آموزش سریع کد ایگنایتر

جلد اول

CodeIgniter

نویسنده : پیروز جنابی

دی ماه ۹۵

WWW.piero.ir

فهرست مطالب

نویسنده : پیروز جنابی

برنامه نویسان حرفه ای PIERO
نویسنده : پیروز جنابی
www.piero.ir

مقدمه و پیشگفتار نویسنده

امروزه نیاز به برنامه نویسی جهت ساخت و سفارشی سازی برنامه های کامپیوتری هر روز بیشتر از قبل است . ضمن اینکه هر روز ما بیشتر به سمت اینترنت گرایش پیدا می کنیم و برنامه ها از حالات تکی و یک جا به سمت برنامه های تحت وب می رود ،که دارای مزایای زیادی است از جمله **وابسته نبودن به جا و مکان ، امنیت بالا ، عدم وابستگی به سیستم عامل خاص ، قدرت مانور بالا و ...**

ار آن رو بر آوردیم که یکی از قوی ترین فریم ورک های پی اچ پی را برای شما آموزش دهیم . اما دو سوال مطرح می شود :

چرا پی اچ پی ؟ نمی توان کدام زبان برنامه نویسی برتر دانست ولی زبان برنامه نویسی پی اچ پی پر کار برد ترین زبان تحت وب می باشد که سالیان زیاد است که شرکتهای بزرگ از آن استفاده می کنند و چون بازمتن می باشد دارای هزاران فریم ورک ، ویرایشگر و ماژول های رایگان در اینترنت می باشد .

چرا کد ایگنایتر ؟ و باز هم نمی توان گفت کدام فریم ورک قوی تر می باشد ولی کد ایگنایتر در بسیاری از آمارهای اینترنتی در مقام اول و یا دوم قرار دارد و این بخاطر راحتی و قدرت مانور بالای آن می باشد . شما با کد ایگنایتر می توانید برنامه های تحت و یا وب سایتهای پر قدرت بسازید.

مدتها پیش سعی بر آن آوردیم بتوانیم کتاب آموزش کامل کدایگنایتر را در اختیار هم وطنان عزیزمان قرار دهیم ، در حال حاضر این کتاب جلد شماره یک می باشد و هر جلد به صورت دوره ای از مطالب وب **سایت پیرو (piero.ir)** انتشار می یابد و شما می توانید جدید ترین نکات کد ایگنایتر را در وب سایت ما ببینید و استفاده کنید. همچنین شما نیز می توانید مقالات خود را برای ما ارسال نمایید تا با اسم شما انتشار یابد.

درصورت بروز مشکل خوشحال می شویم بتوانیم کمکی به شما بکنیم : info@piero.ir – 03136519040

امیدوارم بتوانم به همراه تیم وفادارم خدمتی به شما دوستان کرده باشم – پیروز جنابی (مدیر و موسس پیرو)

فریم ورک چیست و دلیل استفاده از فریم ورک

فریم ورک قالبی است که شما در آن کدهای خود را در حالت و فرم آن فریم ورک می نویسید . به طور عامیانه بخواهم فریم ورک را برسسی کنیم می توانیم به یک سری کد بگوییم که افرادی قبلا روی آن کار کردن عملیاتهایی را برای آن تعریف کردن ، شما می توانید از ان عملیاتها استفاده کنید و کدهای بهتری بسازید.اگر بخواهم در دنیای واقعی مثالی بزنم بهتر است ساخت یک خودرو را مثال بزنم :

مثال ساخت خودرو:

فرض کنید شما می خواهید خودرویی بسازید دو راه در پیش رو دارید :

1- ماشین را از اول بسازید (آهن را ذوب کرده رنگ را تولید کرده و تک تک قطعات را ساخته و بعد مونتاژ کنید). این راه کار ممکن است سالهای طولانی طول بکشد تا شما بتوانید یک ماشین تولید کنید در ضمن ماشین بسیار ضعیفی می سازید که فقط و فقط خودتان از آن اطلاع دارید و خودتان می توانید درستش کنید.

2- از قطعات کمپانی های دیگر به طور مثال اگزوز موتورهای برقی پنجره را از کمپانی بخریم و کمپانی دیگری چراغ های ما را تولید کنند و کیت های الکترونیکی به همین صورت در آخر ما قطعات سفارشی خودمان را بسازیم و ترکیب کنیم خب در این حالت سریع تر به خواسته هایتان می رسید و طبیعتا محصول شما کیفیت بهتری دارد و چون از **استاندارد خاصی** پیروی می کند **کسانی که به این استاندارد مسلط باشند هم می توانند این ماشین را تعمیر کنند و هم توسعه دهند و هم گروهی از افراد به طور همزمان روی این ماشین کار می کنند** .

مقایسه مثال خودرو با برنامه نویسی:

و حالا اگر شما کدتان را از پایه می نویسید در واقع شما مثل این می ماند که ماشین را از اول بسازید(مثال 1) . در واقع خیلی طول می کشد استاندار نیست و ملزوم به استفاده از استاندارد خاصی ندارید ، هم خیلی زمان بر و سخت است قطعات هم که باید خودتان از اول بسازید .

ولی اگر از فریم ورک استفاده کنید مثل راهکار دوم ساخت خودرو می ماند .سریع قطعات را کنار هم گذاشته با کیفیت بهتر و سریعتر تحویل مشتری می دهید . از استاندارد خاصی پیروی می کنید که در حال حاضر این استاندارد ام وی سی (mvc) است . پس هر فردی که به این معماری آشنا باشد می تواند با آن کار کنید و هم قابل توسعه می باشد.در ضمن کدهایی که قبلا ساخته شده و شما از آنها استفاده می کنید توسط متخصصین به بهترین نحو ساخته شده که باعث می شود برنامه شما بسیار بهینه باشد.

چرا کدایگنایتر؟

خب حالا که به این نتیجه رسیدید که باید از فریم ورک استفاده کنید ، بنظر من بهترین فریم ورک برای شما **code igniter** است البته من نمی گویم بهترین فریم ورک دنیا این فریم ورک است ولی سادگی و راحتی آن باعث می شود که شما بسیار راحت فرا بگیرید .

