به نام ایزد یکتا

آشنایی با فریم ورک
CodeIgniter

ارائه شده در همایش سراسری لمپ در ایران :: زنجان

zconf1

http://phpdevelopers.ir

نویسنده : فرید احمدیان

Ahmadian.farid@yahoo.com

۱۳۸۹

فهرست

- مقدمه
- CodeIgniter چیست؟
- CI برای چه کسانی هست؟
- معرفی کوتاه MVC
- روند اجرای برنامه (Application Flow Chart)
- آدرس‌های اینترنتی در کد اگنایتر(Code Igniter URLS)

  بخش‌های آدرس (URI Segment)

  حذف فایل ایندکس از آدرس (Removing the index.Php file)

  اضافه کردن پسوند به آدرس(Adding a URL Suffix)

  Enabling Query String
- کنتـرل کننده ها (controller)

  پاس دادن مقادیر uri به توابع(Passing URI segment to your funition)

  Remapping

  توابع خصوصی (Private function)

  مشخص کردن سازنده کلاس (Class constructors)

  اسامی رزرو شده در توابع(Reserved Function Name)
- Views
- کتابخانه‌ها (Libraries)
- کمک کننده ها (Helper Files)

  توسعه کمک کننده ها (Extending Helper)

  پیشوند خود را برای کمک کننده ها انتخاب کنید(Setting your own prefix)

  Using code igniter library
- Plugins

  Loading a Plugin
- فایل تنظیمات شخصی (Custom Config File)

- Language File

ساختن فایلهای زبان

Loading a Language File

- مباحث مرتبط به کار با پایگاه داده در CI

وصل شدن به دیتابیس

Automatically connection

Manually connection

Models

ساختار Model ها

Loading a model

وصل شدن به پایگاه داده در یک مدل

آشنایی با ابزار Scaffolding

انجام عملیات CRUD در CI

Active Record Class

SELECT

INSERT

UPDATE

DELETE

- Web page caching

کش چگونه کار می کند؟

فعال سازی کش

Deleting Caches

- اجرای یک مثال عملی در CI

دستورات نصب CI

ایجاد صفحه درج اطلاعات

ایجاد صفحه خواندن اطلاعات

ایجاد صفحه حذف مطالب

- کد های مثال

مقدمه

سالهاست php در ایران یک زبان شناخته شده برای طراحی وب است اما متاسفانه قالب‌های کاری (framework) آن در ایران شناخته شده نیستند و بعضا ما شاهد مقایسه‌های اشتباه بین ASP.NET که یه زبان طراحی وب بعلاوه یک قالب کاری هست، با زبان php بدون هیچ کدام از قالب‌های کاریش هستیم . این نشان دهنده تصور قالب در اکثر دانشگاهیان ما هست.

در این نوشته سعی دارم یکی از ساده‌ترین و سریعترین قالب‌های کاری php را معرفی کنم و آن چیزی نیس جز Code Igniter

این نوشته در قسمت اول به مفاهیم بنیادی میپردازد سپس سعی میکند اجزای اصلی این قالب را معرفی کرده و در آخر با آوردن یک مثال عملی ساده ، سعی بر اتمام هدف خود کند.

در جای جای این نوشته از مستندات خود Code Igniter به صورت ترجمه آزاد استفاده شده و در قسمت‌هایی از تجارب شخصیم استفاده کردم ،بدیهی است برای اطلاعات بیشتر و آشنایایی با دیگر مباحث این قالب توصیه می‌شود حتماً از مستندات رسمی آن استفاده شود :

http://codeigniter.com/user_guide/

به امید اینکه این مقاله گامی کوچک در فرهنگ سازی در مورد قالب‌های کاری آزاد باشد.

همچنین توجه به این نکته را لازم میدانم که فرض ما بر تسلط خواننده بر php و html و آشنایی کوچکی با مفاهیمی چون قالب کاری ، MVC و ORM است.

CodeIgniter چیست؟

CodeIgniter که از این به بعد، آن را به اختصار CI صدامیزنیم در حقیقت یه چهارچوب یا Frame Work به زبان php است که دارای دوره یادگیری بسیار کوتاهی هست و برای php کارانی ساخته شده است که به دنبال یک ابزارساده و مناسب و سریع برای ایجادسایتهایی با تمام قابلیتها هستند.

CI واقعاً سریع است به طوری که ایجاد کنندگان آن شما را برای پیدا کردن فریم ورکی با performance بالاتر به چالش میکشند!

در حقیقت CI یکی از بهترین گزینه ها برای اجرا بر روی هاست های به اشتراک گزاشته شده است و برای آنهاییست که از فریم ورک ها یا قالب‌های کاری سنگینو بزرگ که تماماً مستند نشده اند، خسته شده‌اند و به دنبال جایگزین میگردند.

CI از الگوی MVC یا (Model-View-Controller) استفاده میکنم و به این ترتیب لایه منطقی سایت و لایه ارائه از هم جدا می شوند.

این فریم ورک با یک محدوده ی وسیعی از کتابخانه‌هایی که معمولاً مورد استفاده قرار میگیرند عرضه شده این ، به عنوان مثال کتابخانه‌هایی برای:

accessing a database, sending email, validating form data, maintaining sessions, manipulating images, working with XML-RPC data and much more.

و در آخر این فریم ورک به سادگی با استفاده از پلاگین ها و کتابخانه‌های کمک کننده (plugins and helper libraries) قابل توسعه است.

CI برای چه کسانی هست؟

- برای کسانی که به دنبال یک فریم ورک جمع جور میگردند
- برای کسانی که به دنبال یک بادزهی استثنائی هستند
- آن‌هایی که به دنبال یک همخوانی بسیار بالا بین نرم افزارشان و هاست های استاندارد که نسخه های گوناگونی از php را به همراه تنظیمات گوناگون استفاده میکنند.
- برای آن‌هایی که به یک فریم ورک که به کمترین تنظیمات برای راه اندازی احتیاج دارند میگردند.

- به دنبال فریم ورکی میگردین که احتیاجی به خط فرمان نداشته باشد.
- برای آن‌هایی که به کتابخانه‌های بزرگ یکپارچه مانند PEAR علاقه‌ای ندارند.
- برای آن‌هایی که نمیخواهند مجبور شوند با یک زبان ایجاد پوسته(templating language) کار کنند(گر چه به صورت اختیاری در CI وجود دارد)
- برای آن‌هایی که از پیچیدگی اجتناب میکنند و به راه حل‌های ساده علاقه مندند
- و برای آن‌هایی اینکه به مستندات کاملاً واضح وشفاف نیاز دارند.

