<http://www.phperz.com/article/15/1007/161638.html>

https://framework.zend.com/manual/1.12/zh/zend.controller.quickstart.html#zend.controller.quickstart.go

# ZEND FrameWork

## PHP流行的MVC框架介绍

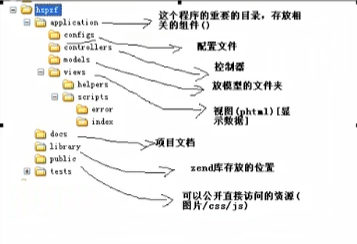
1. Zend framework（zend公司开发，官方产品）
2. ThinkPHP（中国人开发）
3. Yii（外企用的比较多）
4. Laravel

## 开发步骤

1. 下载最新的zend framework库，官方下载
2. 创建zend framework项目，有两个方法：第一个就是手工创建（麻烦），第二个就是使用zf.bat程序来创建：通过shell进入到zf.bat所在目录，然后执行以下命令

Zf.bat create project项目的全路径（d:/hspzf）

创建完，项目的基本结构如下：



1. 使用zend studio ，创建一个空项目htpzf（目前可省了吧）
2. 把刚才创建的项目的结构拷贝到hspzf项目中（目前可省了吧）
3. 引入zf库到hspzf项目中，拷贝到library目录下
4. 开始配置apache，让他支持我们zf框架：开启pdo和pdo相关数据引擎

在php.ini配置文件中，里面有extension=php\_pdo\_mysql.dll去掉前面的分号

开启apache的rewrite模块，apache\conf\httpd.conf，去掉前面#号

LoadModule rewrite module modules/mod\_rewrite.so

并开启识别.htaccess功能

把该httpd.conf文件AllowOverride None 改成All

1. 配置虚拟主机：
2. 如果开发环境是自定义安装的。启动虚拟主机的配置项

Include conf/extra/httpd-vhosts.conf

PHP 数据对象（PDO） 扩展为PHP访问数据库定义了一个轻量级的一致接口。pdo是PHP Data Objects的简写意思为php数据处理对象。所以一般进行数据库操作之后，会返回一个pdo对象。

1. 在httpd-vhosts.conf中，配置自己的虚拟主机。



1. 在hosts中添加dns路由

8.重新启动 apache

9.http//myzf.com

## 访问流程

浏览器只要发出请求，都要经过apache，apache根据重写规则，首先找到public/index.php（总司令）文件，执行Bootstrap方法，初始化所有控制器，index.php可以决定具体去找哪一个控制器下（小队长）的方法，init方法是对当前控制器初始化工作方法。MVC规定，控制器处理完数据之后，就应该引入一个视图显示数据内容，也叫渲染视图。

为什么首先会去找index.php，这是因为对url进行了改写。在php项目中，我们通过对url的重写，可以使整个项目只有一个入口，这样可以防止用户随意访问任意的url或者php文件。我们在项目中添加一个.htaccess文件（Hypertext Access超文本入口），在里面加上规范就可以，并在apache配置文件中启用rewrite机制。

rewriteEngine On ：去掉前面#号

rewriteRule ^. \*$ enter.php 无论是什么文件，都要经过enter.php

rewriteRule ^. \*$\.php enter.php 所有的php文件，都要经过enter.php

rewriteRule ^. \*$\.php enter.html 所有的html文件，都要经过enter2.php

## 网页请求原理分析

实际上服务器是一个管理资源并为用户提供服务的计算机软件，我们也总是把运行服务器软件程序的计算机（host）或硬件（hardware）叫做服务器。服务器（HTTP服务器如apache，nginx，tomcat本质上就是一种应用程序，通常运行在服务器上）的功能主要是接收请求，并依据请求返回对应的文件。当如果遇到动态内容时候，就需要服务器后端语言如php调用数据库，并通过PHP-FPM解析之后在返回给前端。服务器本身不能识别后端语言，需要安装对应的语言解释器。即Fast-CGI，协议翻译机。

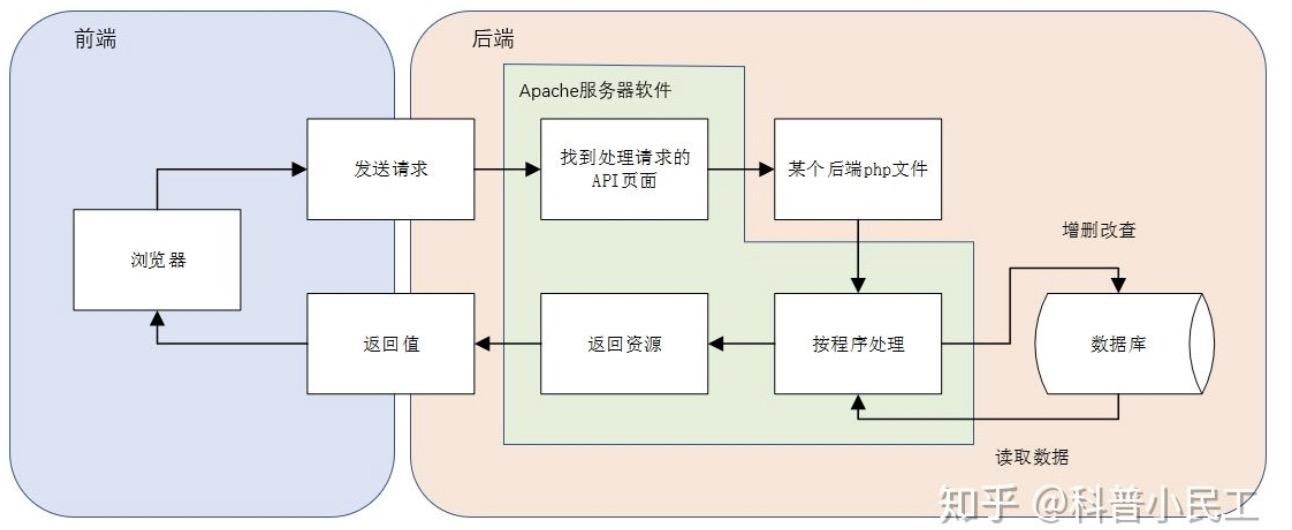
当用户在客户端对服务器SERVER进行网页请求时，早期的 Web 通信只是按照客户端请求将保存在 Web 服务器硬盘中的数据转发过去而已，这种情况下客户端每次获取的信息也是同样的内容（即静态请求，比如图片、样式文件、HTML文档），而随着 Web 的发展，Web 所能呈现的内容更加丰富，与用户的交互日益频繁，比如博客、论坛、电商网站、社交网络等。

这个时候仅仅通过静态资源已经无法满足 Web 通信的需求，所以引入 CGI （common gateway interface，更形象的叫法是“协议翻译机”）以便客户端请求能够触发 Web 服务器运行另一个外部程序来解析动态资源（即动态请求，比如基于 PHP、Python、Java 实现的应用），客户端所输入的数据也会传给这个服务器中的外部程序，该程序运行结束后会将生成的 HTML 和其他数据通过 Web 服务器再返回给HTML。利用 CGI 可以针对用户请求动态返回给客户端各种各样动态变化的信息。