مزایای کد ایگنایتر:

- ✓ بسیار ساده و قابل فهم است و شما به راحتی می توانید کدهایتان را با قدرت زیاد پیاده سازی کنید
- ✓ کد نویسی از پایه
- ✓ سرعت بالا و استفاده بهینه از فضا و رم
- ✓ دارا بودن داکیومنت بسیار کامل.
- ✓ استفاده از معماری **mvc** (که در اینده به آن می پردازیم).

بنابراین برای شما هم اکنون بهترین گزینه کدایگنایتر می باشد همچنین توسط کد ایگنایتر شما ساختار فایلی خود را می توانید تغییر دهید و هیچ محدودیتی وجود ندارد وبسیار **url** های بهینه ای دارد و متد سئو را بسیار خوب پشتیبانی میکند.

معماری MVC

مهماری ام وی سی مخفف **M**odel **V**iew **C**ontroller می باشد این معماری که در حال حاضر محبوب ترین معماری می باشد شامل سه قسمت می باشد:

MODEL

قسمت مدل محل قرار گیری کدای کار با دیتا بیس می باشد و توسط کلاسها تعریف می شود و از کلاس مرجع مدل انشعاب می گیرد به مدل زیر توجه نمایید.

```
Class piero_model extend CI_Model
{
  Function add (){//some code}
  Functiom edit () {//some code}
}
```

در کد ایگنایتر یک سری عملیاتها مانند پرس و جو و کار با دیتابیس به صورت بهینه نوشته شده که شما به راحت ترین راه و بهینه ترین روش می توانید آنها را استفاده نمایید.

VIEW

در این قسمت فایلهای نمایشی قرار دارد در واقع کدهای ...,**angular** ,**jquery** ,**js** , **css** , **html** و هر جیزی که به کاربر نمایش داده می شود.

Controller

کنترلر ها مد لها و ویو ها را ترکیب می کند و عملیاتهای کنترلی را ما بینشان انجام می دهد.

برای توضیح بیشتر در مورد ام وی سی می تواند به وب سایت www.piero.ir **مراجعه نمایید و مقاله ام وی سی آن را دانلود نمایید.**

شروع کار با CI

CIمخفف **Code Egniter**می باشد که در بعضی جاها به صورت اختصار از آن استفاده می کنیم.

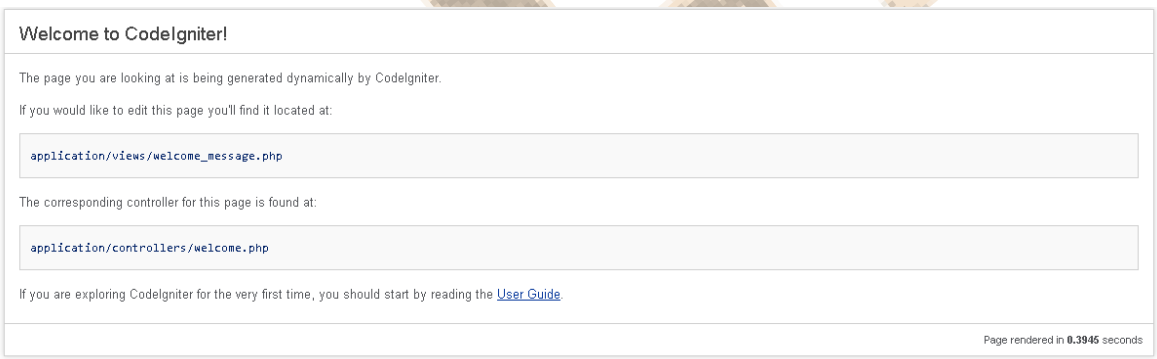
طریقه نصب و راه اندازی

برای نصب کد ایگنایتر کافی است آن را دانلود کرده و در هاست خود بگذارید .

1- دانلود از لینک : https://www.codeigniter.com/download

2- فایل زیپ را استخراج کرده و کدهای آن را اجرا می کنیم .

کار نصب ما تمام شد بعد از اجرا می بینید که صفحه نمایش می دهد که این صفحه نمایانگر مثالی است به تصویر زیر توجه نمایید.



همانطور که می بینید این صفحه یک مثال کاربردی است که نشان می دهد :

برای تغییرات در فایل نمایشی به آدرس زیر مراجعه نمایید :

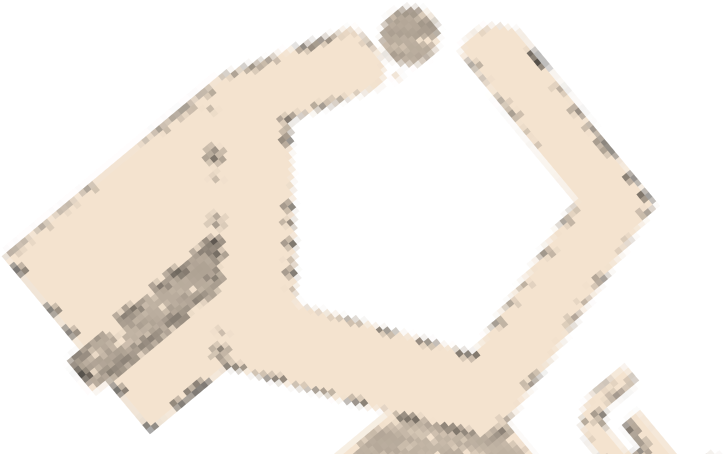
```
application/views/welcome_message.php
```

در این فایل کدهای `html ,css` جهت نمایش این صفحه است.

