# 一、入口文件index.php：

给入口文件添加一个header头声明字符集：

Header(‘Content-Type:text/html;charset=utf8’);

开启调试模式：

Define(‘APP\_DEBUG’,true);

定义应用目录：

Define(‘APP\_PATH’,’./Application/’);

定义一些常量：



引入ThinkPHP框架入口文件：

Require ‘./ThinkPHP/ThinkPHP.php’;

# 二、Controller(控制器)：

## 控制器的创建：

命名规则：控制器名+Controller+.class.php

控制器结构代码：

第一步：声明当前控制器的命名空间

Namespeace Home\Controller

第二步：引入父类控制器

Use ThinkPHP\Controller

第三步：声明控制器并继承父类

Class UserController extends Controller{

}

## 路由形式：

路由：是指访问项目中具体某个方法的URL地址

在ThinkPHP中系统提供了4种路由形式：

普通形式路由(get形式路由)：

http://网址/入口文件?m=分组名&c=控制器&a=方法名&参数名=参数值

Pathinfo形式路由(默认)：

http://网址/入口文件/分组名/控制器/方法名/参数名/参数值

Rewrite形式路由：

http://网址/分组名/控制器/方法名/参数名/参数值

需要配置才能使用:

第一步：需要修改Apache配置文件Apache/conf/httpd.conf，开启重写模块，去掉之前的注释符

LoadModule rewrite\_module modules/mod\_rewrite.so

第二步：需要修改虚拟主机配置文件Apache/conf/extra/httpd-vhosts.conf

AllowOverride all

第三步：重新启动Apache

第四步：将ThinkPHP压缩包中的.htaccess复制到入口文件的同级目录(需注意该文件是不是需要的文件，因为运行模式文件有好几种)

兼容形式路由:

URL\_MODEL路由形式的配置值，不影响我们在地址栏中直接输入其他形式路由访问，影响的是U方法组装生成的url地址形式。

## 控制器中的跳转：

### URL组装

语法：U(‘url路径’，参数数组);

例1：要使用U方法组装出当前控制器下Index方法地址

U(‘index’);

例2：要使用U方法组装出另一个控制器Index下Index方法地址

U(‘Index/index’);

例3：要使用U方法给指定的页面添加参数，给Index控制器下的index方法传递一个id=100

U(‘Index/index’,array(‘id’=>100,’name’=>’smith’));

总结出一个通用的格式：

U(‘[分组名/控制器/]方法名’，array(‘参数1’=>’值1’，‘参数2’=>’值2’));

URL\_MODEL路由形式的配置值，不影响我们在地址栏中直接输入其他形式路由访问，影响的是U方法组装生成的url地址形式。

### 2.系统跳转方法：

在ThinkPHP中系统有2个跳转方法，分别是成功跳转和失败跳转

成功：

$this->success(跳转提示，跳转地址，等待时间);

失败：

$this->error(跳转提示，跳转地址，等待时间);

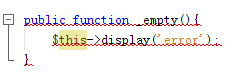
跳转提示参数必须要有，不然会死循环。如果没有指定跳转地址，默认跳转到上一页。

## 空操作：

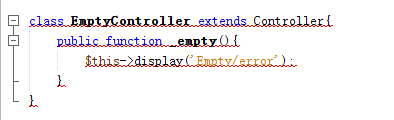
空操作是指系统在找不到指定的操作方法的时候，会定位到空操作方法来执行(针对控制器也是如此)，利用这个机制，我们可以实现错误页面和一些URL的优化。

关于空操作的说明：

1. 空操作方法：在看控制器中可以定义一个操作方法名字叫做\_empty()。创建一个error.html页面放到当前控制器对应的视图文件夹里。



1. 空操作控制器：在ThinkPHP中操作一个空的控制器，当指定的控制器找不到，则会去访问空的控制器，空控制器的文件名叫做EmptyController.class.php。并且需要在View目录下中创建一个文件夹Empty，然后把404页面error.html放入Empty文件夹，当控制器错误时调用\_empty方法，展示error.html页面。



