja mostrar.

Nota: Esta função simplesmente retorna a linha. Ela não eco-lo para você.

Classe Output

A output é uma classe com uma pequena classe principal função: Para enviar a página da web finalizado para o requerente navegador. É igualmente responsável pela caching suas páginas da web, se você usar esse recurso.

Nota: Esta categoria é inicializado automaticamente pelo sistema então não há necessidade de fazê-lo manualmente.

Em circunstâncias normais, mesmo que você não vai notar a classe de output, uma vez que funciona transparente, sem a sua intervenção. Por exemplo, quando você usar a classe Carregador para carregar um ficheiro opinião, ele é automaticamente passado para a classe de output, que será chamado automaticamente pelo Código Igniter no fim do sistema de execução. É possível, no entanto, para que você possa intervir manualmente com a produção se for necessário, usando um dos dois seguintes funções:

$ this-> output-> set_output ();

Permite-lhe definir manualmente a output final da string. Uso exemplo:

$ this-> output-> set_output ($ dados);

Importante: Se você não definir sua output manualmente, deve ser a última coisa a fazer em função de você chamá-lo. Por exemplo, se você criar uma página em um controlador de suas funções, não se ajustar a saída até o fim.

$ this-> output-> get_output ();

Permite-lhe recuperar manualmente a qualquer saída que tenha sido enviado para armazenagem na saída classe. Uso exemplo:

$ string = $ this-> output-> get_output ();

Note que os dados só serão recuperadas a partir desta função, se tiver sido previamente enviado para a saída por uma turma do Código Igniter funções como $ this-> load-> view ().

$ this-> output-> set_header ();

Permite-lhe definir manualmente cabeçalhos servidor, que a saída classe irá enviar para você quando a saída definitiva proferida visor. Exemplo:

```
$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 ();

Permite-lhe activar ou desactivar o Profiler, que irá exibir benchmark e outros dados, na parte inferior de suas páginas para depuração e otimização fins.

Para ativar o profiler lugar as seguintes funções em qualquer lugar dentro do seu controlador funções:

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

Quando ativado, um relatório será gerado e inserido na parte inferior de suas páginas.

Para desativar o profiler você irá usar:

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

Classe Paginação

Código Igniter Paginação da classe é muito fácil de usar, e é 100% personalizável, éter dinamicamente ou via armazenados preferências.

Se você não estiver familiarizado com o termo "paginação", ele se refere a ligações que lhe permite navegar a partir de página a página, como este:

« First < 1 2 3 4 5 > Last »

Exemplo

Aqui está um exemplo simples mostrando como criar uma paginação no seu controlador de funções:

```
$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:

A $config array contém a sua configuração variáveis. É passado para o $this->pagination->initialize função como mostrado acima. Embora existam alguns vinte itens que você pode configurar, no mínimo, você precisa dos três mostrados. Aqui está uma descrição do que representam esses itens:

base_url Este é o URL completo para o controlador class / function contendo a sua paginação. No exemplo acima, é o que aponta para um controlador chamado "Teste" e uma função chamada "página". Tenha em mente que você pode re-encaminhar seus URI se você precisa de uma estrutura diferente.

total_rows Este número representa o total de linhas no resultado que está a criar para a paginação. Tipicamente este será o número total de linhas que seu banco de dados consulta retornou.

per_page O número de itens que você deseja mostrar por página. No exemplo acima, você estaria mostrando 20 itens por página.

O **create_links ()** função retorna uma string vazia quando não há paginação para mostrar.

Definir preferências de um ficheiro de configuração

Se você preferir não definir preferências usando o método acima, você pode colocá-los em vez um arquivo de configuração. Basta criar um novo arquivo chamado o pagination.php, adicione o $config array neste arquivo. Em seguida, salve o arquivo em: config / pagination.php e ele será usado automaticamente. Você não precisa usar o $this->pagination->initialize função se você salvar suas preferências em um arquivo de configuração.

Personalizando a Paginação

O seguinte é uma lista de todas as preferências você pode passar a função de inicialização para adequar o monitor.

$ config ['uri_segment'] = 3;

A paginação função que determina automaticamente o seu segmento de URI contém o número da página. Se você precisa de algo diferente, você pode especificar-lo.

$ config ['num_links'] = 2;

O número de "dígito" links que você gostaria antes e após a página selecionada número. Por exemplo, o número 2 vai colocar dois dígitos em ambos os lados, como no exemplo links na parte superior desta página.

Adicionando juntando Markup

Se você gostaria de cercar toda a paginação com alguma marcação você pode fazê-lo com estas duas preferências:

$ config ['full_tag_open'] = '<p>';

A tag de abertura colocado no lado esquerdo de todo o resultado.

$ config ['full_tag_close'] = '</ p>';

Tag de fechamento colocado do lado direito de todo o resultado.

Personalizando o primeiro link

$ config ['first_link'] = 'First';

O texto que você gostaria mostrado no "primeiro" link do lado esquerdo.

$ config ['first_tag_open'] = ' ';

A tag de abertura para o "primeiro" link.

$ config ['first_tag_close'] = '</ div>';

Tag de fechamento para o "primeiro" link.

Personalizando o último link

$ config ['last_link'] = 'Último';

O texto que você gostaria mostrado no "último" link do lado direito.

$ config ['last_tag_open'] = ' ';

A tag de abertura para o "último" link.

$ config ['last_tag_close'] = '</ div>';

Tag de fechamento para o "último" link.

Personalização do "Next" Link

$ config ['next_link'] = '>';

O texto que você gostaria mostrado na "próxima" página link.

$ config ['next_tag_open'] = ' ';

A tag de abertura para a "próxima" link.

$ config ['next_tag_close'] = '</ div>';

Personalizando o "anterior" Link

$ config ['prev_link'] = '<';

O texto que você gostaria mostrado no programa "anterior" page link.

$ config ['prev_tag_open'] = ' ';

A tag de abertura para o "anterior" link.

$ config ['prev_tag_close'] = '</ div>';

Tag de fechamento para o "anterior" link.

Personalizando a "Página atual" Link

$ config ['cur_tag_open'] = '<b>';

A tag de abertura para o "atual" link.

$ config ['cur_tag_close'] = '</ b>';

Tag de fechamento para o "atual" link.

Personalizando o "Dígito" Link

$ config ['num_tag_open'] = ' ';

A tag de abertura para o "dígito" link.

$ config ['num_tag_close'] = '</ div>';

Tag de fechamento para o "dígito" link.

Classe Sessão

A classe Sessão lhe permite manter um "status" do usuário e acompanhar sua atividade enquanto ele navega por seu site. A classe armazena as informações da sessão para cada usuário em dados dispostos em série (e opcionalmente criptografados) num cookie. Ela pode ainda armazenar os dados da sessão em uma tabela no banco de dados para aumentar a segurança, pois isto permite que a ID da sessão no cookie do usuário tenha ligação, e seja comparável, com o ID da sessão armazenada. Por default, apenas o cookie é salvo. Se você escolher usar a opção de banco de dados, precisará criar a tabela de sessão indicada abaixo.

Nota: A classe Sessão não utiliza as sessões nativas do PHP. Ela gera seus próprios dados de sessão, oferecendo maior flexibilidade para os desenvolvedores.

Inicializando a Sessão

Sessões irão tipicamente rodar globalmente em cada carregamento da página, portanto a classe sessão deve ser ou inicializada em seus construtores dos controllers, ou pode ser carregada automaticamente pelo sistema. Na maioria dos casos a classe sessão irá rodar despercebida no background, portanto sua simples inicialização a fará ler, criar e atualizar sessões.

Para inicializar a classe Sessão manualmente em seu controller, use a função $this->load->library:

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

Uma vez carregada, o objeto estará disponível usando: $this->session

Como as Sessões funcionam?

Quando uma página é carregada, a classe sessão irá checar se um dado válido de sessão existe no cookie de sessão do usuáiro. Se os dados de sessão não existirem (ou houverem expirados) uma nova sessão será criada e salva no cookie. Se a sessão já existir, sua informação e o cookie serão atualizados.

É importante que você entenda que uma vez inicializada, a classe Sessão roda automaticamente. Não há nada que você precise fazer para isto. Você pode, como verá abaixo, trabalhar com os dados de sessão ou mesmo adicionar seus próprios dados para a sessão do usuário, mas o processo de leitura, escrita e atualização da sessão é automático.

O que são os Dados da Sessão?

Uma sessão, no que tange à preocupação do Code Igniter, é simplesmente uma array contendo as seguintes informações:

A ID de Sessão única do usuário (ela é estatisticamente uma string randômica com uma entropia muito forte, "hasheada" com MD5, para portabilidade)

O endereço IP do usuário

Os dados do User Agent do usuário (os primeiros 50 caracteres da string de dados do browser)

O horário da "última atividade" e da "última visita".

Os dados acima são armazenados num cookie numa array em série, com este protótipo:

```
[array]
(
    'session_id'    => random hash,
    'ip_address'    => 'string - endereço IP do usuário',
    'user_agent'    => 'string - dados do User Agent do usuário',
    'last_activity' => horário,
    'last_visit'    => horário
)
```

Se você tiver habilitado a opção de criptografia, a array em séria será criptografada antes de ser armazenada num cookie, tornando os dados bem seguros e impossíveis de serem lidos ou alterados por alguém. Mais informações acerca da criptografia, podem ser encontradas aqui, no entanto a classe Sessão é responsável por inicializar e criptografar os dados automaticamente.

Nota: Cookies de sessão são atualizados apenas de 5 em 5 minutos, para reduzir a carga de processamento. Se você repetidamente recarregar a página, notará que o horário de "última atividade" será atualizado apenas se 5 minutos ou mais se passaram desde a última vez que o cookie foi escrito.

Recuperando os Dados de Sessão

Qualquer pedaço da informação vinda da array de sessão estará disponível através da função:

$this->session->userdata('item');

Onde item é o índice da array correspondente ao item que você deseja recuperar. Por exemplo, para recuperar a ID de sessão, você deve fazer isto:

$session_id = $this->session->userdata('session_id');

Nota: A função retorna FALSO (booleano) se o item que você esta tentando acessar não existir.

Adicionando Dados de Sessão Personalizados

Um aspecto útil da array de sessão é que você pode adicionar seus próprios dados a ela, e ela os armazenará no cookie do usuário. Mas porque você gostaria de fazer tal coisa? Aqui vai um exemplo:

Digamos que um usuário em particular faça o login em seu site. Uma vez autenticado, você poderá adicionar seu nome de usuário e e-mail para o cookie de sessão, tornando os dados globalmente disponíveis sem precisar rodar uma query no banco quando precisar acessá-los.

Adicionar seus dados para a array de sessão envolve passar uma array, contendo seus novos dados, para esta função:

$this->session->set_userdata($array);

Onde $array é uma array associativa contendo seus novos dados. Aqui está um exemplo:

```
$novosdados = array(
            'nome_de_usuario'  => 'fulano',
            'e-mail'    => 'fulano@algum-site.com',
            'esta_logado' => TRUE
          );
$this->session->set_userdata($novosdados);
```

Se você desejar adicionar dados do usuário em um valor por vez, set_userdata() também suporta esta sintaxe:

$this->session->set_userdata('alguma_coisa', 'algum_valor');

Nota: Cookies podem conter até 4KB de dados, por isto tenha cuidado em não exceder tal capacidade. O processo de criptografia, em particular, produz uma string de dados mais longa que a original, por isso mantenha controle de quantos dados você armazenará.

Salvando os dados de Sessão no Banco de Dados

A ID de sessão, contida da array respectiva, só pode ser validada se você a armazenar no banco de dados. Para as aplicações que requeiram pouca ou nenhuma segurança, a validação da ID de sessão pode ser

necessária, mas se sua aplicação exigir segurança, a validação é obrigatória.

Quando os dados de sessão estão disponíveis no banco, toda vez que uma sessão válida for encontrada no cookie do usuário, uma query é executada para compará-los. Se a ID não bater, a sessão é destruída. As ID's não nunca podem ser atualizadas, podem apenas ser geradas quando uma nova sessão é criada.

Para armazenar sessões, você primeiro deve criar a tabela no banco para este propósito. Aqui vai um protótipo básico (para MySQL) requerido pela classe Sessão:

```
CREATE TABLE IF NOT EXISTS `ci_sessions` (
id_sessao varchar(40) DEFAULT '0' NOT NULL,
endereco_ip varchar(16) DEFAULT '0' NOT NULL,
user_agent varchar(50) NOT NULL,
ultima_atividade int(10) unsigned DEFAULT 0 NOT NULL,
PRIMARY KEY (id_sessao)
);
```

Nota: Por default, a tabela chama-se ci_sessions, mas você pode colocar qualquer nome que quiser, desde que atualize o arquivo application/config/config.php para que ele contenha o novo nome. Uma vez criada a tabela, você pode habilitar a opção de banco de dados em seu arquivo config.php como se segue:

$config['sess_use_database'] = TRUE;

Uma vez habilitada, a classe Sessão irá armazenar os dados de sessão no banco.

Tenha certeza de ter especificado o nome da tabela em seu arquivo de configuração:

$config['sess_table_name'] = 'ci_sessions";

Nota: A classe Sessão tem uma coleção lixeira embutida, que limpa as sessões expiradas, por isso você não precisa criar sua própria rotina para fazê-los.

Destruindo uma Sessão

Para limpar a sessão atual:

$this->session->sess_destroy();

Preferências de Sessão

Você encontrará as preferências relacionadas a Sessões em seu arquivo application/config/config.php:

Preferências	Valor Default	Opções	Descrição
sess_cookie_name	ci_session	Nenhum	O nome para a sessão do cookie.
sess_expiration	7200	Nenhum	O número de segundos que você quer que a sessão dure. O valor padrão é 2 horas (7.200 segundos). Se desejar uma sessão que não expire, use o valor zero: 0
sess_encrypt_cookie	FALSE	TRUE/FALSE (boolean)	Se deve criptografar os dados da sessão.
sess_use_database	FALSE	TRUE/FALSE (boolean)	Se deve salvar a sessão no banco. Você deve criar a tabela antes de habilitar esta opção.
sess_table_name	ci_sessions	Qualquer nome válido para uma tabela SQL	O nome da tabela de sessão no banco.

sess_match_ip	FALSE	TRUE/FALSE (boolean)	Se deve verificar o endereço IP do usuário ao ler os dados da sessão. Note que alguns ISPs dinamicamente alterar o IP, portanto se você quiser uma sessão que não expirer, deve usar FALSE.
sess_match_useragent	TRUE	TRUE/FALSE (boolean)	Se deve verificar o User Agent do usuário ao ler os dados da sessão.

Classe Trackback

A Classe Trackback Class provê funções para enviar e receber dados Trackback.

Se você não está familiarizado com Trackbacks, achará mais informações aqui.

Inicializando a Classe

Como a maioria das classes no Code Igniter, a classe Trackback é inicializada, em seu controller, usando a função $this->load->library:

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

Uma vez carregada, seu objeto estará disponível usando: $this->trackback

Enviando Trackbacks

Um trackback pode ser enviado de qualquer função em seus controllers, usando um código similar a este exemplo:

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

$tb_data = array(
        'ping_url'  => 'http://algum-site.com/trackback/456',
        'url'       => 'http://www.meu-site.com/blog/entry/123',
        'title'     => 'O Título do meu Post',
        'excerpt'   => 'O conteúdo do Post.',
        'blog_name' => 'O nome do Meu Blog',
        'charset'   => 'utf-8'
        );

if ( ! $this->trackback->send($tb_data))
{
    echo $this->trackback->display_errors();
}
else
{
    echo 'O Trackback foi enviado!';
}
```

Descrição dos dados da array:

ping_url - A URL do site para o qual você está enviando o Trackback. Você pode enviar Trackbacks para múltiplas URLs separando-as por vírgula..

url - A URL para o SEU site, onde o post do blog poderá ser visto.

title - O nome do título do seu post.

excerpt - O conteúdo de seu post. Nota: a classe Trackback irá automaticamente enviar apenas os primeiros 500 caracteres do post. E também irá retirar todo o HTML.

blog_name - O nome do seu Blog.

charset - A codificação de caracteres na qual seu post foi escrito. Se omitida, a UTF-8 será usada.

A função de envio de Trackbacks retornará TRUE/FALSE (booleano) em sucesso ou falha, respectivamente. Se ela falhar, você pode recuperar a mensagem de erro usando:

$this->trackback->display_errors();

Recebendo Trackbacks

Antes que possa receber Trackbacks, você precisa criar um blog. Se não tem um ainda, não há razão em continuar lendo isto.

Receber Trackbacks é um pouco mais complexo que enviá-los, apenas porque você precisará de uma tabela no banco de dados para armazená-los e precisará validar os dados recebidos. Você é encorajado a implementar um processo de validação completo e consistente para evitar dados duplicados e se proteger de spam. Você pode querer também limitar o número de Trackbacks que deseja receber de um particular IP, dentro de um espaço de tempo, para diminuir a chance de spam. O processo de receber Trackbacks é bem simples; o processo da validação dos mesmos é que toma mais tempo e esforço.

Sua URL Permanente

Para aceitar Trackbacks você precisa mostrar a URL Trackback próxima a seus posts. Está será a URL que as pessoas usarão para enviar Trackbacks para você (iremos nos referenciar a ela como sua "URL Ping").

Sua URL Ping deve apontar para a função de um controller onde o seu código de recepção de Trackbacks está localizado e a URL deve conter o número ID de cada post em particular, para que quando um Trackback for recebido, você poderá associá-lo com o post.

Por exemplo, se sua classe controller chama-se Trackback, e a função de recepção chama-se receive, suas URLs Ping irão se parecer com algo assim:

http://www.seu-site.com/index.php/trackback/receive/entry_id

Onde entry_id representa o número ID individual ID para cada um de seus posts.

Criando uma Tabela Trackback

Para receber Trackbacks, você precisar criar uma tabela para armazená-las. Aqui vai um protótipo básico para tal tabela:

```
CREATE TABLE trackbacks (
 tb_id int(10) unsigned NOT NULL auto_increment,
 post_id int(10) unsigned NOT NULL default 0,
 url varchar(200) NOT NULL,
 titulo varchar(100) NOT NULL,
 excerto text NOT NULL,
 nome_blog varchar(100) NOT NULL,
 tb_data int(10) NOT NULL,
 endereco_ip varchar(16) NOT NULL,
```

```
 PRIMARY KEY (tb_id),
 KEY (entry_id)
);
```

A especificação Trackback apenas requer quarto pedaços de informação para serem enviados (url, título, excerto, nome_blog), mas para tornar os dados mais úteis, adicionamos alguns campos a mais no esquema acima (data, endereço IP etc.).

Processando um Trackback

Aqui está um exemplo mostrando como você irá receber e processar um Trackback. O código abaixo tem a intenção de ser usado dentro da função controller onde você espera receber os Trackbacks.