برای نمایش تغییر در فایل کنترلر به آدرس زیر مراجعه نمایید:

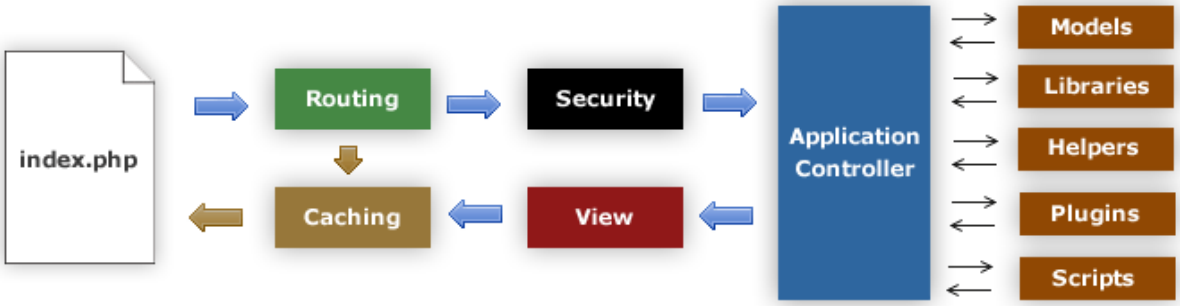
```
application/controllers/welcome.php
```

در این فایل توابع جهت بارگزاری فایل نمایشی وجود دارد



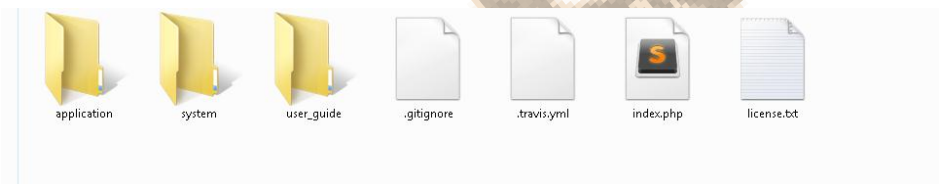
ساختار

به شکل زیر توجه نمایید:



تمامی در خواستها به فایل **index.php** ارسال می شود . توسط **routing** و چک شدن امنیتی به قسمت کنترلر مراجعه می نماید در آنجا اطلاعات از مدل ها یا کتابخانه ها و ... برداشت شده وتوسط **view** کش می شود و به کاربر نمایش داده می شود. این ساختار انجام کار در کد ایگنایتر می باشد.

ساختار پوشه ها



همانطور که مشاهده می کنید ساختار پوشه کد ایگنایتر همانند تصویر بالا است که به اختصار هرکدام را توضیح می دهیم:

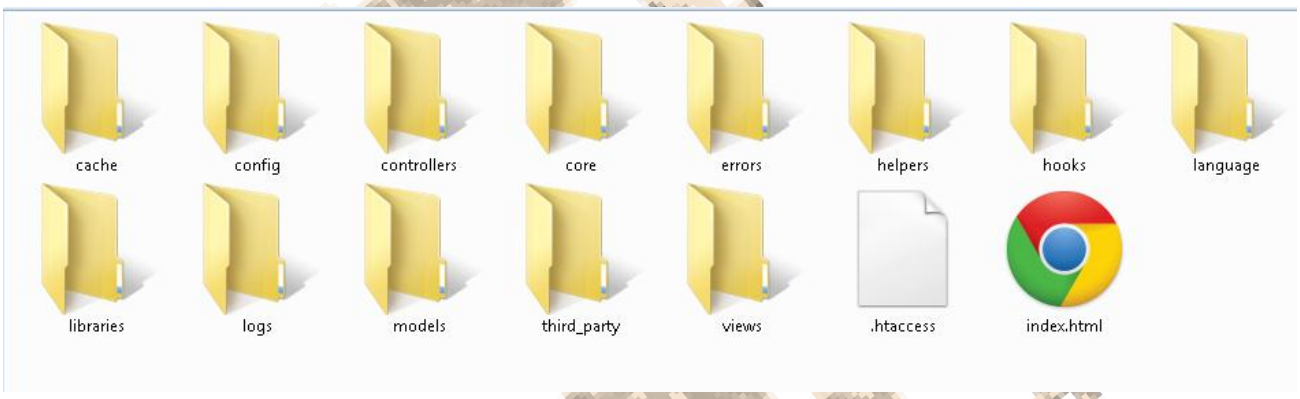
Application: کلیه کد های شما و برنامه شما در این پوشه قرار می گیرد و می توانید از این پوشه چند تا داشته باشید به طور مثال پوشه ای برای قسمت کاربری و پوشعه ای برای مدیریت همچنین میتوانید نامش را عوض کنید و در فایل **index.php** تغییر دهید

System: کلیه هسته برنامه در این پوشه قرار دارد می توان پوشه هسته را در قسمت **root**سایت قرار داد که کاربر دسترسی به این پوشه نداشته باشد .

User guid: کلیه موارد راهنمایی در این پوشه قرار دارد.(اجباری به وجود این فایل نیست و شما می توانید آن را حذف نمایید).

Index.php: فایل اصلی اجرای برنامه

ساختار پوشه Application



:. تنظیمات برنامه در این پوشه قرار دارد به طور مثال تنظیمات اتصال به بانک اطلاعاتی**Config**

:. کلیه فایل های کنترل کننده و هدایت گر در این پوشه قرار می گیرد **Controllers**

:. فایلهای برای تعریف زبانهای مختلف**Languages**

: کلیه فایل های کتابخانه ای به طور مثال کتابخانه قالب و ...**Libraries**

: کلیه فایلهای کار با بانک اطلاعاتی در اینجا قرار می گیرد**Models**

: و کلیه فایلهای نمایشی به کاربر در اینجا قرار می گیرد**Views**

: این فایل که مختص به اپاچی بوده یک سری تنظیمات منحصر به فرد را برای سرور شما فراهم می سازد از جمله تنظیمی جهت **.htaccess** ها که در آینده به آن می پردازیم. **url**بهینه سازی

سیستم مورد نیاز جهت اجرای کد ایگنایتر

PHP version 5.4 or newer is recommended.

دیتابیس هایی که در کد ایگنایتر پشتیبانی می شود:

MySQL (5.1+) via the *mysql* (deprecated), *mysqli* and *pdo* drivers
Oracle via the *oci8* and *pdo* drivers
PostgreSQL via the *postgre* and *pdo* drivers
MS SQL via the *mssql*, *sqlsrv* (version 2005 and above only) and *pdo* drivers
SQLite via the *sqlite* (version 2), *sqlite3* (version 3) and *pdo* drivers
CUBRID via the *cubrid* and *pdo* drivers
Interbase/Firebird via the *ibase* and *pdo* drivers
ODBC via the *odbc* and *pdo* drivers (you should know that ODBC is actually an abstraction layer)

URLدر کد ایگنایتر

URL و یا آدرس ها در کد ایگنایتر به صورت کاملا بهینه می باشد به مثال زیر توجه نمایید:

```
example.com/news/article/my_article
```

این آدرس نشان می دهد که در سایت **example** قسمت **news**و ...