در قسمت‌های بعد با مفاهیم و اجزای اصلی CI آشنا شده و در آخر با یک مثال مقاله را به اتمام میرسانم:

معرفی کوتاه MVC

گفته شد که CI بر پایه‌های الگوی توسعه Model-View-Controller بنا شده است.MVC الگویی است که منطق برنامه را از لایه نمایش جدا میکند.

Model: مدل ساختار اطلاعاتی شما را مشخص میکند. به صورت معمول این لایه حاوی توابعی است که به شما در خواندن / درج / بروزرسانی اطلاعات از پایگاه داده کمک میکند.

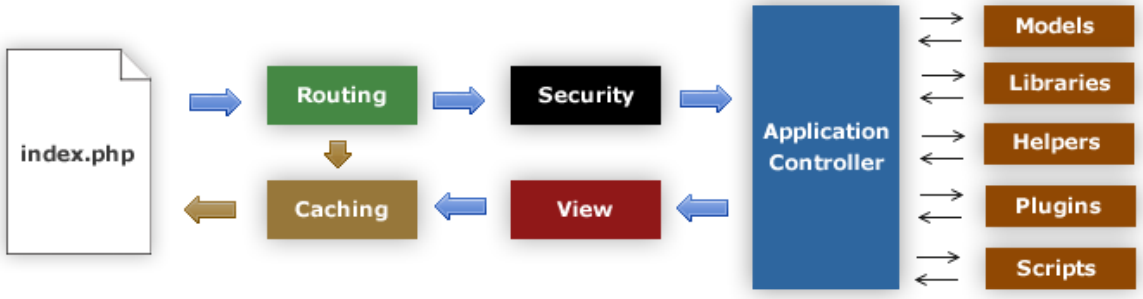
View: ویو اطلاعاتی است که به کاربر نشان داده میشود. معمولاً ویو یک صفحه وب است ولی در CI میتوانید جدا جدا باشد مثلاً header و footer ، همچنین ویو میتواند انواع مختلفی از صفحه باشد مثل صفحه rss.

Controller :کنترلر به عنوان یک میانجی بین view و model و سایر منابعی عمل میکند که برای جواب دادن به درخواست های HTTP و ایجاد صفحات وب لازم است.

نکته: CI در مورد پیاده‌سازی MVC سخت گیری نکرده و پیاده‌سازی model الزامی نیست.

توجه: توصیف دقیق MVC و علل و مزایای استفاده از آن خود، به مقاله‌ای جدا نیازمند است، بنابراین خواهشمندم با کمی جستجو در اینترنت این مطلب را ریشه‌ای تر مورد تحقیق و پژوهش قرار دهید.

روند اجرای برنامه Application Flow Chart



1. فایل index.php کنترلر خط مقدم را آماده می کند و منابع که باید بـرای اجـرای CI مقـدار دهی شوند را مقدار دهی می کند!

2. router درخواست HTTP را برای تصمیم گیری در مورد آن آزمایش می کند.

3. اگر فایل کش موجود بود بدون روند معمولی جواب درخواست فرستاده می سود.

4. قبل از بارگزاری Application controller درخواست HTTP و هر داده ای که کاربر فرستاده برای امنیت بیشتر فیلتر می شود.

5. کنترلر helper, core library, model و .. را بارگزاری می کند.

6. در مرحله آخر view آماده می شود و به مرورگر کاربر برای نمایش ارسال می شود البتـه اگر کش فعال باشد اول کش می شود تا به درخواست های بعدی بتواند جواب دهد.

آدرس‌های اینترنتی در کد اگنایتر(Code Igniter URLS)

به صورت پیش فرض URL ما در CI جوری طراحی شده اند که search- engine and human friendly باشند!

گرچه از رویکرد استاندارد "query string" هم می توان در سیستم داینامیک استفاده کرد امـا CI از یک رویکرد و شیوه segment- base استفاده میکند

مثال:

example.com/news/article/my- article

نکته: می توان به صورت دلخواه از query string هم استفاده کرد

بخش‌های آدرس (URI Segment)

بخش های موجود در URL روش MVC را دنبال می کنند که معمولاً به صورت زیر است:

Example.com/class/function/ID

1. قسمت اول segment نمایانگر کنترل کلاس مورد نظر است.

2. دومین قسمت segment نمایانگر تابع مورد نظر کلاس یا متود مورد نظر است.

3. سومین قسمت و قسمت های بعدی متغییرهایی است که به کنترلر داده می شوند.

کلاس URL Helper شامل توابعی است که کار با داده های داخل URL را آسان می کند. بعلاوه URLهای شما می توانند remap شوند که این کار توسط URL Routing انجام می شود! برای انعطاف بیشتر!!!

حذف فایل ایندکس از آدرس (Removing the index.Php file)

به صورت پیش فرض فایل index.php در URL شما قرار دارد

Example.com/index.php/news/article/my- article

شما می توانید به راحتی به کمک فایل htaecess این فایل را از URL حذف کنید در پایین یک مثال آمده که از روی روش negative استفاده می کنید و همه چیز و به جز موارد خاص redirect میکند!

```
RewriteEngine on

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

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

در مثال بالا هر درخواست HTTP به جز آن‌هایی که index.php و robots.txt و images هستند به یک درخواست برای index.php تبدیل می شود.

اضافه کردن پسوند به آدرس(Adding a URL Suffix)

در داخل فایل config/config.php شما می توانید یک پسوند برای URLهای که توسط CI ساخته می شوند مشخص کنید!

برای مثال:

```
example.com/index.php/products/view/shoes
```

که بعد از اضافه کردن پسوند:

```
example.com/index.php/products/view/shoes.html
```

Enabling Query String

در بعضی موارد شما خواهان استفاده از query string هستید.

