بسم الله الرحمن الرحیم

# مقدمه

## بررسی اجمالی

با نام و یاد خدا شروع می کنیم. این هسته توسط آقای محمد حسن صالحی حاجی آبادی نوشته شده و توسط آقایان جواد عوض‌زاده کاکرودی و سامان سلطانی و با همکاری آقایان رضا محیطی و احمد کریمی توسعه داده شده است.

## ویژگی ها

این سامانه ویژگی های زیادی خواهد داشت! که در آینده تکمیل خواهد شد.

## فهرست

[مقدمه 1](#_Toc400467976)

[بررسی اجمالی 1](#_Toc400467977)

[ویژگی ها 1](#_Toc400467978)

[فهرست 1](#_Toc400467979)

[اصل مطلب :) 1](#_Toc400467980)

[شاخصه‌های فایل‌ها 1](#_Toc400467981)

[Controller 1](#_Toc400467982)

[Model 1](#_Toc400467983)

[View 2](#_Toc400467984)

[Display 2](#_Toc400467985)

[چیدمان و توابع فایل‌ها 2](#_Toc400467986)

[مشترکات فایل‌ها 2](#_Toc400467987)

[پیشوند‌ها 2](#_Toc400467988)

[توابع مخصوص کنترلرها 3](#_Toc400467989)

[پیش به سوی بی‌نهایت 4](#_Toc400467990)

[توابع مخصوص نمایش(ویو) 4](#_Toc400467991)

[توابع مخصوص مدل 4](#_Toc400467992)

[کاربا دستوران کوئری 5](#_Toc400467993)

[کاربا کتابخانه sql\_maker 5](#_Toc400467994)

[ساخت کوئری خصوصی 6](#_Toc400467995)

[ساخت کلاس کوئری عمومی 6](#_Toc400467996)

[کار با فرم‌ها 8](#_Toc400467997)

[کار با کتاب‌خانه form\_maker 8](#_Toc400467998)

[ساخت فرم از طریق متدهای پیش‌تعریف 9](#_Toc400467999)

[ساخت فرم از طریق دیتابیس 9](#_Toc400468000)

# اصل مطلب :)

هر ماژول ممکنه دارای 4 فایل باشه. Controller, model, view, display ماژول های مادر در زیرشاخه دوم از پوشه ها قرار می‌گیرند. همانند

home/home

user/login

user/logout

پوشه‌های home همان index صفحات هستند که پوشه home/home صفحه سایت بدون اسلش است یعنی همان index و پوشه‌های زیرشاخه‌ای homeصفحات index پوشه هستند همانند profile/home که اگر در آدرس‌بار صفحه profile بخورد به این پوشه هدایت می‌شود.

# شاخصه‌های فایل‌ها

## Controller

فایل مادر هر ماژول است، و بودنش به معنی استقلال ماژول به حساب می‌آید. اگر ماژولی دارای این فایل نباشد از کنترلر اولین ماژول home استفاده می‌کند و اگر در شاخه پوشه home نبود، از پوشه home/home استفاده می‌کند.

به عنوان مثال ماژول user/logout دارای فایل controller نیست. پس سیستم در فایل user/home به دنبال کنترلر می‌گردد و اگر این پوشه نیز وجود نداشت یا فایل controller نداشت به پوشه home/home منتقل می‌شود و از آن پوشه کنترلر را درخواست می‌کند.

همانگونه که از اسم این فایل مشخص است، وظیفه این فایل به عهده گیرنده پردازش‌های اصلی و کنترل کننده فایل است، از قبیل توابع مشترک در model, view و کنترل سطوح دسترسی، کنترل آدرس‌بار صفحه و دیگر خصیصه‌های اصلی.

## Model

این فایل تنها زمانی فراخوانی می‌شود که دستور post به صفحه ارسال شود و یا نیازداشته باشیم در view از کوئری خاصی استفاده کنیم، قابل توجه است که اگر قصد هدایت به این فایل را داشتیم و دستور ما از نوع post نبود، می‌توان در listen کنترلر یک post مجازی ایجادکرد.

## View

این فایل کنترلر نمایشی صفحه سمت سرور را به عهده می‌گیرید. از قبیل تنظیم فایل‌ها و کوئری‌ها.

## Display

این فایل که با پسوند html ذخیره می‌شود تم و حالت نمایشی صفحه را به عهده می‌گیرد که با ساختار نوشتاری twig به متغیرهای view متصل می‌شود.

# چیدمان و توابع فایل‌ها

## مشترکات فایل‌ها

‌‌‌‌config()

همان constructor ما می‌باشد که عملیات‌های ابتدایی و اصلی در این تابع صورت می‌گیرد.