این آدرس هم کاربر پسند می باشد و هم ایده ال برای پیاده سازی ،این آدرس به راحتی توسط کد ایگنایتر قابل پیاده سازی می باشد .

```
example.com/class/function/ID
```

همان گونه که مشاهده می کنید ساختار آدرسها در کد ایگنایتر بسیار کار آمد می باشد که در زیر شرح می دهیم:

1- پارامتر اول اشاره دارد به کلاس که در قسمت کنترلر قرار دارد.
2- پارامتر دوم اشاره دارد به تابع که در ان کلاس فرار دارد..
3- وقسمت سوم که اشاره دارد به مقداری که به آن تابع ارسال می شود.

توجه:کتابخانه **URL** (*URI Library*) تمامی توابع را برای مدیریت راحت تر آدرسها فراهم می سازد.

حذف فایل index.php از آدرسها

به طور پیش فرض در سرورها آدرس به صورت زیر می باشد

```
example.com/index.php/news/article/my_article
```

که شما به راحتی با تنظیم وب سرور آپاچی می توانید از نمایش فایل **index.php** جلوگیری نمایید و آدرسی عرضه نمایید که از دید کاربر بسیار بهتر و مورد پسند تر می باشد.

کافی است که اطلاعات زیر را در فایل **httaccess.** قرار بدهید:

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
```

افزودن پسوند به آدرسها

شما به راحتی می توانید به آدرسهای خود پسوند اضافه نمایید به طور مثال به آخر آدرس ها **html.** اضافه کنیدهمانند مثال زیر:

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

فقط کافی است به داخل فایل **config.php** رفته و پارامتر زیر را مقدار دهی نمایید.

```
$config['url_suffix'] = '.htm;'
```

فعال سازی query string

بعضی از مواقع شما نیاز دارید از فرمت **query string** استفاده نمایید.

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

در فایل **config .php** مقادیر زیر را وارد نمایید

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

بعد از انجام عملیات فوق آدرس ها به صورت زیر در می آید.

```
index.php?c=controller&m=method
```

کنترلر ها در کد ایگنایتر

کنترلر ها کلاسی هستند که رابط کاربر با برنامه می باشند و آدرسها را مدیریت میکنند همچنین رابط بین **model** و **view**می باشند .

```
example.com/index.php/blog/
```

در مثال بالا فایل **Blog.php**که در پوشه **controller**قرار دارد را بارگزاری می کند و هچنین تابع **index**که به صورت پیش فرض می باشد. در واقع وقتی ما آدرسی را بدون تابع صدا بزنیم تابع **index**فراخوانی می شود.

```
<?php
class Blog extends CI_Controller {

        public function index()
        {
                echo 'Hello World!';
        }
```

```
}
```

در صورتی که شما فایل **Blog.php**را همانند مثال بالا بسازید و آدرس زبر را صد بزنید مقدار **hello world** چاپ می شود.

```
example.com/index.php/blog/
```

توجه: نام کنترلرها حتما باید با اسم بزرگ شروع شود 'B'

```
<?php
class Blog extends CI_Controller {


}
```

متد ها در کنترلر ها

همان گونه که در مثال بالا مشاهده نمودید ، کده درون تابع **index()** نوشته می شود ، درواقع تابع پیش فرض ما برای با رگزاری **index**می باشد ولی شما قابلیت دارید هر تابعی را درون فایل **CONTROLLER**بارگزاری نمایید .به مثال زیر توجه نمایید.

```php
<?php
class Blog extends CI_Controller {

        public function index()
        {
                echo 'Hello World!';
        }

        public function comments()
        {
                echo 'Look at this!';
        }
}
```

درا ین مثال ما دو تابع داریم به نام های **index**و **comments** برای صدا زدن تابع **comments**در زیر را بارگزاری می نمایین.

```
example.com/index.php/blog/comments/
```

با بارگزاری آدرس فوق شما باید متن جدید را مشاهده کنید .

ارسال مقدار به توابع و متدها

شما امکان ارسال مقدار به متدها را دارید به طور مثال وقتی بخواهید شناسه ای را به توابع بفرستید و طبق اطلاعات آن شناسه مطلبی را نشان دهد .

```
example.com/index.php/products/shoes/sandals/123
```

همان گونه که می بینید می خواهیم به تابعی به نام **shoes** که کفشها را نشان میدهد ، مقدار **sandal**و **123** را ارسال کنیم.

```php
<?php
class Products extends CI_Controller {

        public function shoes($sandals, $id)
        {
                echo $sandals;
                echo $id;
        }
}
```

همانگونه که مشاهده میکنید تابع **shoes**درمتغیر **sandal**و **id**را دریافت و چاپ میکند بعد از اجرای برنامه **sandal**و **123** چاپ می شود و درون تابع مقدارهای **$sandals**و **$id** مقدار می گیرند.

تعریف کنترلر پیش فرض

کنترلر پیش فرض : کنترلی است که در صورتی که ما برنامه را بدون آدرسی صدا بزنیم آن کنترل بارگزاری می شود . که مقدار آن را در فایل **application/config/routes.php** می توانید تغییر دهید.

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

توجه: در مقدار دهی بالا فقط کافی است کنترلر را بنوسید بدون هیچ آدرس دیگری در صورتی که مقدار تابع را مشخص نکنید تابع **index()** در آن تکنترلر بارگزاری میشود.

طبقه بندی کنترلرها با استفاده از پوشه بندی

شما به راحتی می توانید فایلهای کنترلرها را طبقه بندی کنید و آنها را با فولدها جدا سازی کنید به طور مثال:

```
application/controllers/products/Shoes.php
```

وبه صورت زیر میتوانید از ادرس زیر استفاده کنید.

```
example.com/index.php/products/shoes/show/123
```

سازنده های کلاس

هنگامی که شما می خواهید کلاسی سفارشی بسازید و تمامی اجزای کلاس ریشه آنرا بارگزاری نمایید از سازنده ها استفاده می کنیم.