```
$this->load->library('trackback');
$this->load->database();

if ($this->uri->segment(3) == FALSE)
{
   $this->trackback->send_error("Não foi possível determinar o ID do post");
}

if ( ! $this->trackback->receive())
{
   $this->trackback->send_error("O Trackback não contém dados válidos");
}

$dados = array(
          'tb_id'     => '',
          'entry_id'  => $this->uri->segment(3),
          'url'       => $this->trackback->data('url'),
          'title'     => $this->trackback->data('titulo'),
          'excerpt'   => $this->trackback->data('excerto'),
          'blog_name' => $this->trackback->data('nome_blog'),
          'tb_date'   => time(),
          'ip_address' => $this->input->endereco_ip()
          );

$sql = $this->db->insert_string('trackbacks', $dados);
$this->db->query($sql);

$this->trackback->send_success();
```

Notas:

É esperado que o número ID do post esteja no terceiro segmento da URL. Isto baseia-se no exemplo de URI que mostramos anteriormente:

http://www.seu-site.com/index.php/trackback/receive/entry_id

Note a entry_id no terceiro URI, o qual você pode recuperar usando:

$this->uri->segment(3);

No código de recebimento do Trackback, acima, se o terceiro segmento estiver faltando, iremos disparar um erro. Sem uma ID válida, não há razão para continuar.

A função $this->trackback->receive() é simplesmente uma função de validação que pegará os dados recebidos e ter a certeza que eles contêm os quatro pedaços de dados obrigatórios (url, titulo, excerto,

nome_blog). Ela retornará TRUE/FALSE em sucesso ou falha, respectivamente. Se ela falhar, você receberá uma mensagem de erro.

Os dados do Trackback recebido podem ser recuperados usando esta função:

$this->trackback->data('item')

Onde item representa um destes quatro pedaços de informação: url, título, excerto ou nome_blog.

Se os dados do Trackback forem recebidos com sucesso, você poderá enviar uma mensagem de sucesso usando:

$this->trackback->send_success();

Nota: O código acima não contém validação de dados. Você é encorajado a incluí-la.

Classe Template Parser

Esta classe lhe habilita analisar sintaticamente (to parse) pseudo-variáveis contidas dentro de seus arquivos View. Ela pode analisar simples variáveis ou pares de tag variáveis. Se você nunca usou um template engine, pseudo-variáveis se parecem com isto:

```
<html>
<head>
<title>{blog_title}</title>
</head>
<body>
<h3>{blog_heading}</h3>
{blog_entries}
<h5>{title}</h5>
<p>{body}</p>
{/blog_entries}
</body>
</html>
```

Esas variáveis não são variáveis PHP na verdade, mas apenas representações em texto puro, que lhe permitem eliminar o PHP de seus templates (view files).

Nota: o Code Igniter não requer que você use esta classe, pois usar PHP puro em suas páginas view as deixa rodar um pouco mais rápido. No entando, alguns desenvolvedores preferem usar um template engine se estiverem trabalhando com designers, que podem ter dificuldades em trabalhar com PHP.

Note também: A Classe Template Parser não é uma solução à toda prova. Nós a mantivemos bem concisa em seu propósito, para manter máxima performance.

Inicializando a Classe

Como a maioria das classes no Code Igniter, esta é inicializada, em seu controller, usando a função $this->load->library:

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

Uma vez carregada, seu objeto estará disponível usando: $this->parser

As seguintes funções estão disponíveis para esta biblioteca:

Esta variável aceita um nome e uma array de dados de um template como entrada, e gerar a versão analisada (parsed). Exemplo:

```
$this->load->library('parser');
$dados = array(
        'titulo_blog' => 'Meu Título do Blog',
        'cabecalho_blog' => 'Meu Cabeçalho do Blog'
        );
$this->parser->parse('blog_template', $dados);
```

O primeiro parâmetro contém o none do arquivo view (neste exemplo, o arquivo poderia se chamar blog_ template.php), e o segundo parâmetro contém a array associativa com os dados a serem substituídos no template. No exemplo acima, o template conteria duas variáveis: {titulo_blog} e {cabecalho_blog}

Não é preciso dar "echo" ou fazer outra coisa com os dados retornados pela $this->parser->parse(). Eles são automaticamente passados à Classe Output que dará saída para o browser. No entando, se você quiser os dados retornados, no lugar de enviá-los à Classe Output, pode passar TRUE (booleano) para o terceiro parâmetro:

```
$string = $this->parser->parse('blog_template', $dados, TRUE);
```

Pares de Variáveis

O exemplo acima permite que simples variáveis sejam substituídas. Mas e se você quiser que um bloco inteiro de variáveis seja repetido, com cada interação contendo valores novos? Considere o exemplo que mostramos no topo da página:

```
<html>
<head>
<title>{blog_title}</title>
</head>
<body>
<h3>{blog_heading}</h3>
{blog_entries}
<h5>{title}</h5>
<p>{body}</p>
{/blog_entries}
</body>
</html>
```

No código acima, você notará o par de variáveis: {blog_entries} dados... {/blog_entries}. Neste caso, o montante de dados entre estes pares seria repetido múltiplas vezes, correspondendo ao números de linhas em um resultado.

A análise sintática de pares de variáveis é feita usando o mesmo código parar analisar variáveis simples mostrado acima, exceto que você adicionar uma array multidimensional correspondente aos dados do seu par de variáveis. Considere este exemplo:

```
$this->load->library('parser');
$dados = array(
        'blog_title'  => 'Meu Título do Blog',
        'blog_heading' => 'Meu cabeçalho do Blog',
        'blog_entries' => array(
                        array('title' => 'Título 1', 'body' => 'Conteúdo 1'),
                        array('title' => 'Título 2', 'body' => 'Conteúdo 2'),
                        array('title' => 'Título 3', 'body' => 'Conteúdo 3'),
                        array('title' => 'Título 4', 'body' => 'Conteúdo 4'),
                        array('title' => 'Título 5', 'body' => 'Conteúdo 5')
                        )
        );
$this->parser->parse('blog_template', $dados);
```

Se os dados do seu "par" estão vindo de um resultado do banco de dados, que já é uma array multidimensional, você pode simplismente usar a função de resultado de banco:

```
$query = $this->db->query("SELECT * FROM blog");
$this->load->library('parser');
$dados = array(
        'blog_title'   => 'Meu Título do Blog',
        'blog_heading' => 'Meu cabeçalho do Blog',
        'blog_entries' => $query->result_array()
       );
$this->parser->parse('blog_template', $dados);
```

Classe Unit Testing

Unit testing, ou Teste Unitário, é uma abordagem em desenvolvimento de software na qual testes são escritos para cada função em sua aplicação. Se você não está familiarizado com este conceito, poderia fazer uma busca rápida sobre o assunto.

A classe Unit Testing do Code Igniter é bem simples, consistindo de uma função de avaliação e duas de resultado. Ela não tem a intenção de ser uma suite de testes gigante mas sim um simples mecanismo de avaliar seu código para determinar se ele está produzindo os corretos tipos de dados e resultados.

Inicializando a Classe

Como a maioria das classes no Code Igniter, esta classe é inicializada, em seu controller, usando a função $this->load->library:

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

Uma vez carregada, seu objeto estará disponível usando: $this->unit

Rodando Testes

Rodar um teste envolve fornecer um teste e o resultado esperado à seguinte função:

$this->unit->run(teste, resultado esperado, 'nome do teste');

Onde teste é o resultado do seu código que você deseja testar, resultado esperado é o tipo de dado que você espera, e nome do teste é um nome opcional que você pode dar ao seu teste. Exemplo:

```
$teste = 1 + 1;
$resultado_esperado = 2;
$nome_do_teste = 'Somar um mais um';
$this->unit->run($teste, $resultado_esperado, $nome_do_teste);
```

O resultado esperado fornecido por você pode ser uma comparação ligeral, ou uma comparação de tipos de dados. Aqui vai um exemplo de uma literal:

$this->unit->run('Foo', 'Foo');

Aqui vai um exemplo de uma comparação de tipos de dados:

$this->unit->run('Foo', 'is_string');

Notou o uso do "is_string" no segundo parâmetro? Isto diz à função para avaliar se seu teste está produzindo uma string como resultado. Aqui vai uma lista de tipos permitidos de comparações:

is_string	is_false	is_float	is_null
is_bool	is_int	is_double	
is_true	is_numeric	is_array	

Gerando relatórios

Você pode tanto mostrar resultados após cada teste, ou pode rodar vários testes e gerar um relatório no final. Para mostrar um relatório diretamente, simplesmente dê um echo ou retorne a função run:

echo $this->unit->run($teste, $resultado_esperado);

Para rodar um relatório completo de todos os testes, use isto:

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

O relatório será formatado em uma tabela HTML para visualização. Se você preferir os dados crus, pode recuperar um array usando:

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

Modo Estrito

Por default, a classe unit test avalia comparações literais de forma pouco rigorosa. Considere este exemplo:

$this->unit->run(1, TRUE);

O teste irá avaliar um inteiro, mas o resultado esperaro é booleano. O PHP, no entando, devido a seus tipos-de-dados genéricos, irá avaliar o código acima como VERDADEIRO, usando um teste de igualdade normal:

if (1 == TRUE) echo 'Isto é avaliando como VERDADEIRO';

Se você preferir, pode colocar a classe unit test em modo estrito, no qual ela irá comprar os tipos de dados assim como os valores:

if (1 === TRUE) echo 'Isto é avaliando como FALSO';

Para habilitar o modo estrito, use isto:

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

Habilitando/Desabilitando o Unit Testing

Se você quiser manter alguns testes dentro de seus scripts, mas não quer rodá-los a menos que precise, pode desabilitar o unit testing usando:

$this->unit->active(FALSE)

Criando um Template

Se você quiser seus resultados de testes formatados de forma diferente da padrão, pode montar seu próprio template. Aqui vai um exemplo de um simples template. Note as pseudo-variáveis exigidas:

```
$str = '
<table border="0" cellpadding="4" cellspacing="1">
  {rows}
    <tr>
    <td>{item}</td>
    <td>{result}</td>
    </tr>
  {/rows}
</table>';

$this->unit->set_template($str);
```

Nota: O seu template deve estar declarado antes de rodar o processo de unit testing.

Classe URI

A Classe URI provê funções que o ajudarão recuperar informações a partir de suas string URI. Se você usa o roteamento URI, pode ainda recuperar informações sobre os segmentos re-roteados.

Nota: Esta classe é inicializada automaticamente pelo sistema, por isso não é necessário fazê-lo.

$this->uri->segment(n)

Permite que você recupere um segmento específico. Onde n é o número do segmento que você deseja recuperar. Segmentos são numeros da esquerda pra direita. Por exemplo, se sua URL completa for assim:

http://www.seu-site.com/index.php/noticias/locais/metro/crime_aumenta

O número dos segmentos seriam estes:

1. noticias

2. locais

3. metro

4. crime_aumenta

Por default, a função retorna FALSE (booleano) se o segmento não existir. Há um segundo parâmetro adicionar que lhe permite configurar seu próprio valor padrão se o segmento estiver faltando. Por exemplo, isto diria à função retornar o número zero no advento de uma falha:

$product_id = $this->uri->segment(3, 0);

Isto ajuda evitar escrever código como desse jeito:

```
if ($this->uri->segment(3) === FALSE)
{
  $product_id = 0;
}
else
{
  $product_id = $this->uri->segment(3);
}
```

$this->uri->rsegment(n)

Esta função é idência a anterior, exceto que ela lhe permite recuperar um segmento específico em seus URI re-roteados, se você estiver usando a funcionalidade URI Routing do Code Igniter.

$this->uri->slash_segment(n)

Esta função é praticamente idêntica a $this->uri->segment(), exceto que ela adiciona uma barra antes e/ou depois baseado no segundo parâmetro. Se o parâmetro não for usado, a barra é adicionada antes. Exemplo:

```
$this->uri->slash_segment(3);
$this->uri->slash_segment(3, 'leading');
$this->uri->slash_segment(3, 'both');
```

Retorna:

1. segmento/

2. /segmento

3. /segmento/

$this->uri->slash_rsegment(n)

Esta função é idência a anterior, exceto que ela lhe permite adicionar barras a um segmento específico seus URI re-roteados, se você estiver usando a funcionalidade URI Routing do Code Igniter.

$this->uri->uri_to_assoc(n)

Esta função lhe permite tranformar segmentos URI em uma array associativa de pares chave/valor. Considere este URI:

index.php/usuario/busca/nome/joe/localizacao/UK/sexo/masculino

Usando esta função, você pode transformar a URI em uma array associatia usando este protótipo:

```
[array]
(
  'nome' => 'joe'
  'localizacao' => 'UK'
  'sexo' => 'masculino'
)
```

O primeiro parâmetro da função lhe permite configurar a partir de qual segmento deve começar o processamento. Por default, ela começara do 3, já que seu URI normalmente irá conter um controller/ função no primeiro e segundo parâmetro respectivamente. Exemplo:

```
$array = $this->uri->uri_to_assoc(3);
echo $array['nome'];
```

O segundo parâmetro lhe permite configurar os nomes de chave padrões, para que a array retornada pela função sempre contenha indíces esperados, mesmo se faltantes na URI. Exemplo:

```
$default = array('nome', 'sexo', 'localização', 'tipo', 'espécie');
$array = $this->uri->uri_to_assoc(3, $default);
```

Se o URI não contiver um valor no segmento 3, um índice na array será adicionadado com esse nome, com o valor FALSE.

Por último, se um valor correspondente a uma dada chave não for encontrado para (se hover um número ímpar de segmentos URI), o valor será marcado como FALSE (booleano).

$this->uri->ruri_to_assoc(n)

Esta função é idêntica a anterior, exceto que ela cria uma array associativa usando seus URI re-roteados, se você estiver usando a funcionalidade URI Routing do Code Igniter.

$this->uri->assoc_to_uri()

Pega uma array associativa como entrada e gera uma string URI a partir dela. As chaves da array serào incluídas na string. Exemplo:

```
$array = array('produto' => 'sapatos', 'tamanho' => 'grande', 'cor' => 'vermelho');

$str = $this->uri->assoc_to_uri($array);

// Produz: produto/sapatos/tamanho/grande/cor/vermelho

$this->uri->uri_string()
```

Retorna uma string com o URI completo. Por exemplo, se esta é sua URL completa:

http://www.seu-site.com/index.php/noticas/local/345

A função retornaria isto:

noticas/local/345

$this->uri->ruri_string(n)

Esta função é idêntica a anterior, exceto que ela retorna seus URI re-roteados, se você estiver usando a funcionalidade URI Routing do Code Igniter.

$this->uri->total_segments()

Retorna o número total de segmentos.

$this->uri->total_rsegments(n)

Esta função é idêntica a anterior, exceto que ela retorna o número total de segmentos em seus URI re-roteados, se você estiver usando a funcionalidade URI Routing do Code Igniter.

$this->uri->segment_array()

Retorna uma array contendo os segmentos URI. Por exemplo:

```
$segs = $this->uri->segment_array();

foreach ($segs as $segment)
{
  echo $segment;
  echo '<br />';
}
```

$this->uri->rsegment_array(n)

Esta função é idêntica a anterior, exceto que ela retorna a array contendo os segmentos em seus URI re-roteados, se você estiver usando a funcionalidade URI Routing do Code Igniter.

Classe User Agent

A Classe User Agent provê funções que auxiliam a identificar informações sobre o navegador, dispositivo móvel, ou o robô que esteja visitando seu site. Você também pode usar esta Classe para obter informações de referrer (de onde vem o usuário), idioma e conjunto de caracteres suportado (charset)

Inicializando a Classe

Da mesma maneira como a maioria das outras classes no Code Igniter, a classe User Agent é inicializada no seu controller usando a função $this->load->library:

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

Uma vez carregado, o objeto estará disponível usando: $this->agent

Definições de User Agent

As definições de nome do user agent estão localizadas no arquivo de configuração localizado em application/config/user_agents.php. Você pode adicionar itens às várias arrays do user agent se necessário.

Exemplo

Quando a classe User Agent é inicializada ela tentará determinar se o user agent que está visitando seu site é um navegador (web browser), um dispositivo móvel ou um robô(spider).Também tentará obter a informação da plataforma, caso esteja disponível.

```
$this->load->library('user_agent');
if ($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(); // Informação de Platforma (Windows, Linux, Mac, etc.)
```

Referências de Funções

$this->agent->is_browser()

Retorna TRUE/FALSE (booleano) se o user agent é um navegador conhecido.

$this->agent->is_mobile()

Retorna TRUE/FALSE (booleano) se o user agent é um dispositivo móvel conhecido.

$this->agent->is_robot()

Retorna TRUE/FALSE (booleano) se o user agent é um robô conhecido.

Nota: A biblioteca user agent contem apenas as definições de robôs mais comuns. Não é uma lista completa destes. Existe centenas deles, portanto ficar procurando cada um deles pode não algo eficiente (e prático). Caso você ache que algum destes robôs que regularmente visitam seu site está fora da lista, adicione-o no arquivo application/config/user_agents.php.

$this->agent->is_referral()

Retorna TRUE/FALSE (booleano) se o the user agent foi referenciado de um outro site (está vindo de algum outro site).

$this->agent->browser()

Retorna uma string que contem o nome do navegador que está visitando seu site.

$this->agent->version()

Retorna uma string que contem o número de versão number do navegador que está visitando seu site.

$this->agent->mobile()

Retorna uma string que contem o nome do dispositivo móvel que está visitando seu site.

$this->agent->robot()

Retorna uma string que contem o nome do robô que está visitando seu site.

$this->agent->platform()

Retorna uma string que contem a platforma que está sendo usada para visitar seu site (Linux, Windows, OS X, etc.).

$this->agent->referrer()

O referrer, se o user agent está vindo de algum outro site. Normalmente você usará esta função da seguinte maneira:

```
if ($this->agent->is_referral())
{
   echo $this->agent->referrer();
}
```

$this->agent->agent_string()

Retorna uma string que contem a identificação completa do user agent . Normalmente algo assim:

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()

Permite verficar se o user agent aceita um idioma particular. Exemplo:

```
if ($this->agent->accept_lang('en'))
{
   echo 'Você suporta Inglês!';
}
```

Nota: Esta função não é muito confiável isto porque alguns navegadore não provêem infomação de idioma, e os que o fazem, não fornecem a informação de modo preciso.

$this->agent->accept_charset()

Permite verficar se o user agent aceita um conjunto de caracteres (charset) particular. Exemplo:

```
if ($this->agent->accept_charset('utf-8'))
{
   echo 'Seu browser suporta UTF-8!';
}
```

Nota: Esta função não é muito confiável isto porque alguns navegadore não provêem infomação de conjunto de caracteres, e os que o fazem, não fornecem a informação de modo preciso.

Validação de Formulário

Antes de explicar a abordagem do Code Igniter's com a validação de dados, vamos descrever o cenário ideal:

1. Um formulário é exibido.
2. Você preenche e o submete.
3. Se você submeteu alguma informação inválida ou esqueceu algum item obrigatórioa, o formulário é re-exibido contendo seus dados e uma mensagem de erro descrivendo o problema.
4. Este processo continuará até você submeter um formulário totalmente válido.

No lado de recebimento dos dados, o script deve:

1. Conferir os dados que são obrigatórios.
2. Verifique que os dados são do tipo correto e correspondem, respectivamente, com os critérios corretos. (Por exemplo, se um nome de usuário é submetido, ele deve ser validado de modo a conter apenas caracteres permitidos. Deve ter um número mínimo de caracteres e não exceder o número máximo. O nome de usuário não pode ser um já existente ou nem mesmo uma palavra reservada. Etc.)
3. Limpar os dados para segurança (exemplo: limpar SQL injections).
4. Pré-formatar os dados caso precise (É necessário retirar algum espaço em branco? Codificar/Encodar HTML ? Etc.)
5. Preparar os dados para inserção na base de dados.

Embora não haja nada de complexo no processo acima, ele frequentemetne requer um considerável números de linhas de código e para exibição de mensagens de erro, várias estruturas de controle são normalmente colocadas dentro do formulário HTML. Valdação de Formulário, embora simples de criar, é geralmente um pouco desorganizado e tedioso de implementar.

Code Igniter provê uma estrutura detalhada de validação que de fato minimiza a quantidade de linhas de código que você teria de escrever. Também remove toda as estruturas de controle de seu HTML, permitindo que seja limpo e livre de código.

Visão Geral

De modo a implementar a validação de formulário do Code Igniter você precisará de três coisas:

1. Um arquivo de View contendo o formulário.
2. Um arquivo de View contendo a mensagem de "successo" para ser exibida após o formulário ser enviado.
3. Uma função controller para receber e processar os dados que forem submetidos pelo formulário.

Vamos criar estas três coisas usando como exemplo um formulário de cadastro de usuário.

O Formulário

Usando um editor de texto, crie um formulário chamado myform.php. Nele coloque este código e salve-o na sua pasta applications/views/:

```
<html>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/validation.
html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:59 GMT -->
<head>
<title>My Form</title>
</head>
<body>
<?=$this->validation->error_string; ?>
<?=form_open('form'); ?>
<h5>Usuário</h5>
<input type="text" name="username" value="" size="50" />
<h5>Senha</h5>
<input type="text" name="password" value="" size="50" />
<h5>Confirmação de Senha</h5>
<input type="text" name="passconf" value="" size="50" />
<h5>Endereço de Email</h5>
<input type="text" name="email" value="" size="50" />
<div><input type="submit" value="Submit" /></div>
</form>
</body>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/validation.
html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:59 GMT -->
</html>
```

A Página de Sucesso

Usando um editor de texto, crie um formulário chamado formsuccess.php. Nele coloque este código e salve-o na sua pasta applications/views/:

```
<html>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/validation.
html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:59 GMT -->
<head>
<title>Meu Formulário</title>
</head>
<body>
<h3>Seu formulário foi enviado com sucesso!</h3>
<p><?=anchor('form', 'Try it again!'); ?></p>
</body>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/validation.
html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:59 GMT -->
</html>
```

O Controller

Usando um editor de texto, crie um formulário chamado form.php. Nele coloque este código e salve-o na sua pasta 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');
		}
	}
}
?>
```

Execute-o!