## 在ThinkPHP中获取ip信息：

在ThinkPHP的系统函数库文件中封装了一个方法来获取ip:get\_client\_ip()。

语法：

Get\_client\_ip(可选参数数字)

如果参数是0的话或者不写(默认)，则表示返回正常的ipv4地址

如果参数是1则表示返回ipv4地址对应的数字地址。

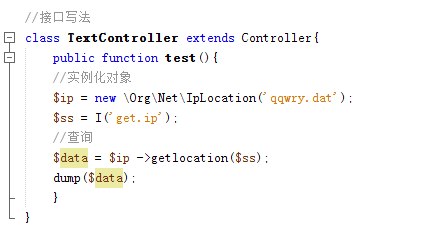
如何把ip地址转化成物理地址：

在ThinkPHP中系统提供了一个工具类，能实现转化，但是系统不提供转化所使用的数据，也就是说需要自己去寻找对应的数据库。数据库可以从纯真官网去寻找(http://www.cz88.net)。下载安装包setup.zip后安装。下载安装的程序后，在其安装目录中可以找到qqwry.dat文件，该危机就是后续所使用到的数据库文件。

ThinkPHP提供的类：Org/Net/IPlocation.class.php

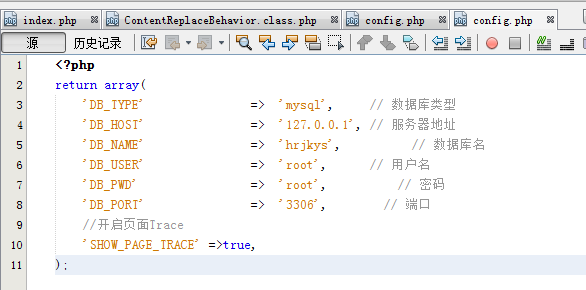
在实例化的时候可以传递一个文件名，文件名所在的位置和当前的类是同级目录，也就是需要把qqwry.dat文件复制到Org/Net目录下，和Iplocation.class.php在同一个目录。

Getlocation：



# 三、Model(模型)：

## 配置数据库连接：



不要修改TP框架的配置文件，把需要修改的内容复制后，在项目目录的配置文件里修改，如上。

## 模型的创建：

模型是MVC三大组成部分的M(Model),作用是负责与数据库的数据交互(CURD)。

视图展示模板(V)

调用模型处理业务逻辑

模型返回数据

对应的控制器(C)

模型(M)

控制器、方法名称以及参数

数据库

模型的创建：

命名规范：模型名(要求是不带前缀的表名并且首字母大写)+Model关键词+.class.php

代码结构规范：

第一步：声明命名空间

第二步：引入父类模型Model.class.php

第三部：声明模型并且继承父类模型

空模型也可以进行数据表的基本CURD操作，因为模型继承了父类，所以可以执行基本的操作。

## 模型的实例化：

模型创建完成后，由于模型本质是一个类，类在使用的时候需要实例化操作

### 普通实例化：

通过自己编写代码来new一个对象

$obj = new 类名();

在控制器中定义一个方法来实例化模型，使用的是普通方式实例化：

### 快速实例化方法：

TP框架为了简单、快速、高效开发，提供了两个快速方法对模型实例化操作：

M()方法;

$obj = M([‘不带前缀的表名’]);

表达的含义：直接实例化父类模型(Model.class.php)。如果指定了表面，则实例化父类模型的时候管理指定的表，如果没有指定表面(没有传递参数)则不关联表，一般用于执行原生的sql语句(M()->query(原生sql语句))。

D()方法:

$obj = D([‘模型名’]);

表达的含义：实例化我们自己创建的模型(分组/Model目录中)；如果传递了模型名，则实例化指定的模型，如果没有指定或者模型名不存在，则直接实例化父类模型(Model.class.php)。