这样一来，PHP-FPM 就好理解了，FastCGI（CGI的一个升级版本） 只是一个协议规范，需要每个语言具体去实现，PHP-FPM 就是 PHP 版本的 FastCGI 协议实现，有了它，就是实现 PHP 脚本与 Web 服务器（通常是 Nginx）之间的通信，同时它也是一个 PHP SAPI，从而构建起 PHP 脚本与 Web 服务器之间的桥梁。PHP-FPM只解析<?php ?>里面的内容，所以如果在前台页面使用php调用数据库传递过来数据时或者在遍历，判断语句的时候，需要使用<?php ?>进行封装。这样php解析器才能够识别，然后将封装好的数据传递给前端。

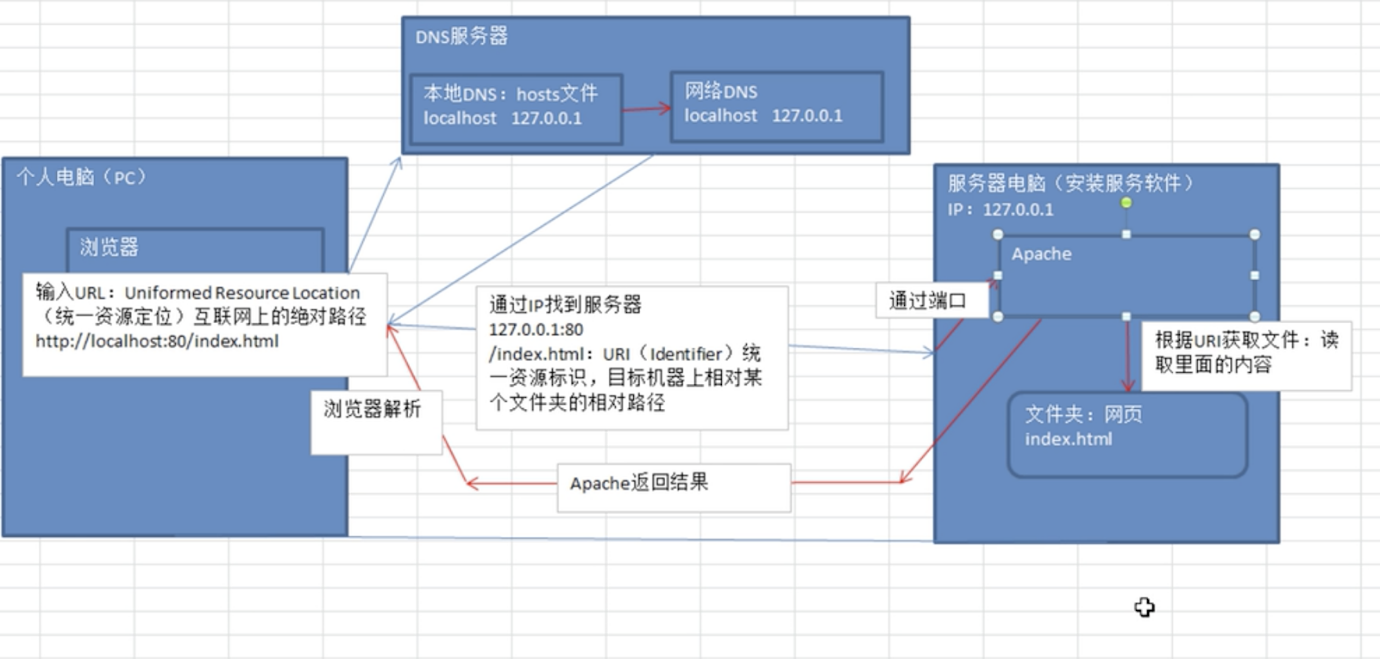
例如当文件中包含.php代码时，Apache不会直接返回给客户，因为客户端无法解析php code，（所以我们说，php 是一个Server-side language，因为其是在server上运行的），Apache服务器会将代码传给php application（PHP-FPM），让其解析，在解析过程中可能会遇到query，会从数据库中调用数据，然后生成用户需要的网页内容，传给客户。

所以综上，我们需要在安装服务器的硬件上安装3个软件，PHP-FPM，Apache，MySQL Server。当我们在计算机中安装了Mamp时，它是一个集成环境，将以上3个软件环境集成到一起，这样相当于我们在本地就有一个server，即local server，所以存储文件夹叫localhost。此时的server就是一个硬件概念，安装了apacha软件服务器的硬件，我们可以在那里找到文件，并直接运行。目前最主流的三个Web服务器是Apache、 Nginx 、IIS。