Para executar seu formulário, visite seu site usando uma URL similar a esta:

www.seu-site.com/index.php/form/

Se você enviar o formulário verá apenas ele ser recarregado. Isso porque você ainda não configurou regras de validação, regras estas que em breve veremos.

Explanação

Você perceberá muitas coisas a respeito das páginas acima:

O formulário (myform.php) é um formulário padrão, com algumas exceções:

1. Ele usa um assistente de formulário para criar a abertura do formulário. Tecnicamente isto não é necessário. Você poderia criar o formulário usando um HTML padrão. Entretando, o benfício de usar um assistente é que ele gera a URL de 'action' para você, baseado na URL em seu arquivo de configuração. Isto torna sua aplicação mais portável e flexível na eventualidade de mudança de suas URLs.

2. No topo do formulário você notará a seguinte variável:

   <?=$this->validation->error_string; ?>

   Esta variável exibrá qualquer mesagem de erro retornada pelo validador. Se não existirem mensagens, ele não retornará nada.

O controller (form.php) tem uma função: index(). Esta função inicializa a classe de validação e carrega o assistente de formulário e o assistente de URL usados pelos seus arquivos de view. Ela também executa a rotina de validação. No caso da validação ter sido bem sucedida, pode ser apresentado o formulário ou a página de sucesso de envio.

Partindo do pressuposto que você não tenha dito à classe de validação para validar coisa alguma ela retornará "false"(false booleano) por padrão. A função run() apenas retorna "true" se tiver aplicado suas regras com sucesso sem nenhum tipo de falha.

Configurando Regras de Validação

Code Igniter permite que você configure quantas regras você precisar para determinado campo, cascateando-as em ordem, e permite até mesmo que você trate e pré-processe os datos do campo concomitantemente. Vejamos isso em ação, explicaremos isso em seguida.

Em seu controller (form.php), adicione este código logo abaixo da função de inicialização de validação:

```
$rules['username'] = "required";
$rules['password'] = "required";
$rules['passconf'] = "required";
$rules['email'] = "required";

$this->validation->set_rules($rules);
```

Seu controller agora irá se parecer com algo assim:

```
<?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');
		}
	}
}
?>
```

Agora submeta o formulário com os campos em branco e você deverá ver a mensagem de erro. Se você submeter o formulário com todos os campos preenchidos, verá então a página de envio com sucesso.

Nota: Os campos do formulário ainda não estão sendo re-populados com os dados quando ocorre o erro. Vamos chegar nesta parte em breve, uma vez que ainda estamos falando sobre as regras de validação.

Alterando os Delimitadores de Erro

Por padrão, o sistema adiciona uma tag de parágrafo (<p>) em volta de cada mensagem de erro exibida. Você pode facilmente alterar estes delimitadores com este código, colocado em seu controller:

```
$this->validation->set_error_delimiters('<div class="error">', '</div>');
```

Neste exemplo nós trocamos para tags DIV.

Regras em Cascata

Code Igniter permite concatenar múltiplas regras. Vamos experimentar. Altere sua array de regras para isto:

```
$rules['username'] = "required|min_length[5]|max_length[12]";
$rules['password'] = "required|matches[passconf]";
$rules['passconf'] = "required";
$rules['email'] = "required|valid_email";
```

O código acima requer que:

1. O campo username não seja menor que 5 caracteres nem maior que 12.
2. O valor do campo senha(password) deve corresponder ao valor do campo de confirmação de senha (passconf).
3. O campo email deve conter um endereço de email válido.

Experimente

Nota: Há muitas regras disponíveis que você pode ler sobre na referência de validação.

Tratando os Dados

Juntamente com as funções de validação como as que usamos acima, você também pode tratar seus dadoas de vários modos. Por exemplo, você pode configurar regras como esta:

```
$rules['username'] = "trim|required|min_length[5]|max_length[12]|xss_clean";
$rules['password'] = "trim|required|matches[passconf]|md5";
$rules['passconf'] = "trim|required";
$rules['email'] = "trim|required|valid_email";
```

Acima, estamos "aparando"("trimming") os campos, covertendo o password para MD5, e executando o username através da função "xss_clean", que remove dados maliciosos.

Qualquer função nativa do PHP que aceite um parâmetro pode ser usada como regra, como, por exemplo,: htmlspecialchars, trim, MD5, etc.

Nota: Geralmente você irá querer usar as funções de tratamento depois das regras de validação, pois caso exista algum erro os dados originais serão mostrados no formulário.

Callbacks(Rechamadas): Suas próprias Funções de Validação

O sistema de validação suporta callbacks(rechamadas) para suas próprias funções de validação. Isto permite que você extenda a classe de validação para suas necessidades. Por exemplo, se você precisa executar uma query no banco de dados para ver se o usuário escolheu um nome de usuário(username) único, você pode criar uma função de callback que faça isso. Vamos criar um exemplo simples.

Em seu controller, altere a regra de "username" para isto:

$rules['username'] = "callback_username_check";

Então adicone uma nova função chamada username_check ao seu controller. Aqui está como deve ficar seu controller:

```
<?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;
                }
        }

}
?>
```

Recarregue seu formulário e submeta a palavra "test" como Nome de usuário(username). Você pode ver que os dados do campo do formulário foram passados a sua função de callback(rechamada) para você processar.

Para chamar um callback apenas coloque o nome da função em uma regra com "callback_" como prefixo da regra.

A mensagem de erro foi configurada usando a função $this->validation->set_message. Lembre-se que a chave da mensagem (o primeiro parâmetro) deve coincidir com o nome de sua função.

Nota: Você pode aplicar suas próprias mensagens de erros personalizadas a qualquer regra, apenas configurando a mensagem similarmente. Por exemplo, para alterar a mensagem para a regra "required" (obrigatório) você fará isto:

$this->validation->set_message('required', 'Your custom message here');

Re-populando o formulário

Até aqui estivemos apenas lidando com erros . Chegou a hora de repopular os campos do formulário com os dados enviados. Isto é feito similarmente às regras. Adicione o seguinte código a seu controller, logo abaixo de suas regras:

```
$fields['username'] = 'Nome de usuário';
$fields['password'] = 'Senha';
$fields['passconf'] = 'Confirmação de Senha';
$fields['email'] = 'Endereço de Email';
$this->validation->set_fields($fields);
```

As chaves do array são os nomes atuais dos campos do formulário, o valor representa o nome completo que você quer mostrar na mensagem de erro.

A função index de seu controller irá se parecer com algo assim:

```
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');
	}
}
```

Agora abra seu arquivo de view myform.php e atualize o valor em cada campo de modo que cada um tenha um objeto correspondente a seu nome:

```
<html>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/validation.
html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:59 GMT -->
<head>
<title>My Form</title>
</head>
<body>
<?=$this->validation->error_string; ?>
<?=form_open('form'); ?>
<h5>Nome de usuário</h5>
<input type="text" name="username" value="<?=$this->validation->username;?>" size="50" />
<h5>Senha</h5>
<input type="text" name="password" value="<?=$this->validation->password;?>" size="50" />
<h5>Confirmação de Senha</h5>
<input type="text" name="passconf" value="<?=$this->validation->passconf;?>" size="50" />
<h5>Endereço de Email</h5>
<input type="text" name="email" value="<?=$this->validation->email;?>" size="50" />
<div><input type="submit" value="Submit" /></div>
</form>
</body>
<!--Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/libraries/validation.
html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 19:59:59 GMT -->
</html>
```

Agora recarregue sua página e envie o formulário de modo que isso irá disparar um erro. Seu formulário deverá estar populado e as mensagens de erro conterão um nome de campo mais relevante.

Exibindo Erros Individualmente

Se você preferir mostrar uma mensagem de erro próxima a cada campo do formulário, ao invés de uma lista, você pode alterar seu formulário de modo que ele fique parecido com algo assim:

```
<h5>Nome de usuário</h5>
<?=$this->validation->username_error; ?>
<input type="text" name="username" value="<?=$this->validation->username;?>" size="50" />

<h5>Senha</h5>
<?=$this->validation->password_error; ?>
<input type="text" name="password" value="<?=$this->validation->password;?>" size="50" />

<h5>Confirmação de Senha</h5>
<?=$this->validation->passconf_error; ?>
<input type="text" name="passconf" value="<?=$this->validation->passconf;?>" size="50" />

<h5>Endereço de Email</h5>
<?=$this->validation->email_error; ?>
<input type="text" name="email" value="<?=$this->validation->email;?>" size="50" />
```

Se não houverem erros, nada será mostrado. Se houver um erro, a mensagem aparecerá dentro dos delimitadores que você configurou (tag <p> por padrão).

Nota: Para exibir erros desta maneira você deve lembrar de configurar seus campos usando a função$this->validation->set_fields descrita anteriormente. Os erros serão transformados em variáveis que tenham "_error" após o nome do seu campo. Por exemplo , seu erro para "username" estará disponível em:

$this->validation->username_error.

Referência de Regras

O que segue é uma lista de todas as regras nativas que estão disponíveis para uso:

Regra	Parâmetro	descrição	Exemplo
required	Não	Retorna FALSE se o elemento do formulário estiver vazio.	
matches	Sim	Retorna FALSE se o elemento do formulário não corresponder com o elemento no parâmetro.	matches[form_item]
min_length	Sim	Retorna FALSE se o elemento do formulário for mais curto/menor que o valor do parâmetro.	min_length[6]
max_length	Sim	Retorna FALSE se o elemento do formulário for mais longo/maior que o valor do parâmetro.	max_length[12]
exact_length	Sim	Retorna FALSE se o elemento do formulário não for exatamente igual ao valor do parâmetro.	exact_length[8]
alpha	Não	Retorna FALSE se o elemento do formulário contiver alguma outra coisa que não sejam caracteres alfabéticos(letras).	
alpha_numeric	Não	Retorna FALSE se o elemento do formulário contiver alguma outra coisa que não sejam caracteres alfanuméricos.	
alpha_dash	Não	Retorna FALSE se o elemento do formulário contiver alguma outra coisa que não sejam caracteres alfanuméricos, underscores ou traços(hífen).	
numeric	Não	Retorna FALSE se o elemento do formulário contiver alguma outra coisa que não sejam caracteres numéricos(números).	
valid_email	Não	Retorna FALSE se o elemento do formulário não contiver um endereço de email válido.	
valid_ip	Não	Retorna FALSE se o IP fornecido não for válido.	

Nota: Estas regras também podem ser chamadas através de funções discretas. Por exemplo:

$this->validation->required($string);

Nota: Você também pode usar quaisquer funções nativa do PHP que permitam um parâmetro.

Referência de Tratamento de Dados

O que segue é uma lista de todas as funções de tratamento que estão disponíveis para uso:

Nome	Parâmetro	descrição
xss_clean	Não	Executa os dados através da função de filtragem XSS, descrita na página Classe Input.
prep_for_form	Não	Converte caracteres especiais de modo que dados HTML pode ser mostrados em um campo de formulário sem quebrá-lo.
prep_url	Não	Adiciona "http://" às URLs caso não tenham.
strip_image_tags	Não	Retira o código HTML das tags de imagem deixando apenas a URL bruta de endereço da mesma.
encode_php_tags	Não	Converte tags PHP em 'entities'.

Nota: Você também pode usar quaisquer funções nativa do PHP que permitam um parâmetro, comotrim, htmlspecialchars, urldecode, etc.

Configurando Mensagens de Erros Personalizadas

Todas as mensagens de erro estão localizadas no seguinte arquivo de idioma: language/english/validation_lang.php

Para configurar sua própria mensagem personalizada você pode: ou editar esse arquivo ou usar a seguinte função:

$this->validation->set_message('regra', 'Mensagem de Erro');

Onde regra corresponde ao nome de uma regra particular, e Mensagem de Erro é o texto que deverá ser exibido.

Tratando com Menus Select, Radio Buttons e Checkboxes

Se você usa menus select, radio buttons or checkboxes, irá, então, querer que o estado destes itens seja guardado quando ocorrer um erro. A classe Validação tem três funções que o auxiliam a fazer isto:

set_select()

Permite que vocÊ exiba o item do menu que foi selecionado. O primeiro parâmetro deve conter o nome do menu select, o segundo parâmetro deve conter o valor de cada item. Exemplo:

```
<select name="meuselect">
<option value="um" <?= $this->validation->set_select('meuselect', 'um'); ?> >Um</option>
<option value="dois" <?= $this->validation->set_select('meuselect', 'dois'); ?> >Dois</option>
<option value="tres" <?= $this->validation->set_select('meuselect', 'tres'); ?> >Três</option>
</select>
```

set_checkbox()

Permite que você exiba um checkbox no estado em que foi enviado. O primeiro parâmetro deve conter o nome do checkbox, o segundo parâmetro deve conter seu valor. Exemplo:

<input type="checkbox" name="meucheck" value="1" <?= $this->validation->set_checkbox('meucheck', 1); ?> />

set_radio()

Permite que você exiba radio buttons no estado em que forem enviados. O primeiro parâmetro deve conter o nome do radio button, o segundo parâmetro deve conter seu valor. Exemplo:

<input type="radio" name="meuradio" value="1" <?= $this->validation->set_radio('meuradio', 1); ?> />

Classes XML-RPC e Servidor de XML-RPC

As classes de XML-RPC do Code Igniter permitem que você envie requisições a outro servidor, ou configurar seu próprio servidor de XML-RPC para receber requisições.

O que é XML-RPC?

Simplesmente é um modo de dois computadores comunicarem-se pela internet usando XML. Um computador, que chamaremos de de cliente, envia uma requisição XML-RPC a outro computadore, que chamaremos de servidor. Uma vez que o servidore receba e processe a requisição ele enviará de volta uma resposta ao cliente.

Por exemplo, usando a API MetaWeblog, um Cliente XML-RPC (normalmente uma ferramenta de desktop publishing) enviará uma requisição para um Servidor XML-RPC rodando em seu site. Esta requisição poderia ser uma nova postagem de um weblog sendo enviada para publicação, ou poderia ser uma requisição de uma postagem editada. Quando o Servidor XML-RPC recebe esta requisição ele o examinará para determinar qual classe/método deve ser chamado para processar esta requisição. Uma vez processada, o servidor enviará de volta uma mensagem de resposta.

Para especificações detalhadas, visite o site XML-RPC.

Inicializando a Classe

Assim como a maioria das classe no Code Igniter, as classes XML-RPC e XML-RPCS são inicializadas em seu controller usando a função $this->load->library:

Para carregar a classe XML-RPC você usará:

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

Uma vez carregado, o objeto biblioteca xml-rpc estará disponível usando: $this->xmlrpc

Para carregar a classe Servidor de XML-RPC você usará:

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

Uma vez carregado, o objeto biblioteca xml-rpcs estará disponível usando: $this->xmlrpcs

Nota: Ao usar a classe Servidor de XML-RPC você deve carregar tanto a classe XML-RPC quanto a classe Servidor de XML-RPC.

Enviando Requisições XML-RPC

Para enviar uma solicitação a um servidor XML-RPC você deve especificar os seguintes dados:

- O URL do servidor

- O método no servidor que você deseja chamar

-A solicitação de dados (explicado abaixo).

Aqui está um exemplo básico que envia uma simples Weblogs.com ping para o Ping-o-Matic

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

$this->xmlrpc->server('http://rpc.pingomatic.com/', 80);
$this->xmlrpc->method('weblogUpdates.ping');

$request = array('My Photoblog', 'http://www.my-site.com/photoblog/');
$this->xmlrpc->request($request);

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

Explicação

O código acima inicializa o XML-RPC aula, conjuntos URL do servidor eo método a ser chamado (weblogUpdates.ping). O pedido (neste caso, o título eo URL de seu site) é colocada em uma matriz de transporte, e compilado com o pedido () função. Por último, o pedido é enviado completo. Se o send_ request () retorna falso método que irá exibir a mensagem de erro foi enviado de volta a partir do XML-RPC Server.

Anatomia de uma Solicitação

Uma solicitação XML-RPC é simplesmente o que você está enviando os dados para o servidor XML-RPC. Cada pedaço de dados no âmbito de um pedido se refere a um pedido como parâmetro. O exemplo acima tem dois parâmetros: o título eo URL do seu site. Quando o servidor XML-RPC recebe o seu pedido, ele irá procurar por parâmetros necessários.

Pedido parâmetros devem ser colocados em uma matriz de transporte, e cada parâmetro pode ser um dos sete tipos de dados (strings, números, datas, etc.) Se os seus parâmetros são algo diferente de cordas você terá que incluir o tipo de dados no pedido array.

Aqui está um exemplo simples de um array com três parâmetros:

```
$request = array('John', 'Doe', 'www.some-site.com');
$this->xmlrpc->request($request);
```

Se você usar outros tipos de dados do que cordas, ou se você tiver vários tipos de dados, você irá colocar cada um dos parâmetros para a sua própria matriz, com o tipo de dados na segunda posição:

```
$request = array (
            array('John', 'string'),
            array('Doe', 'string'),
            array(FALSE, 'boolean'),
            array(12345, 'int')
          );
$this->xmlrpc->request($request);
```

A secção abaixo os tipos de dados tem um lista completa de tipos de dados.

Criando um XML-RPC Server

Um XML-RPC Server funciona como uma espécie de polícia de tráfego, à espera de receber os pedidos e reorientando-os para as funções para tratamento adequado.

Para criar o seu próprio XML-RPC envolve inicialização do servidor XML-RPC Server classe em seu controlador quando você espera que o próximo pedido para aparecer, então, que cria um array com mapeamento instruções, para que receber os pedidos podem ser enviados para a classe eo método adequado para

processamento.

Aqui está um exemplo para ilustrar:

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

$config['functions']['new_post']; = array('function' => 'My_blog.new_entry');
$config['functions']['update_post'] = array('function' => 'My_blog.update_entry');

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

O exemplo acima contém um array especificando dois pedidos método permite que o servidor. Os métodos são permitidas no lado esquerdo da matriz. Quando um desses dois são recebidos, eles serão mapeados para a classe e método do lado direito.

Em outras palavras, se um cliente XML-RPC envia um pedido para que a new_post método, o servidor irá carregar o My_blog classe e chamar a new_entry função. Se o pedido é para o update_post método, o servidor irá carregar o My_blog classe e chamar a update_entry função.

A função nomes no exemplo acima, são arbitrários. Você vai decidir o que deve ser chamado em seu servidor, ou se você estiver usando APIs padronizado, como o Blogger ou MetaWeblog API, você vai usar os seus nomes de função.

Processamento servidor pedidos

Quando o XML-RPC Server receba um pedido e carrega a classe / método de tratamento, ele irá passar um objeto para que o método que contém os dados enviados pelo cliente.

Usando o exemplo acima, se o método new_post é solicitado, o servidor irá esperar uma classe de existir com este protótipo:

```
class My_blog extends Controller {
  function new_post($request)
  {

  }
}
```

A variável $request é um objeto compilado pelo servidor, que contém os dados enviados pelo cliente XML-RPC. Usando esse objeto, você terá acesso ao pedido parâmetros para que você possa processar o pedido. Quando tiver concluído irá enviar uma resposta de volta para o Cliente.