**D()和M()方法的区别：**D方法是实例化自定义模型，如果自定义模型不存在，则实例化父类模型；M方法本身就是直接实例化父类模型。两者的差异就是实例化的对象不一样。

在实际开发中，如果需要使用的操作在父类模型中已经封装好了，则可以直接实例化父类模型，使用M方法，如果父类中方法不能满足开发需求而需要自定义方法，则使用D方法实例化自定义模型。

## 增加操作：

在TP框架中封装好方法：

$model -> add(一维数组);

一维数组要求必须是键值(关联)数组，键必须和数据表中字段名要匹配，如果不匹配则在增加的时候会被TP过滤掉。

add方法返回值是新增记录的主键id。

如果需要添加多个记录：

1. 可以循环使用add方法;
2. 使用addAll方法，语法：$model->addAll(二维数组)，这里的二维数组也必须是关联数组。外层数组必须是下标从0开始的连续索引数组。

## 修改操作：

在TP框架封装好了方法：

$model->save(一维关联数组);

条件需要一维关联数组必须要有主键信息。如果没有主键信息，则相当于批量修改，这在TP中是不允许的，目的是为了防止误操作。

如果没有指定主键信息，返回值是false,表示修改操作并没有执行，而不是指Mysql执行失败。

正常执行的返回值：表示受到影响的行数。int(1)

## 查询操作：

在TP中封装好了方法：select方法和find方法

select方法语法：

$model->select(); 表示查询全部的信息

$model->select(id); 表示查询指定id的信息

$model->select(‘id1,id2,id3...’); 表示查询指定Id集合的信息。

find()方法语法：

$model->find(); 表示查询当前表中第一条记录

$model->find(id)； 表示查询表中指定id的记录

返回值：

select方法返回值是一个二维数组，即使查询的是一条记录返回也是二维数组；find返回值是一维数组。

## 删除操作：

在TP框架中封装好了一个方法：delete();

$model->delete(); 不能使用，删除方法必须有条件

$model->delete(id); 删除指定id对应的记录

$model->delete(‘id1,id2,id3...’); 删除多个id对应的记录

说明：

删除分为两种删除：

物理删除：真删除

逻辑删除：假删除，本质是修改操作。在数据表中定义一个状态字段，比如说是status,取值是0或者1，在查询的时候读取状态是1的；当用户点击删除之后触发修改操作，将状态从1修改成0。之后状态是0的就不会被显示。

# 四、View(视图):

## 什么是视图：

视图就是MVC三大组成部分中V(view),主要负责信息的输出和展示

## 视图的创建：

创建的位置需要是在分组目录下的View目录下与控制器同名的目录中View/Index/index

## 视图的展示：

在smarty中展示模板使用display方法，在ThinkPHP中同样也是display方法

语法格式：

$this->display(); 展示当前控制器下与当前请求方法名称一致的模板文件

$this->display(‘模板文件名[不带后缀]’); 展示当前控制器下的指定模板文件

$this->display(‘View目录下的目录名/模板文件名[不带后缀]’); 展示指定控制器目录下的指定模板文件

## 变量分配(初阶)：

在实际开发中不仅仅是展示模板，往往还需要数据的输出。这个时候变量还在控制器的方法中，需要将数据给传递到模板中并且展示，这个过程叫做变量分配。

在ThinkPHPzhong系统封装好了一个变量的分配方法，这个方法叫做assign()方法。语法如下：

$this->assign(‘模板中变量名’,$php中的变量名);

一般情况，两个参数的变量名都是一样的。

目前已经将var变量传递给了模板文件，在模板文件中展示数据如下：

{$模板中变量名}

例如：

$var = date(‘Y-m-d H:i:s’,time());

$this->assign(‘var’,$var);

$this->display();