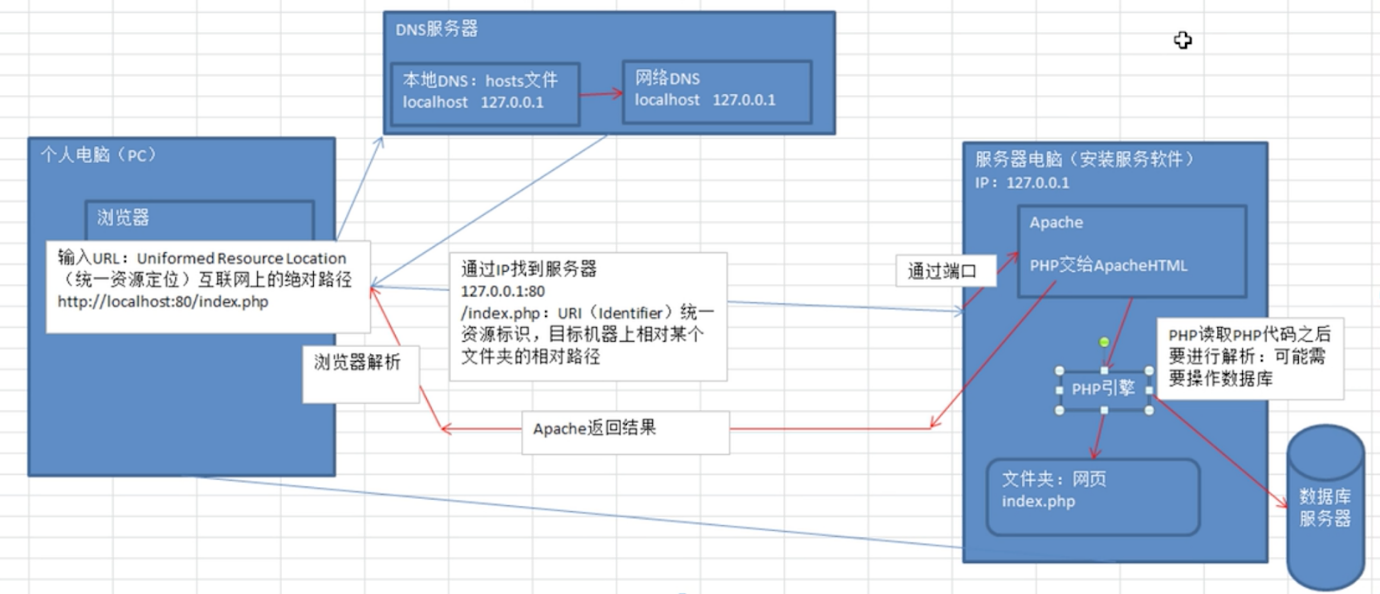


Web分为两类： 静态访问和动态访问

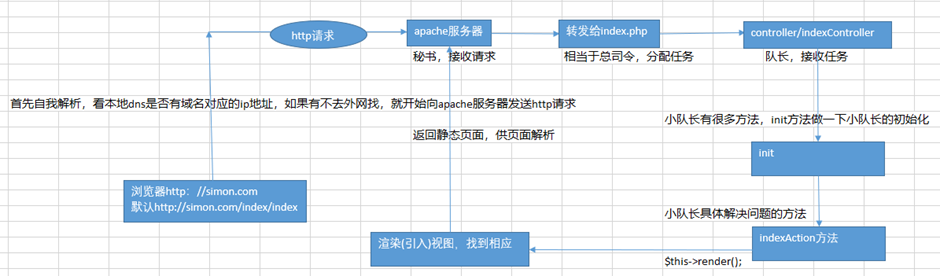
静态访问过程



 动态访问过程：需要apache通过CGI调用外部程序（php-fpm）解析，过程中可能会调用数据库。



## Zend请求流程



浏览器只要发出请求，都要经过apache，apache根据重写规则，首先找到public/index.php（总司令）文件，执行Bootstrap方法，初始化所有控制器，index.php可以决定具体去找哪一个控制器下（小队长）的方法，init方法是对当前控制器初始化工作方法。MVC规定，控制器处理完数据之后，就应该引入一个视图显示数据内容，也叫渲染视图。

为什么首先会去找index.php，这是因为对url进行了改写。在php项目中，我们通过对url的重写，可以使整个项目只有一个入口，这样可以防止用户随意访问任意的url或者php文件。我们在项目中添加一个.htaccess文件（Hypertext Access超文本入口），在里面加上规范就可以，并在apache配置文件中启用rewrite机制。

rewriteEngine On ：去掉前面#号

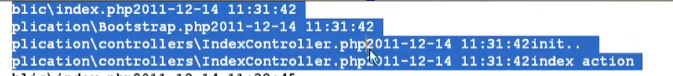
rewriteRule ^. \*$ enter.php 无论是什么文件，都要经过enter.php

rewriteRule ^. \*$\.php enter.php 所有的**php**文件，都要经过enter.php

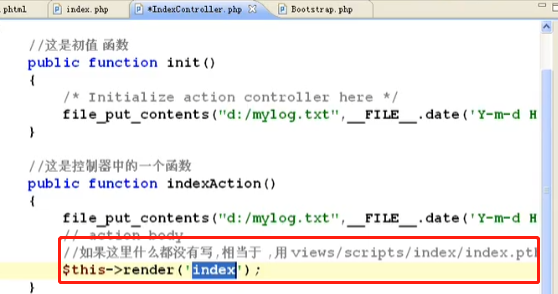
rewriteRule ^. \*$\.php enter.html 所有的**html**文件，都要经过enter.php

Bootstrap是对所有的控制器做初始化工作，控制器里面的init方法只对当前控制器做初始化工作。

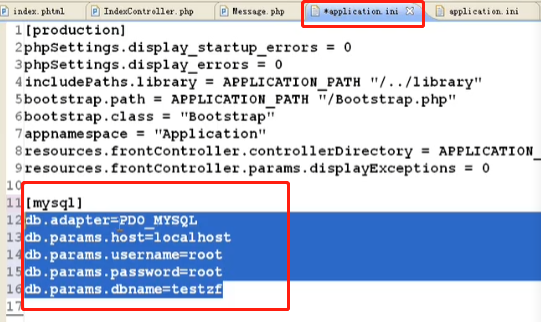
抓包测试，一个请求，会访问以下四个路由，即index.php—》bootstrap.php—》IndexController.php.init—》indexaction



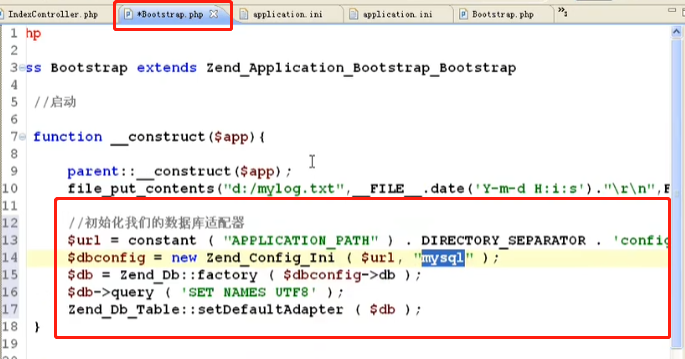
如果没有写渲染哪一个视图，ViewRenderer根据控制器的名称（例如index）和当前的action的名称（例如index）来决定使用哪个模板文件。默认情况下，模板文件使用.phtml作为扩展名，我们将使用index/index.phtml模板文件。另外，ViewRenderer自动假设views目录与控制器目录平级，作为视图层的基础目录，而实际的模板文件则放置在views/scripts/ 子目录下。因为我们的模板文件应该是 application/views/scripts/index/index.phtml。



在application.ini中配置数据库



在Bootstrap.php中配置以下内容，初始化数据库。但并不是所有的controller都是需要数据库，所以最好在相应的controller下init方法下写数据库的配置。



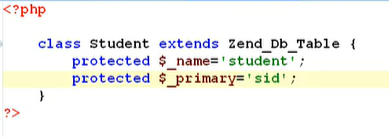
如何从控制器中获取数据，给当前的对象的视图一个属性，并给属性赋值。



## 综合案例1

### 表模型M

表模型必须继承于类Zend\_Db\_Table ，否则就不是表模型。建立了表模型之后，才可以通过相应的方法对表进行增删改查。默认表名是class\_name , 如果主键是id，可以不用写primary属性。一般在变量前添加一个下划线，如\_request，\_name等，是为了同函数里面的参数和局部变量相区分。通常也是表示private和protected变量的一种概念而已，与普通变量没有什么区别。



### 调用视图V

$this理解

首先了解$this的概念，$this->是指向实例化后的具体对象的一个指针，在实例化后确定指向，我们一般在一个类的内部使用本类或父类的属性或方法时，就使用$this->去调用，例如在控制器中，我们经常会调用$this->\_request->getParam()，就是调用了父类中的action中的\_request属性（其是Request\_Abstract的实例）。

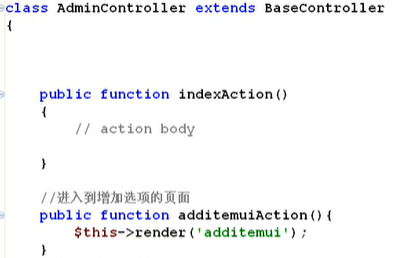
#### this与parent，self区别

方法的调用：$this->方法名();如果在类中有该方法则调用的是类中的方法或属性，若没有则是调用父类中的方法或属性。self是对类本身的一个引用，一般用来指向类中的静态变量；parent是对父类的引用，一般使用parent来调用父类的构造函数。

parent:: 则始终调用的是父类中的方法。

#### 分配视图

默认情况下，ViewRenderer会被启用。这意味着只需要简单地下定义一个action方法，就会自动指定一个对应的View脚本。在AdminController的方法indexAction里面，ViewRenderer会自动调用views/scripts/admin文件夹下，再找到名为index的视图，例如application/views/scripts/admin/index.phtml。



方法里面尽管什么都没有写，但是默认其调动了scripts/admin目录下的模型index.phtml，即默认存在$this->render(‘index’);如果想引用其他的页面，就不可以使用render方法，需要使用$this->\_redirect(‘home/add.phtml’);render函数也是Controller\_Action的一个方法，