```
parent::__construct();
```

در کد ایگنایتر شما کنترلری می سازید و از ریشه ان سازنده می سازید همانند کد زیر بعد از آن شما تمامی کنترلرها را می توانید از آن کنترلر ریشه بگیرید و به این صورت شما می توانید اجزای سفارشی خود را در کلاس ریشه ساخته و هر موقع نیاز داشتید بارگزاری نمایید

```php
<?php
class Blog extends CI_Controller {

        public function __construct()
        {
                parent::__construct();
                // Your own constructor code
        }
}
```

توجه:جهت دریافت لیست نامهای رزرو شده به آدرس زیر مراجعه نمایید.

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

View

متشکل از فایلهایی می باشد که نمایش برای کاربر را فراهم می سازد درواقع شما نیاز دارید به نمایش اطلاعات به کاربر مثل سربرگ ، جداول و ... که این کار در فایل های **view**انجام می شود.

توجه داشته باشید که فایلهای **view**به تنهایی صدا زده نمی شود و شما می بایست حتما در **controller** فایلهای **view**را صدا بزنید .به همین منظور به شما توصیه می شود مطالب فصل قبل را به خوبی تسلط داشته باشید.

ساخت view

جهت ساخت **view** یک فایل ساده **html**را در آدرس *application/views/* بسازید، به طور مثال فایلی با محتوای زیر.

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

بارگزاری view

جهت صدا زدن **view**در **controller**کد زیر را فراخوانی می کنیم.

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

nameاسم فایل شما می باشد.

نویسنده : پیروز جنابی

مثال کاربردی

1–فایل کنترلر با محتوای زیر را در آدرس **controller/blog.php** می سازیم

```
<?php
class Blog extends CI_Controller {

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

2– وحالا فایل **view**را در آدرس **views/blogview.php**

3– آدرس زیر را فراخوانی می کنیم.

```
example.com/index.php/blog/
```

بارگزاری چندین فایل view در یک کنترلر

برای بارگزاری چندین فایل **view** تنها کافی است که کد مربوطه را چندین بار صدا بزنیم .

در مثال بالا توسط آرایه **data**مقادیری را به فایل **content** ارسال کردیم که در آینده به طور کامل شرح می دهسم.

مرتب سازی فایلهای view با استفاده از پوشه بندی

همان گونه که در فصل قبلی مرتب سازی را برای فایلهای کنترلر توضیح دادیم برای فایلهای **view**هم صدق می کند و شما می توانید همانند دستور زیر استفاده کنید.

```
$this->load->view('directory_name/file_name');
```

```php
<?php

class Page extends CI_Controller {

        public function index()
        {
                $data['page_title'] = 'Your title';
                $this->load->view('header');
                $this->load->view('menu');
                $this->load->view('content', $data);
                $this->load->view('footer');
        }

}
```

ارسال مقادیر به فایلهای view

و اما مهمترین چیز در این قسمت ارسال مقادیر به فایلهای **view**می باشد . در واقع شما بعد از انجام عملیاتها میخواهید طلاعاتی را به قسمت **view**برای نمایش بفرستید ، به طور مثال می خواهید آرگومانی را که در کنترلر دریافت نموده اید با عدد **2** جمع کنید و به فایل **VIEW**برای چاپ بفرستید. شما می بایست فایل کنترلر را همانند زیر بسازید.

```php
<?php

class Page extends CI_Controller {

        public function index($id)
        {
                $data['id'] = $id;
                $this->load->view('content', $data);
        }

}
```

همانگونه که می بینید آرایه ای داریم به نام **data** و همچنین عنصری به نام **id** ، که مقدار را از آرگومانها که به طور کامل در قسمت کنترلرها توضیح داده شده است ، به **content**می فرستیم و شما در فایل **view**آن را می توانید چاپ کنید.

توجه داشته باشید شما برای چاپ مقدار می توانید از فرمت کوتاه شده آن استفاده نمایید.**<?=$id?>**

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

ساخت حلقه در view

فرض کنید آرایه ای متشکل از مقدارهای مختلف را به فایل **view**می فرستید و می خواهید چاپ کنید ، شما می بایت حلقه ای در فایل **view** ایجاد کرده و مقادیر را چاپ کنید به مثال زیر توجه نمایید.

```
<?php

class  Blog  extends  CI_Controller  {

        public  function  index()
        {
                $data['todo_list']  =  array('Clean House',  'Call Mom',  'Run Errands');

                $data['title']  =  "My Real Title";
                $data['heading']  =  "My Real Heading";

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

همانگونه که می بینید در فایل کنترلر آرایه **data**داریم که در آن ،آرایه ای به نام **todo_list**داریم و آن را به **blogview** می فرستیم .

همانگونه که می بینید با استفاده از یک حلقه **forech** مقادیر را چاپ می کنیم.

توضیح **forech** : یک حلقه میباشد که تعدادش را به صورت اتوماتیک از آرایه ای که در پارا متر اول می خواند و در ادامه با استفاده ازکلمه کلیدی **as** مقادیر آرایه را درون متغیری که بعد از **as**می آید می ریزیم .

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

        <h3>My  Todo  List</h3>

        <ul>
        <?php  foreach  ($todo_list  as  $item):?>

                <li><?php  echo  $item;?></li>

        <?php  endforeach;?>
        </ul>

</body>
</html>
```

خروجی پذیری view

فکر کنید می خواهید خروجی یک فایل **view**را داخل متغیری بریزید و چاپ نکنید ، با دستور زیر می توانید این کار را بکنید.

```
$string = $this->load->view('myfile', '', TRUE);
```

مدل ها در کد ایگنایتر

مدل در کد ایگنایتر به منظور ارتباط با بانک اطلاعاتی به کار می رود ، شما تصور کنید وب سایتی فروشگاهی داریدکه نیازدارید محصولات را از بانک اطلاعاتی بگیرد و نشان دهد ، این کدها در قسمت مدل نوشته می شود.

به مثال زیر توجه کنید:

توجه : این مثال با استفاده از **query builder** ساخته شده است که درآینده به طور کل به آن می پردازیم .