默认的变量左右分隔符是{和}，是可以修改的。在配置文件中找到具体配置项TMPL\_L\_DELIM和TMPL\_R\_DELIM。

## 变量分配(进阶)：

### 一维数组的输出：

在ThinkPHP中变量的分配都是使用assign方法。



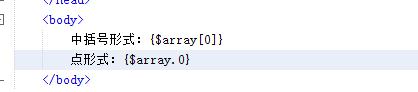
在php中输出数组的具体元素可以通过下标的形式输出：$array[key]

关于数组在模板中输出的语法格式：

支持中括号形式：{$array[key]}

点形式：{$array.key}

定义模板并且输出：



两种形式在输出的效果上没有什么区别，根据自己的喜好使用。

### 二位数组的输出：

在方法中定义一个二维数组，并分配到模板。



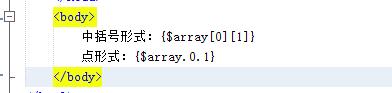
关于数组在模板中输出的语法格式：

支持中括号形式：{$array[key1][key2]}

点形式：{$array.key1.key2}

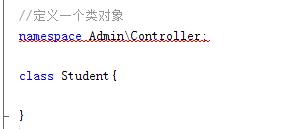
两种形式在输出的效果上没有什么区别，根据自己的喜好使用。

定义模板并且输出：



### 对象变量的输出：

对象在实例化之后一般会保持到一个变量中，这个变量也可以被分配到模板当中去。



实例化对象：

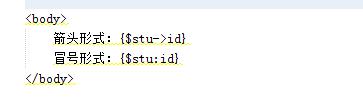


在php中通过$obj->attr或者$obj::attr形式输出对象的属性.

在ThinkPHP的模板中输出属性的值，可以通过下面的两种方式来实现：

支持箭头形式：{$obj->attr}

支持冒号形式：{$obj::attr}



切记：在模板中输出对象的属性时，不能使用点形式。

### 系统变量：

在ThinkPHP中系统提供了一下几个系统变量(超全局变量在模板中的使用)：

$Think.server 等价于$\_SERVER,获取服务器的相关信息

$Think.get 等价于$\_GET,获取get请求的信息

$Think.post 等价于$\_POST,获取post请求的信息

$Think.request 等价于$\_REGUEST,获取get和post请求的信息

$Think.cookie 等价于$\_COOKIE,获取cookie中的信息

$Think.session 等价于$\_SESSION获取session中的信息

$Think.config 获取ThinkPHP中所有配置文件的一个综合，如果后面指定了元素，则获 取指定的配置。

语法格式：

{$Think.xxx.具体的元素下标}

例如：

获取get请求中的id

{$Think.get.id}

案例：

{$Think.server.path}

{$Think.get.id}

{$Think.post.id}

{$Think.request.id}

{$Think.config.DEFAULT\_MODULE}

## 视图中使用函数：

在实际开发中，有些变量在模板中不能直接使用，比如在数据表中存储的时间戳，必须在视图中使用函数的方式来解决这个问题：

语法格式：

{$var|fun1|fun2=参数1，参数2...}

参数说明：

$var ： 变量

|： 变量修饰符

Fun1: 表示需要使用的第一个函数

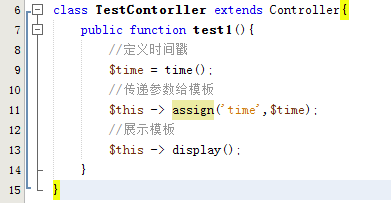
Fun2: 表示需要使用的第二个函数

参数1、参数2： 函数2的参数

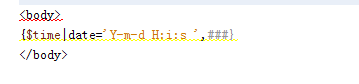
=：表示该函数有参数

###:表示变量的自身

案例1：时间戳的格式化。



模板中应用：



输出结果：

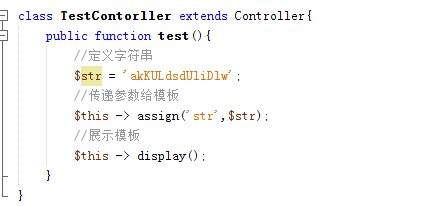
2017-3-14 15:10:22

特别说明：

第一：###什么时候该写，什么时候不该写。当需要使用的函数只有一个参数并且是变量自身的时候，###可以省略。当需要使用的函数有多个参数，但是其第一个参数是变量自身的时候，也可以省略不写###。