```
index.php?c=products&m=view&id=345
```

CI به صورت اختیاری این امکان رو پشتیبانی می کند که می توان از مسیر /application config.php آن را فعال کرد.

اگر شما فایل فوق را باز کنید این خطوط را می بینید:

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

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

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

اگر شما enable- query را به TRUE تغییر دهید این ویژگی فعال می شود.

و از آن پس function , Controller شما توسط کلمات trigger قابل دسترسی هستند.

index.php? c=controller &m= method

نکته: اگر شما از روش query string استفاده کنید دیگر نمی توان از URL helperها و بعضی helperهای دیگر که URLتولید می کنند مثل From helperها استفاده کرد چون برای کار با Segment Base URLها طراحی شده است!

کنترل کننده ها (controller)

کنترلها قلب برنامه شما هستند و آنها هستند که تصمیم میگیرند که درخواستهای HTTP چگونه باید بکار گرفته شود.

در حقیقت کنترل یک کلاس است و وقتی که نام یک کنترل در اولین قسمت URL می آید، آن فراخوانی (Load) میشود. مثلاً:

Example.com/index.php/blog

که در مثال بالا یک کنترل به نام blog.php بارگذاری میشود.

کنترلها در مسیر application/controllers قرار دارند.

نکته: همانند Java هر فایل که در کنترل میسازیم باید شامل یک کلاس باشد که همنام با فایل ما باشد و آن ذکر میکند این کلاس سازنده است، همچنین این کلاس باید از کلاس Controller مشتق یا extend شود!

مثال:

```
<? php
```

```
Class Blog extends Controller {
        Function index ()
          {
              Echo 'Hello World';
           }
    }
?>
```

تابع index در هر کلاسی به صورت Default اجرا میشود.

مثلاً برای دیدن خروجی تابع بالا باید از آدرس:

Localhost/CodeIgniter/index.php/blog

استفاده شود.

نکته: فرم کلی آدرس به صورت index.php/classname/function است. البته گفته شد که در صورت ننوشتن function به صورت پیشفرض از index استفاده میشود و همچنین index.php را میتوان به سادگی با htaccess پاک کرد.

در این مرحله اگر ما آدرس Localhost/CodeIgniter را بزنیم با خطای 404 مواجه میشویم. بنابراین، باید مشخص کنیم به صورت پیشفرض کدام کلاس نمایش داده شود برای

اینکار به پوشه config رفته و فایل routes.php را باز میکنیم و در آنجا Default_Controller را تغییر میدهیم.

نکته: نام کلاس باید با یک حروف بزرگ شروع شود مانند Blog/Function همان طور که گفته شد function پیش فرض index است و اگر دومین قسمت از urIخالی باشد آن صدا زده می-شود و در حقیقت دومین قسمت urI مشخص کننده function ای است که Load می شود مثلاً:

example.com/index.php/blog/Comments

تابع comments از کلاس blog را صدا می زند!

پاس دادن مقادیر uri به توابع(Passing URI segment to your funition)

اگر urI شما بیش از دو قسمت داشته باشد، قسمت های بعدی به صورت پارامتر به تابع شما پاس داده می شوند!

برای مثال example.com/index.php/prodacts/shoes/sandal/123

به تابع شما قسمت های سوم و چهارم uri یعنی sandals و 123 پاس داده خواهد شد.

نکته: اگر شما از ویژگی urI Routing استفاده کنید مطالب بالا کمی متفاوت می شود!

Remapping

همانطور که دانستیم دومین قسمت از urI موجب بارگزاری تابع مورد نظر توسط کنترل مـی شود. CI امکان لغو این رفتار پیش فرض را با استفاده از تابع remap()- به ما می دهد.

نکته: اگر کنترل شما دارای تابع –remap باشد، همیشه از چیزی که در urI می آید صرف نظـر می کند و با لغو کردن رفتار پیش فرض به شما اجازه می دهد قوانین مسـیردهی خودتـان را ایجاد کنید!

توابع خصوصی (Private function)

در بعضی مواقع ممکن است قصد ایجاد تابعی را داشته باشیم که در دسـترس عمـوم نباشـد برای ایجاد یک تابع خصوصی کافی است علامث – یـا underscore را اول نـام تـابع مـان قـرار دهیم و آن تابع دیگر با استفاده از urI قابل دسترسی نیست!

اگر شما در حال ساخت یک برنامه بزرگ باشید شـاید مناسـب ببینیـد کـه بـرای سـازماندهی کنترل ها را درون زیر پوشه ها ببرید!

CI این اجازه را به شما می دهد که به سادگی با سـاختن پوشـه هـایی در مسـیر /application/ controllers این کار را انجام دهید.

نکته: وقتی شما این کار را بکنید، اولین قسمت urI شما به پوشه اشاره می کند

Example.com/index.php/products/shoes/show/123
_____ _____ _____

مشخص کردن سازنده کلاس (Class constructors)

اگر شما قصد قرار دادن یک سازنده درون کلاس تان دارید باید کد زیر را در آن قرار دهید!

Parent:: controller ();

علت ابتکار در آن است که سازنده اصلی کنترل توسط سازنده شـما تغییـر پیـدا مـی کنـد و برای همین نیاز است تا به صورت دستی صدا زده شود!

نکته: سازنده ها مفید هستند اگر شما ایجاد مقادیر اولیه پیش فرض یـا انجـام پـردازش هـای اولیه را داشته باشید، سازنده فاقد توانایی بازگرداندن مقدار هستند ولی مـی توانیـد کارهـای پیش فرض را انجام دهند!

اسامی رزرو شده در توابع(Reserved Function Name)

تا زمانی که کلاس های کنترل های ما از کلاس کنترل اصلی مشتق می شوند باید مواظب باشیم برای آنها از نامهای یکسانی با نام های کلاس اصلی یکی هستند استفاده نکنیم در غیر این صورت توابع محلی توابع اصلی را تغییر می دهند! برای دیدن لیست این توابع رزرو شده

http://codeigniter.com/user_guide/general/reserved_names.html

نکته: پر استفاده ترین رزرو شده ها index است.

Views

نکته: در هنگام ایجاد viewها در پوشه views زیباست که نامگذاری آنها مشخص کننده ی کنترل آنها باشد. مثلاً برای مثال فایل blog_view.php استفاده شود.

نکته: ما میتوانیم برای سازماندهی بهتر از subfolder ها در پوشه ی viewها استفاده کنیم..

نکته: برای وصل کردن view به control درون function موردنظر کنترل، از کد:

$this → load → view ('blog_view');

نکته: در هنگام عمل بالا ما میتوانیم از نام کامل فایل به همراه پسوند آن هم استفاده کنیم.

برای پاس دادن مقادیر از کنترل به view:

<!- فایل blog.php که در پوشه controllers است-->

```
<? php
    Class Blog extends controller {
        Function index () {
            $ data ['title']="My Blog title";
            $ data ['heading']="My Blog heading";
            $ this → load → view ('blog_view', data);
                }
            }
?>
```

<!-فایل blog_view.php که در پوشه view است-->

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

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

نکته: ما در فرستادن مقادیر به ویو محدود نیستیم مثلاً میتوانستیم آرایه هم پاس دهیم.