#### 分配变量

$this->view->err=”您的用户名或密码错误”，view是父类中Zend\_Controller\_Action.php的一个属性，控制器通过此属性给err变量赋值，并将err变量分配给调用的视图，默认为和action同名的视图。这样在相应的视图中，就可以通过$this->err调用此变量进行显示。如果调用视图的类中有该变量则调用的是类中的变量，若没有则调用的是父类中的。

### 前台显示数据

在前端页面中，如果我们想显示后台传递给该视图的变量，那么我们如下所示:

首先遍历二维数组，将二维数组遍历成为一维数组，这样就可以利用一维数组的属性取获取数据了。如果只是想获取二维数组里面的一个属性值，可以使用array[0][3]或者array[0][‘name’]获取其中一个值。

<?php foreach($this->items as $item) ?>

<td><? =$item[‘name’] ?></td>

<?php endforeach ?>

### 控制器C

控制器必须继承于Zend\_Controller\_Action，如果没有继承，那么就是一个普通的类，那么就不会通过路由名调用其对应的方法了，也就没有render，forward等方法了。

class testController extends Zend\_Controller\_Action {

public function init() {

//始终会被调用的方法

}

public function loginAction() {

echo ‘user login’;

$this->render(“login”);

}

}

在scripts目录下建立test文件夹，此文件夹中视图专门对应testController对应的视图，xx.phtml文件.

优化代码：做一个父类，此类也同样继承于框架中Zend\_Controller\_Action类，专门供其它的控制器来继承。因为有些控制器，需要操作数据库，有些不需要操作数据库，因此我们可以写一个父类，在其init方法里连接数据库，然后所有其他的需要连接数据库的都需要继承此父类，如果不需要连接数据库，就继承框架中的Zend\_Controller\_Action类。

### 数据交互

后台Controller利用sql语句和相关封装的方法，从数据库获取相应的数据，然后将数据分配给变量，将变量分配给视图。然后前台视图通过变量显示数据。同时前台有时候也需要向后台传值，主要是通过name属性（用户输入），还有url后面附带参数传值（url链接），两种方式传值。无论哪种传值也都需要后台controller->getRequest->getParam方法进行接收传过来的值。

### 前台向后台传值

1.get方法：<a href=’/vote/?userid=<?=$user[‘id’]?>’>投票</a>

2.post方法: 通过<input type=”text” name=”username” value=”Simon” />, 属性来传值, form还有一个方法是input数组提交:

<input type=”text” name=”username[ ]” value=”Jason” />

<input type=”text” name=”username[ ]” value=”Simon” />

<input type=”text” name=”username[ ]” value=”Goe” />

我们在name属性后面加一个[]，后台获取数据的时候，就可以拿到一个数组：$\_POST[‘username’] = [‘Jason’,’Simon’,’Goe’].或者zend中，$usernames=$this->getRequest()->getparam(‘username’);

$usernames[0]=Jason, $usernames[1]=Simon, $usernames[2]=Goe,也可以使用for循环遍历。

### 后台接收值

1.接收get传递过来的，$this->getRequest()->getParam(‘userid’,’no’);，其中no为default值，如果没有传递过来任何值情况下发挥作用。

2.接收post传递过来的，$this->getRequest()->getParam(‘description’);

3.获取所有参数，$this->getAllParams

### 获取用户ip

$ip = $this->getRequest()->getServer(‘REMOTE\_ADDR’);

### 图片等资源引入

在zend framework项目中，把图片，css，js，flash…都同一放在public目录下管理

<script type=’text/javascript’ src=’/js/my.js’></script>

<link rel=’stylesheet’ href=’/css/my.css’ type=’text/css’ />

<img src=’/images/logo.png’/>

### 验证输入

在数据接口的地方，前端输入，后端接收时候，数据比较容易出问题，在客户端有javascript正则表达式进行验证，在服务器端仍然有php代码处理。

### ViewHelper视图助手

如果在视图中，如果经常要执行某些固定复杂的函数，例如格式化日期，生成表格元素或者显示动作链接等，就可以使视图助手，或者插件，类去执行这些行为。一个helper其实就是一个实现了Zend\View\Helper\HelperInterface的一个类。

<https://framework.zend.com/manual/2.1/en/modules/zend.view.helpers.html>

不难理解，视图，就是View层，前端展示页面，助手，就是辅助工具，辅助类或者一个功能。这个功能是抽象出来的一个组件，可以在必要的时候调用这个助手，就可以在View上展现指定的内容。例如。现在有个职业类型的选择下拉列表，我们就可以把这个下拉列表做成一个助手类，在使用的时候，调用这个助手类就可以了。并且便于修改和替换。<https://blog.csdn.net/mengxiangbaidu/article/details/7479109>

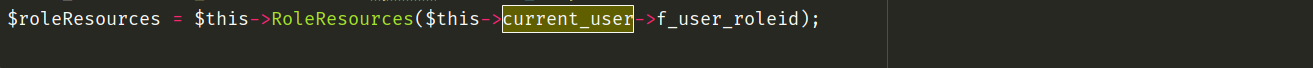
#### 调用视图助手

在视图脚本中，你可以用$this->helperName()来调用helper。这时Zend\_View会加载 Zend\_View\_Helper\_HelperName类，建立一个对象实例，并调用它的helperName()方法。对象的实例会在 Zend\_View的实例内一直存在，并可以被$this->helperName()重复调用。

在项目中，视图助手目前可发现存在下面两个地方，很多地方会在视图中直接应用到一下视图助手的方法。例如经常见到的ShowDiValues()，ShowMessage()，RoleResources(),例如下面例子中，就直接调用了default中的视图助手。Current\_user是action祖先类中定义好的视图变量。

/home/wwwroot/salemessage/library/DealCore/views/helpers

/home/wwwroot/salemessage/web/application/modules/default/views/helpers



Text

Description automatically generated

#### ViewRender视图解析助手

Note: 如果手动执行\_forward()、redirect、或者render时，不会发生自动解析。因为执行这些动作时，等于告诉ViewRenderer，你要自己确定输出结果。

Note: ViewRenderer助手默认启用。

### render

直接渲染phtml页面文件

不可以跳到其他视图中去，只能渲染script里面和当前controller同名文件夹中里的视图，例如IndexController中方法引用了$this->render(‘ok’);那么其会去找script中index文件夹中的ok.ptml。

### forward

可以跳转到本控制器中某个函数，也可以跳到别的控制器的某个函数。在本网站(项目)中控制器跳来跳去。

以下代码表示跳转到GlobalController下的okAction方法。在okAction方法中，调用ok.phtml视图，一般都是因为这是一个全局的视图（即所有控制器可以共享的视图），比如跳转成功，跳转失败等这样所有控制器可以公用的视图，但是必须需要建立global控制器，和对应的方法。如果不写第二个参数，那么默认为当前控制器下okAction。

$this->\_forward(‘ok’,‘global’)；

### redirect

Redirect可以在本网站的控制器转向url，同时也可以跳转到搜狐等外网url。

Eg：$this->\_redirect(‘http://www.sohu.com’);

$this->\_redirect(/hoge/foo);

### Create=C





https://framework.zend.com/manual/1.12/zh/zend.controller.actionhelpers.html

### zendframework 禁用view和layout

<https://my.oschina.net/tenking/blog/23798>

https://blog.csdn.net/mengxiangbaidu/article/details/7669700

在zend framework中使用layout布局之后，有些action中不需要view或者layout所以可以用下面方法禁用view和layout。layout代表很多视图view的公共部分，例如，页面和页脚，默认情况下，layout存储在view/layout文件夹中。

$this->\_helper->viewRenderer->setNoRender();

$this->\_helper->layout->disableLayout();

#### 在 Zend\_Controller 中禁用 view

#### 在 Action 级别禁用 view：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  class FooController extends Zend\_Controller\_Action  {       public function barAction()       {            $this ->\_helper->viewRenderer->setNoRender();       }  }  ?> |

在执行当前 action 的时候会不会展示 view .