第二：关于函数名的说明，函数名对应的函数必须是php内置的函数或者是在函数库文件中定义好的函数，其他主观臆造的函数不能使用。

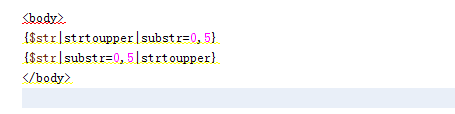
案例2：定义一个字符串，截取其中的前5个字符，并将其转化为大写。



分析：截取字符串使用函数substr,转化大写使用strtoupper。

Substr(变量，起始位置，长度);

模板中使用：



这里的substr函数由于第一个参数是变量$str本身，所以###省略不写，写全因该是{$str|substr=###,0,5}

## 默认值：

默认值：就是当某个变量不存在或者为空的时候，就会显示默认的字符，默认的字符就是变量的默认值。

语法：

{$变量名|default=默认值}

Default是TP中封装的一个函数，默认值是函数的参数。

## 运算符：

{$a+$b}

{$a-$b}

{$a\*$b}

{$a/$b}

{$a%$b}

{$a++}或{++$a}

{$a--}或{--$a}

在方法中定义变量a和变量b传递给模板，在模板中运算。

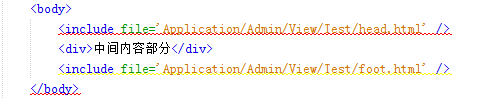
## 文件引入/包含：

在实际开发的时候一般情况下会把网站的公共部分，如头部、尾部等可以单独存放到一个文件中，在后期的时候可以直接引入该部分。维护方便，代码不重复。

在TP中系统提供了一个模板标签，可以让我们引入一些公共部分的代码文件，这个标签是include标签：

<include file=’需要引入的模板文件’>

说明：路径可以是相对路径，但是相对于入口文件的。



说明：在实际开发的时候，上述的路径很长不容易记，一般采用另外一种比较简单的方法：

<include file=’View目录名/模板文件名’ />



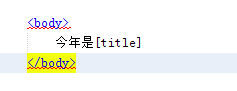
除了使用include标签来引入文件外，include标签还有另外一个用法：传递参数给引入文件

<include file=’文件路径’ 参数名=’参数值’ />



在目标文件中使用参数：

[参数名]



说明：如果目标文件中的参数[title]不存在，则[title]会被原样输出到浏览器上。

## 循环遍历：

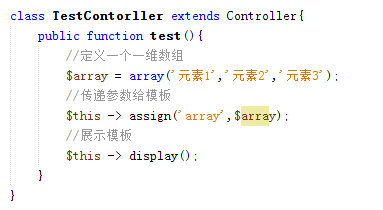
在TP中系统提供了2个标签来实现数组在模板中的遍历：

### Volist标签：

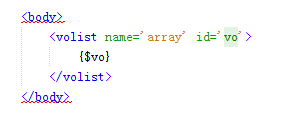
<volist name=’需要遍历的模板变量’ id=’当前遍历到的元素’>

</volist>

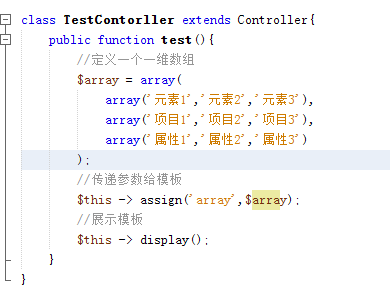
案例1：一维数组



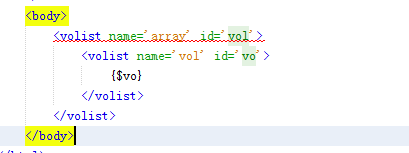
模板中遍历一维数组：



案例2：二维数组



模板中遍历二维数组：



### foreach标签：

<foreach name=’需要遍历的模板变量’ item=’当前遍历到的元素’>

</foreach>

两个标签大体上是一样的，区别在于：volist除了上述的name和Id属性外，还支持更多的属性对，如Mod、key、length等。而foreach除了上述的name和item外只支持key属性对，可以理解为foreach标签是volist标签的一个简化版本。建议使用volist标签。