Abaixo está um exemplo do mundo real, utilizando o Blogger API. Um dos métodos, no Blogger API é getUserInfo (). Usando esse método, um XML-RPC Client Server pode enviar a um nome de usuário e senha, em contrapartida o servidor envia de volta informações sobre o usuário especial (apelido, ID do usuário, endereço de e-mail, etc.) Aqui está a forma como a função de transformação possa parecer:

```
class My_blog extends Controller {

  function getUserInfo($request)
  {
    $username = 'smitty';
    $password = 'secretsmittypass';

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

    $parameters = $request->output_parameters();

    if ($parameters['1'] != $username AND $parameters['2'] != $password)
    {
      return $this->xmlrpc->send_error_message('100', 'Invalid Access');
    }

    $response = array(array('nickname'  => array('Smitty','string'),
                'userid'   => array('99','string'),
                'url'      => array('http://yoursite.com','string'),
                'email'    => array('jsmith@yoursite.com','string'),
                'lastname'  => array('Smith','string'),
                'firstname' => array('John','string')
                ),
              'struct');

    $this->xmlrpc->send_response($response);
  }
}
```

Notas:

O output_parameters () recupera uma função array indexada corresponde ao pedido parâmetros enviados pelo cliente. No exemplo acima, os parâmetros de saída será o nome de usuário e senha.

Se o nome de usuário ea senha enviada pelo cliente não eram válidas, ea mensagem de erro é retornado usando send_error_message ().

Se a operação foi bem sucedida, o cliente será enviada de volta uma resposta array contendo as informações do usuário.

Formatar uma Resposta

Semelhante aos pedidos, respostas devem ser formatado como um array. No entanto, ao contrário de pedidos, a resposta é uma matriz que contenha um único item. Este item pode ser um array com vários outros arranjos, mas só pode haver um índice primário array. Em outras palavras, o protótipo básica é esta:

$ pedido = array ('Response dados', array ');

As respostas, no entanto, costumam conter várias peças de informação. A fim de realizar esta tarefa que temos de colocar a resposta em sua própria matriz de modo a que a principal matriz continua a conter uma única peça de dados. Aqui está um exemplo que mostra a forma como esta poderá ser realizada:

```
$request = array (
            array(
                'first_name' => array('John', 'string'),
                'last_name' => array('Doe', 'string'),
                'member_id' => array(123435, 'int'),
                'todo_list' => array(array('clean house', 'call mom', 'water plants'), 'array'),
                ),
            'struct'
            );
```

Note que a matriz acima é formatado como uma struct. Este é o tipo mais comum de dados para as respostas.

Tal como acontece com Pedidos, pode ser uma resposta a uma das sete tipos de dados listados na seção de tipos de dados.

Enviando uma resposta de erro

Se você precisar enviar ao cliente um erro resposta que você irá usar o seguinte:
regressar $ this-> xmlrpc-> send_error_message ('123 ", dados solicitados não estão disponíveis');
O primeiro parâmetro é o número de erro enquanto o segundo parâmetro é a mensagem de erro.

Criar o seu próprio cliente e servidor

Para ajudá-lo a entender tudo o que tenha coberto até agora, vamos criar uma jovem controladores que atuam como XML-RPC cliente e servidor. Você vai usar o Cliente para enviar uma solicitação para o servidor e receber uma resposta.

O Cliente

Usando um editor de texto, crie um controlador chamado xmlrpc_client.php. Nele, colocar esse código e guarde-o para seus aplicativos / controladores / pasta:

```
<?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: No código acima estamos a usar um "ajudante url". Você pode encontrar mais informações na página Funções Auxiliares.

O Servidor

Usando um editor de texto, crie um controlador chamado xmlrpc_server.php. Nele, colocar esse código e guarde-o para seus aplicativos / controladores / pasta:

```
<?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);
        }
}
?>
```

Experimente!

Agora, visite o seu site usando uma URL semelhante a esta:

www.your-site.com/index.php/xmlrpc_client/

Agora você deve ver a mensagem que você enviou para o servidor, e sua resposta de volta para você.

O cliente envia uma mensagem que você criou ("Como vai se passa?") Para o servidor, junto com um reqest para o "Saudações" método. O servidor recebe o pedido e mapas-a ao "processo" função, no caso de a resposta é enviada de volta.

XML-RPC função referência

$this->xmlrpc->server()

Define o URL e o número da porta do servidor para o qual um pedido está a ser enviada:

$this->xmlrpc->server('http://www.sometimes.com/pings.php', 80);

$this->xmlrpc->timeout()

Defina um tempo de descanso período (em segundos) após o qual o pedido será cancelado:

$ this-> xmlrpc-> timeout (6);

$this->xmlrpc->method()

Define o método que será solicitada a partir do servidor XML-RPC:

$ this-> xmlrpc-> método ('método');

Quando método é o nome do método.

$ this-> xmlrpc-> pedido ()

Toma um conjunto de dados e constrói pedido para ser enviada ao servidor XML-RPC:

```
$request = array(array('My Photoblog', 'string'), 'http://www.yoursite.com/photoblog/');
$this->xmlrpc->request($request);
```

$ this-> xmlrpc-> send_request ()

O pedido enviando função. Retorna boolean TRUE ou FALSE baseada no sucesso para a falha, permitindo que seja usado condicionalmente.

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

Habilita depuração, que irá mostrar uma variedade de informações e dados úteis erro durante o desenvolvimento.

$ this-> xmlrpc-> display_error ()

Retorna uma mensagem de erro como uma string caso o seu pedido falhou por algum motivo.

echo $ this-> xmlrpc-> display_error ();

$ this-> xmlrpc-> display_response ()

Retorna a resposta do servidor remoto uma vez pedido é recebido. A resposta vai ser tipicamente um array associativo.

$ this-> xmlrpc-> display_response ();

$ this-> xmlrpc-> send_error_message ()

Essa função permite que você envie uma mensagem de erro do seu servidor para o cliente. Primeiro parâmetro é o número de erro enquanto o segundo parâmetro é a mensagem de erro.

return $this->xmlrpc->send_error_message('123', 'Requested data not available');

$ this-> xmlrpc-> send_response ()

Permite-lhe enviar a resposta do seu servidor para o cliente. Uma matriz de valores válidos dados devem ser enviados com esse método.

```
$response = array(
        array(
            'flerror' => array(FALSE, 'boolean'),
            'message' => "Thanks for the ping!")
          )
        'struct');
return $this->xmlrpc->send_response($response);
```

Tipos de Dados

De acordo com a especificação XML-RPC, existem sete tipos de valores que poderá ser enviado via XML-RPC:

- int or i4
- boolean
- string
- double
- dateTime.iso8601
- base64
- struct (contains array of values)
- array (contains array of values)

Classe codificação ZIP

Código Igniter a classe Codificação Zip permitem que você crie arquivos Zip. Arquivos podem ser baixados para o seu desktop ou gravados em um diretório.

Inicialização da classe

Tal como a maioria dos outros ramos do Código Igniter, o CEP em sua classe é inicializado utilizando o controlador $ this-> load-> biblioteca função:

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

Uma vez carregado, o objeto Zip biblioteca estará disponível usando: $ this-> zip

Exemplo de uso

Este exemplo demonstra como a comprimir um ficheiro, guarde-o para uma pasta no seu servidor, e baixá-lo para o seu desktop.

```
$name = 'mydata1.txt';
$data = 'A Data String!';
$this->zip->add_data($name, $data);
// Write the zip file to a folder on your server. Name it "my_backup.zip"
$this->zip->archive('/path/to/directory/my_backup.zip');

// Download the file to your desktop. Name it "my_backup.zip"
$this->zip->download('my_backup.zip');
```

Function Reference

$this->zip->add_data()

Permite-lhe adicionar os dados para o arquivo Zip. O primeiro parâmetro deve conter o nome que você gostaria dado ao processo, o segundo parâmetro deve conter os dados do arquivo como uma string:

```
$name = 'my_bio.txt';
$data = 'I was born in an elevator...';

$this->zip->add_data($name, $data);
```

Você é permitido várias chamadas para esta função, a fim de acrescentar vários ficheiros para o seu arquivo. Exemplo:

```
$name = 'mydata1.txt';
$data = 'A Data String!';
$this->zip->add_data($name, $data);

$name = 'mydata2.txt';
$data = 'Another Data String!';
$this->zip->add_data($name, $data);
```

Ou você pode passar vários arquivos usando um array:

```
$data = array(
          'mydata1.txt' => 'A Data String!',
          'mydata2.txt' => 'Another Data String!'
        );

$this->zip->add_data($data);

$this->zip->download('my_backup.zip');
```

Se você gostaria que seu comprimido dados organizados em sub-pastas, incluir o caminho como parte do nome de arquivo:

```
$name = 'personal/my_bio.txt';
$data = 'I was born in an elevator...';

$this->zip->add_data($name, $data);
```

O exemplo acima my_bio.txt terá lugar dentro de uma pasta chamada pessoal.

$ this-> zip-> add_dir ()

Permite-lhe adicionar um diretório. Normalmente esta função é desnecessária uma vez que você pode colocar os seus dados em pastas quando utiliza $ this-> zip-> add_data (), mas se você gostaria de criar uma pasta vazia você pode fazê-lo. Exemplo:

$ this-> zip-> add_dir ('myfolder'); / / Cria uma pasta chamada "myfolder"

$ this-> zip-> read_file ()

Permite que você comprima um arquivo que já existe algum lugar no seu servidor. Abastecimento um caminho de arquivo zip e da turma vai lê-lo e adicioná-lo ao arquivo:

```
$path = '/path/to/photo.jpg';

$this->zip->read_file($path);

// Download the file to your desktop. Name it "my_backup.zip"
$this->zip->download('my_backup.zip');
```

Se você quiser o arquivo Zip para manter a estrutura de diretórios de arquivos está em, passar TRUE (boolean) no segundo parâmetro. Exemplo:

```
$path = '/path/to/photo.jpg';

$this->zip->read_file($path, TRUE);

// Download the file to your desktop. Name it "my_backup.zip"
$this->zip->download('my_backup.zip');
```

No exemplo acima, photo.jpg serão colocadas dentro de duas pastas: path/to/

$ this-> zip-> read_dir ()

Permite que você comprima uma pasta (eo seu conteúdo) que já existe algum lugar no seu servidor. Abastecimento um caminho de arquivo para o diretório e da classe zip irá recursivamente lê-lo e recriá-la como um arquivo Zip. Todos os arquivos contidos dentro do caminho será fornecido codifica, como se qualquer sub-pastas contidas dentro dela. Exemplo:

```
$path = '/path/to/your/directory/';

$this->zip->read_dir($path);

// Download the file to your desktop. Name it "my_backup.zip"
$this->zip->download('my_backup.zip');
```

$ this-> zip-> archive ()

Escreve o arquivo Zip-codificado para um diretório em seu servidor. Enviar um servidor válido caminho que termina no nome do arquivo. Verifique se o diretório é gravável (666 ou 777 é normalmente OK). Exemplo:

$this->zip->archive('/path/to/folder/myarchive.zip'); / / Cria um arquivo chamado myarchive.zip

$ this-> zip-> download ()

Faz com que o arquivo Zip para ser transferido para o seu servidor. A função deve ser passado o nome que você gostaria que o arquivo zip chamado. Exemplo:

$ this-> zip-> download ('latest_stuff.zip'); / / arquivo será nomeado "latest_stuff.zip"

Nota: Não exibir quaisquer dados no âmbito do controlador em que você chamar esta função, uma vez que envia várias servidor cabeçalhos que causam o download do arquivo de acontecer e de ser tratado como binário.

$ this-> zip-> get_zip ()

Retorna o arquivo compactado Zip-dados. Normalmente você não precisará desta função a não ser que você quer fazer alguma coisa única com os dados. Exemplo:

```
$name = 'my_bio.txt';
$data = 'I was born in an elevator...';

$this->zip->add_data($name, $data);

$zip_file = $this->zip->get_zip();
```

$ this-> zip-> clear_data ()

O Zip classe caches seu zip dados para que ele não precisa de recompilar o arquivo Zip para cada função que você usa acima. Se, no entanto, é necessário criar múltiplas Fecha, cada um com dados diferentes, você pode limpar a cache entre chamadas. Exemplo:

```
$name = 'my_bio.txt';
$data = 'I was born in an elevator...';

$this->zip->add_data($name, $data);
$zip_file = $this->zip->get_zip();

$this->zip->clear_data();

$name = 'photo.jpg';
$this->zip->read_file("/path/to/photo.jpg"); // Read the file's contents


$this->zip->download('myphotos.zip');
```

Classe banco de dados

Código Igniter vem com uma full-featured e muito rápido captadas classe base que suporta tanto as estruturas tradicionais e Active Record padrões. A base de dados oferecem funções claras, simples de sintaxe.

Banco de Dados início: exemplo de código

A página seguinte contém exemplo de código mostrando como a base de dados classe é usada. Para maiores detalhes leia as páginas individuais descrevendo cada função.

Inicializando a classe banco de dados

O código a seguir carrega e inicia o banco de dados baseados em sua classe configurações:

$this->load->database();

Uma vez carregada a classe está pronto para ser usado como descrito abaixo.

Nota: Se todas as suas páginas requerem acesso banco de dados você pode se conectar automaticamente.

Veja a página ligar para mais detalhes.

Standard query com vários resultados (versão Object)

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

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

echo 'Total Results: ' . $query->num_rows();
```

O resultado acima result() retorna um array de objetos. Exemplo: $ row-> título

Standard query com vários resultados (versão Array)

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

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

O acima result_array() função retorna um array de array padrão índices. Exemplo: $row ['título']

Testes de Resultados

Se você executar consultas que não poderia produzir um resultado, você é incentivado a ensaio para um resultado usando o primeiro num_rows () Função:

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

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

Standard Query com o único resultado

```
$query = $this->db->query('SELECT name FROM my_table LIMIT 1');

$row = $query->row();
echo $row->name;
```

A função row() retorna um objeto. Exemplo: $ row-> name

Com o único resultado Standard Query (versão Array)

```
$query = $this->db->query('SELECT name FROM my_table LIMIT 1');

$row = $query->row_array();
echo $row->['name'];
```

O acima row_array () função retorna um array. Exemplo: $ row-> ['name']

Inserir Standard

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

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

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

Active Record Query

O Active Record Padrão dá-lhe uma simplificação dos meios de recuperar dados:

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

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

O acima get () recupera todas as função dos resultados a partir da tabela fornecida. O Active Record classe contém um elogio completo de funções para trabalhar com os dados.

Active Record inserir

```
$data = array(
        'title' => $title,
        'name' => $name,
        'date' => $date
       );

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

// Produces: INSERT INTO mytable (title, name, date) VALUES ('{$title}', '{$name}', '{$date}')
```

Configuração de banco de dados

Código Igniter tem um ficheiro de configuração que permite que você armazene seu banco de dados conexão valores (nome de usuário, senha, nome de dados, etc.) O arquivo de configuração está localizado em:

application/config/database.php

As definições de configuração são armazenadas em um array multi-dimensional com este protótipo:

```
$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']['active_r'] = TRUE;
```

A razão para utilizar um array multi-dimensional, em vez de uma maneira mais simples é a de permitir que você opcionalmente armazenar vários conjuntos de ligação valores. Se, por exemplo, você executar múltiplos ambientes (desenvolvimento, produção, testes, etc) ao abrigo de uma única instalação, você pode configurar uma conexão para cada grupo e, em seguida, alternar entre os grupos, conforme necessário. Por exemplo, a criação de um "teste" ambiente que você pode fazer isso:

```
$db['test']['hostname'] = "localhost";
$db['test']['username'] = "root";
$db['test']['password'] = "";
$db['test']['database'] = "database_name";
$db['test']['dbdriver'] = "mysql";
$db['test']['dbprefix'] = "";
$db['test']['pconnect'] = TRUE;
$db['test']['db_debug'] = FALSE;
$db['test']['active_r'] = TRUE;
```

Em seguida, para o sistema globalmente dizer que a utilização grupo, você poderá criar essa variável localizada no ficheiro de configuração:

active_group $ = "test";

Nota: O nome "teste" é arbitrário. Pode ser o que quiser. Por padrão temos usado a palavra "padrão" para a conexão primária, mas ela também pode ser renomeado para algo mais relevante para o seu projeto.

Explicação de Valores:

hostname - O nome do host do seu banco de dados. Muitas vezes isto é "localhost".

username - O nome de usuário usado para conectar ao banco de dados.

password - A senha usada para conectar ao banco de dados.

database - O nome do banco de dados que deseja ligar.

dbdriver - A base de dados tipo. ou seja: mysql, POSTGRE, obdc, etc devem ser especificados em minúsculas.

dbprefix - Uma mesa opcional prefixo que será adicionado ao nome da tabela quando correr Active Record buscas. Isto permite múltiplas Código Igniter instalações de partilhar uma base de dados.

pconnect - verdadeiro / falso (boolean) - a possibilidade de utilizar uma conexão persistente.

db_debug - verdadeiro / falso (boolean) - Quer erros de dados deve ser exibido.

active_r - verdadeiro / falso (boolean) - Quer a carga do Active Record Class. Se você não estiver usando o registro ativo classe você pode ter se omitido quando o banco de dados aulas são inicializadas a fim de utilizar menos recursos.

port - A base de dados número da porta. Actualmente, apenas utilizado com o POSTGRE condutor.

Nota: Dependendo do que você está usando plataforma de dados (MySQL, POSTGRE, etc) todos os valores não serão necessários. Por exemplo, quando se utilizam SQLite não será necessário fornecer um nome de usuário ou senha, eo nome do banco de dados será o caminho para o seu arquivo de dados. As informações acima assume que você está usando o MySQL.

Conectando a sua base de dados

Existem duas formas de se conectar a um banco de dados:

Ligar automaticamente

O "auto conectar" recurso irá carregar e exemplificam a base de dados com todas as páginas classe carga. Para ativar a "auto conexão", adicionar a palavra banco de dados para o núcleo array, como indicado na seguinte arquivo:

application/config/autoload.php

Conectar manualmente

Se apenas algumas das suas páginas requerem conectividade de dados manualmente você pode conectar ao seu banco de dados através da adição desta linha de código em qualquer função onde ela é necessária, ou em sua classe construtor de tornar a base de dados disponível a nível mundial em que a classe.

$ this-> load-> database ();

Se a função acima não contém qualquer informação no primeiro parâmetro ele irá conectar-se ao grupo em seu banco de dados especificado o arquivo de configuração. Para a maioria das pessoas, este é o método preferido de uso.

O primeiro parâmetro desta função pode opcionalmente ser utilizado para especificar um grupo particular de dados a partir do seu arquivo config, ou você pode até apresentar valores para a conexão de uma base de dados que não está especificado em seu arquivo de configuração. Exemplos:

Para escolher um grupo específico de seu arquivo config você pode fazer isto:

$ this-> load-> database ('group_name');

Quando group_name é o nome da conexão de seu grupo arquivo de configuração.

Para conectar manualmente para um banco de dados desejados você pode passar um conjunto de valores:

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

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

Para obter informações sobre cada um desses valores, por favor veja a página de configuração.

Ou você pode enviar sua base de dados valores como uma fonte de dados Nome. DSNs devem ter este protótipo:

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

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

Note que se você usar um DSN que você não será capaz de indicar alguns dos valores padrão como você pode se usar uma conexão array.

Conectando-se a múltiplas bases de dados

Se você precisa se conectar a mais do que um banco de dados em simultâneo, pode fazê-lo do seguinte modo:

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

Nota: Alterar as palavras "group_one" e "group_two" para o grupo específico que você está se conectando a nomes (ou você pode passar a ligação valores como indicado acima).