#### 在 Controller 级别禁用 view：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  class FooController extends Zend\_Controller\_Action  {       public function init()       {            $this ->\_helper->viewRenderer->setNoRender();       }  }  ?> |

在执行当前 controller 下的所有 action 的时候都不会展示 view .

#### 全局级别禁用 view：

|  |  |
| --- | --- |
| 1  2  3 | <?php  Zend\_Controller\_Front::getInstance()->setParam( 'noViewRenderer' , true);  ?> |

在整个程序的执行过程中都不会展示 view .

#### 在 Controller 中禁用或改变 layout

#### 禁用 layout

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  class FooController extends Zend\_Controller\_Action  {       public function barAction()       {           $this ->\_helper->layout->disableLayout();       }  }  ?> |

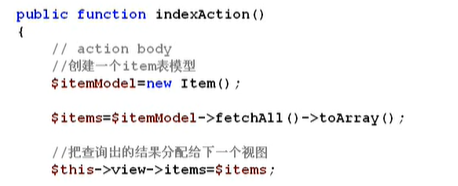
在此 action 执行的时候将不会使用 Zend\_Layout 。

#### 改变 layout

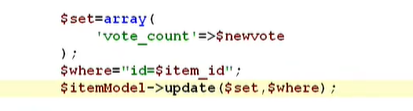
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  class FooController extends Zend\_Controller\_Action  {       public function barAction()       {           $this ->\_helper->layout->setLayout( 'other' );       }  }  ?> |

在此 action 执行的时候将使用名为 other 的 layout 。

### Read=R



### Update=U





### Delete=D

## 综合案例2

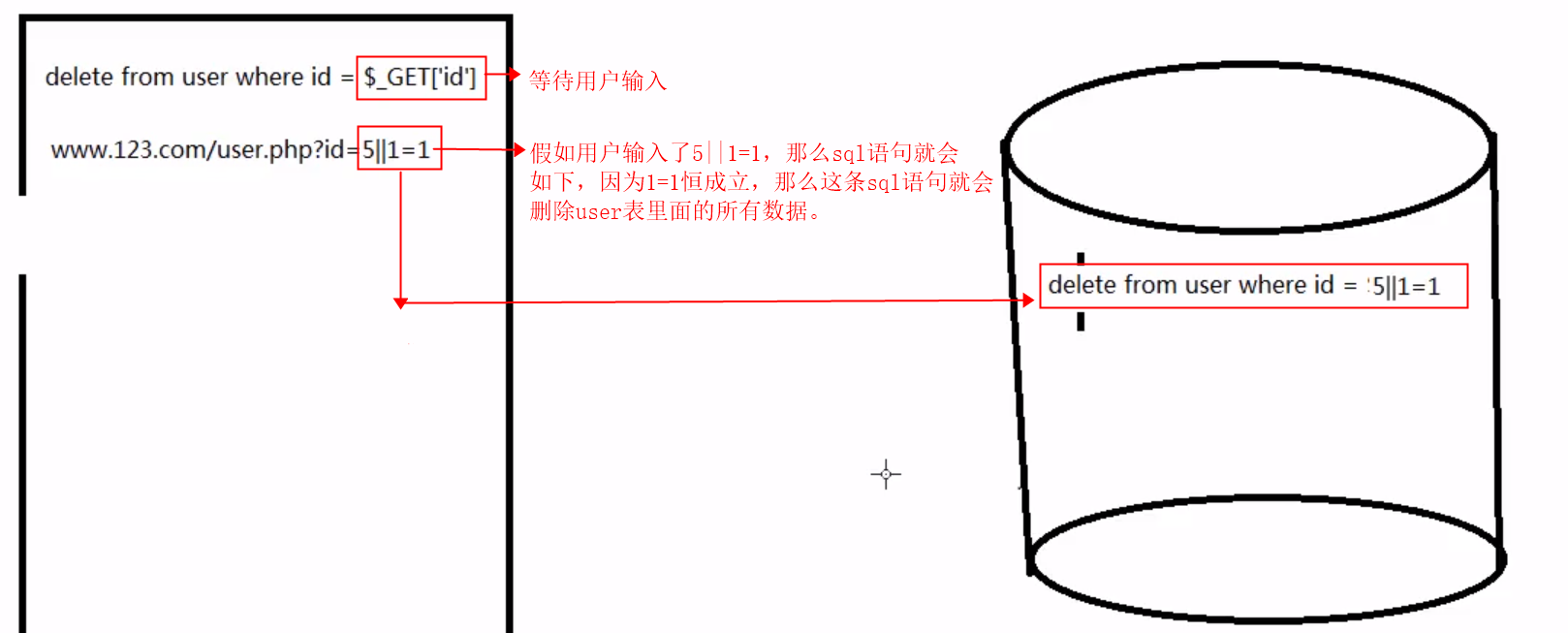
学生选课系统

建立三张表，学生表，课程表，选课表，并建立对应的model。建立controller和视图文件夹。

在table模型下和adapter类下，都有一些fetch方法用来获取结果集数据，这两者是不同的，adapter下 的fetchAll(),可以在括号里面输入sql语句，table下的fetchAll()，需要在括号中输入参数，其都是对PHP中PDO对象的一个扩展。

#### SQL Injection

如果您通过网页获取用户输入的数据并将其插入一个MySQL数据库，那么就有可能发生SQL注入安全的问题。所谓SQL注入，就是通过把SQL命令输入到Web表单递交，或输入域名，或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令，也就是说SQL注入是用户输入的数据，在拼接SQL语句的过程中，超越了数据本身，成为了SQL语句查询逻辑的一部分，然后这样被拼接出来的SQL语句被数据库执行，产生了开发者预期之外的动作。那么quote等方法的目的，就是使用sql引号规则来限制定界符，使一些特殊的字符对于一个表名或者列名来说，就是一个标识符，而不是sql关键字或者特殊字符，这样就防止了sql注入的问题。

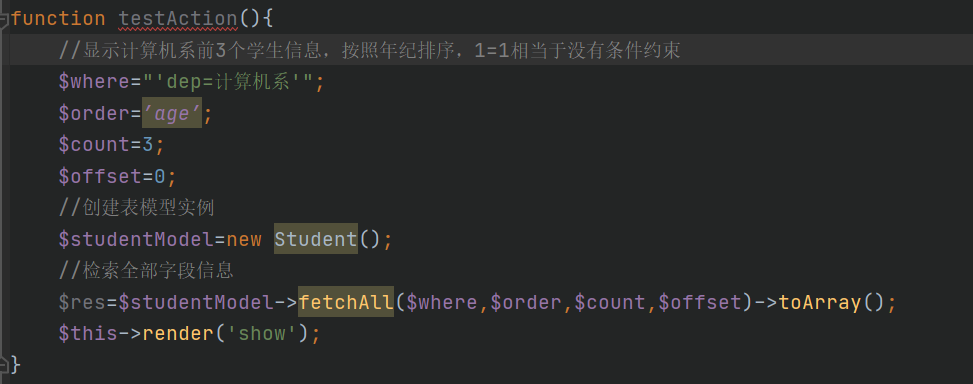


<https://www.runoob.com/mysql/mysql-sql-injection.html>

#### 检索全部字段信息

使用语句可以检索出表中的所有字段信息：$tableModel->fetchall()，这是db-table-abstract表模型下的方法，不是db-adapter-abstract下方法。