شرح مثال:

ابتدا کلاس مدل را همانند کنترلر را می سازیم با این تفاوت که وارث **CI_MODEL**می باشد و باید در سازنده آن سازنده والد آن را صدا بزنیم.

```
class Shop_model extends CI_Model
{
    public $title;
    public $content;
    public $date;

    public function __construct()
    {
        // Call the CI_Model constructor
        parent::__construct();
    }

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

    public function insert_entry()
    {
        $this->title   = $_POST['title']; // please read the below
        $this->content = $_POST['content'];
        $this->date    = time();

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

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

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

}
```

و در گام بعد توابع مورد نیاز جهت اعمال مورد نظر را می سازیم.

همچنین توجه داشته باشید فایل شما باید در جای خود ساخته شود به طور مثال فایل بالا در آدرس زیر ساخته می شود.

```
application/models/Shop_model.php
```

صدا زدن مدل

جهت صدا زدن مدل ابتدا شما باید کلاس آن را صدا بزنید :

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

و در گام بعدی با استفاده از اسم کلاس مدل تابع مربوطه را صدا بزنید:

```
$this->model_name->method();
```

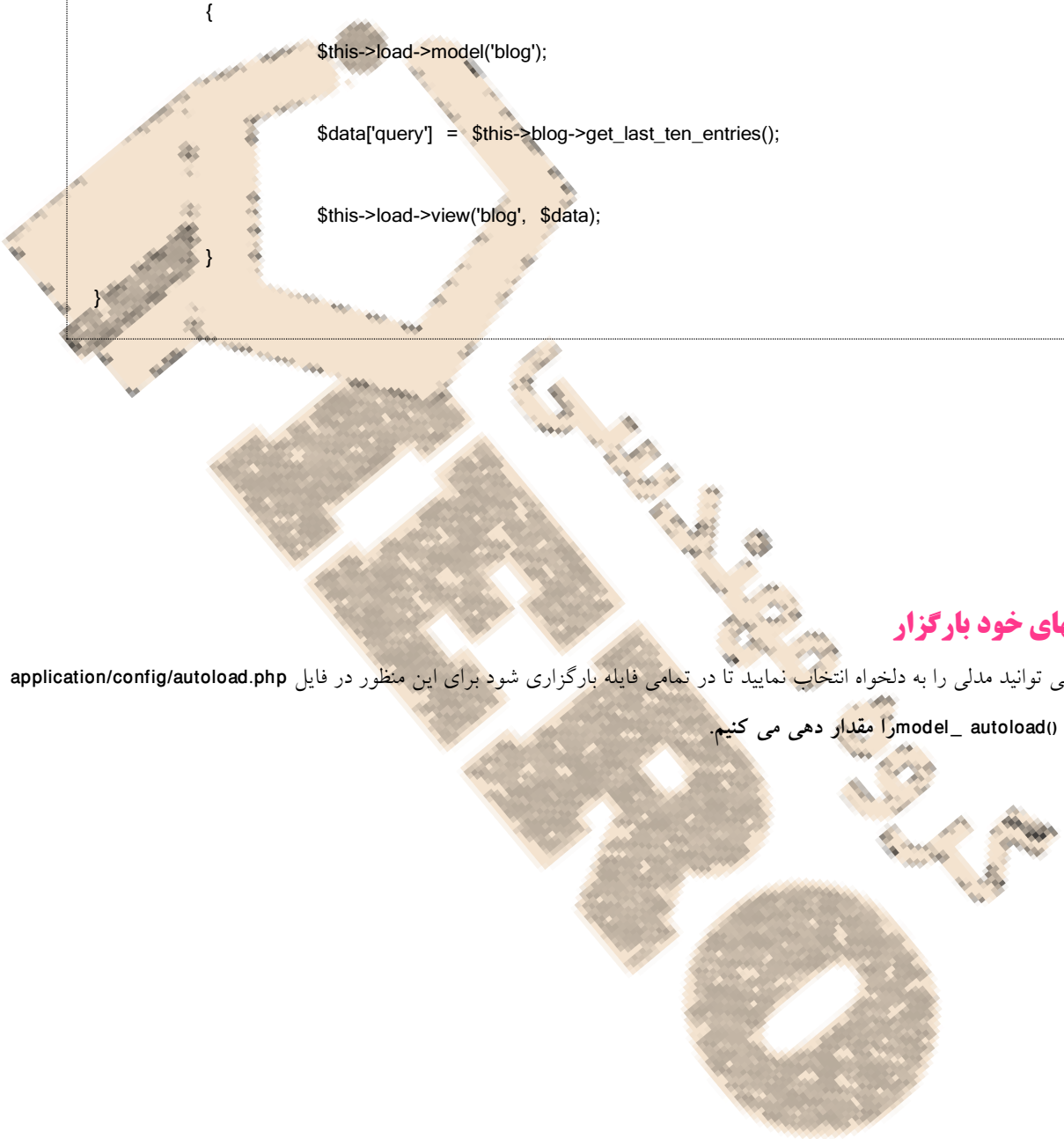
توجه نمایید شما می توانید در کنترلر خود چندین مدل را صدا بزنید.

در صورتی که شما بخواهید کلاس مدلی را صدا بزنید و نامی منحصر به آن بدهید و در صدا زدن توابع از آن نام استفاده کنید روند زیر را انجام می دهید:

```
$this->load->model('model_name',  'foobar');

$this->foobar->method();
```