$this->sql()

همانگونه که از اسمش مشخص است یک شی از کوئری را برایمان می سازد.

این تابع اگر در دو فایل controller, view خوانده شود باید آرگومانی با پیشوند @,. داشته باشد و اگر در مدل خوانده شود می‌تواند بدون پسوند آرگومان بگیرد یا اصلا آرگومان نگیرد.

اگر در مدل به این تابع آرگومانی اختصاص یابد، در هرکجای ماژول بعد از ساخته شدن این کوئری می‌توان بایکتای آن این کوئری را فراخوانی کرد که برای سیستم پلاگین بندی مفیداست.

## پیشوند‌ها

@: بعد از این پیشوند نام تابعی در مدل نوشته می‌شود که آن تابع با پیشوند sql\_ در مدل تعریف شده است.

View: $this->sql("@Uid");

Model: function sql\_Uid();

می‌توان در خط فراخوانی آرگومان‌های اضافی داد که آرگومان‌های دوم تا n به تابع اصلی ارجاع داده می‌شوند.

View: $this->sql("@Uid", "baravak", true);

Model: function sql\_Uid($name, $syntax);

.(نقطه): به کوئری‌های عمومی ارجاع داده می‌شوند. این کوئری‌ها در تمامی صفحات قابل دسترسی هستند.

مثال:

View: $this->sql(".Uid")

OR Model: $this->sql(".Uid")

OR Controller: $this->sql(".Uid")

Uid Query: class Uid function config()

در این حالت نیز می‌توان آرگومان‌هایی ارسال کرد.

توجه داشته باشین که اگر تنها یک نقطه گذاشته شود کوئری از تابع config درخواست می‌شود، اما اگر با دو نقطه مشخص گردد کوئری در کلاس دنبال نام تابع می‌گردد.

مثال:

View,Model,Controller: $this->sql(".Uid.login");

Uid Query: class Uid function login()

توضحیات تکمیلی در عنوان مخصوص به دیتابیس‌ها آمده است.

$this->redirect()

یک ریدایرکتور برای صفحه ایجاد می کند که می‌تواند بدون آرگومان باشد یا تا دو آرگومان بگیرد

آرگومان اول: به صورت دستی نام صفحه ای که قرار است ریدایرکت شود را مشخص می کنید.

$this->redirect("/profile");

آرگومان دوم مقدار بولین می‌گیرد که صحیح یعنی از طریق کد php و تابع header صفحه ریدایرکت می‌شود و مقدار false از طریق متاتگ‌های html این کار صورت می‌گیرد.

توضحیات تکمیلی در عنوان مخصوص به منتقل کننده‌ها آمده است.

## توابع مخصوص کنترلرها

$this->addProperty()

این تابع یک متغیر سراسری در کلاس تعریف می‌کند که در فایل‌های مدل و ویو نیز قابل دسترسی هستند.

برای این‌کار یک متغیر در کلاس ایجاد می‌کنید و در تابع config کنترلر مقدار متغیر را به آرگومان این تابع می‌دهید

Controller:

public $version = 2.6;

function config()

{

$this->addProperty("version");

}

View Or Model:

$this->version;

$this->addMethod()

همان کار addProperty را انجام می‌دهد با این تفاوت که یک تابع را سراسری می‌کند.

$this->model(), $this->view()

این‌ها مدل‌ها و ویوهای صفحات را بازگشت می‌دهند. قابل توجه اینکه در ویو و مدل هرصفحه می توانید با شی $this->controller به این کنترلر دسترسی داشته باشین.

$this->listen()

این تابع برروی ساختار آدرس صفحه آنالیزها و دستوراتی را انجام می‌دهد که به صورت قانون در می‌آید.

به صورت پیشفرض، صفحات نمی‌توانند دایرکتوری‌های مجازی داشته باشند مثلا فرض کنید صفحه users/profile موجود است، حال اگر در ادامه اضافه کنیم users/profile/110 سیستم خطای 404 می‌دهد.

یا اینکه فرض کنید می‌خواهید آدرس های منطقی در آدرس‌بار صفحه بسازید مثلا از لحاظ چینشی فایل‌های مربوط به عضویت و ورود باید در پوشه users قرار بگیرد: users/login, users/logout, users/register اما از لحاظ کاربردپزیری و سهولت دسترسی باید در آدرس‌های اصلی آدرس بار باشند: /login, /logout, /register برای اینکار کنترلرها می‌توانند واسطه شوند و صفحات مجازی را به ماژول‌های حقیقی منتقل کنند.