## IF标签：

语法格式：

<if condition=’条件1’>

输出结果1

<elseif condition=’条件2’>

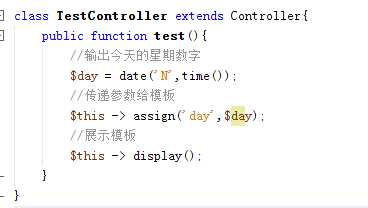
输出结果2

<else/>

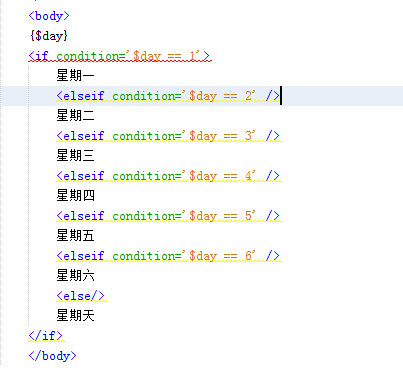
最后一个输出结果

</if>

案例：输出今天的星期，传递给模板



在模板中使用：



## Php标签：

Php标签即使指在模板中使用php语句。

TP支持2中形式：

第一种：PHP内置的php标签；语法格式<?php php代码?>

第二种：TP内置的php标签;语法格式<php>php代码<php>

在世界开发的时候一般情况下不建议在模板中使用php标签。在配置项中有一个配置项可以禁用php标签，配置项叫做：TMPL\_DENY\_PHP,只是禁用php原生代码，并不会禁用TP封装的php标签。默认是false，禁用改为true。而且禁用后，会出错。建议不禁用。

## 模板常量替换机制：

在实际开发中会出现一个这样的问题：在引入图片、CSS和JS文件的时候，往往需要写一些比较复杂的路径，这个时候可以考虑使用模板常量替换的机制。

在ThinkPHP中系统默认提供了几个常用的模板常量(模板常量不能在控制器中使用)。

\_\_MODULE\_\_ 表示从域名后面开始一直到分组名结束的路由

\_\_CONTROLLER\_\_ 表示从域名后面开始一直到控制器结束的路由

\_\_ACTION\_\_ 表示从域名后面开始一直到方法名结束的路由

\_\_PUBLIC\_\_ 站点根目录下的Public目录的路由

\_\_SELF\_\_ 表示从域名后面开始一直到路由的最后(如果没有参数则和\_\_ACTION\_\_所输出的内容是一样的)。

在ThinkPHP中“模板常量”是通过模板内容替换机制来实现的，并非是常量的定义，替换机制可以查看行为文件ControllerReplace.Behaviour.class.php



其模板常量的实现核心就是字符串的替换。

为了后期使用的方便，我们可以在配置文件中定义模板常量。



## 模板内容获取方法：

在ThinkPHP中有一个方法和display()有点相似，叫做fetch方法

Display()方法：$this->display() 展示模板

Fetch()方法:$this->fetch(); 获取模板(有返回值)

在底层实现上的差异：

Display()方法：替换模板中常量/变量->获取模板内容->输出模板内容

Fetch()方法：替换模板中常量/变量->获取模板内容

Display()方法的前2步操作实际上是通过fetch方法实现的。

例如；

$str = $this->fetch();

Echo $str;