```
<!- blog.php -->
{
    $data ['todo']= array('clean hous', 'eat lunch', 'call mom');
}

<!- blog_view.php -->
{
<OL>
<? php foreach ($todo as $item){ ?>
<Li> <? echo $item; ?> </Li>
</OL>
}  <? php } ?>
```

نکته: درون کلاس کنترل ما اگر function ای هـمنـام بـا کلاس سـاخته شـود مـوجب خطـا در هنگام نمایش میشود که فکر کنیم به خاطر تداخل در سازنده هاست. به هر حال بـا گذاشـتن کد parent controller; درون آن فانکشن مشکل حل میشود!

-

کتابخانه‌ها (Libraries)

کلاسهایی که در مسیر system/Libraries و در system/application/Libraries هستند، در این دسته قرار دارند.

مثال برای Load:

```
$autoload['Libraries']= array('database','session');
```

کمک کننده ها (Helper Files)

همانطور که از نام helper برمیآید کدهایی هستند که ما را در انجام کمک میکنند. هر فایل helper یک مجموعه از توابع در گروهبندی خاص است. برای مثال:

URL helper یاری کننده در ساخت پیوندها

Form helper یاری کننده در ساخت فرمها

Text helper فراهم کنندهی روالهای فرمت دادن به متن

Cookie helper ایجاد و خواندن کوکیها

File helper به شما در کار با فایلها کمک میکند

نکته: متاسفانه برخلاف سیستمهای دیگر در CI کمک کنندهها با مِتُد شیگرایی نوشته نشده-اند. آنها به صورت ساده و ساختیافته توسط مجموعهای از functionها نوشته شدهاند. هر function یک وظیفه ی خاص را انجام میدهد و هیچ وابستگیای به سایر توابع ندارد.

کمک کنندهها در مسیر system/helpers و در system/application/helpers ذخیره میشوند. CI اول مسیر system/application/helpers را چک میکند، اگر فایل موردنظر را پیدا نکرد داخل مسیر system/helpers را چک میکند.

برای load یک helper یا از autoload فایل autoload.php استفاده میکنیم یا بهصورت موردی در کنترل بهصورت:

$this → load → helper('name');

مثال:

$this → load → helper(URL);

نکته: اسم بالا به طور کامل به صورت URL_helper.php است که هنگام load کردن بدون پسوند و قسمت helper صدا زدن میشود.

نکته: برای load چندین helper به صورت زیر میتوان عمل کرد:

$this → load → helper(array('helper1', 'helper2' , 'helper3'));

نکته: بعد از load کمک کننده موردنظر شما میتوانید از آن مثل یک تابع استاندارد php استفاده کنید. مثلاً برای ساختن پیوند:

<? php echo anchor ('blog/comments' , 'click here'); ?>

توسعه کمک کننده ها (Extending Helper)

برای توسعه helperها باید یک نام با همان نام helper که قصد توسعه ی آن را داریم اما با پیشوند My_ قبل از نام آن ایجاد کنیم و در مسیر application/helpers قرار دهیم. برای اطلاعات بیشتر به فایل helpers.html مراجعه کنید.

پیشوند خود را برای کمک کننده ها انتخاب کنید(Setting your own prefix)

ما میتوانیم پیشوند دلخواه خود را به جای My_ تعیین کنیم. برای اینکار به فایل application/config/config.php میرویم و مقدار

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

عوض میکنیم.

توجه به این نکته الزامی است که این پیشوند برای گسترش کلاسهای core و helper استفاده میشود.

نکته: توجه داشته باشید که تمام کتابخانههای محلی CI با پیشوند CI_ شروع میشوند؛ پس لطفاً از این پیشوند استفاده نکنید.

Using code igniter library

علاوه بر روشهای گفته شده، مثل سایر موارد گفته شده اگر نخواهیم بهصورت کلی کتابخانه ها را load کنیم از کد:

$this → load → library ('classname');

<!- Creating_Libraries.html -->

استفاده میکنیم.

Plugins

Plugin ها هم کار helper ها را میکنند. تفاوت اساسی در این است که Plug-in معمولاً یک تابع را فراهم میکنند ولی کمک کننده ها معمولاً یک مجموعه از توابع را در اختیار ما می-گذارند.

CI برای دسترسی به یک Plug-in مسیر system/application/ Plugins را چک میکند و اگر در آنجا چیزی یافت نکرد به مسیر system/ Plugins مراجعه میکند.

Loading a Plugin

برای بارگذاری یک Plugin استفاده میکنیم.

$this → load → Plugin ('captcha');

نکته: در مثال بالا نام کامل فایل captcha_pi.php است که از آوردن پسوند _pi خودداری می‌کنیم.

نکته: میتوان Plugin علاوه بر کنترل در ویو هم صدا زد، گرچه توصیه نمیشود.

نکته: برای بارگزاری چند پلاگین با هم (Loading Multiple Plugin) از خط زیر استفاده می‌کنیم.

$this → Load → Plugin (array(' Plugin 1',' Plugin 2'));

نکته: پس از بارگذاری، پلاگین موردنظر همانند یک تابع php معمولی قابل استفاده است.

فایل تنظیمات شخصی (Custom Config File)

به صورت پیشفرض CI یک فایل تنظیمات اصلی به آدرس

Application/config/Config.php

را دارا میباشد. تکتک تنظیمات در این فایل در آرایهای به نام $Config ذخیره میشوند.

ما میتوانیم Config item های خود را در این فایل اضافه کنیم یا فایل خود را ایجاد کرده و در پوشهی Config ذخیره کنیم.

نکته: اگر فایل خود را ساختید سعی کنید و حتماً باید به همان روش فایل اصلیConfig itemها را بسازیم یعنی در آرایهای به نام $Config قرار دهیم. CI بهصورت هوشمندانه این فایلها را مدیریت میکنند و از تداخل جلوگیری میکند.

نکته: برای علاوهبر روش اتوماتیک میتوان از Config File یک Load
$this → config → Load ('File name');

استفاده کنیم که File name اسم فایل بدون پسوند .php است.

نکته: برای به دست آوردن مقدار یک item config فایلها از دستور زیر استفاده میکنیم.

$this → config → item ('item name');

نکته: اگر آیتمی که سعی بر بدست آوردن مقدار شی داریم وجود نداشته باشد به ما مقدار FALSE بازگشت داده میشود.

برای تنظیم یک آیتم config فایل از روش زیر استفاده میکنیم:

$this → config → set-item ('item_name','item_value');

کلاس config کمک کنندههای زیر را فراهم میکنند.

$this → config → File_URL ();

این تابع URL سایت ما را به همراه index ی که در config مشخص کردیم میدهد.

$this → config → system_URL ();

این تابع آدرس "URL system folder" را به ما برمیگرداند.

Language File

کلاس فایل language توابعی را برای بین المللی سازی در اختیار ما میگذارد. فایلهای زبان به طور نمونه در مسیر system/ language ذخیره میشوند.

گرچه میتوان پوشه ی language را در مسیر system/application ساخت و فایلهای زبان را در آن ذخیره کرد. CI اول مسیر بالا را چک میکند اگر نبود به مسیر اصلی میرود.

نکته: فایلهای هر زبان باید درون یک پوشه ذخیره شوند برای مثال فایلهای زبان انگلیسی در مسیر system/ language/English هستند.

ساختن فایلهای زبان

فایلهای زبان باید دارای پسوند _lang.php باشند.

مثلاً: error_lang.php

درون این فایل هر خط اشارهای دارد به یک item آرایهای به نام $lang به این صورت:

$lang [' language_key'] = "the actual to be show";

نکته: برای جلوگیری از تداخل بهتر است نام item ها را با یک پیشوند مثلاً: error _ مشخص کنیم:

$lang ['error_email_missing'] = 'you must submit';

Loading a Language File

ما با کمک config.php به صورت پیشفرض زبانی را به طور کلی load میکنیم یا از

$this → lang → load ('File name',' language ');

استفاده میکنیم. File name فایلی است که میخواهیم load به درون php کنیم.

Language: and language is the language set containing it English

نکته: اگر پارامتر دوم را ندهیم بر اساس پیشفرض config مقداردهی میشود.

وقتی که فایل زبان load شد از روش زیر به خطوط آن دسترسی داشته باشیم.

$this → lang → line ('language_key');

مباحث مرتبط به کار با پایگاه داده در CI

وصل شدن به دیتابیس

پس از ایجاد دیتابیس و جداول، باید به دیتابیس وصل شویم. برای وصل شدن از مسیر system → application → config → database.php پارامترهای لازم را تنظیم میکنیم و CI به صورت اتوماتیک به DB وصل میشود. البته این وصل شدن به DB به دو صورت است:

وصل شدن به DB

Automatically connection :

اگر از ارتباط DB میخواهیم در تمام یا بیشتر صفحات استفاده کنیم از ویژگی auto connect در فایل application/config/autoload.php استفاده میکنیم. برای فعال کردن auto connection کلمه ی database را در فایل مذکور به صورت زیر اضافه میکنیم:

$autoload ['core']= array ('database');

برای اینکه تا حد ممکن CI سبک نگه داشته شود فقط یک سری منابع کوچک به صورت پیشفرض load میشوند. به وسیله ی فایل application/config/autoload.php میتوان منابع مورد نیاز را به صورت پیشفرض در تمام قالب load کرد.

چیزهایی که میتوان در قالب load کرد:

1- Libraries
2- Helper Files
3- Plug ins
4- Custom Config Files
5- Language Files
6- Models

Manually connection :

به صورت دستی Manually connecting می توانیم از کد زیر در هر تابع و یا در سازنده کلاس برای در دسترس قرار دادن DB در کل کلاس استفاده کنیم.

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

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

- **نکته:** اطلاعات بیشتر در مورد وصل شذن به DB در Connect.html

Models

Models به صورت اختیاری در دسترس کسانی است که میخواهند الگوی آنها بیشتر شبیه MVC باشد.

Model کلاس php است که برای کار با دیتابیس طراحی شده است. فرض کنید ما از CI برای ساخت یک blog استفاده میکنیم و میخواهیم از کلاس Model که حاوی توابعی برای درج، بازیابی و بروزرسانی است، استفاده کنیم.

مثال:

```
class Blogmodel extends Model {


   var $title   = '';

   var $content = '';

   var $date    = '';


   function Blogmodel()

   {

      // Call the Model constructor

      parent::Model();

   }


   function get_last_ten_entries()
```

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

    function insert_entry()
    {
        $this->title   = $_POST['title']; // please read the below note
        $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']));
    }

}
```

نکته: برای حفظ سادگی در این مثال ما از $_post به صورت مستقیم استفاده کردیم ولـی فرم صحیح‌تر آن:

$this → input → post('title')

کلاسهای Model در مسیر application/model ذخیره میشود و حتی میتوانید درون این مسـیرها از پوشه استفاده کنید.

ساختار Model ها:

```
class Model_name extends Model {


   function Model_name()

   {

      parent::Model();

   }

}
```

نکته: حرف اول کلاس Model شـما بایـد بـزرگ باشـد و بقیـه کوچـک و کلاس حتمـاً از کلاس Model مشتق شده باشد.

فایل مثال قبل /application/ model/ user- model.php

Loading a model

برای بار گزاری یک model می توان درون توابع controller از کد زیر استفاده کرد:

$this -> Load -> model ('model- name');

نکته: اگر model شما درون زیر پوشه ای قرار دارد برای مثال:

Applieation/ models/ blog/ queries.php

به صورت زیر load میکنیم

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

هنگاهی که Model مورد نظر بارگزاری شود می توانید به توابـع Model مـورد نظـر از طریـق object ای هم نام با Model بارگزاری شده از آن استفاده کنید:

```
$this -> load-> model ('Model- name');
```

```
$this->model- name->function;
```

مثال کلی:

```
class Blog_controller extends Controller {


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


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


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

اگر از یک model در برنامه مان خیلی استفاده میکنیم می توانیم توسط فایل /application/ config/ autoload.php اون رو به صورت کلی بارگزاری می کنیم!

وصل شدن به پایگاه داده در یک مدل

هنگامی که یک مودل بارگزاری می شود، اون به صورت خودکار به DB وصل نمی شود. گزینه های زیر برای وصل شدن در دسترسی هستند.

می توان از طریق استاندارد گفته شده از کلاس controller یا model بارگزاری کرد

می توان با TRUE کردن سومین پارامتر ورودی modelبه آن گفت که با استفاده از تنظیمات موجود در فایل تنظیمات DB به DB وصل شود.

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

شما می توانید به صورت دستی پارامترهای وصل شدن به DB را بفرستید

آشنایی با ابزار **Scaffolding**

ویژگی scaffolding فریم ورک cod igniter یک راه راحت را برای انجام اعمال اصلی "درج، حذف، به روز رسانی" اطلاعات در DB در طول روند توسعه است.

نکته بسیار مهم: Scaffolding فقط برای استفاده طراحان سایت است!

اون امنیت بسیار پائینی در حد یک secret word فراهم می کند بنابراین هر کسی که به سایت CI شما دسترسی داشته باشد می تواند بر روی اطلاعات DB شما تغییراتی بده ! بنا بر این در هنگام انتشار کار این ویژگی باید خاموش باشد.

اگر شما از ویژگی Scaffolding استفاده می کنید، بسیار ضروری است که بعد از استفاده اون رو غیر فعال کنید! و حتماً قبل از استفاده هم Secret word را تنظیم کنید:

برای تنظیم secret word به فایل application/ config/ routes.php مراجعه کرده و بعدبه متغیر مورد نظر مقدار مطلوب را بدهید. مثلاً:

```
$route [' scaffolding- trigger']= "scaffolding";
```

بعد از تنظیم secret word برای استفاده از scaffolding باید اون رو در تابع سازنده کلاسمون مقدار دهی کنیم، مثلاً:

```
<?php

class Blog extends Controller {


    function Blog()

    {

        parent::Controller();


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

    }

}

?>
```

نکته: table name نام جدولی است که ما قصد کار بر روی آن را داریم.

- پس از مقدار دهی بالا می توانیم با استفاده از الگوی داشته باشیم.

/Localhost/index.php/class/secret- word

برای مثال:

/Localhost/index.php/blog/scaffolding

نکته: Scaffolding فقط می تواند با جداولی که دارای primary key هستند کار کند!

نکته: برای کار با Scaffolding فرض شد کد شما با مباحث controllers آشنا هستید و به DB می توانید وصل شوید.

انجام عملیات CRUD در CI

پس از آشنایی با ساختار کلی و اعمال تنظیمات اولیه ما اکنون قادریم که تحت ساختار دلخواه (controll/model) شروع به مکالمه با دیتابیس نماییم! مکالمه با دیتابیس شامل

Create و Retrieve , Update و Delete است که به اختصار CRUD گفته میشود.

CI برای انجام CRUD به صورت پیش‌فرض از **Active Record** استفاده میکند که در حقیقت یک ORM است.

Active Record Class

CI از یک ورژن تغییر یافته الگوی Active Record استفاده می کند این الگو به ما اجازه بازیابی، بروز رسانی، درج و حذف اطلاعات جداول بانک اطلاعاتی را بوسیله اسکریپت های کوچک می دهد. در بعضی مواقع یک یا دو خط که برای یک عمل DB ضروری است! CI یک رابطه ساده برای کار با جداول Data Base فراهم می کند! علاوه بر سادگی ، استفاده از **Active Record** یک منفعت بسیار بزرگ هم دارد و آن بعد برنامه های مستقل از DB هست.

همچنین ویژگی دیگر آن امن کردن درخواست های شما توسط بررسی خودکار مقادیر ورودی به query است!

نکته: اگر شما قصد استفاده از query های خود را دارید می توانید این کلاس را در فایل config دیتا بیسی غیر فعال کنید!

نکته:

در پایین برای هر عمل در Active Record یک مثال آورده شده است اما برای دیدن لیست کامل آن به http://codeigniter.com/user_guide/database/active_record.html

مراجعه شود همچنین فرض گرفته شده است که شما بر روی SQL تسلط کافی دارید.

SELECT:

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


// Produces: SELECT * FROM mytable
```

```
$query = $this->db->get('mytable', 10, 20);


// Produces: SELECT * FROM mytable LIMIT 20, 10 (in MySQL. Other databases have slightly different syntax)
```

برای اضافه کردن یک شرط به دستورات بالا:

```
$query = $this->db->get_where('mytable', array('id' => $id), $limit, $offset);
```

INSERT:

```
$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')
```

UPDATE:

```
$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
```

DELETE:

```
$this->db->delete('mytable', array('id' => $id));


// Produces:

// DELETE FROM mytable

// WHERE id = $id
```

Web page caching

CI به شما اجازه کشی کردن صفحات وب برای دستیابی به بیشترین بازدهی رو می دهد گرچه CI کاملاً سریع است، اما فاکتورهایی مثل مقدار اطلاعات پویایی که نمایشی می دهیم، مرتبط است با منابع سرور، حافظه، مقدار Cpu مورد استفاده که بر روی سرعت Loud صفحات تاثیر می گذارد. بوسیله کش کردن صفحات تا زمانی که تغییر نکرده اند می تواند Performanee را نزدیک به صفحات ایستاد نگه داشت!!!

کش چگونه کار می کند؟

کش می تواند برای هر صفحه فعال شود و شما می توانید مقدار زمانی طول می‌کشد تا کش منقضی شود را معین کنید. وقتی صفحه ای برای بار اول loud می شود فایل کش در مسیر system/ cache نوشته می شود. در Loud بعدی درخواست کاربر از طریق کش جواب داده خواهد شد البته اگر منقضی شده باشد فایل کش پاک شده و درخواست مستقیم جواب داده خواهد شد.

نکته: تگ Bench mark کش نمی شود و شما می توانید در حالت کش هم Loud سرعت را ببینید.

فعال سازی کش

برای فعال سازی کش خط زیر را در کنترولر مورد نظر درج کنید:

$this->out put->cache (n);

N مدت زمان نگه داری کش است.

نکته: بخاطر نوع شیوه ذخیره سازی خروجی در CI کش کردن فقط در زمانی کار می کند که ما از view برای نمایش استفاده کنیم.

نکته: قبل از به کار انداختن سیستم کش ما باید مجوز نوشتن به مسیر System/ cache بدهیم.

Deleting Caches

اگر شما دیگر نمی خواهید کش در کنترلر شما باشد کافی است تگ آن را بردارید و فایل کش بعد از منقضی شدن پاک می شود اما اگر می خواهید این اتفاق سریع بیوفتد خود دستی باید فایل کش را از فولدر کش پاک کنید

اجرای یک مثال عملی در CI

در این مثال ما سعی داریم ، CI را راه اندازی کرده و تحت معماری MVC عملیات CRUD را بر روی یک جدول انجام داده و با کلیات کار در CI آشنا شویم.

در این مرحله ما یک پایگاه داده به اسم test_ci ساخته و یک جدول به اسم entries درون آن ایجاد میکنیم که شامل سه فیلد id , title و body است.

Entries

id
title
body

کد دستور SQL :

```
CREATE TABLE `test_ci`.`Entries` (

`id` INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
```

```
`title` VARCHAR( 200 ) NOT NULL ,

`body` TEXT NOT NULL

) ENGINE = MYISAM ;
```

دستورات نصب CI

0. دریافت CI از سایت اصلی آن (http://codeigniter.com)

1. Extract کردن پکیج
2. فایل ها و فولدر ها را در server آپلود کنید، به صورت معمولی index.php باید در شاخه اصلی باشد.
3. فایل system/application/config/config.php را باز کرده و base URL را تنظیم می کنیم. اگر شما قصد استفاده از session یا encryption را داریدآنها را تنظیم کنید.
4. اگر شما قصد استفاده از DB را دارید، فایل application/config/database.php را باز کرده و تنظیمات لازم را اعمال کنید.