1. $tableModel->fetchall().不附加如何条件
2. $tableModel->fetchall($where,$order,$count,$offset),其中count代表取多少条，offset表示从第几条开始取。



1. 通过adapter来解决可读性和参数注入的问题。

一般where条件删选都是用户输入的，那么用户很可能输入的是sql语句，这样可能会欺骗server对数据库产生更改。通过db-adapter下的quoteInfo可以过滤。一个条件约束情况下，本例只是用了db-adapter下的quoteInto方法过滤where输入条件，仍然使用的是表模型中fetchAll方法调用数据，其并没有使用db-adapter调用数据。



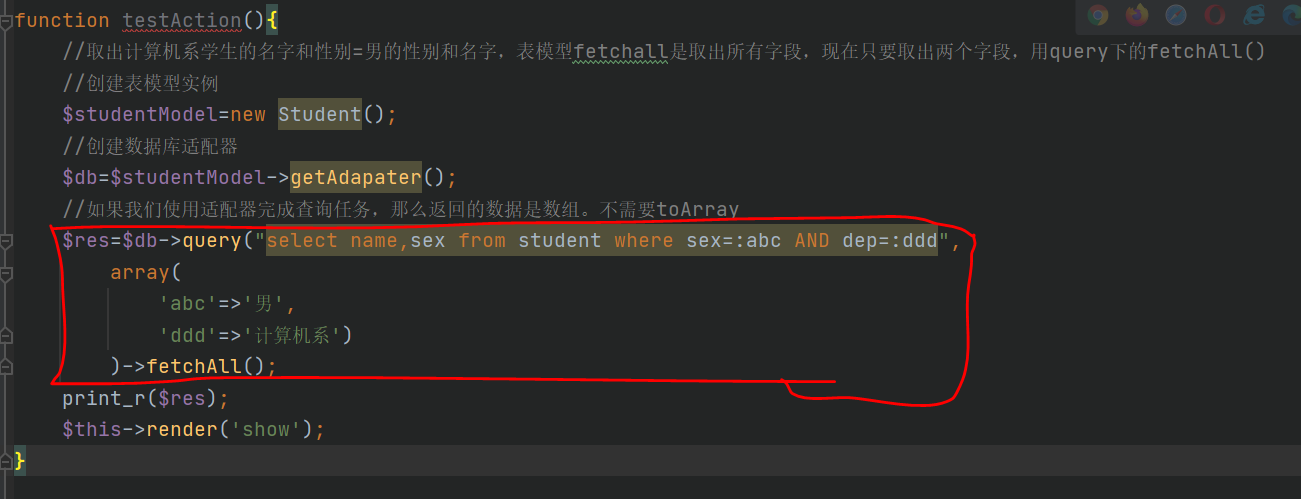
如果是多个条件约束，要像下面一样一个一个的注入，方法笨拙。有更好的方案吗？



有，如果使用db-table下的方法，只能向上面一样，一个参数，一个参数的防止sql注入，如果我们使用adapter中的query方法，此方法可以直接 执行sql语句进行查询，返回PDOSO，那么就可以通过数组方式防止sql注入。

#### 获取部分字段信息query

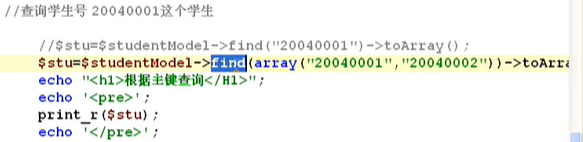
在以上模型实例的fetchAll()方法中，直接就是where条件筛选，并没有让我们写select语句来筛选部分字段，所以我们要使用适配器的query方法来筛选部分字段，并过滤条件字段，然后再取出。



对于删改查，需要对条件从句进行加引号处理，考虑注入的问题，那么就需要使用表的getadapter去处理一下。因为对于增加的时候，并不需要条件从句。但需要考虑xss攻击问题.

#### 取出一条记录

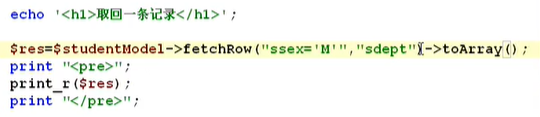
1.根据主键查询记录



2.取回一条记录

如果预期查询只有一条记录返回来，那么使用fetchRow方法，找到了一条就会返回，不在继续查找。

以下表示取回一条性别为男记录，通过系排名。



#### Adapter使用规则

1. 如果查询部分字段，使用Adapter,通过表模型获取适配器。
2. 如果多表查询Adapter
3. 如果考虑sql注入Adapter： $adapter->query()->fetchAll();
4. 如果是增加、删除、修改、查询某张单表的全部字段，即没有条件约束，直接使用表模型fetchAll()就可以了。

### Zend\_Db\_Select

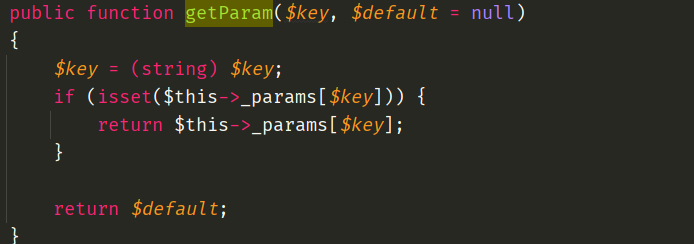
使用Zend\_Db\_Select方法是一种不受数据库约束构建select的sql语句的工具 （注：用户可以使用该方法生成查询的sql语句，而不需要考虑各种数据 库sql语句的差别）。虽然该方法明显还不完善，但是的确为我们提供一种方 法，帮助我们在不同的后台数据库进行相同的查询工作。除此之外，它还可以避免sql语句攻击。

<http://www.phperz.com/article/15/1007/161612.html#zend.db.select.%E7%AE%80%E4%BB%8B>