## 视图中的注释：

视图中的注释特指ThinkPHP中视图的注释。

Html中的注释<!--注释内容-->，不会在网页中显示，但是会在页面的源代码中被输出。

ThinkPHP中的模板注释：

行注释：{//行注释内容}

块注释：{/\*块注释内容\*/}

ThinkPHP中的模板注释内容，不会在网页中显示，也不会在页面的源代码中被输出。

普通的html注释属于客户端注释，会在浏览器的源代码中输出；而ThinkPHP中的模板注释则属于服务器端的注释，不会被浏览器输出。

注意：行注释不要当作块注释来写，在行注释和块注释当中(大括号里面)不要再出现大括号(模板变量)

# 五、扩展

## Ueditor富文本编辑器：

Ueditor是百度公司开发的，还有一款类似的插件，叫做CKeditor。

上百度搜索Ueditor下载PHP版本的压缩包，使用可以参考demo.html。

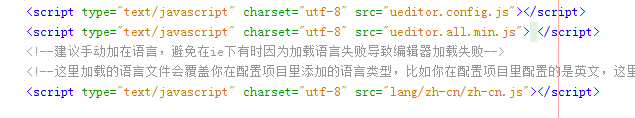
使用UE的步骤：

第一步：引入外部的资源文件(javascript文件)，实际应用中需要修改路径

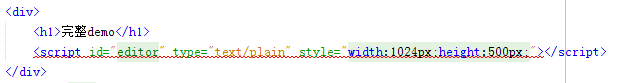
Ueditor.config.js

Ueditor.all.min.js

Zh-cn.js

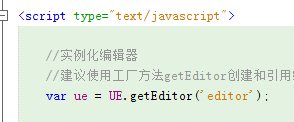


第二步：指定标签，设置容器的位置(编辑器显示的位置)



上述代码是通过id来指定容器的名字，后期实例化容器的时候需要指定的Id。

第三步：实例化容器，生成编辑器效果。



**使用UE编辑器替换掉视图文件中的textarea。**

第一步，将utf8-php解压目录复制到公共静态资源文件目录下的插件目录(plugin)

第二步，在目标模板文件中引入三个需要的js文件。

第三步，设置标签，定义富文本编辑器的位置。

第四步，实例化容器

问题：

1. UE编辑器默认的name值是editorVlue，默认值和数据表中的字段名是不一样的，这就需要我们给当前容器的标签定义一个name值。
2. 在Ue的源码中的一些样式会被ThinkPHP中的I()方法进行转化成实体字符。

UE使用说明：

1. 防止sql注入和xss：通过I方法是不能全部解决的，使用一个插件htmlpurifiy来对指定的标签进行过滤。
2. 关于UE中的表情使用需要联网。
3. 关于图片上传，该功能需要配置，配置文件ue/php/config.json,需要指定路径



在模板文件中读取数据库内容的时候需要将数据表中实体字符进行还原，可以使用函数htmlspecialchars\_decode。

## 验证码：

验证码：captcha(全自动识别机器与人类的图灵测试)。

常见的验证码分为三类：页面上的图片形式、短信验证码、语音验证码。

在ThinkPHP中，系统封装了一个验证码类：Verify.class.php

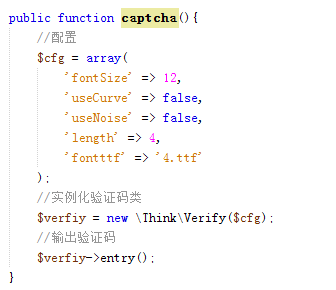
**构造方法：**\_\_confstruct()在实例化的时候可以传递一个数组，用于和其他成员属性config进行合并，生成新的配置