نکته: اگر شما مایل به افزایش ضریب امنیت بوسیله پنهان کردن فایل های CI هستید کافی است پوشه system را تغییر نام دهید، اگر شما این کار را کردید باید فایل index.php را باز کرده و مقدار متغییر system- folder را با نامی که به پوشه دادید مقدار دهی کنید.

ایجاد صفحه درج اطلاعات

خوب در این مرحله ما قصد داریم تا اول صفحه ای بسازیم تا اطلاعات را در جدولمان درج کند پس ما می‌آییم مدل خود را به سایت اضافه میکنیم:

system/application/controllers/blog.php

و بعد درون آن کدهای اولیه لازم را بنا به مستنداتی که قبلاً در این مقاله مطالعه کردیم اضافه میکنیم:

```
class blog extends Controller {

    function blog(){

        parent::Controller();

                                }

}
```

خوب در این مرحله به کلاسمون تابع ای که قرار هست عملیات درج را انجام دهد را اضفه میکنیم و داخل آن در گام اول صفحه view این تابع را که قرار است ساخته شود را صدا میزنیم:

```
function insert(){

$this->load->view('blog/view_insert');

}
```

خوب بیایید صفحه view مورد نظر را بسازیم، برای ایجاد آن درون مسیر system/application/views یک پوشه به نام blog ایجاد کرده و دورن آن فایل view_insert.php را میسازیم و کدهای زیر را داخل آن درج میکنیم:

```
<html>

	<head>

		<title>

			Blog Insert Page

		</title>

	</head>

	<body>

		<p>Hello World!</p>

	</body>

</html>
```

خوب برای آزمایش کافیه مسیر پروژه را به اضافه اسم کلاس و نام تابع مان در مرورگر بزنیم:

http://localhost/CodeIgniter_1.7.2/index.php/blog/insert

اگه همه چیز درست باشد شما باید پیغام Hello World را در صفحه مشاهده کنید.

خوب در گام بعد ما سعی میکنیم ک فرم ورود اطلاعات را به صفحه خود اضافه کنیم ، برای این کار ما میخواهیم از کمک کننده ساخت جدول استفاده کنیم (form helper) . برای این کار در مرحله اول، در تابع insert کلاسمان باید کمک کننده فرم را بارگزاری کنیم:

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

حالا میتوانیم از ابزاری که این کمک کننده در اختیار ما گذاشته استفاده کنیم ، به جای Hello World فایل View کدهای زیر را درج میکنیم:

```
  <?php

     echo form_open('blog/insert');
```

```
echo form_hidden('id',0);
echo form_label('title', 'title');
   echo('<br>');
echo form_input('title');
echo('<br>');
echo form_label('body', 'body');
   echo('<br>');
   $textarea = Array ("name" => "body", "cols" => "70");
   echo Form_textarea($textarea);
echo('<br>');
echo form_submit('', 'Submit new post!');
echo('</form><br><hr>');
?>
```

مجموعه کدهای بالا برای ما یک فرم ورود اطلاعات میسازد که برای ساخت آن تماماً از کمک کننده CI استفاده شدهو فکر میکنم به اندازه کافی گویا هستند ولی برای آشنایی کامل با توابعی در قسمت بالا استفاده شده و همچنین دیگر امکانات این کمک کننده به مرجع اصلی آن مراجعه کنید :

http://codeigniter.com/user_guide/helpers/form_helper.html

و در آخرین مرحله ، ما یک model اضافه کرده و در آن عمل درج در پایگاه داده را هم پیاده میکنیم.

نکته : توجه کنید برای کار کردن مدلتان نکاتی که در طول مقاله گفته شد باید حتماً رعایت شوند ، آن نکات عبارت اند از تنظیم فایل database و فایل autoload تا CI بتواند در مرحله اول به پایگاه داده وصل شود و بعد از آن فرامین مدل را اجرا کند.

خوب بعد از تنظیم این دو فایل ما به مسیر /system/application/models رفته و فایل BlogModel.php را ایجاد کرده و کد های زیر را در آن درج میکنیم:

```
<?php
class BlogModel extends Model {
   function BlogModel()
   {
```

```
        // Call the Model constructor
        parent::Model();
    }

    function insert()
    {
            $this->db->insert('Entries', $_POST);
        }
}
?>
```

تا جای ممکن کد های بالا در قسمت 'مباحث مرتبط به کار با پایگاه داده در CI ' توضـیح داده شده برای اطلاعات بیشتر به مستندات رسمی مراجعه نمایید.

اما ما هنوز مدلمان را به کنترولر معرفی نکردیم برای این کار باید اول مدل خود را در کنترلر بارگزاری کرده :

```
    function blog(){
        parent::Controller();
            $this->load->model('BlogModel');
            }
```

و سپس تابع مورد نظر را صدا می زنیم:

```
            function insert(){
                $this->load->helper('form');
                $this->BlogModel->insert();
                $this->load->view('blog/view_insert');
                    }
```

اگه تمام مراحل بالا با موفقیت انجام شود ، اطلاعات بـه درسـتی در جـدولمان درج میشـود. توجه به این نکته ضروری است که ما در این مثال سعی داریـم تـا جـای ممکـن ایـن مثـال را ساده نگه داشته و از پیچیدگی آن جلو گیری کنیم و لزوماً روشـهای کـه مـا بـرای رسـیدن بـه هـدف اتخـاب کردیـم صـحیح‌ترین دوش نیسـت. در اینجـا هـم مشـاهده میکنیـم پـس از درج اطلاعات کاربر در همان صفحه میماند ! در این قسمت قصد ما راهنمـایی کـاربر پـس از درج اطلاعات به صفحه مدیریت محتویات وبلاگ است که هنوز آن را ایجاد نکرده ایم.

پس باید چک کنیم اگر اطلاعات فرم فرستاده شده بودن پس از درج آن را با استفاده از تابع redirect کمک کننده url به صفحه مورد نظر منتقل کنیم.