Ao definir o segundo parâmetro para TRUE (boolean) a função irá retornar o banco de dados objeto.

```
Quando você ligar, desta forma, você irá usar o seu nome ao objeto emitir comandos mais do que a sintaxe usada durante todo este guia. Em outras palavras, em vez de emitir comandos:
$this->db->query();
$this->db->result();
etc...

You will instead use:

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

Query

$ this-> db-> query ();

Para enviar uma consulta, use as seguintes funções:

$ this-> db-> query ("a sua consulta AQUI ');

A função query() retorna um banco de dados objeto resultado quando a "ler" tipo buscas são feitas, que você pode usar para mostrar seus resultados. Quando "escrever" tipo buscas são feitas é simplesmente retorna true ou false dependendo do êxito ou fracasso. Ao recuperar os dados que você normalmente atribuir a consulta para a sua própria variável, como este:

$ query = $ this-> db-> query ("a sua consulta AQUI ');

$ this-> db-> simple_query ();

Esta é uma versão simplificada da função $this-> db-> query (). Ele só retorna verdadeiro / falso, sobre o sucesso ou fracasso. Não retorna um banco de dados resultado conjunto, nem definir o temporizador consulta, ou vincular compilar dados, ou guardar a sua consulta para depuração. Ela simplesmente permite-lhe enviar uma consulta. A maioria dos usuários raramente utilize esta função.

Escape de Consultas

É uma boa prática de segurança para escapar os seus dados antes de enviá-lo em seu banco de dados. Código Igniter tem duas funções que ajudam você a fazer isso:

$ this-> db-> escape () Esta função determina o tipo de dados para que possa escapar apenas string dados. Ele também adiciona automaticamente aspas simples em torno dos dados, para que você não tem que:

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

$ this-> db-> escape_str () Esta função escapa os dados passaram a ela, independentemente do tipo. A maior parte do tempo você vai usar a função acima esta um pouco depois. Use a função desta forma:

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

Query Ligações

Bindings permitem-lhe simplificar a sua consulta sintaxe ao deixar o sistema de colocar a consultas entre si para si. Considere o seguinte exemplo:

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

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

Os pontos de interrogação na consulta são substituídos automaticamente com os valores na tabela, no segundo parâmetro da função Consulta.

O benefício da utilização da liga secundária é que os valores são automaticamente escapou, produzindo mais segura buscas. Você não tem que se lembrar de manualmente escapar dados; que é que o motor automaticamente para você.

Gerando Resultados de Uma Query

Existem várias formas de gerar resultados query:

Result ()

Esta função retorna a consulta como resultado de uma série de objetos, ou um array vazio em fracasso. Normalmente você vai usar isso em um loop foreach, como este:

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

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

A função acima é um apelido de result_object ().

Se você executar consultas que não poderia produzir um resultado, você é incentivado a testar o primeiro resultado:

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

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

result_array()

Esta função retorna o resultado como uma pura query array, ou FALSE em fracasso. Normalmente você vai usar isso em um loop foreach, como este:

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

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

row()

Esta função retorna um único resultado row. Se a sua consulta tem mais de uma linha, ele retorna apenas a primeira row. O resultado é retornado como um objeto. Aqui está um exemplo de utilização:

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

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

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

Se você quiser uma linha específica retornou você pode enviar a linha número como um dígito no primeiro parâmetro:

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

row_array ()

Idêntico ao da linha acima (), exceto que retorna um array. Exemplo:

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

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

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

Se você quiser uma linha específica retornou você pode enviar a linha número como um dígito no primeiro parâmetro:

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

Além disso, você pode andar forward/backwards/first/last através de seus resultados usando estas variações:

$row = $query->first_row()

$row = $query->last_row()

$row = $query->next_row()

$row = $query->previous_row()

Por padrão ela retornar um objeto a menos que você coloque a palavra "array" no parâmetro:

$row = $query->first_row('array')

$row = $query->last_row('array')

$row = $query->next_row('array')

$row = $query->previous_row('array')

Resultado Helper funções

$ query-> num_rows ()

O número de linhas retornadas pela consulta. Nota: Neste exemplo, $ query é a variável que a consulta resultado é atribuído ao objeto:

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

$ query-> num_fields ()

O número de campos (columns) retornado pela consulta. Certifique-se de chamar a função utilizando a sua consulta resultar objeto:

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

$ query-> free_result ()

Ela libera a memória associada com o resultado e suprime o resultado dos recursos ID. Normalmente PHP libera sua memória automaticamente no final do script execução. No entanto, se você estiver executando uma série de consultas, em particular um script você pode querer libertar o resultado depois de cada consulta resultado foi gerado, a fim de reduzir os consumos de memória. Exemplo:

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

foreach ($query->result() as $row)
{
  echo $row->title;
}
$query->free_result(); // The $query result object will no longer be available

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

$row = $query2->row();
echo $row->name;
$query2->free_result(); // The $query2 result object will no longer be available
```

Query Helper funções

$ this-> db-> insert_id ()

O número de ID de inserir dados no desempenho insere.

$ this-> db-> affected_rows ()

Mostra o número de linhas afetadas, quando fazendo "escrever" tipo queries (inserir, atualizar, etc.)

Nota: No MySQL "DELETE FROM TABLE" retorna 0 linhas afectadas. A base de dados classe tem um pequeno truque que lhe permite voltar a correcta do número de linhas afetadas. Por padrão esse truque está ativado, mas ele pode ser desligado no banco de dados condutor arquivo.

$ this-> db-> count_all ();

Permite-lhe determinar o número de linhas em uma determinada tabela. Enviar o nome da tabela no

primeiro parâmetro. Exemplo:

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

// Produces an integer, like 25
```

$ this-> db-> platform ()

Saídas a base de dados plataforma que está sendo executada (MySQL, MS SQL, POSTGRE, etc ..):

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

$this->db->version()

Saídas a base de dados versão que você está executando:

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

$this->db->last_query();

Retorna a última consulta que foi executado (a string de consulta, e não o resultado). Exemplo:

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

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

Os dois seguintes funções ajudam a simplificar o processo de escrita de dados inserções e atualizações.

$this->db->insert_string();

Esta função simplifica o processo de escrita inserção de dados. Ele retorna uma string inserir SQL formatado corretamente. Exemplo:

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

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

O primeiro parâmetro é o nome da tabela, o segundo é um array associativo com os dados que devem ser inseridos. O exemplo acima produz:

INSERT INTO table_name (name, email, url) VALUES ('Rick', 'rick@your-site.com', 'www.your-site.com')

Nota: Os valores são automaticamente escapou, produzindo mais segura buscas.

$this->db->update_string();

Esta função simplifica o processo de escrita database updates. Ele retorna uma string atualização SQL formatado corretamente. Exemplo:

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

$where = "author_id = 1 AND status = 'active'";

$str = $this->db->update_string('table_name', $data, $where);
```

O primeiro parâmetro é o nome tabela, o segundo é um array associativo com os dados que devem ser inseridos, eo terceiro parâmetro é o "onde" cláusula. O exemplo acima produz:

UPDATE exp_weblog SET name = 'Rick', email = 'rick@your-site.com', url = 'www.your-site.com' WHERE author_id = 1 AND status = 'active'

Nota: Os valores são automaticamente escapou, produzindo mais segura buscas.

Active Record classe

Código Igniter usa uma versão modificada do Active Record Database Padrão. Este padrão permite a informação a ser recuperada, inserido, e atualizados em seu banco de dados com o mínimo de scripting. Em alguns casos, apenas uma ou duas linhas de código são necessários para a execução de uma base de dados acção. Código Igniter não exige que cada banco de dados será o seu próprio quadro classe arquivo. É invés fornece uma interface mais simplificada.

Para além simplicidade, um grande benefício para a utilização do Active Record características é que ele permite que você crie banco de dados independente aplicações, desde a consulta sintaxe é gerada por cada banco de dados adaptador. Permite também mais seguro para consultas, uma vez que os valores são escapou automaticamente pelo sistema.

Nota: Se você tem a intenção de escrever as suas próprias consultas você pode desativar esta classe em seu banco de dados config arquivo, permitindo que o banco central biblioteca e adaptador para utilizar menos recursos.

Selecting Data

As seguintes funções permitem que você crie SQL SELECT declarações.

Nota: Se você estiver usando o PHP 5 você pode utilizar o método de encadeamento mais compacto sintaxe. Esta situação é descrita no final da página.

$ this-> db-> get ();

Executada a selecção de consulta e retorna o resultado. Pode ser usado por si só para recuperar todos os registros de uma tabela:

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

// Produces: SELECT * FROM mytable
```

O segundo e terceiro parâmetros permitir que você faz fixar um limite e offset cláusula:

```
$query = $this->db->get('mytable', 10, 20);

// Produces: SELECT * FROM mytable LIMIT 20, 10 (in MySQL. Other databases have slightly different syntax)
```

Você verá que o que precede função é atribuída a uma variável chamada $ query, que pode ser usada para mostrar os resultados:

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

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

Por favor, visite a página de funções resultado de uma ampla discussão quanto resultado geração.

$this->db->getwhere();

Idêntico ao da função acima exceto que ela permite que você adicione um "where" cláusula no segundo parâmetro, em vez de usar o db-> where () Função:

$ query = $ this-> db-> getwhere ('mytable', array (id => $ id), US $ limite de US $ offset);

Por favor leia sobre o caso da função abaixo para obter mais informações.

$this->db->select();

Permite-lhe escrever o SELECT porção de sua consulta:

```
$this->db->select('title, content, date');
$query = $this->db->get('mytable');
// Produces: SELECT title, content, date FROM mytable
```

*Nota: Se você estiver selecionando todos (*) a partir de uma tabela que você não precisa usar esta função. Quando omitido, Código Igniter assume que você deseja SELECT **

$this->db->from();

Permite-lhe escrever a partir porção de sua consulta:

```
$this->db->select('title, content, date');
$this->db->from('mytable');
$query = $this->db->get();
// Produces: SELECT title, content, date FROM mytable
```

Nota: Tal como demonstrado anteriormente, a partir porção de sua consulta pode ser especificado no $this-> db-> get (), então use independentemente do método que preferir.

$ this-> db-> join ();

Permite-lhe escrever o JOIN porção de sua consulta:

```
$this->db->select('*');
$this->db->from('blogs');
$this->db->join('comments', 'comments.id = blogs.id');
$query = $this->db->get();
// Produces:
// SELECT * FROM blogs
// JOIN comments ON comments.id = blogs.id
```

Múltipla função chamadas podem ser feitas se você precisa de vários junta em uma consulta.

Se você precisar de alguma coisa que não um natural JOIN você pode especificá-lo através do terceiro parâmetro da função. As opções são: esquerda, direita, exterior, interior, exterior à esquerda, direita e exterior.

```
$this->db->join('comments', 'comments.id = blogs.id', 'left');
// Produces: LEFT JOIN comments ON comments.id = blogs.id
```

$ this-> db-> where ();

Essa função permite que você defina cláusulas WHERE utilizando uma das quatro modalidades:

Nota: Todos os valores passados para essa função são escapou automaticamente, produzindo mais segura buscas.

1. Simple chave / valor patrimonial (Simple key/value method):

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

// Produces: WHERE name = 'Joe'
```

Repare que a igualdade de sinal é adicionado para você.

Se você usar múltiplas funções que serão chamadas encadeada juntamente AND entre eles:

```
$this->db->where('name', $name);
$this->db->where('title', $title);
$this->db->where('status', $status);

// WHERE = 'Joe' AND title = 'boss' AND status = 'active'
```

2. Custom chave / valor patrimonial (Custom key/value method):

Você pode incluir um operador no primeiro parâmetro, a fim de controlar a comparação:

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

// Produces: WHERE name != 'Joe' AND id < 45
```

3. Array associativo método (Associative array method):

```
$array = array('name' => $name, 'title' => $title, 'status' => $status);
$this->db->where($array);

// Produces: WHERE name = 'Joe' AND title = 'boss' AND status = 'active'
```

Você pode incluir seus próprios operadores que utilizam este método também:

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

4. Custom texto (Custom string):

Você pode escrever as suas próprias cláusulas manualmente:

```
$where = "name='Joe' AND status='boss' OR status='active'";
$this->db->where($where);
```

$ this-> db-> orwhere ();

Esta função é idêntica à anterior, excepto as várias instâncias estão ligadas por OR:

```
$this->db->where('name !=', $name);
$this->db->orwhere('id >', $id);
// Produces: WHERE name != 'Joe' OR id > 50
```

$this->db->like();

Essa função permite-lhe gerar LIKE cláusulas, útil para fazer pesquisas.

Nota: Todos os valores passados para essa função são automaticamente escapou.

1. Simple chave / valor patrimonial (Simple key/value method):

```
$this->db->like('title', $match);
// Produces: WHERE title LIKE '%match%'
```

Se você usar múltiplas funções que serão chamadas encadeada juntamente AND entre eles::

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

// WHERE title LIKE '%match%' AND body LIKE '%match%'
```

2. Array associativo método (Associative array method):

```
$array = array('title' => $match, 'page1' => $match, 'page2' => $match);
$this->db->like($array);
// WHERE title LIKE '%match%' AND page1 LIKE '%match%' AND page2 LIKE '%match%'
```

$ this-> db-> orlike ();

Esta função é idêntica à anterior, excepto as várias instâncias estão ligadas por OR:

```
$this->db->like('title', $match);
$this->db->orlike('body', $match);
// WHERE title LIKE '%match%' OR body LIKE '%match%'
```

$this->db->groupby();

Permite-lhe escrever o GROUP BY porção de sua consulta:

```
$this->db->groupby("title");
// Produces: GROUP BY title
```

Você também pode passar uma matriz de múltiplos valores tão bem:

```
$this->db->groupby(array("title", "date");
// Produces: GROUP BY title, date
```

$this->db->having();

Permite-lhe escrever o HAVING porção de sua consulta:

```
$this->db->having('user_id = 45');
// Produces: HAVING 'user_id = 45'
```

Você também pode passar uma matriz de múltiplos valores tão bem:

```
$this->db->having(array('title =' => 'My Title', 'id <' => $id));
// Produces: HAVING title = 'My Title', 'id < 45'
```

$this->db->orderby();

Permite-lhe definir uma cláusula ORDER BY. O primeiro parâmetro contém o nome da coluna que você gostaria de por fim. O segundo parâmetro permite-lhe definir o rumo do resultado. As opções são asc ou desc ou RAND ().

```
$this->db->orderby("title", "desc");
// Produces: ORDER BY title DESC
```

Você também pode passar a sua própria string no primeiro parâmetro:

```
$this->db->orderby('title desc, name asc');
// Produces: ORDER BY title DESC, name ASC
```

$this->db->limit();

Permite limitar o número de linhas que você gostaria retornado pela consulta:

```
$this->db->limit(10);
// Produces: LIMIT 10
```

O segundo parâmetro permite que você defina um resultado compensado.

```
$this->db->limit(10, 20);
// Produces: LIMIT 20, 10 (in MySQL. Other databases have slightly different syntax)
```

$ this-> db-> count_all ();

Permite-lhe determinar o número de linhas em uma determinada tabela. Enviar o nome da tabela no primeiro parâmetro. Exemplo:

```
echo $this->db->count_all('my_table');
// Produces an integer, like 25
```

Inserting Data

$this->db->insert();

Gera inserir uma string com base nos dados que você fornecer, e executa a consulta. Você pode passar um array ou um objeto para a função. Aqui está um exemplo usando um array:

```
$data = array(
        'title' => 'My title' ,
        'name' => 'My Name' ,
        'date' => 'My date'
      );
$this->db->insert('mytable', $data);
// Produces: INSERT INTO mytable (title, name, date) VALUES ('My title', 'My name', 'My date')
```

O primeiro parâmetro irá conter o nome da tabela, o segundo é um array associativo de valores.

Aqui está um exemplo usando um objeto:

```
/*
   class Myclass {
      var $title = 'My Title';
      var $content = 'My Content';
      var $date = 'My Date';
   }
*/
$object = new Myclass;
$this->db->insert('mytable', $object);
// Produces: INSERT INTO mytable (title, content, date) VALUES ('My Title', 'My Content', 'My Date')
```

O primeiro parâmetro irá conter o nome da tabela, o segundo é um array associativo de valores.

Nota: Todos os valores são automaticamente escapou produzindo mais segura buscas.

$this->db->set();

Esta função lhe permite definir valores para inserções ou atualizações.

Pode ser usado em vez de passar um array dados diretamente para a inserir ou atualizar funções:

```
$this->db->set('name', $name);
$this->db->insert('mytable');
// Produces: INSERT INTO mytable (name) VALUES ('{$name}')
```

Se você usar vários função chamada eles serão montados corretamente baseada em saber se você está fazendo uma inserção ou de uma actualização:

```
$this->db->set('name', $name);
$this->db->set('title', $title);
$this->db->set('status', $status);
$this->db->insert('mytable');
```

Você também pode passar um array associativo para esta função:

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

Ou um objeto:

```
/*
   class Myclass {
      var $title = 'My Title';
      var $content = 'My Content';
      var $date = 'My Date';
   }
*/
$object = new Myclass;
$this->db->set($object);
$this->db->insert('mytable');
```

Updating Data

$this->db->update();

Gera uma atualização string e roda a consulta na base de dados que você fornecer. Você pode passar um array ou um objeto para a função. Aqui está um exemplo usando um array:

```
$data = array(
          'title' => $title,
          'name' => $name,
          'date' => $date
        );
$this->db->where('id', $id);
$this->db->update('mytable', $data);
// Produces:
// UPDATE mytable
// SET title = '{$title}', name = '{$name}', date = '{$date}'
// WHERE id = $id
```

Ou você pode fornecer uma objeto:

```
/*
   class Myclass {
      var $title = 'My Title';
      var $content = 'My Content';
      var $date = 'My Date';
   }
*/
$object = new Myclass;
$this->db->where('id', $id);
$this->db->update('mytable', $object);
// Produces:
// UPDATE mytable
// SET title = '{$title}', name = '{$name}', date = '{$date}'
// WHERE id = $id
```

Nota: Todos os valores são automaticamente escapou produzindo mais segura buscas.

Você pode observar a utilização dos $this-> db-> where (), permitindo-lhe definir o WHERE. Você pode opcionalmente passar essas informações diretamente na atualização funcionar como uma string:

$this->db->update('mytable', $data, "id = 4");

Ou como um array:

$this->db->update('mytable', $data, array('id' => $id));

Você também pode usar o $this-> db-> set () função acima descrito quando realizar atualizações.

Deleting Data

$this->db->delete();

Gera uma string SQL apagar e executa a consulta.

```
$this->db->delete('mytable', array('id' => $id));
// Produces:
// DELETE FROM mytable
// WHERE id = $id
```

O primeiro parâmetro é o nome da tabela, onde a segunda é a cláusula. Você também pode usar the where() or orwhere() funciona em vez de cruzar os dados para o segundo parâmetro da função:

```
$this->db->where('id', $id);
$this->db->delete('mytable');
// Produces:
// DELETE FROM mytable
// WHERE id = $id
```

Nota: Todos os valores são automaticamente escapou produzindo mais segura buscas.

Método encadeamento (Method Chaining)

Método de encadeamento lhe permite simplificar sua sintaxe, ligando várias funções. Considere este exemplo:

```
$this->db->select('title')->from('mytable')->where('id', $id)->limit(10, 20);
$query = $this->db->get();
```

Nota: Método encadeamento só funciona com o PHP 5.

Transações (Transactions)

Código Igniter banco de dados do abstração permite a você usar transações com bases de dados que suportam a operação segura tabela tipos. No MySQL, você precisa estar executando InnoDB ou BDB sim, então a tabela tipos mais comuns MyISAM. A maior parte das outras plataformas apoiar operações de dados nativa.

Se você não está familiarizado com operações recomendamos que você encontrar um bom recurso on-line para aprender sobre eles para o seu banco de dados particular. As informações a seguir assume que você tenha um entendimento básico de operações.

Código Igniter a abordagem da Transações

Código Igniter utiliza uma abordagem de transações que é muito semelhante ao processo utilizado pela classe ADODB base popular. Temos que a abordagem escolhida, uma vez que simplifica muito o processo de executar operações. Na maioria dos casos, tudo o que é exigido são duas linhas de código.