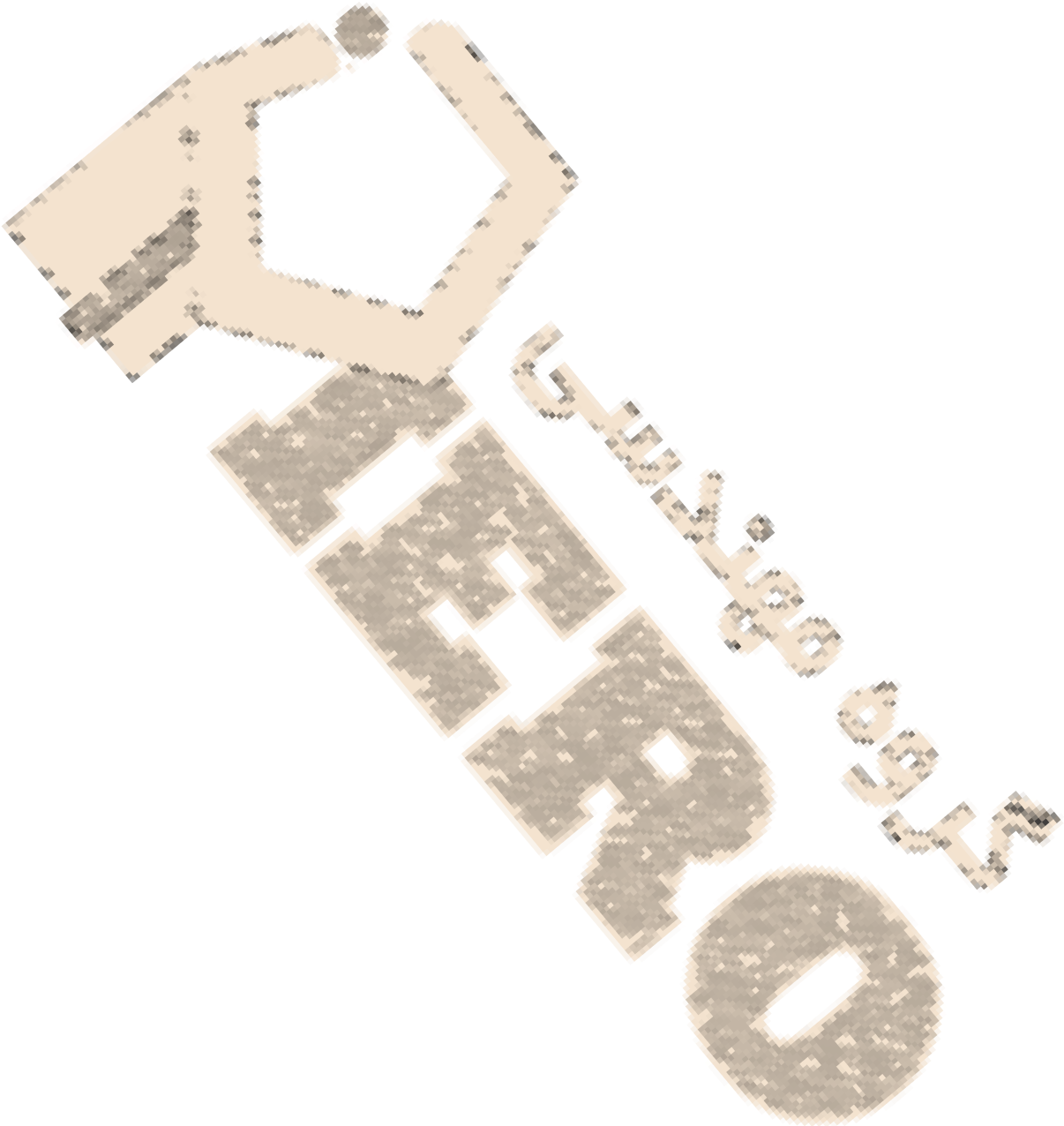
و در کل می توانید با توجه به مثال زیر کنترلر ها و مدلها را ترکیب و به ویو جهت نمایش ارسال کنید.

```
class Blog_controller extends CI_Controller {

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

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

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



مدلهای خود بارگزار

شما می توانید مدلی را به دلخواه انتخاب نمایید تا در تمامی فایله بارگزاری شود برای این منظور در فایل **application/config/autoload.php** آرایه **autoload()_model** را مقدار دهی می کنیم.



برنامه نویسان حرفه ای
PIERO
نویسنده : پیروز جنابی
www.piero.ir

Helper در کد ایگنایتر

همانطور که از اسم آن پیداست ، وقتی شما به کمک آنها نیاز دارید **helper** ها به کمک شما می آیند مثلا در مواقعی که به اجزای فرم نیاز دارید کمک کننده فرم به کمک شما می آید و اجزای فرم را برای شما فراهم می سازد.

توجه داشته باشید کمک کننده ها در پوشه **helper**ساخته می شوند هم در **system**و هم در **application** و در هنگام صدا زدن ابتدا پوشه **application/helper** چک می شود و بعد پوشه **system/helper** ، به طور ساده می توان گفت که فایلهای کمک کننده های خود را درون **application/helper**بسازید و استفاده کنید.

همانند اجزای دیگر کد ایگنایتر برای استفاده از کد ایگنایتر باید آنها را صدا بزنیم به طور مثال :

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

بارگزاری چندین helperبه صورت همزمان

به منظور بارگزاری چندین فایل کمک کننده به طور همزمان بجای یک فایل در تابع صداکننده چندین فایل را در یک آرایه صدا می زنیم همانند مثال زیر:

```
$this->load->helper(
        array('helper1', 'helper2', 'helper3')
);
```

همچنین برای لود شدن اتوماتیک یک کمک کننده می توانید آن را در فایل **application/config/autoload.php** در قسمت **helper**صدا بزنید.

استفاده از کمک کننده ها

استفاده از کمک کننده ها بسیار راحت و جذاب است ، بعد از صدا زدن کمک کننده ، تابع مورد نظر را در هرجایی که بخواهیم صدا می زنیم مانند مثال زیر:

```
<?php echo anchor('blog/comments', 'Click Here');?>
```

وراثت در کمک کننده ها Helpers "Extending"

برای ارث گرفتن یک کمک کننده ابتدا فایل مورد نظر را درون **application/helpers/ folder** با همان اسم کمک کننده اصلی بسازید ولی باید اسمش با **MY_** شروع شود که این اسم درون تنظیمات قابل تغییر است.

نکته : درون فایل **config** برای تغییر پیشوند کلاس مقدار روبه رو را مقدار دهی کنید :

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

توجه نمایید اگر نیاز دارید کمک کننده ها را طبق سلیقه و نیاز خود تغییر بدهید بهتر است دست به فایل اصلی نزنید و از روش بالا برای تغییر آن کمک کننده استفاده نمایید.

به مثال زیر توجه کنید:

برای بازسازی کمک کننده آرایه به ثرتیب زیر ئیش می رویم:

فایل مورد نظر را ساخته و تابع آن را بازنویسی می کنیم **application/helpers/MY_array_helper.php**

کتابخانه ها (library) در کد ایگنایتر

کتابخانه ها از پر مصرف ترین عناصر کد ایگنایتر می باشد ، شما می توانید کتابخانه ای بسازید و در آن توابعی معرفی کنید و هر موقع خواستید آن را صدا بزنید و از ان استفاده کنید و یا از کتابخانه های پیش فرض خود کد ایگنایتر را بار گزاری و استفاده کنید یا از کتابخانه های رایج در پی اچ پی استفادده کنید به طور مثال می توانید از کتابخانه **tcpdf** در کتابخانه استفاده کنید .

کتابخانه های سیستم در ***system/libraries/*** *موجود می باشد که شما به طریق زیر می توانید آنها را فراخوانی نمایید.*

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

به طور مثال کتابخانه ای که برای اعتبار سازی فرمها استفاده می شود که در آینده به طور کامل به آن می پردازیم

```php
// any_in_array() is not in the Array Helper, so it defines a new function
function any_in_array($needle, $haystack)
{
	$needle = is_array($needle) ? $needle : array($needle);

	foreach ($needle as $item)
	{
		if (in_array($item, $haystack))
		{
			return TRUE;
		}
	}

	return FALSE;
}

// random_element() is included in Array Helper, so it overrides the native function
function random_element($array)
{
	shuffle($array);
	return array_pop($array);
}
```

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

همچنین شما می توانید با استفاده از آرایه های چندین کتابخانه را بارگزاری نمایید مانند مثال زیر:

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

ساخت کتابخانه

به منظور ساخت کتابخانه یک کلاس پی اچ پی که با حرف بزرگ شروه شود در پوشه **application/libraries** می سازیم ، تو جه داشته باشید شما می توانید با اسم کلاسی که در پوشه سیستم کد ایگنایتر وجود دارد بسازید و هنگامی که اینکار را می کنید کتابخانه ای که در پوشه **application/libraries** ساختید الویت دارد به سیستم و ان را بارگزاری می نماید در این زمان شما می توانید آن را ورثه کتابخانه اصلی نمایید تا بتواانید از ویژگی های کتابخانه اصلی استفاده نمایید.

توجه داشته باشید برای نام گزاری به قواعد زیر توجه نمایید:

- حتما با حرف بزرگ شروع شود مثل Myclass.php
- نام کلاس و نام فایل باید یکسان باشد

برای ساخت کتابخانه ابتدا ساختار آن را همانند زیر تعریف کنید

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

class Someclass {

        public function some_method()
        {
        }
}
```

استفاده از کلاس

ابتدا باید کتابخانه را صدا بزنیم

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

و بعد هرجا آن را می توانیم صدا بزنیم

```
$this->someclass->some_method();
```

ارسال پارامتر به یک کتابخانه

شما می توانید یک کتابخانه را بسازید و در سازنده آن پارا متر های مورد نظر را دریافت و پردازش نمایید.

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

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

در کتابخانه :

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

class Someclass {

        public function __construct($params)
        {
                // Do something with $params
        }
}
```

نکته مهم در کتابخانه ها

توجه داشته باشید شما نمی توانید از همه عناصری که قبلا در کنترلر و مدل استفاده می کردید استفاده کنید مانند فرم و یا بانک اطلاعاتی به این منظور شما باید متد شبیه ساز **$this** را صدا بزنید و از ان استفاده کنید.

```
$CI =& get_instance();

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

به طور مثال

جاگزاری کتابخانه شما با کتابخانه اصلی

فرض کنید می خواهید کتابخانه بسازید که جایگزین کتابخانه **email**شود ، به همان ترتیب معمولی فایلش رامی سازیم ***application/libraries/Email.php*** *و بعد کلاسش را تعریف می کنیم*

```
class CI_Email {

}
```

```
class Example_library {

        protected $CI;

        // We'll use a constructor, as you can't directly call a function
        // from a property definition.
        public function __construct()
        {
                // Assign the CodeIgniter super-object
                $this->CI =& get_instance();
        }

        public function foo()
        {
                $this->CI->load->helper('url');
                redirect();
        }

        public function bar()
        {
                echo $this->CI->config->item('base_url');
        }
```

تو جه داشته باشید کلاسش حتما باید با **CI_**شروع شود .

و برای صدا زدن نیز به صورت معمولی صدایش می زنیم

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

ارث بری در توابع جایگزین

شما می توانید وقتی تابعی را جایگزین می کنید ، کاری کنید که از تابع اصلی آن ارث گرفته و تمامی متد آن را بارگزاری نماید . توجه داشته باشید باید اسم کلاستون حتما با **MY_** شروع شود (البته این تنظیم قابل تغییر است[]).

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

به عنوان مثال

```
class MY_Email extends CI_Email {

}
```

و در گام بعد نیاز است در سازنده آن سازنده والد را صدا بزنیم

```
class MY_Email extends CI_Email {

        public function __construct($config = array())
        {
                parent::__construct($config);
        }

}
```

صدا زدن کلاس ساخته شدن همانند صدا زدن عادی می باشد .

متن پایانی

با تشکر از اینکه این کتاب را خوانده اید و امید واریم توانسته باشیم پیشرفتی در علم کشور عزیزمان برداشته باشیم . توجه نمایید این نشریه ، ادامه دار می باشد و این نسخه شماره 1 می باشد . در ضمن توجه داشته باشید ممکن است خطا ها و کمبود هایی در این نشریه باشد ، امیدواریم شما با بزرگواری خود ما را در رفع این کاستی ها یاری نمایید.

به منظور ارتباط با ما می توانید با ایمیل info@piero.ir ارتباط بر قرار نمایید .

شما می توانید در لینک زیر جدید ترین مطالب کد ایگنایتر را ببینید . http://www.piero.ir/category/free-learn/code-egniter-learn /

کانال تلگرام(هنر کدینگ:آموزش حرفه ای کد ایگنایتر و برنامه نویسی) :https://telegram.me/itnew4u

با تشکر پیروز جنابی