برای این کارها اول کمک کننده url را در کنترلر خود بارگزاری کرده:

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

و سپس قسمت فراخوانی تابع insert را به صورت زیر تغییر می‌دهیم :

```
if(isset($_POST['id']))

{

	$this->BlogModel->insert();

	redirect('blog/main');

}
```

این کد باعث می‌شود پس از درج کاربر به صفحه main منتقل شود که البته آن را هنوز نساخته ایم! البته بهتر بود برای چک کردن id از کلاس Input Class استفاده میکردیم که برای کاهش پیچیدگی از آن صرف نظر کردیم. در این قسمت عمل درج ما تمام شده و به سراغ صفحه main می‌رویم تا عملیات خواندن اطلاعات و مدیریت آن‌ها را پیاده‌سازی کنیم.

ایجاد صفحه خواندن اطلاعات

خوب برای ایجاد این صفحه تابع main را به کنترل و تابع show را به مدل اضافه میکنیم و برای نمایش دادن اطلاعات فایل view_main.php را به پوشه blog قسمت view اضافه میکنیم:

```
function main(){

	$data['query'] = $this->BlogModel->show();

	$this->load->view('blog/view_main', $data);

}
```

در اینجا ما خروجی تابع show مدل را که شامل فیلدهای جدول Entries را درون اندیس query آرایه data میریزیم و سپس در هنگام فراخوانی view آن را به ویو خود پاس میدهیم ، تابع show مدل ما:

```
function show()

{

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

	return $query->result();
```

```
}
```

و در آخر کدهای صفحه view_main.php ما:

```
<html>
    <head>
        <title>
            Blog Main Page
        </title>
    </head>
    <body>
    <h1>Content Management</h1>
    <ul>
        <?php
            foreach ($query as $row){
                echo "<li>";
                echo "<h3>".$row->title."</h3>";
                echo "<p>".$row->id."</p>";
                echo "</li>";
            }
    ?>
    </ul>
    </body>
</html>
```

ایجاد صفحه حذف مطالب

خوب در این زمان ما توانستیم اطلاعات لازم را نمایش دهیم، هدف بعـدی مـا پـاس دادن ID محتوای مورد نظر به یک صـفحه دیگـر بـرای پـاک کـردن آن اسـت! بنـابر ایـن لازم اسـت تـا قسمت ID را در صفحه نمایش فوق به یک لینک تبدیل کنیم ، برای ایـن کـار از یکـی دیگـر از ابزاری که کمک کننـده url در اختیـار مـا میگـزارد اسـتفاده میکنیـم و آن ابـزار anchor اسـت! کافیه به جای خطی که ID را چاپ میکرد خط زیر را جای گزاری کنید:

```
echo anchor('blog/delet/'.$row->id, 'Delete');
```

برای اطلاع از تمام جزییات کمک کننده url و آشنایی دقیق با تمام توابع آن به لینک زیر مراجعه نمایید:

http://codeigniter.com/user_guide/helpers/url_helper.html

و در پرده آخر این نمایش که من قرار است آن را اجرا کنم! باید آن صفحه‌ای را بسازیم که لینک فوق به آن اساره میکند . برای این کار باز هم کافیست به model مان یک تابع اضافه کنیم :

```
function delete(){

        $this->db->delete('Entries', array('id' => $this->uri->segment(3)));

       redirect('blog/main');

}
```

خوب در صفحه delete قرار نیست چیزی به کاربر نمایش داده شود پس فاقد view است و همچنین در این مقاله اشاره شد که CI اسراری در استفاده از model ندارد! پس در این تابع ما مستقیماً بدون فراخوانی هیچ مدلی اقدام به پاک کردن رکورد مورد نظر از پایگاه داده می نماییم !

این هم مثال این مقاله ، اما برای اینکه این مقاله جنبه عملی هم داشته باشد توصیه میکنم مثال بالا را تکمیل کرده و به امکانات بالا امکان بروزرسانی را هم اضافه نمایید و یک صفحه جدا برای نمایش کامل محتویات ایجاد نمایید ، سپس با اضافه کردن یک جدول امکان نظر دادن به هر مطلب را طراحی کنید، کدهای کامل مثال بالا در انتها زمینه می‌شود.

با آرزوی اینکه این مقاله توانسته باشد به هدف خود که همانا آشناکردن و علاقه مند کردن شما به قالب‌های کاری php و به خصوص Code Igniter بود ، رسیده باشد . در همین راستا خواهش میکنم مشکلات موجود در این مقاله و پیشنهادها و انتقادات خود را به پست الکترونیکی Ahmadian@pitm.net ارسال نمایید و در هر چه بهتر نمودن این مقاله به این بنده حقیر یاری نمایید.

کد های مثال:

فایل system/application/controllers/blog.php

```
<?php
class blog extends Controller {
	function blog(){
	   parent::Controller();
		$this->load->model('BlogModel');
		$this->load->helper('url');
				}

		function main(){
			$data['query'] = $this->BlogModel->show();
			$this->load->view('blog/view_main', $data);
		}

		function delete(){
			 $this->db->delete('Entries', array('id' => $this->uri->segment(3)));
			redirect('blog/main');
		}


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

```
                if(isset($_POST['id']))
                {
                        $this->BlogModel->insert();
                        redirect('blog/main');
                }
                        $this->load->view('blog/view_insert');
                                                }
}
?>
```

کدهای system/application/models/blogmodel.php

```
<?php

class blogmodel extends Model {



   function blogmodel()
   {
      // Call the Model constructor
      parent::Model();
   }

   function show()
   {
                        $query = $this->db->get('Entries');
                return $query->result();
        }
```

```
    function insert()
    {
                $this->db->insert('Entries', $_POST);


        }
}
?>
```

کدهای فایل system/application/views/blog/view_insert.php

```
<html>
        <head>
                <title>
                        Blog Insert Page
                </title>
        </head>
        <body>
                <?php
      echo form_open('blog/insert');
      echo form_hidden('id',0);
      echo form_label('title', 'title');
         echo('<br>');
       echo form_input('title');
       echo('<br>');
       echo form_label('body', 'body');
         echo('<br>');
         $textarea = Array ("name" => "body", "cols" => "70");
```

```
    echo Form_textarea($textarea);
    echo('<br>');
    echo form_submit('', 'Submit new post!');
    echo('</form><br><hr>');
    ?>
    </body>
</html>
```

کدهای فایل system/application/views/blog/view_main.php

```
<html>
    <head>
        <title>
            Blog Main Page
        </title>
    </head>
    <body>
    <h1>Content Management</h1>
    <ul>
        <?php
            foreach ($query as $row){
                echo "<li>";
                echo "<h3>".$row->title."</h3>";
                echo anchor('blog/delete/'.$row->id, 'Delete');
                echo "</li>";
            }
    ?>
    </ul>
```

```
        </body>
</html>
```