Tradicionalmente, as operações têm exigido um número razoável de trabalho para implementar uma vez que exigem que você se manter a par das suas consultas e determinar se a cometer ou repetir a basear-se o sucesso ou o fracasso de suas buscas. Isto é particularmente pesado com nested queries. Em contrapartida, temos implementado um sistema inteligente transacção que é que isto automaticamente para você (você também pode gerenciar suas transações se você optar por manualmente, mas há realmente nenhuma vantagem).

Executando Transações

Para executar suas consultas usando transações que você irá usar o $this->db->trans_start() e $this->db->trans_complete() funciona como segue:

```
$this->db->trans_start();
$this->db->query('AN SQL QUERY...');
$this->db->query('ANOTHER QUERY...');
$this->db->query('AND YET ANOTHER QUERY...');
$this->db->trans_complete();
```

Você pode executar tantas as perguntas que você quer entre o início / funções completas e eles serão todos empenhados ou retirada baseado em sucesso ou fracasso de uma determinada consulta.

Gerenciando Erros

Se você tiver ativado em seu relato erro config / database.php arquivo que você verá uma mensagem de erro se a norma cometer foi vencida. Se depuração é desligado, você pode gerenciar seus próprios erros como este:

```
$this->db->trans_start();
$this->db->query('AN SQL QUERY...');
$this->db->query('ANOTHER QUERY...');
$this->db->trans_complete();
if ($this->db->trans_status() === FALSE)
{
   // generate an error... or use the log_message() function to log your error
}
```

A ativação Transações

As transacções são ativadas automaticamente o momento em que você usar $ this-> db-> trans_start (). Se você quiser desativar transações você pode fazer isso usando $ this-> db-> trans_off ():

```
$this->db->trans_off()
$this->db->trans_start();
$this->db->query('AN SQL QUERY...');
$this->db->trans_complete();
```

Quando as transacções são deficientes, as suas questões serão auto-commited, como já acontece quando executando consultas sem transações.

Modo de teste

Você pode opcionalmente colocar o sistema em operação "Modo de teste", o que fará com que suas consultas para ser retirada - mesmo que as consultas produzir um resultado válido. Para usar o modo teste basta definir o primeiro parâmetro no $ this-> db-> trans_start () a função TRUE:

```
$this->db->trans_start(TRUE); // Query will be rolled back
$this->db->query('AN SQL QUERY...');
$this->db->trans_complete();
```

Executando transações manualmente

Se você gostaria de executar operações manualmente você pode fazê-lo do seguinte modo:

```
$this->db->trans_begin();
$this->db->query('AN SQL QUERY...');
$this->db->query('ANOTHER QUERY...');
$this->db->query('AND YET ANOTHER QUERY...');
if ($this->db->trans_status() === FALSE)
{
  $this->db->trans_rollback();
}
else
{
  $this->db->trans_commit();
}
```

Nota: Certifique-se de usar $ this-> db-> trans_begin () quando estiver executando transações manual, e NÃO $ this-> db-> trans_start ().

Campo de dados (Field Data)

$ this-> db-> list_fields ()

Retorna uma matriz contendo os nomes campo. Essa consulta pode ser chamado duas maneiras:

1. Você poderá fornecer o nome da tabela e chamá-lo a partir de US $ this-> db-> object:

```
$fields = $this->db->list_fields('table_name')
foreach ($fields as $field)
{
  echo $field;
}
```

2. Você pode recolher o campo nomes associados com qualquer dúvida que correm ao chamar a função de sua consulta resultar objeto:

```
$query = $this->db->query('SELECT * FROM some_table')
foreach ($query->list_fields() as $field)
{
  echo $field;
}
```

$this-> db-> field_exists ()

Algumas vezes é útil saber se existe um campo particular antes de executar uma ação. Retorna um boolean verdadeiro / falso. Uso exemplo:

```
if ($this->db->field_exists('field_name', 'table_name'))
{
  // some code...
}
```

Nota: Substitua field_name com o nome da coluna que você está procurando, e substituir table_name com o nome da tabela que você está procurando.

$ this-> db-> field_data ()

Retorna um array de objetos contendo informações campo.

Algumas vezes é útil para reunir o campo nomes ou outros metadados, como a coluna tipo, comprimento máximo, etc

Nota: Nem todos os bancos de dados fornecem meta-dados.

Uso exemplo:

```
$fields = $this->db->field_data('table_name')
foreach ($fields as $field)
{
  echo $field->name;
  echo $field->type;
  echo $field->max_length;
  echo $field->primary_key;
}
```

Se você tiver executado uma consulta já pode utilizar o resultado objeto em vez de fornecer o nome da tabela:

```
$query = $this->db->query("YOUR QUERY")
$fields = $query->field_data()
```

Os dados a seguir está disponível a partir desta função se for suportada pelo seu banco de dados:

name - coluna nome

max_length - tamanho máximo da coluna

primary_key - 1 se a coluna é uma chave primária

type - o tipo da coluna

Função convites personalizados (Custom Function Calls):

$this-> db-> call_function ();

Essa função permite que você faça chamadas de dados PHP funções que não estão incluídas no Código nativamente Igniter, uma plataforma de forma independente. Por exemplo, permite afirmar que pretende chamar a mysql_get_client_info (), o que não é apoiado pelo Código Igniter nativamente. Você poderia fazê-lo desta forma:

$ this-> db-> call_function ('get_client_info');

Você deve fornecer o nome da função, sem o prefixo mysql_, no primeiro parâmetro. O prefixo é adicionado automaticamente com base em dados motorista que está sendo usada. Isto permite-lhe executar a mesma função em diferentes plataformas de dados. É evidente que nem todas as funções chamadas são idênticos entre plataformas, por isso existem limites à forma como esta função pode ser útil em termos de portabilidade.

Quaisquer parâmetros necessários pela função que está a ligar será acrescentado ao segundo parâmetro.

$ this-> db-> call_function ('some_function', $ param1, $ param2, etc.);

Muitas vezes, você quer necessidade de abastecer um banco de dados ou ligação ID resultado de uma base de dados ID. A ligação pode ser acessado utilizando ID:

$ this-> db-> conn_id;

O resultado ID podem ser acessados a partir de objeto dentro do seu resultado, como este:

```
$query = $this->db->query("SOME QUERY");
$query->result_id;
```

Database Utility Class

O Banco de dados Utility Class contém funções que ajudam a gerenciar sua base de dados.

Inicializando o utilitário de classe

Importante: Para inicializar o utilitário classe, seu banco de dados condutor devem ser já em execução, uma vez que os utilitários classe baseia-lo.

Coloque o Utilitário classe como segue:

$ this-> load-> dbutil ()

Uma vez que você terá inicializado utilizando o acesso a funções $ this-> dbutil objeto:

$ this-> dbutil-> some_function ()

$ this-> dbutil-> create_database ('db_name')

Permite-lhe criar o banco de dados especificada no primeiro parâmetro. Retorna verdadeiro / falso baseado no sucesso ou falha:

```
if ($this->dbutil->create_database('my_db'))
{
   echo 'Database created!';
}
```

$ this-> dbutil-> drop_database ('db_name')

Permite-lhe a queda da base de dados especificada no primeiro parâmetro. Retorna verdadeiro / falso baseado no sucesso ou falha:

```
if ($this->dbutil->drop_database('my_db'))
{
   echo 'Database deleted!';
}
```

$ this-> dbutil-> list_databases ()

Retorna uma matriz de nomes de dados:

```
$dbs = $this->dbutil->list_databases();

foreach($dbs as $db)
{
   echo $db;
}
```

$ this-> dbutil-> optimize_table ('table_name');

Nota: Este características está disponível apenas para MySQL / MySQLi bases de dados.

Permite-lhe para otimizar uma tabela utilizando o nome da tabela especificada no primeiro parâmetro. Retorna verdadeiro / falso baseado no sucesso ou fracasso:

```
if ($this->dbutil->optimize_table('table_name'))
{
   echo 'Success!';
}
```

Nota: Nem todas as plataformas de dados apoio tabela otimização.

$ this-> dbutil-> repair_table ('table_name');

Nota: Este características está disponível apenas para MySQL / MySQLi bases de dados.

Permite que você reparar uma tabela utilizando o nome da tabela especificada no primeiro parâmetro. Retorna verdadeiro / falso baseado no sucesso ou falha:

```
if ($this->dbutil->repair_table('table_name'))
{
   echo 'Success!';
}
```

Nota: Nem todas as plataformas de dados apoio tabela reparos.

$ this-> dbutil-> optimize_database ();

Nota: Este características está disponível apenas para MySQL / MySQLi bases de dados.

Permite que você otimize o seu banco de dados DB está actualmente ligado à classe. Retorna uma matriz contendo o status DB mensagens ou FALSE em falha.

```
$result = $this->dbutil->optimize_database();
if ($result !== FALSE)
{
   print_r($result);
}
```

Nota: Nem todas as plataformas de dados apoio tabela otimização.

$ this-> dbutil-> csv_from_result ($db_result)

Permite a você gerar um arquivo CSV a partir de uma consulta resultado. O primeiro parâmetro da função deve conter o resultado objeto de sua consulta. Exemplo:

```
$this->load->dbutil();
$query = $this->db->query("SELECT * FROM mytable");
echo $this->dbutil->csv_from_result($query);
```

O segundo e terceiro parâmetros permite que você defina o delimitador e newline caracteres. Por padrão guias são utilizados como o delimitador e "\ n" é utilizada como uma nova linha. Exemplo:

```
$delimiter = ",";
$newline = "\r\n";
echo $this->dbutil->csv_from_result($query, $delimiter, $newline);
```

Importante: Esta função não gravará o arquivo para você. Ela simplesmente cria o layout CSV. Se você precisa para gravar o arquivo utilizar o File Helper.

$ this-> dbutil-> xml_from_result ($db_result)

Permite a você gerar um arquivo XML a partir de uma consulta resultado. O primeiro parâmetro espera uma consulta objeto resultado, o segundo pode conter uma variedade de configurações opcionais parâmetros. Exemplo:

```
$this->load->dbutil();
$query = $this->db->query("SELECT * FROM mytable");
$config = array (
              'root'    => 'root',
              'element' => 'element',
              'newline' => "\n",
              'tab'     => "\t"
            );
echo $this->dbutil->xml_from_result($query, $config);
```

Importante: Esta função não gravará o arquivo XML para você. Ela simplesmente cria o esquema XML. Se você precisa para gravar o arquivo utilizar o File Helper.

$ this-> dbutil-> backup ()

Permite-lhe fazer backup completo banco de dados ou tabelas individuais. O backup de dados pode ser comprimido em zip ou qualquer formato GZIP.

Nota: Este características está disponível apenas para MySQL / MySQLi bases de dados.

Nota: Devido ao tempo limitado execução e de memória disponível para o PHP, apoiando-se muito grandes bases de dados podem não ser possível. Se a sua base de dados é muito grande pode precisar de backup diretamente do seu servidor SQL através da linha de comando, ou que tenham o seu servidor admin fazer isso para você se você não tem privilégios root.

Exemplo de uso:

```
/ / Carregue o utilitário classe DB
$this->load->dbutil();
/ / Backup todo o seu banco de dados e associá-lo a uma variável
$backup =& $this->dbutil->backup();
/ / Carga do arquivo helper e escrever o arquivo para o servidor
$this->load->helper('file');
```

```
write_file('/path/to/mybackup.gz', $backup);
/ / Carregue o download ajudante e enviar o arquivo para o seu desktop
$this->load->helper('download');
force_download('mybackup.gz', $backup);
```

Definir preferências de backup

São estabelecidas através da apresentação de uma série de valores para o primeiro parâmetro da função de backup.

Exemplo:

```
$prefs = array(
        'tables'     => array('table1', 'table2'),  // Array of tables to backup.
        'ignore'     => array(),         // List of tables to omit from the backup
        'format'     => 'txt',          // gzip, zip, txt
        'filename'   => 'mybackup.sql',    // File name - NEEDED ONLY WITH ZIP FILES
        'add_drop'   => TRUE,           // Whether to add DROP TABLE statements to backup file
        'add_insert' => TRUE,           // Whether to add INSERT data to backup file
        'newline'    => "\n"            // Newline character used in backup file
      );

$this->dbutil->backup($prefs);
```

Descrição das Preferências de backup

	Valor Default	Opções	Descrição
tables	empty array	None	Uma série de tabelas que você quer cópia de segurança. Se for deixado em branco todas as mesmas serão exportados.
ignore	empty array	None	Uma série de tabelas que você deseja que a rotina de backup para ignorar.
format	gzip	gzip, zip, txt	O formato do arquivo de exportação do arquivo.
filename	the current date/time	None	O nome da backups de arquivo. O nome é necessária apenas se você estiver usando compressão zip.
add_drop	TRUE	TRUE/FALSE	Seja para incluir declarações DROP TABLE em seu arquivo SQL exportação.
add_insert	TRUE	TRUE/FALSE	Se a incluir em suas declarações INSERT SQL exportação arquivo.
newline	"\n"	"\n", "\r", "\r\n"	Tipo de newline para uso em seu arquivo SQL exportação.

Classe Database caching

O Banco de dados permite que você cashing Classe cache suas buscas de arquivos como texto base reduzida carga.

Importante: Esta categoria é inicializado automaticamente pelo banco de dados quando caching condutor está habilitado. Não ative esta classe manualmente.

Além disso, note: Nem todas as funções resultado consulta estão disponíveis quando você usa caching. Por favor, leia atentamente esta página.

A ativação cashing

Caching é ativado em três etapas:

- Crie um diretório gravável em seu servidor onde o cache ficheiros podem ser armazenados.
- Defina o caminho para o seu esconderijo em sua aplicação pasta / config / database.php arquivo.
- Ativar cache do recurso, quer a nível global através da definição de preferência na sua candidatura / config / database.php arquivo, ou manualmente, como descrito a seguir.

Uma vez ativado, caching irá acontecer automaticamente sempre que uma página é carregada que contém banco de dados SQL.

Como o cache funciona?

Código Igniter's query caching sistema dinamicamente acontece quando as suas páginas são visualizadas. Quando cache está ativado, a primeira vez que uma página da Web é carregada, resultado da consulta será objecto serializada e armazenadas em um arquivo de texto em seu servidor. A próxima vez que a página é carregado o arquivo de cache será usado em vez de acessar seu banco de dados. Seu banco de dados de utilização podem efetivamente ser reduzida a zero para quaisquer páginas que foram armazenadas na memória cache.

Apenas leia-tipo (SELECT) as consultas podem ser armazenadas em cache, uma vez que estas são o único tipo de consultas que produzem um resultado. Escreva-tipo (INSERT, UPDATE, etc) as consultas, uma vez que eles não geram um resultado, não será armazenada pelo sistema.

Cache ficheiros não expiram. Todas as consultas que foram em cache irá permanecer armazenadas até você excluí-los. O sistema permite que você caching claro caches associados a páginas individuais, ou você pode apagar todo o cache coleção de arquivos. Normalmente você vai usar a casa para funções descritas a seguir para apagar arquivos cache após certos eventos ocorrem, como quando você adicionar novas informações ao seu banco de dados.

Caching irá melhorar o desempenho do seu site?

Conseguir um desempenho ganho como resultado de caching depende de muitos fatores. Se você tem um banco de dados altamente otimizado com muito pouca carga, você provavelmente não irá ver um desempenho impulso. Se seu banco de dados está em uso pesado você provavelmente verá uma melhor resposta, assumindo o seu sistema de arquivos não é excessivamente tributado. Lembre-se que caching simplesmente muda o modo como suas informações são recuperadas, deslocando-lo a partir de uma base de dados a ser uma operação de um sistema de arquivos.

Em alguns ambientes agregadas servidor, por exemplo, caching pode ser prejudicial ao sistema de arquivo desde as operações são tão intensa. A única servidores em ambientes compartilhados, caching provavelmente será benéfico. Infelizmente, não existe uma única resposta para a questão de saber se você deve cache do banco de dados. É realmente depende da sua situação.

Como são Cache arquivos armazenados?

Código Igniter coloca o resultado de cada consulta em seu próprio arquivo de cache. Conjuntos de cache de arquivos são mais organizados em sub-pastas correspondentes às suas funções controlador. Para ser preciso, a sub-pastas com nomes idênticos para os dois primeiros segmentos do seu URI (o controlador classe nome e função nome).

Por exemplo, digamos que você tem um blog chamado controlador com uma função chamada comentários que contém três buscas. O cache vai criar um sistema de cache pasta chamada blog + comentários, em que ele irá escrever três cache arquivos.

Se você usar consultas que mudar dinâmico baseado nas informações de seu URI (quando se utiliza paginação, por exemplo), cada instância da consulta irá produzir o seu próprio arquivo de cache. É possível, portanto, para acabar com cache muitas vezes mais do que arquivos você tem buscas.

Gerenciando o cache arquivos

Desde cache ficheiros não termina, você precisa construir supressão rotinas em sua candidatura. Por exemplo, digamos que você tem um blog que permite que os usuários comentando. Sempre que for apresentado um novo comentário que você quer apagar a memória cache arquivos associados com o controlador função que serve o seu comentário. Você encontrará duas funções descritas abaixo excluir que o ajudam a limpar os dados.

Nem todas as funções funcionam com base caching

Por último, temos de salientar que o resultado é que o objeto em cache é uma versão simplificada do resultado completo objeto. Por essa razão, algumas das funções resultado da consulta não estão disponíveis para uso.

As seguintes funções não estão disponíveis quando se utiliza um resultado em cache objeto:

num_fields ()

field_names ()

field_data ()

free_result ()

Além disso, os dois recursos de dados (result_id e conn_id) não estão disponíveis quando caching, uma vez que apenas se referem a recursos resultar em tempo de execução operações.

Função Referência

$this->db->cache_on() / $this->db->cache_off()

Manualmente permite / desabilita caching. Isto pode ser útil se você quiser manter certas consultas de ser armazenado em cache. Exemplo:

```
/ / Rode a caching
$this->db->cache_on();
$query = $this->db->query("SELECT * FROM mytable");
/ / Turn off caching para uma consulta
$this->db->cache_off();
$query = $this->db->query("SELECT * FROM members WHERE member_id = '$current_user'");
/ / Retorna caching
$this->db->cache_on();
$query = $this->db->query("SELECT * FROM another_table");
```

$ this-> db-> cache_delete ()

Apaga o cache arquivos associados com uma página em particular. Isto é útil se você precisa limpar a cache depois de você atualizar o seu banco de dados.

O sistema salva o cache caching arquivos para pastas que correspondem à URI da página que você está vendo. Por exemplo, se você está vendo uma página em www.your-site.com/index.php/blog/comments, o sistema de caching cache irá colocar todos os arquivos associados a ele em uma pasta chamada blog + comentários. Para apagar esses arquivos cache especial que você usará:

$ this-> db-> cache_delete ('blog', 'Comentários');

Se você não usa nenhum dos actuais parâmetros URI será utilizado para determinar qual deve ser apuradas.