# پیش به سوی بی‌نهایت

شما می‌توانید انواع کتابخانه‌ها و کلاس‌های پروژه خود را در توابع کنترلر تعریف کنید و در تمامی ماژول‌ها و پلاگین‌ها از آن‌ها استفاده کنید.

## توابع مخصوص نمایش(ویو)

$this->form()

این تابع کار فرم‌سازی پویا را انجام می‌دهد. که می‌تواند با پیشوند‌@ یا بدون پیشوند بیاید. اگر دارای پیشوند@ بود نام جدول پایگاه داده‌ها را می‌گیرد و فرم آن جدول را می‌سازد و اگر بدون پیشوند بود فرمی با نوع آن مقدار می‌سازد.

## توابع مخصوص مدل

$this->commit(), $this->rollback()

این توابع زمانی اجرا می‌گردند که دستور مدل ما به پایان رسیده است. سیستم آنالیز می‌کند که اگر پروسه دارای خطا بود rollback و اگر بدون خطا بود commit را اجرا می‌کند. هر دوی این توابع اولین متغیرشان باید از نوع تابع باشد و مابقی متغیر ها اگر تعریف شود به عنوان آرگومان برای تابع اول می‌آیند.

Exp one:

$this->commit(function(){

echo "commit";

});

Exp two:

$age = 13;

$this->commit(function($argAge){

echo $argAge;

}, $age);

# کاربا دستوران کوئری

## کاربا کتابخانه sql\_maker

شما می توانید به جای فراخوانی این کتاب‌خانه در فایل model با استفاده از $this->sql() این کتاب‌خانه را فراخوانی کنید.

شروع به کار:

$query = $this->sql()->tableUser()

->fieldAll()

->whereUser\_id(12)

->andPassword(md5(111111))

->select();

$this->sql()

در این حالت ما یک شی از کتاب‌خانه sql\_maker را در فایل model ساختیم.

->tableTableName()

این تابع جدول مقصد کوئری مارا مشخص می‌کند.

توجه داشته باشین که در سرتاسر این کتاب‌خانه اولین حرف بزرگ تعیین کننده است. همانگونه که در بالا و مثال بالا مشاهده می‌کنید، پارامترهای تعریفی را از اولین حرف بزرگ شروع کردیم و نام اصلی تابع قبل اولین حرف بزرگ است، به عنوان مثال در بالا tableTableName() نام تابع table است که TableName نام جدول پایگاه داده‌ای است.

مثال:

->tableUsers() // fn = table; table = users; SELECT \* FROM users

->tablePermisions() // fn = table; table=permisions; SELECT \* FROM permisions

در خط اول مثال بالا ما جدول users را تعریف کردیم و در خط دوم جدول permisions را مشخص نمودیم.

->setField(insert)

این تابع فیلد field را برابر insert قرار می‌دهد.

مثال:

->setUsrname("ali"); SELECT … SET username="ali"

->setPassword(md5(110)) ; SELECT … SET password=md5(110);

->setEmail("ali@world.com"); SELECT … SET email="[ali@world.com](mailto:ali@world.com)"

حال متنی که برای کوئری ساخته می‌شود اگر برای دستور select باشد خط اول متن زیر را تولید می کند و اگر برای دستور update باشد خط دوم متن زیر تولید می‌شود.

// SELECT … (username, password, email) VALUES ("ali", md5(110), "[ali@world.com](mailto:ali@world.com)");

// UPDATE … SET username="ali", password=md5(110), email="[ali@world.com](mailto:ali@world.com)"

$this->where, if, or, and, like, orlike, andlike

این دستورات شرط کوئری را می سازند.

->whereId(10)->andName("ali")->orlikeEmail("%.com")

// where id=10 and name = "ali" or email like "%.com"

->groupOpen, groupClose

این دستورارت شرط ما را داخل یک پرانتز قرار می‌دهند برای شروط پویا

->wehreId(10)

->groupOpen()

->andName("ali")

->orlikeEmail("%.com")

->groupClose()

->orCode(114)

// where id = 10 and (name = "ali" or email like "%.com") or code = 114

->limit(len), limit(start, len)

این تابع اگر یک آرگومان داشته باشد مقدار limit کوئری را از 0(صفر) تا len ادامه می‌دهد و اگر دو آرگومان داشته باشد مقادر limit کوئری را از start تا len ادامه می‌دهد.

->limit(20) // … LIMIT 0, 20

->limit(20, 40) LIMIT 20, 40

->orderId()

این تابع نحوه چینش کوئری را مشخص می‌کند.