**check方法：**校验验证码，需要传递参数

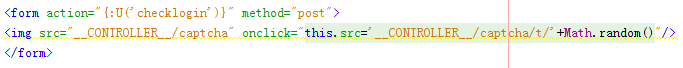
**entry:方法：**输出图片，保持验证码到session中

**步骤：**

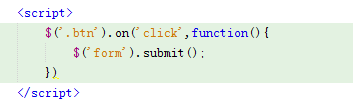
**第一步：在控制器中创建captcha方法，用于输出验证码**



**第二步：在模板文件login.html中输出验证码，修改模板文件，调整大小和样式，增加点击事件刷新验证码。**



第三步：<a>连接提交的话，需要修改Jquery代码段

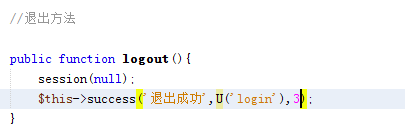


第四步：编写checklogin方法用于处理用户登录

先验证验证码是否正确，如果正确再验证用户名和密码。



退出登录：



关于中文验证码的几点说明：

第一：不到万不得已的情况下不要使用中文验证码

第二：中文验证码需要中文字体的支持，中文字体可以在自己的计算机中找到。

第三：使用中文验证码必须开启php的扩张mbstring

## 上传类：

ThinkPHP中系统封装了一个上传类：Upload.class.php

方法:

构造方法：可以在实例化的时候传递一个配置数组，有内部进行合并配置操作

getError：用户获取最后一次上传的错误信息。

$upload->getError();

uploadone:上传单个文件

参数是$\_FILES中的子元素，返回值是上传的结果。成功返回的时候具有九个元素的一维数组，失败返回false。

upload:上传多个文件，参数通常是$\_\_FILES整个数组，成功返回上传的结果，是二维数组。失败返回false。

注意点：

第一：表单必须有enctype=”multipart/form-data”属性

第二：表单中的文件域type类型必须是file

第三：表单提交方式必须是Post。

**案例：**

**第一步：控制器里创建add方法，调用模型里的saveDate方法保存数据**

为了符合MVC的设计规范，我们需要自定义一个模型，然后将文件上传以及数据的保持在模型中封装一个方法，由这个方法执行数据的保存。

控制器修改：



**第二步：创建模型：DocModel.class.php，并创建saveDate方法。**



如果地址是给服务器端脚本使用的，则可以使用相对于入口文件的相对路径，也可以使用带盘符的绝对路径。

如果地址是给客户端浏览器用的，则地址应该写成”/”形式，相对于站点域名后面开始找。

在上传的案例中整个路径都不会传递给客户的，则属于上述第一种情况。在上传的时候，保存路径建议写成带盘符的路径。

在开发的时候可以将带盘符的绝对路径进行拆分：

例如：D:\WWW\ITCAST\Public\Upload

可以拆分成：D:\WWW\ITCAST(工作目录) \Public\Upload(上传根目录)

而工作目录可以由魔术常量\_\_DIR\_\_表示，则上述的2个路径都可以用常量表示。

常量可以在入口文件中定义



特别注意：保存上传文件的路径的时候，在数据表中千万不要写带盘符的路径，因为上传的文件一般都是浏览器使用，如果使用了带盘符的路径，则会导致Http协议和file协议冲突。

说明：如果后期再写CURD操作，只是简单的基本操作，则可以写在控制器中。如果数据需要处理，则最好是写在模型中进行数据的CURD操作。

## 分页：

数据分页它是通过limit语法来实现的。

在ThinkPHP中系统封装了一个分页类：Page.class.php

方法：

构造方法：有三个参数，但是至少得传递第一个参数(总的记录数)，一般还要指定第二个参数(每页显示的记录数，如果不指定则默认是20).

SetConfig方法：通过Public类型的SetConfig来设置私有属性config

Show方法：生成页面及页面页码上的URL连接。

实际应用步骤：

第一步：查询出总的记录数

第二步；实例化分页类，由于地层实现要求实例化的时候至少需要传递总数，所有在实例化的时候传递参数。

第三步：(可选步骤)定制显示分页提示的文字

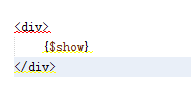
第四步：通过show方法输出分页页码的连接

第五步：使用Limit方法进行分页查询，注意其参数是page分类的属性

第六步：使用assign将查询的数据和分页连接数据传递给模板

第七步：输出模板

模板中展示分页：



控制器中展示内容的方法：