$ this-> db-> cache_delete_all ()

Apaga todos os arquivos existentes cache. Exemplo:

$ this-> db-> cache_delete_all ();

Referências dos Assistentes

Assistente de Array

O arquivo Assistente de Array contem funções que auxiliam no trabalho com arrays.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('array');

As seguinte funções estão disponíveis:

element()

Permite que você busque um item dentro de um array. A função testa o índice do array e seu valor. Se um valor existe ele então é retornado. Se um valor não existe ele retorna FALSE, ou qualquer outra mensagem que você tenha especificado como padrão no terceiro parâmetro. Exemplo:

```
$array = array('cor' => 'vermelho', 'forma' => 'redondo', 'size' => '');

// retorna "vermelho"
echo element('cor', $array);

// retorna NULL
echo element('tamanho', $array, NULL);
```

random_element()

Pega um array como entrada e retorna um elemento randômico dele. Exemplo de uso:

```
$citacoes = array(
        "I find that the harder I work, the more luck I seem to have. - Thomas Jefferson",
        "Don't stay in bed, unless you can make money in bed. - George Burns",
        "We didn't lose the game; we just ran out of time. - Vince Lombardi",
        "If everything seems under control, you're not going fast enough. - Mario Andretti",
        "Reality is merely an illusion, albeit a very persistent one. - Albert Einstein",
        "Chance favors the prepared mind - Louis Pasteur"
        );
echo random_element($citacoes);
```

Assistente de Cookie

O arquivo Assistente de Cookie contem funções que auxiliam no trabalho com cookies.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('cookie');

As seguinte funções estão disponíveis:

set_cookie()

Configura um cookie contendo os valores que você especificar. Existe dois modos de passar a informação a esta função de modo que o cookie pode ser configurado como: Método de Array, e Parâmetros Discretos(Discreet Parameters):

Método de Array

Usando este método, um array associativo é passado ao primeiro parâmetro:

```
$cookie = array(
            'name'   => 'O Nome do Cookie',
            'value'  => 'O Valor',
            'expire' => '86500',
            'domain' => '.algum-dominio.com',
            'path'   => '/',
            'prefix' => 'meuprefixo_',
          );
set_cookie($cookie);
```

Notas: Apenas o nome(name) e valor(value) são obrigatórios.

O tempo de expiração(expire) do cookie é ajustado em segundos, que será adicionado à hora atual.Não inclua a hora, apenas os segundos a partir do momento que você deseja que o cookie seja válido . Se o tempo de expiração estiver configurado como zero o cookie durará apenas o tempo em que o browser estiver aberto.

Para apagar um cookie configure o tempo de expiração em branco/sem valor.

Para cookies que abranjam todo site não importando como este seja requisitado, adicione sua URL ao domain começando com um perído, como este: .seu-dominio.com

O caminho (path) é normalmente desnecessário desde que a função configure um caminho raiz.

O prefixo é apenas necessário se você precisar evitar colisões de nomes dos cookies no seu servidor.

Parâmetros Discretos(Discreet Parameters)

Se você preferir, pode configurar o cookie passando os dados usando parâmetros individuais:

set_cookie($name, $value, $expire, $domain, $path, $prefix);

get_cookie()

Permite que você busque um cookie. O primeiro parâmetro conterá o nome do cookie que vocês está buscando:

get_cookie('algum_cookie');

A função retornará FALSE (booleano) se o item que você estiver tentando recuperar não existir.

O segundo parâmetro opcional permite você executar os dados através do filtro XSS. Este estará habilitado ao configurar o segundo parâmetro para TRUE;

get_cookie('algum_cookie', TRUE);

delete_cookie()

Permite que você apague um cookie. A não ser que você tenha configurado um caminho personalizado ou outros valores, apenas o nome do cookie é necessário:

delete_cookie("nome");

Esta função, por sua vez, é idêntica à set_cookie(), exceto pelo fato de não ter os parâmetros de valor (value) e expiração(expire). Você pode submeter um array de valores no primeiro parâmetro ou pode configurar parâmetros discretos(discreet parameters).

delete_cookie($name, $domain, $path, $prefix)

Assistente de Data

O arquivo Assistente de Data contem funções que auxiliam no trabalho com datas.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('date');

As seguinte funções estão disponíveis:

now()

Retorna a hora atual no format Unix timestamp, referente ou ao seu servidor local ou a hora GMT, baseada na "hora referência"("time reference") configurada em seu arquivo de configuração. Se você não pretende configurar sua hora mestre conforme a hora GMT (que normalmente é o que você faz se vocÊ permite que cada usuário ajuste suas próprias configurações de local e data) não haverá benfício em usar esta função ao invés da função time() do PHP.

mdate()

Esta função é idêntica à função date() do PHP, a não ser pelo fato que ela lhe permite usar códigos de data no estilo do MySQL, onde cada código é precedido por uma sinal de percentual: %Y %m %d etc.

O benfício de fazer datas desta maneira é que você não precisa se preocupar em das escape em nenhum caracter que não seja um código de data como você normalmente faria se usasse a função date(). Exemplo:

```
$stringdedata = "Ano: %Y Mês: %m Dia: %d - %h:%i %a";
$data = time();
echo mdate($stringdedata, $data);
```

Se o timestamp não estiver incluído no segundo parâmetro a hora atual será usada.

standard_date()

Permite gerar uma string de data em um dos muitos formatos padronizados. Exemplo:

```
$formato = 'DATE_RFC822';
$data = time();
echo standard_date($formato, $data);
```

O primeiro parâmetro deve conter o formato, o segundo deve conter a data como Unix timestamp.

Formatos suportados:

- DATE_ATOM
- DATE_COOKIE
- DATE_ISO8601
- DATE_RFC822
- DATE_RFC850
- DATE_RFC1036
- DATE_RFC1123
- DATE_RFC2822
- DATE_RSS
- DATE_W3C

local_to_gmt()

Pega um Unix timestamp como entrada e retonrna como um GMT. Exemplo:

```
$agora = time();
$gmt = local_to_gmt($agora);
```

gmt_to_local()

Pega um Unix timestamp (referente ao GMT) como entrada e o converte para um timestamp local basead no fuso e horário de verão enviados. Exemplo:

```
$timestamp = '1140153693';
$fuso = 'UM8';
$horario_de_verao = TRUE;
echo gmt_to_local($timestamp, $fuso, $horario_de_verao);
```

Nota: Para uma lista de fusos horários veja a referência no fina desta página.

mysql_to_unix()

Pega um MySQL Timestamp como entrada e o retorna como Unix. Exemplo:

```
$mysql = '20061124092345';
$unix = mysql_to_unix($mysql);
```

unix_to_human()

Pega um Unix timestamp como entrada e o retorna num formato legível ao homem com este protótipo:

YYYY-MM-DD HH:MM:SS AM/PM

Pode ser útil se você precisar exibir uma data num campo de formulário para envio.

A hora pode ser formatada com ou sem segundos e pode ser ajustada no formato Europeu ou Americano. Se apenas o timestamp for submetido ele retornará a hora sem segundos no formato Americano. Exemplo The time can be formatted with or without seconds, and it can be set to European or US format. If only the timestamp is submitted it will return the time without seconds formatted for the U.S.:

```
$agora = time();
echo unix_to_human($agora); // Formato de hora Americano, sem segundos
echo unix_to_human($agora, TRUE, 'us'); // Formato Americano com segundos
echo unix_to_human($agora, TRUE, 'eu'); // Formato Europeu com segundos
```

human_to_unix()

Oposto à função acima.Pega a hora "humana" como entrada e retorna no formato Unix. Esta função é útil se você recebe a hora formatada pelo padrão "humano" através de formulário. Retornará FALSE (booleano) se a string de hora passada a ela não estiver formatada como indicado acima. Exemplo:

```
$agora = time();
$humano = unix_to_human($agora);
$unix = human_to_unix($humano);
```

timespan()

Formata o unix timestamp de modo que ele fique similar a isto:

1 Ano, 10 Meses, 2 Semanas, 5 Dias, 10 Horas, 16 Minutos

O primeiro parâmetro deve conter um Unix timestamp. O segundo deve conter um timestamp maior ao primeiro timestamp. Se o segundo parâmetro estiver vazio, a hora atual será usada. A finalidade mais comum para esta função é mostrar quanto tempo se passou desde algum ponto no passado até agora. Exemplo:

```
$hora_passada = '1079621429';
$agora = time();
echo timespan($hora_passada, $agora);
```

Nota: O texto gerado por esta função é encontrado no seguinte arquivo de idioma:language/<seu_idioma>/date_lang.php

days_in_month()

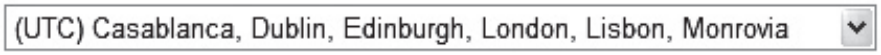
Retorna o número de dias em um dado mês/ano. Returns the number of days in a given month/year. Computa anos bissextos na contagem. Exemplo:

echo days_in_month(06, 2005);

Se o segundo parâmetro estiver vazio, o ano atual será usado.

timezone_menu()

Geral um menu pull-down de fusos horários, como este:



Este menu é útil se possui um site de cadastros onde cada usuário pode ajustar seu próprio fuso local.

O primeiro parâmetrp permite configurar o item selecionado do menu. Por exemplo, para deixar selecionado o fuso Pacific como padrão você deve fazer isto:

echo timezone_menu('UM8');

Por favor veja a referência de fusos abaixo para ver os valores deste menu.

O segundo parâmetro permite configurar um class name de CSS para o menu.

Nota: O texto contido no menu pode ser encontrado no seguindo arquivo de idioma: language/<seu_idioma>/date_lang.php

Referência de Fuso-horário

A tabela a seguir mostra cada fuso com sua localização.

Fuso Horário	Localização
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

Assistente de Diretório

O arquivo Assistente de Diretório contem funções que auxiliam no trabalho com diretório.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('directory');

As seguinte funções estão disponíveis:

directory_map('diretorio fonte')

Esta função lê o caminho de diretório especificado no primeiro parâmetro e monta um array representativo dele e de seus arquivos contidos. Exemplo:

$mapa = directory_map('./meudiretorio/');

Nota: Caminhos são em sua maiorira relativos ao seu arquivo principal index.php.

Subdiretórios contidos dentro do diretório serão mapeados também. Se desejar mapear apenas o diretório de nível mais alto ajuste o segundo parâmetro para true (booleano):

$mapa = directory_map('./meudiretorio/', TRUE);

Cada nome de diretório será um índice do array, enquanto que seus arquivos serão indexados numericamente. Aqui está um exemplo de um array comum:

```
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
)
```

Assistente de Download

O Assistente de Download permite que você baixe dados para seu desktop.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('download');

As seguinte funções estão disponíveis:

force_download('nomedoarquivo', 'dados')

Gera cabeçalhos de servidor que forçam os dados a serem baixados para seu desktop. Útil para downloads de arquivos. O primeiro parâmetro é o nome que você quer que o arquivo a ser baixado tenha, o segundo parâmetro são os dados do arquivo. Exemplo:

```
$dados = 'Aqui está algum texto!';
$nome = 'meutexto.txt';
force_download($nome, $dados);
```

Caso queira baixar algum arquivo existente de seu servidor você precisará lê-lo dentro de uma string:

```
$dados = file_get_contents("/caminho/para/foto.jpg"); // Lê o conteúdo do arquivo
$nome = 'minhafoto.jpg';
force_download($nome, $dados);
```

Assistente de Arquivo

O arquivo Assistente de Diretório contem funções que auxiliam no trabalho com arquivos.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('file');

As seguinte funções estão disponíveis:

read_file('caminho')

Retorna os dados contidos dentro do arquivo especificado no caminho. Exemplo:

$string = read_file('./caminho/para/arquivo.php');

O caminho pode ser relativo ou absoluto no servidor. Retornará FALSE (booleano) caso falhe.

Nota: O caminho é relativo ao seu arquivo index.php principal, NÃO ao seus arquivos controller ou view. O Code Igniter usa um front controller de modo que sempre será relativo ao index principal do site.

Se o seu servidor estiver executando uma restrição open_basedir esta função talvez não funcione se tentar acessar um arquivo que esteja um nível acima ao script chamado.

write_file('caminho', $dados)

Escreve os dados no arquivo especificado no caminho. Caso o arquivo não exista ele será criado. Exemplo:

```
$dados = 'Alguns dados de arquivo';

if ( ! write_file('./caminho/para/arquivo.php', $dados))
{
    echo 'Não foi possível escrever no arquivo';
}
else
{
    echo 'Dados escritos!';
}
```

Você pode configurar opcionalmente o modo de escrita através do terceiro parâmetro:

write_file('./caminho/para/arquivo.php', $dados, 'r+');

O modo padrão é wb Por favor veja o Guia de Usuário do PHP para as opções de modo disponíveis.

Nota: Para que esta função escreva os dados no arquivo, este deve ter suas permissões configuradas para 'escrita' (666, 777, etc.). Se o arquivo não existir ainda, então o diretório que o conterá deve ter essas permissões de escrita.

Nota: O caminho é relativo ao seu arquivo index.php principal, NÃO ao seus arquivos controller ou view. O Code Igniter usa um front controller de modo que sempre será relativo ao index principal do site.

delete_files('caminho')

Apaga TODOS os arquivos dentro do caminho fornecido. Exemplo:

delete_files('/caminho/para/diretorio/');

Se o segundo parâmetro estiver configurado como true, qualquer diretório dentro do caminho fornecido também será apagado. Exemplo:

delete_files('/caminho/para/diretorio/', TRUE);

Nota: Os arquivos devem ter permissão de escrita ou pertencerem ao sistema de modo que possam ser apagados.

get_filenames('caminho/para/diretorio/')

Pega um caminho do servidor como entrada e retorna um array contendo os nomes de todos os arquivos dentro dele. O caminho do arquivo pode opcionalmente ser adicionado aos nomes de arquivos ao configurar o segundo parâmentro como TRUE.

Assistente de Formulário

O Assistente de Formulário contem funções que auxiliam no trabalho com formulário.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

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

As seguinte funções estão disponíveis:

form_open()

Cria uma tag de abertura de formulário com a URL base montada em cima de suas preferências de configuração. Permitirá que, opcionalmente, você adicione atributos e campos 'hidden'.

A principal vantagem de usar esta tag ao invés de codificar manualmente seu próprio HTML é que ela

permite que seu site seja mais fácil de portar no caso de sua URL ser alterada.

Aqui está um exemplo simples:

echo form_open('email/send');

O exemplo acima criará um formulário que aponta para sua URL base mais os segmentos "email/send", desta maneira:

<form method="post" action="http:/www.seu-site.com/index.php/email/send" />

Adicionando Atributos

Atributos podem ser adicionados ao passar um array associativo como segundo parâmetro, desta maneira:

```
$attributes = array('class' => 'email', 'id' => 'meuform');
echo form_open('email/send', $attributes);
```

O exemplo acima criará um formulário semelhante a este:

<form method="post" action="http:/www.seu-site.com/index.php/email/send" class="email" id="meuform" />

Adicionando Campos Hidden

Campos Hidden podem ser adicionados ao passar um array associativo como terceiro parâmetro, desta maneira:

```
$hidden = array('username' => 'Jose', 'id_membro' => '234');
echo form_open('email/send', '', $hidden);
```

O exemplo acima criará um formulário semelhante a este:

```
<form method="post" action="http:/www.seu-site.com/index.php/email/send" class="email" id="meuform" />
<input type="hidden" name="username" value="Jose" />
<input type="hidden" name="id_membro" value="234" />
```

form_open_multipart()

Esta função é totalmente idêntica à tag form_open() acima, exceto pelo daoto de adicionar o atributo 'multipart', que é necessário se você deseja usar um formulário para fazer upload de arquivos.

form_hidden()

Permite gerar campos hidden. Você pode também enviar uma string nome/valor para criar um campo:

```
form_hidden('username', 'joaoninguem');

// Produzirá:

<input type="hidden" name="username" value="joaoninguem" />
```

Ou pode enviar um array associativo para criar múltiplos campos:

```
$dados = array(
          'name'  => 'João Ninguém',
          'email' => 'joao@algum-site.com',
          'url'   => 'http://www.algum-site.com'
        );
echo form_hidden($dados);

// Produzirá:
<input type="hidden" name="name" value="João Ninguém" />
<input type="hidden" name="email" value="joao@algum-site.com" />
<input type="hidden" name="url" value="http://www.algum-site.com" />
```

form_input()

Permite gerar um campo input text padrão. Você pode passar o 'name' e o 'value' do campo no primeiro e segundo parâmetros:

echo form_input('username', 'joaoninguem');

Ou passar um array associativo contendo algum dado que deseje que o fomulário contenha:

```
$dados = array(
          'name'      => 'username',
          'id'        => 'username',
          'value'     => 'joaoninguem',
          'maxlength'  => '100',
          'size'      => '50',
          'style'     => 'width:50%',
        );
echo form_input($dados);

// Produzirá:
<input type="text" name="username" id="username" value="joaoninguem" maxlength="100"
size="50" style="width:50%" />
```

Se quiser que o formulário contenha algum dado adicional, como JavaScript, você pode passar isso como uma string no terceiro parâmetro:

```
$js = 'onClick="some_function()"';
echo form_input('username', 'joaoninguem', $js);
```

form_password()

Esta função é idêntica em todos os aspectos à função form_input() acima exceto pelo fato de gerar um campo como type="password" (campo senha).

form_upload()

Esta função é idêntica em todos os aspectos à função form_input() acima exceto pelo fato de configurar o campo como type="file" (campo arquivo), permitindo o envio de arquivos ao servidor.

form_textarea()

Esta função é idêntica em todos os aspectos à função form_input() acima exceto pelo fato de gerar um campo "textarea". Nota: Ao invés dos atributos "maxlength" e "size" no exemplo acima, você especificará

"rows" e "cols".

form_dropdown()

Permite criar um campo select padrão (drop-down). O primeiro parâmetro contém o nome do campo, o segundo parâmetro conterá um array associativo de opções, e o terceiro parâmetro conterá o valor que você deseja deixar como selecionado. Exemplo:

```
$opcoes = array(
            'pequena'  => 'Camisa Pequena',
            'media'    => 'Camisa Média',
            'grande'   => 'Camisa Grande',
            'extragrande' => 'Camisa Extra-Grande',
           );
echo form_dropdown('camisas', $opcoes, 'grande');

// Produzirá:
<select name="camisas">
<option value="pequena">Camisa Pequena</option>
<option value="media">Camisa Média</option>
<option value="grande" selected>Camisa Grande</option>
<option value="extragrande">Camisa Extra-Grande</option>
</select>
```

Se quiser que a tag de abertura <select> contenha dados adicionais, como JavaScript, você pode passá-los como uma string no quarto parâmetro:

```
$js = 'onChange="some_function()"';
echo form_dropdown('camisas', $opcoes, 'grande', $js);
```

form_checkbox()

Permite gerar um campo checkbox. Exemplo simples:

```
echo form_checkbox('newsletter', 'aceitar', TRUE);
// Produzirá:
<input type="checkbox" name="newsletter" value="aceitar" checked="checked" />
```

O terceiro parâmetro conter um TRUE/FALSE boolenao que determina se o campo deverá vir marcado ou não.

Semelhante às outras funções de formulário neste assistente, você pode passar um array de atributos à função:

```
$dados = array(
          'name'      => 'novidades',
          'id'        => 'novidades',
          'value'     => 'aceitar',
          'checked'   => TRUE,
          'style'     => 'margin:10px',
         );
echo form_checkbox($dados);
// Produzirá:
<input type="checkbox" name="novidades" id="novidades" value="aceitar" checked="checked"
style="margin:10px" />
```

Como em outras funções, se quiser que a tag contenha dados adicionais, como JavaScript, você pode passá-los como uma string no quarto parâmetro:

```
$js = 'onClick="some_function()"';
echo echo form_checkbox('novidades', 'aceitar', TRUE, $js)
```

form_radio()

Esta função é idêntica em todos os aspectos à função form_checkbox() acima exceto pelo fato de configurar o campo(input) como type="radio".

form_submit()

Permite você gerar um botão submit padrão. Exemplo simples:

```
echo form_submit('meusubmit', 'Enviar!');
// Produzirá:
<input type="submit" name="meusubmit" value="Enviar!" />
```

Semelhante a outras funções, você pode enviar um array associativo no primeiro parâmetro se preferir configurar seus próprios atributos. O terceiro parâmetro permite adicionar dados extar ao seu formulário, como Javascript.

form_close()

Gera a tag de fechamento </form>. A única vantagem de é que permite que você passe informações que serão adicionadas após a tag. Por exemplo:

```
$string = "</div></div>";
echo form_close($string);
// Produzirá:
</form>
</div></div>
```

form_prep()

Permite que use com segurança HTML e caracteres como "aspas" dentro dos elementos do formulário sem quebrá-los para fora do formulário. Considere este exemplo:

```
$string = 'Aqui está uma string contendo um texto com "aspas".';
<input type="text" name="meuform" value="$string" />
```

Mesmo que a string acima contenha "aspas" ela não causará a quebra do formulário. A função form_prep convert o HTML de modo que pode ser usado com segurança:

```
<input type="text" name="meuform" value="<?php echo form_prep($string); ?>" />
```

Nota: Se você usar alguma das funções do assistente de formulário listadas nesta página os valores do formulário serão "convertidos" automaticamente, então não é preciso chamar esta função. Use somente se você estiver criando seus próprios elementos de formulário.