->orderId() // ORDER BY id ASC

->orderId("ASC") // ORDER BY id ASC

->orderId("DESC") // ORDER BY id DESC

->groupbyYear()

این تابع کوئری را گروه‌بندی می‌کند.

->groupbyYear() // … GROUP BY year

->select(), insert(), update(), delete()

زمانی که این توابع نوشته می‌شود دیگر نمی‌توانید به کتاب‌خانه sql\_maker دسترسی داشته باشید و در اصل این توابع ویرایش کوئری را بسته و کوئری را به سمت پایگاه داده‌ها ارسال می‌نمایند. که در توضیحات کتاب‌خانه db\_connection نحوه دریافت کوئری و دیگر تنظیمات خواهد آمد.

## ساخت کوئری خصوصی

برای این کار باید حتما فایل model در فولدر ماژول شما موجود باشد. قرار نیست کار سختی را انجام دهید، کافیست یک تابع با پیشوند sql\_ در مدل خود ایجاد کنید. و در آن مستقیما کوئری بزنید و از هرکجای ماژول که نیاز بود دسترسی داشته باشین.

مثال

Model:

function sql\_userName($id){

retrun $this->sql()->tableUser()->whereId($id)->assoc('username');

}

View:

$uname = $this->sql("@userName", 14);

## ساخت کلاس کوئری عمومی

برای اینکه در تمامی صفحات و ماژول‌ها قادر باشید که یک کوئری یا یک مجموعه کوئری یکسان را انجام دهید لازم است به پوشه cls/query رفته و یک کلاس تعریف کنید. در هنگام فراخوانی نیز با پیشوند .(نقطه) می‌توانید به آن کلاس دسترسی داشته باشید.

cls/query/user.cls.php

class query\_user\_cls{

function config($id){

return retrun $this->sql()->tableUser()->whereId($id)->assoc('username');

}

function userPassword($id){

return retrun $this->sql()->tableUser()->whereId($id)->assoc('password');

}

View:

$this->sql(".user", 14);

$this->sql(".user.userPassword", 13);

# کار با فرم‌ها

## کار با کتاب‌خانه form\_maker

این کتاب‌خانه با نام $this->form() نیز در view قابل دسترسی است.

$this->form(type)

نوع فرمی را که قصد ایجادش را داریم مشخص می‌کند

$this->form("checkbox")// <input type="checkbox">

->attr(name, value)

خصیصه ای با نام name و مقدار value به فرم اضافه می‌کند.

->attr("data-name", "ignore") // <input … data-name="ignore">

توجه داشته باشید که خصیصه‌ها را نیز می‌توان مستقیم با قراردادن نام تابع وارد کرد.

->tabindex("0") // <input … tabindex="0">

->style("color:red") // … style="color:red">

->classname(cls)

کلاس فرم را با مقدار cls ایجاد می‌کند.

->classname("tabvil") / … class="tabvil">

->addClass(clsn)

کلاس clsn را به کلاس فرم اضافه می‌کند و اگر فرم کلاسی نداشت مقدار کلاس را برابر clsn قرار می‌دهد.

->classname("one") // class="one"

->addClass("two") // class="one two"

->removeClass(clsn)

کلاس clsn را از کلاس‌های فرم حذف می‌کند.

->classname("one two") // class="one two"

->removeClass("two") // class="one "

->pl(pl)

مقدار pl را برابر placeholder فرم قرار می‌دهد. شما می‌توانستید این خصیصه را دستی وارد کنید اما برای سهولت مخفف آن را تعبیه کرده‌ام.

->label(lb)

مقدار lb را برابر label فرم قرار می‌دهد.

->child()

یک فرزند به فرم اضافه می‌کند. این تابع برای نوع‌های radio و select لازم است. درضمن اگر از این تابع استفاده نمودید از کلاس formMaker خارج شده و یک کلاس جدید ایجاد میشود برای فرزند.

$x = $this->form("radio")

->name("gender")

->child()

->value("male")

->label("male");

$x->child()

->value("female")

->label("female");

$x->label("gender")

<label for="gender">gender</label>

<lable for="male">male</label> <input type="radio" id="male" name="male" value="male">

<lable for="female">male</label> <input type="radio" id="female" name="female" value="female">

## ساخت فرم از طریق متدهای پیش‌تعریف

شما می‌توانید در آرگومانی که نام فرم را تعریف می‌کنید علامت # را قرار دهید تا از متدهای پیش‌تعریف که در کلاس forms\_Extends\_cls نوشته شده استفاده کنید.

## ساخت فرم از طریق دیتابیس

با قرار دادن علامت @ و نام جدول پایگاه داده‌ها میتوانید لیستی از فرم‌های آن جدول را تهییه کنید.