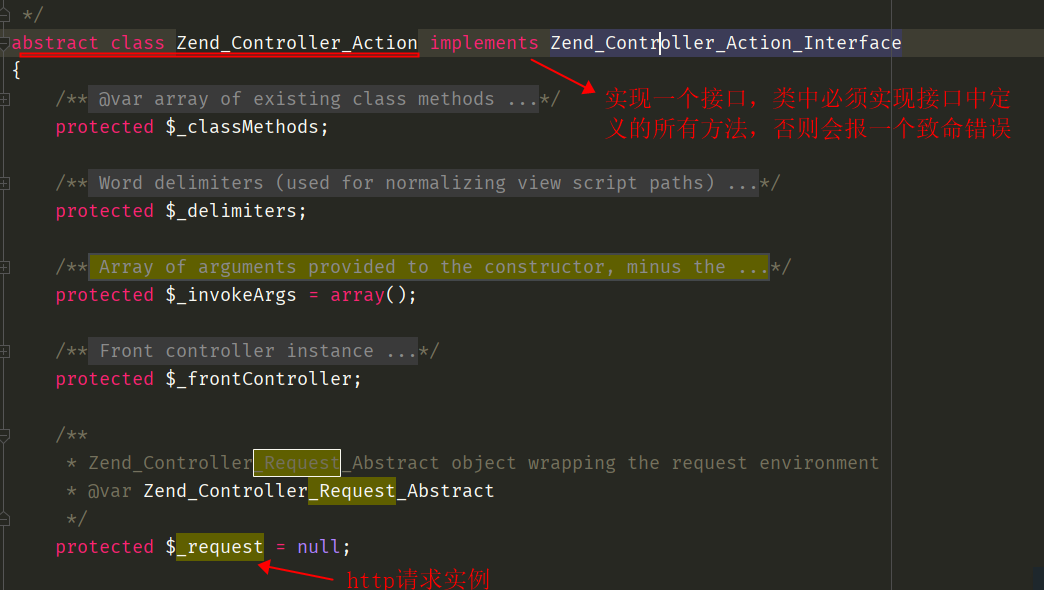
### ZendController基础

ZendController工作流用若干组件来实现。虽然不需要完全理解所有组件的基础知识来使用它，但是拥有工作流程的知识很有帮助。

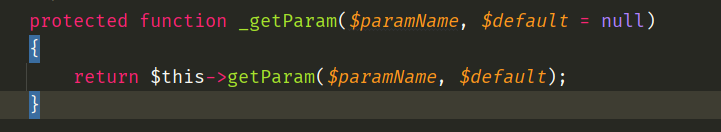
Zend\_Controller\_Request\_Abstract是http请求对象抽象层，允许路由器从请求环境中读出信息。例如在类中有getParam方法，用来获取动作请求参数。

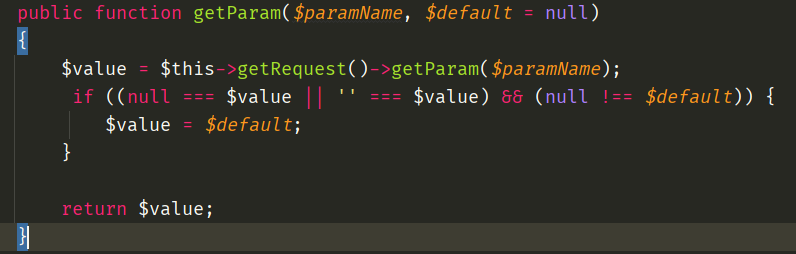


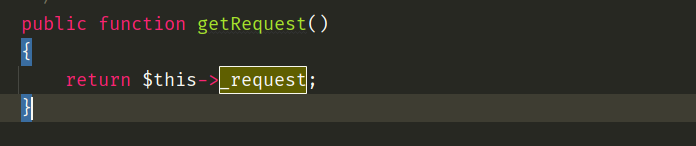
Zend\_Controller\_Router\_Interface是一个端口，而不是类。类需要实例化后才可以使用，端口需要实现（implement）后才可以使用。此端口用来定义路由器。路由是一个过程，在这个过程中它通过检查请求环境来决定哪个控制器和哪个控制器中的动作应该接受请求，而Zend\_Controller\_Action就是Zend\_Controller\_Router\_Interface的端口实现。如下图所示：



可见其中\_request则是Zend\_Controller\_Request\_Abstract的一个实例，那么就可以调用请求对象抽象层中的getParam方法，即$this->\_request->getParam()来获取http请求中的参数。但控制器如果继承于Zend\_Controller\_Action，也可以直接通过$this->\_getParam()来获取参数，因为在Zend\_Controller\_Action定义了如下方法。







综上，故$this->\_request->getParam(),$this->getRequest->getParam(),$this->\_getParam()三个函数都是一个意思，最终都是会调用请求对象抽象层中的getParam方法，但综合起来看，最后一个最简单。

# 对象注册表

$db = Zend\_Registry::get('db');

以上返回数据库访问层对象，就是一个PDO对象，然后就可以调用数据库对象下的方法。原始的PDO对象就有3种方法，见PHP笔记。而在zend中，其对PDO对象进行了封装，直接可以使用在php中PDOSO结果集对象下的方法，如fetchAll等等。尽管你可以使用query()方法直接对数据库进行操作，但是通常情况 下，仍然还是需要选择数据行并返回结果。以fetch开头的一系列的 方法可以实现这个要求。对于每一种 fetch系列 的方法来说，你需 要传送一个select的sql语句。

Graphical user interface, text, application, email

Description automatically generated

也可以是$db->select()->fetchAll()，因为select之后返回的是结果集对象。这里的fetchAll（）里面参数可以为空。而通过$db->fetchAll()调用的里面需要是一个sql语句。

# 代码上划线

不建议使用的方法，很可能方法已经过时，注解为deprecated，一般过时的方法可以找到替代的方法。

Text

Description automatically generated

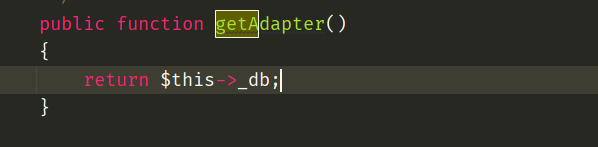
### Zend\_Db\_Adapter\_Abstract 数据库适配器抽象层

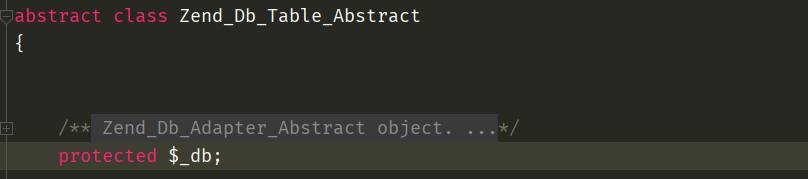
Adapter\_Abstract是zendfrmaework的数据库抽象层api，基于pdo, 你可以使用Adapter\_Abstract连接和处理多种 数据库里面有操作数据库的常用方法，而Table\_Abstract是表的抽象类，其中$\_db是Table\_Abstract一个属性，$\_db是Adapter\_Abstract的一个实例，所以我们当通过$this->\_db（和里面的getAdapter方法调用是一个效果，如下图所示）就会调用Db\_Adapter\_Abstract中的关于PDOSO等fetch系列方法进行数据库操作。同时在Table\_Abstract也有一些PDOSO的fetch方法，和Adapter\_Abstract中的不同，其不会做sql注入检查。

$res = $this->\_db->fetchRow("SELECT \* FROM ".$this->\_name." where type= '".$type."'");