Assistente de HTML

O Assistente de HTML contem funções que auxiliam no trabalho com HTML.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('html');

As seguinte funções estão disponíveis:

heading()

Permite criar tags HTML <h1>, O primeiro parâmetro irá conter os dados, o segundo o tamanho do heading. Exemplo:

echo heading('Bem-vindo!', 3);

O código acima produzirá: <h3>Bem-vindo!</h3>

ol() and ul()

Permite gerar listas HTML ordenadas ou nã-ordenadas de um array simples ou multi-dimensional. Exemplo:

```
$this->load->helper('html');
$lista = array(
        'vermelho',
        'azul',
        'verde',
        'amarelo'
        );
$atributos = array(
              'class' => 'listanegritada',
              'id'    => 'minhalista'
              );
echo ul($lista, $atributos);
```

O código acima produzirá isto:

```
<ul class="listanegritada" id="minhalista">
 <li>vermelho</li>
 <li>azul</li>
 <li>verde</li>
 <li>amarelo</li>
</ul>
```

Aqui está um exemplo mais complexo, usando um array multi-dimensional:

```
$this->load->helper('html');
$lista = array(
        'cores' => array(
                    'vermelho',
                    'azul',
                    'verde'
                  ),
        'formas' => array(
                    'curva',
                    'quadrado',
                    'circulos' => array(
                                  'elipse',
                                  'oval',
                                  'esfera'
                                  )
                  ),
        'humor'   => array(
                    'feliz',
                    'chateado' => array(
                                  'derrotado' => array(
                                                 'rejeitado',
                                                 'desolado',
                                                 'deprimido'
                                                 ),
                                  'irritado',
                                  'atravessado',
                                  'brabo'
                                )
                  )
        );
echo ul($lista);
```

O código acima produzirá isto:

```
<ul class="listanegritada" id="minhalista">
 <li>cores
  <ul>
   <li>vermelho</li>
   <li>azul</li>
   <li>verde</li>
  </ul>
 </li>
 <li>formas
  <ul>
   <li>curva</li>
   <li>quadrado</li>
   <li>circulos
    <ul>
     <li>elipse</li>
     <li>oval</li>
     <li>esfera</li>
    </ul>
   </li>
```

```
  </ul>
 </li>
 <li>humor
  <ul>
   <li>feliz</li>
   <li>chateado
    <ul>
     <li>derrotado
      <ul>
       <li>rejeitado</li>
       <li>desolado</li>
       <li>deprimido</li>
      </ul>
     </li>
     <li>irritado</li>
     <li>atravesado</li>
     <li>brabo</li>
    </ul>
   </li>
  </ul>
 </li>
</ul>
```

nbs()

Gera non-breaking spaces ( ) baseada no número que você enviar. Exemplo:

echo nbs(3);

O código acima produzirá:

br()

Gera tags de quebra de linha (
) baseada no número que você enviar. Exemplo:

echo br(3);

O código acima produzirá:

Assistente de Segurança

O Assistente de Segurança contem funções funções relacionadas à segurança.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('security');

As seguinte funções estão disponíveis:

xss_clean()

Prove um filtro "Cross Site Script Hack". Esta função é um apelido/pseudônimo para a classe Input . Maiores informações podem ser encontradas através do link fornecido.

dohash()

Permite criar um um modo de hash SHA1 ou MD5 apropriado para encriptar senhas. Irá usar o SHA1 por

padrão. Exemplos:

```
$str = dohash($str); // SHA1
$str = dohash($str, 'md5'); // MD5
```

strip_image_tags()

Esta é uma função de segurança que irá tirar as tags de imagem de uma string. Deixando apenas,em texto puro, a URL da imagem.

$string = strip_image_tags($string);

encode_php_tags()

Esta é uma função de segurança que converte tags PHP em entities. Nota: Se você usar a função de filtro XSS ele fará isso automaticamente.

$string = encode_php_tags($string);

Assistente de Smiley

O Assistente de Smiley contem funções que permitem que vocÊ tenha controle sobre smileys (emoticons).

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('smiley');

Visão Geral

O Assistente de Smiley possui um renderizador que pega smileys de texto puro como **:-)** e transforma em representação gráfica como  .

Permite também que seja exibido um conjunto de imagens de emoticons que quando clicados serão inseridos num campo de formulário. Por exemplo, se você tiver um blog que permita o comentário de usuários poderá, então, exibir os emoticons perto do campo de comentários. Seus usuários poderão clicar no emoticon e com a ajuda de algum Javascript este será colocado no campo do formulários.

Tutorial de Smileys(Emoticons) Clicáveis

Aqui está um exemplo que demonstra como você pode criar um conjunto de emoticons clicáveis próximos do campo do formulário. Este exemplo requer que você, primeiramente, baixe e instale as imagens de smiley, então, depois, crie um controller e uma View como descritos.

Importante: Antes de começar, por favor baixe os emoticons e coloque-os em um local de acesso público em seu servidor. Este assistente também assume que você tenha o array de substituição de smiley localizado em application/config/smileys.php

O Controller

Em sua pasta application/controllers/, crie um arquivo chamado smileys.php e cole o código abaixo nele.

Importante: Mude a URL na função get_clickable_smileys() abaixo de modo que aponte para sua pasta smiley.

Você notará que além do assistente de smiley também estamos usando a Classe Table.

```
<?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);
	}
}
```

Em sua pasta application/views/, crie um arquivo chamado smiley_view.php e cole este código nele:

```
<html>

<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/helpers/smiley_
helper.html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 20:00:00 GMT -->
<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>Clique para inserir um smiley!</p>a

<?php echo $smiley_table; ?>

</body>

<!-- Mirrored from www.plasmadesign.com.br/codeigniter/user_guide-pt_BR/helpers/smiley_
helper.html by HTTrack Website Copier/3.x [XR&CO'2007], Wed, 15 Aug 2007 20:00:00 GMT -->
</html>
```

Depois de criar o controller e a view acima, carregue-os visitando http://www.seu-site.com/index.php/smileys/

Referência da Função

get_clickable_smileys()

Retorna um array contendo suas imagens de smiley dentro de um link. Você deve fornecer a URL para sua pasta de smiley através do primeiro parâmetro:

$image_array = get_clickable_smileys("http://www.seu-site.com/images/smileys/");

js_insert_smiley()

Gera o javascript que permite as imagens serem clicadas e inseridas dentro do campo de formulário. O primeiro parâmetro deve conter o nome de seu formulário, o segundo deve conter o nome do campo. Esta função é feita para ser colocada dentro da seção <head> de sua página web.

<?php echo js_insert_smiley('blog', 'comments'); ?>

parse_smileys()

Pega uma string de texto como entrada e substitui o texto pelos smileys/emoticons equivalentes. O primeiro parâmetro deve conter sua string, o segundo deve conter a URL de sua pasta de smiley:

$str = 'Aqui estão alguns emoticons: :-) ;-)'; $str = parse_smileys($str, "http://www.seu-site.com/images/smileys/"); echo $str;

Assistente de String

O Assistente de String contem funções que auxiliam no trabalho com strings.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('string');

As seguinte funções estão disponíveis:

random_string()

Gera uma string randômica baseada no tipo e no comprimento que você especificar. Útil para criar senhas ou hashes randômicos.

O primeiro parâmetro especifica o tipo de string, o segundo especifica o comprimento. As seguintes opções estão disponíveis:

- alnum: String alfa-numérica com caracteres em caixa alta e baixa.

- numeric: String numérica.

- nozero: String numérica sem zeros.

- unique: Encriptada com MD5 e uniquid(). Nota: O parâemtro de comprimento não é disponível para este tipo. Retorna uma string de comprimento fixo de 33 caracteres.

Exemplo de Uso:

echo random_string('alnum', 16);

alternator()

Permite que dois ou mais itens alternem-se entre si quando dentro de um loop. Exemplo:

```
for ($i = 0; $i < 10; $i++)
{
   echo alternator('string um', 'string dois');
}
```

Você pode adicionar quantos parâmetros quiser e a cada interação de seu loop o próximo ítem será retornado.

```
for ($i = 0; $i < 10; $i++)
{
   echo alternator('um', 'dois', 'três', 'quatro', 'cinco');
}
```

Nota: Para usar chamadas múltiplas separadas a esta função basta chamar a função sem argumento para reinicializá-la.

repeater()

Gera cópias repetidas dos dados que vocÊ enviar. Exemplo:

```
$string = "\n";
echo repeater($string, 30);
```

O código acima gerará 30 novas linhas.

Assistente de Texto

O Assistente de Texto contem funções que auxiliam no trabalho com texto.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('text');

As seguinte funções estão disponíveis:

word_limiter()

Trunca uma string para o número de palavras especificado. Exemplo:

```
$string = "Aqui temos um belo texto de oito palavras.";
$string = word_limiter($string, 4);
// Retorna: Aqui temos um belo…
```

O terceiro parâmetro é um sufixo opcional adicionado à string. Por padrão adiciona-se reticências.

character_limiter()

Trunca uma string para o número de caracteres especificado. Mantem a integridade das palavras de modo que a contagem pode ter uma pequena variação para mais ou para menos do que você especificar. Exemplo:

```
$string = "Aqui temos um belo texto de oito palavras.";
$string = character_limiter($string, 20);
// Retorna: Aqui temos um belo …
```

O terceiro parâmetro é um sufixo opcional adicionado à string. Por padrão adiciona-se reticências.

ascii_to_entities()

Converte valores ASCII em character entities, incluindo high ASCII e caracteres MS Word que podem causar problemas quando usado em uma página da web, de modo que possam ser mostrados sem erros não obstante configurações do browser ou serem armazenados sem problemas em uma base de dados. Eiste algumas dependências no seu conjunto de caracteres suportados pelo servidor, portanto isto pode não ser 100% confiável, mas na maioria dos casos ele identificará corretamente os caracteres foram do escopo normal (acentuados por exemplo). Exemplo:

$string = ascii_to_entities($string);

entities_to_ascii()

Esta função faz o oposto da anterior, transforma character entities de volta em ASCII.

word_censor()

Habilita a censura de palavras de uma determinada string. O primeiro parâmetrp conterá a string original. O segundo conterá uma array de palavras proibidas. O terceiro (opcional) conterá um valor substituto para as palavras. Se não especificado elas serão subsituidas por sinal de "jogo da velha": ####. Exemplo:

```
$proibidas = array('tóxico', 'maconha', 'cocaina', 'cigarro');
$string = word_censor($string, $proibidas, 'eeeeeeepppaaaa!');
```

highlight_code()

Colore uma string de código(PHP, HTML, etc.). Exemplo:

$string = highlight_code($string);

A função usa a função highlight_string() do PHP, as cores usadas são as especificadas no seu arquivo php. ini.

highlight_phrase()

Destacará uma frase dentro de uma string de texto. O primeiro parâmetro conterá a string original, o segundo a frase que deseja destacar. Os terceiro e quarto parâmetros conterão as tags HTML de abertura e fechamento onde a frase estará contida. Exemplo:

```
$string = "Aqui está um belo texto sobre nada em especial.";
$string = highlight_phrase($string, "belo texto", '<span style="color:#990000">', '</span>');
```

O código acima retornará:

Aqui está um belo texto sobre nada em especial.

word_wrap()

Quebra a linha de texto conforme o número de caracteres especificados mantendo as palavras inteiras. Exemplo:

```
$string = "Aqui temos um string simples de texto que nos ajudará a demonstrar esta função";
echo word_wrap($string, 25);
// Produzirá:
Aqui temos um string simples
de texto que nos ajudará
a demonstrar esta função
```

Assistente de Tipografia

O Assistente de Tipografia contem funções que auxiliam a formatar texto de modo semanticamente relevante.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('typography');

As seguinte funções estão disponíveis:

auto_typography()

Formata o texto de modo a ficar tipograficamente e semanticamente correto o HTML. Pega uma string como entrada e retorna com a seguinte formatação:

- Envolve os parágrafos com (procura por duas linhas em branco para identificar parágrafos).

- Converte quebras de linha em
, exceto aquelas que aparecerem dentro de tags <pre>.

- Elementos block, como tags , não são envolvidos dentro de tags parágrafos, mas seus conteúdos de texto sim caso sejam parágafos.

- Aspas são convertidas para suas entities corretas, exceto aquelas que aparecerem dentro de tags.

- Apóstrofos são convertidos para entities.

- Traços duplos (como -- esse ou como--esse) são convertidos para traços—assim.

-Três períodos consecutivos precedendo ou seguintes a uma palavra são convertidos para reticências…

- Espaços duplos seguindo sentenças são convertidos para non-breaking spaces   para imitar o espaço.

Exemplo de Uso:

$string = auto_typography($string);

Nota: Formatação tipográfica pode causar sobrecarga de processamento, principalmente se você possuir muito conteúdo a ser formatado. Se optar por usar esta função talvez queira considerar fazer caching de suas páginas.

nl2br_except_pre()

Converte novas linhas em tags
a não ser que estas estejam dento de tags <pre>. Esta função é idêntica à função nl2br() nativa do PHP, exceto pelo fato de esta última ignorar tags <pre>.

Exemplo de Uso:

$string = nl2br_except_pre($string);

Assistente de URL

O Assistente de XML contem funções que auxiliam no trabalho com URLs.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

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

As seguinte funções estão disponíveis:

site_url()

Retorna a URL de seu site, conforme especificada no seu arquivo de configuração. O arquivo index.php (ou algum outro que seja a index_page de seu confrome seu arquivo de configuração) será adicionado à URL, como qualquer segmento URI que vocÊ passar para a função.

Sinta-se encorajado a usar esta função sempre que precisar gerar uma URL local de modo que suas páginas tornem-se muito mais portáveis na eventualidade de mudanças em sua URL.

Segmentos podem ser passados (opcionalmente) para a função como uma string ou array. Aqui está um exemplo de string:

echo site_url("novidades/local/123");

O exemplo acima retornará algo assim: http://www.seu-site.com/index.php/novidades/local/123

Aqui está um exemplo de segmentos passados como um array:

```
$segmentos = array('novidades', 'local', '123');
echo site_url($segments);
```

base_url()

Retorna a URL base do seu site, conforme especificado no seu arquivo de configuração. Exemplo:

echo base_url();

index_page()

Retorna a página "index" do seu site, conforme especificado no seu arquivo de configuração. Exemplo:

echo index_page();

anchor()

Cria um link HTML baseado na URL local de seu site:

Clique aqui

A tag tem três parâmetros opcionais:

anchor(segmentos uri, texto, atributos)

O primeiro parâmetro pode conter qualquer segmento que você queira adicionar à URL. Como com a função site_url() acima, segmentos podem ser uma string ou um array.

Nota: Se você estiver montando links internos para sua aplicação não inclua a URL base (http://...). Isto será adicionado automaticamente conforme informação especificada no seu arquivo de configuração. Inclua apenas os segmentos de URI que você deseje adicionar à URL.

O segundo parâmetro é o texto que o link vai mostrar como clicável. Se deixado em branco então a URL será usada.

O terceiro parâmetro pode conter uma lista de atributos que você gostaria de adicionar ao link(tag). Os atributos podem ser uma simples string ou um array associativo.

Aqui estão alguns exemplos:

echo anchor('novidades/local/123', 'Minhas Novidades');

Produzirá: Minhas Novidades

echo anchor('novidades/local/123', 'Minhas Novidades', array('title' => 'As melhores!'));

Produzirá:<ahref="http://www.seu-site.com/index.php/novidades/local/123"title="Asmelhores!">Minhas Novidades

anchor_popup()

Quase idêntico à função anchor() exceto que abrirá a url em uma nova janela. Você pode especificar atributos Javascript no terceiro parâmetro para controlar como a janela será aberta. Se o terceiro parâmetro não estiver configurado então a nova janela será aberta conforme as configurações do navagador do usuário. Aqui está um exemplo com atributos:

```
$atributos = array(
        'width'     => '800',
        'height'    => '600',
        'scrollbars' => 'yes',
        'status'    => 'yes',
        'resizable' => 'yes',
        'screenx'   => '0',
        'screeny'   => '0'
      );
echo anchor_popup(novidades/local/123, 'Click Me!', $atributos);
```

Nota: Os atributos acima são funções padrão sendo que você somente precisará declará-las se precisar de algo diferente disso. Se você quiser que a função use todos os valores padrões basta passar um array vazio como terceiro parâmetro:

echo anchor_popup('novidades/local/123', 'Clique me!', array());

mailto()

Cria uma tag link para e-mail. Exemplo de uso:

echo mailto('eu@meu-site.com', 'Clique aqui para falar comigo');

Como com a tag anchor() acima, você pode declarar atributos usando o terceiro parâmetro.

safe_mailto()

idêntico à função acima exceto por escrever uma versão ofuscada da tag mailto usando números escritos com Javascript para ajudar a bloquear o rastreio dos emails por robôs de spam.

auto_link()

Transforma URLs e emails (que estão em strings) automaticamente em links. Exemplo:

$string = auto_link($string);

O segundo parâmetro determina se URLs e emails serão convertidos ou somente um ou outro. O padrão é a transformação de ambos se o parâmetro não for especificado

Converte apenas URLs:

$string = auto_link($string, 'url');

Converte apenas endereços de Email:

$string = auto_link($string, 'email');

O terceiro parâmetro determina se os links serão aberto em uma nova janela. O valor pode ser TRUE ou FALSE (booleano):

$string = auto_link($string, 'both', TRUE);

url_title()

Pega uma string como entrada e cria uma string URL amigável. É útil se, por exemplo, você tiver um blog no qual gostaria de usar o título de seus artigos na URL. Exemplo:

```
$title = "O que há de errado com o CSS?";
$url_title = url_title($title);
// Produz: o-que-ha-de-errado-com-o-css
```

O segundo parâmetro determina o delimitador de palavra. Por padrão traços serão usados. As opções são The second parameter determines the word delimiter. By default dashes are used. Options are: traço, ou underscore:

```
$title = "O que há de errado com o CSS?";
$url_title = url_title($title, 'underscore');
// Produces: o_que_ha_de_errado_com_o_css
```

prep_url()

Esta função adicionará http:// se estiver faltando na URL. Passe a string URL para a função deste modo:

```
$url = "www.algum-site.com";
$url = prep_url($url);
```

redirect()

Faz um "header redirect" (redirecionamento pelo servidor) para a URI local especificada. Assim como outras funções neste assistente, esta é designada para redirecionar a uma URL local dentro do seu site. Você não especificará a URL absoluta do seu site, mas somente os segmentos URI do controller que você quer abrir. A função montará a URL baseada nos valores declarados no seu arquivo de configuração.

O segundo parâmetro permite que você escolha entre o método "location" ou o método "refresh". O "location" é mais rápido, porém em servidores Windows pode ser um problema algumas vezes. Exemplo:

```
if ($logged_in == FALSE)
{
    redirect('/login/form/', 'refresh');
}
```

Nota: Para esta função funcionar ela deve ser declarada antes que qualquer coisa que gere uma saída ao browser, pois ela usa cabeçalhos de servidor(server headers).

Assistente de XML

O Assistente de XML contem funções que auxiliam no trabalho com dados XML.

Carregando este Assistente

Este assistente é carregado usando o seguinte código:

$this->load->helper('xml');

As seguinte funções estão disponíveis:

xml_convert('string')

Pega um string como entrada e converte os seguintes caracteres reservados para XML em entities:

- E comercial(&): &
- Maior que e menor que em caracteres: < >
- Aspas simples e aspas duplas: ' "
- Traços: -

Esta função ignora "Es comerciais"(&) se eles forem parte de entities de caracteres existentes. Exemplo:

$string = xml_convert($string);