返回一个stdClass对象，表示是一个标准类，是一个空类。并没有其他实际意义。

<https://blog.csdn.net/linu_cat/article/details/78237442>关于stdClass。





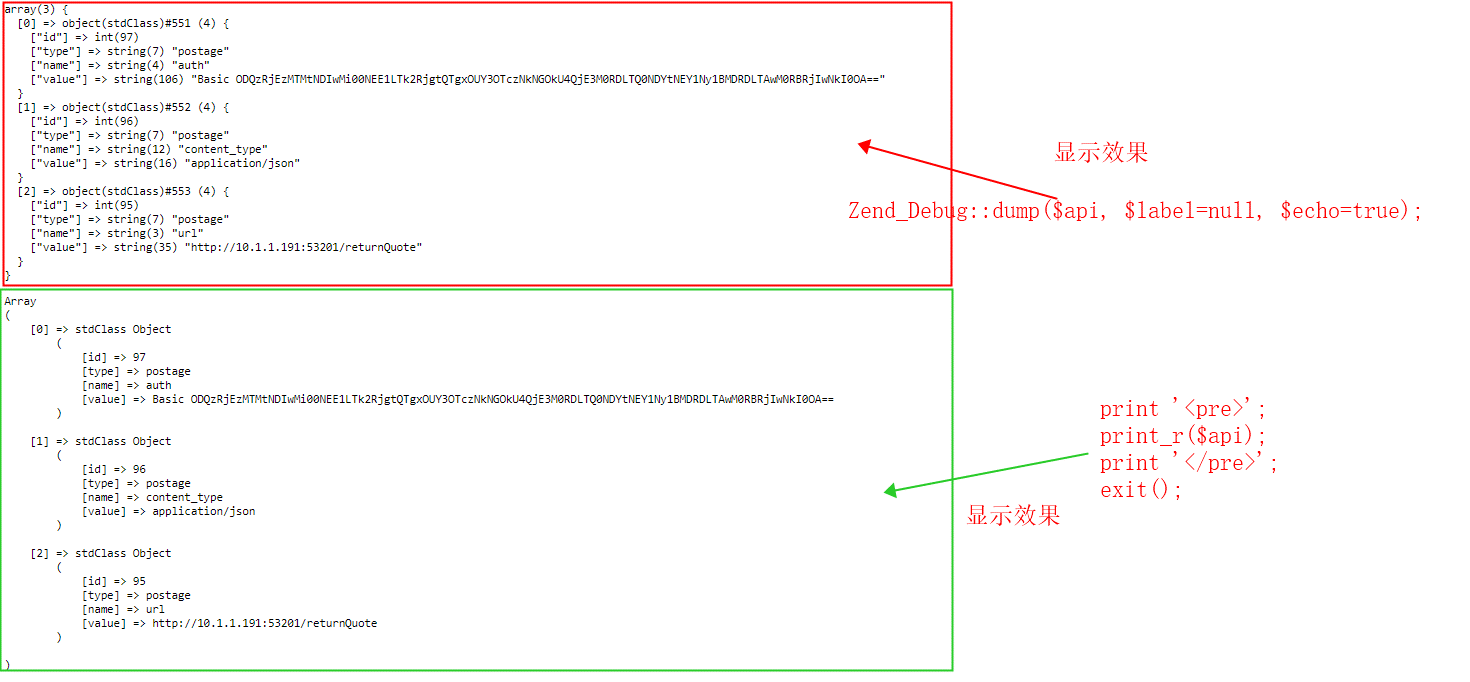
#### Db—adapter query()

Db-adapter下query()方法会prepare 和execute sql语句，这个query方法已经不是php中PDO下的query方法，已经考虑了sql注入，并可以绑定值，其运行结果会返回PDOSO，这个时候可以通过调用PDOSO下的方法去处理结果集对象。具体的获取内容在php中PDOSO里面有讲解。下面的fetch（）方法就是PDOSO下面的fetchAll（）。

$res = $this->\_db->query("SELECT \* FROM ".$this->\_name." where type = '".$type."'")->fetchAll();

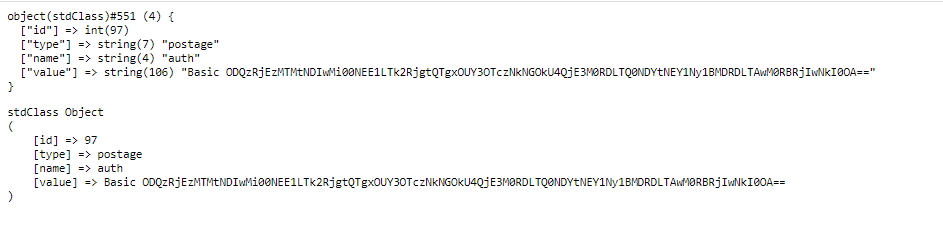
#### Db-adapter下fetchAll()

Db-adapter下fetchAll()方法，相当于将adapter中的query和与PDOSO的fetchAll方法集合到了一起，这样就可以直接获取到处理后的结果集。对于每一种 fetch系列 的方法来说，你需 要传送一个select的sql语句。



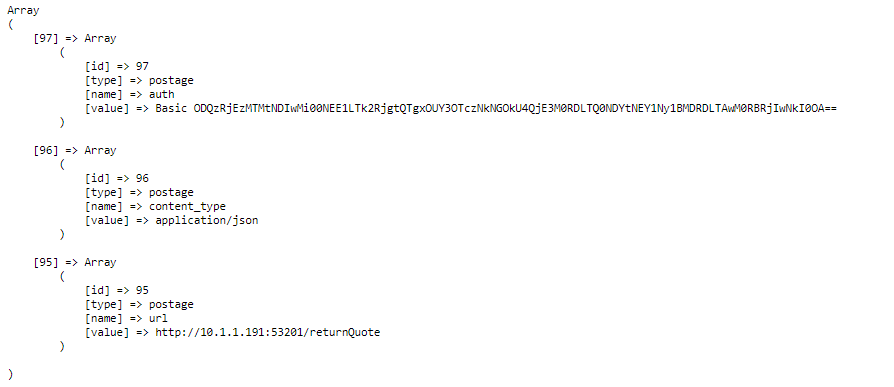
#### Db-adapter下fetchRow()

Db-adapter下fetchRow()方法，相当于将query和与PDOSO的fetch()方法集合到了一起，这样就可以直接获取到处理后的结果集。



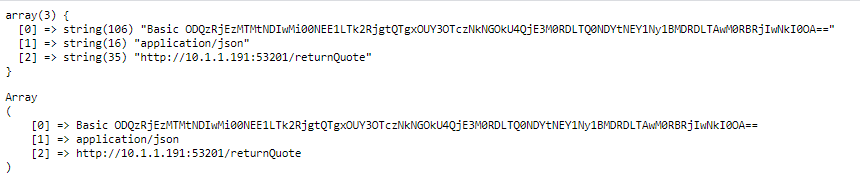
#### Db-adapter下fetchAssoc()

Db-adapter下fetchAssoc()方法，获取到二维数组。区别fetchAll获取的数组里面包含的是标准类对象，即stdClass Object。



#### Db-adapter下fetchCol()

Db-adapter下fetchCol()方法，获取结果集的第一列数据作为数组返回。



#### Db-adapter下fetchOne()

Db-adapter下fetchOne()方法，获取结果集的第一列第一行数据作为一个单独值返回。

对于insert，update，delete增改查等方法，在中文手册里面有说明。

zend中的db对象



PDO就是PHP把操作数据库的函数封装成一个pdo类，其间做了安全验证而已，方便针对不同数据库进行调用。、

zend中的db对象，就是一个pdo类。可以调用pdo下面的方法，以上图片已经验证，如prepare sql之后，可以execute执行。不过已经封装了update等方法，更加方便。

## Zend\_Registry

<https://www.php.cn/php-weizijiaocheng-403846.html>