iWebShop
开源电子商务平台（网店）

www.aircheng.com

iWebShop 内核技术

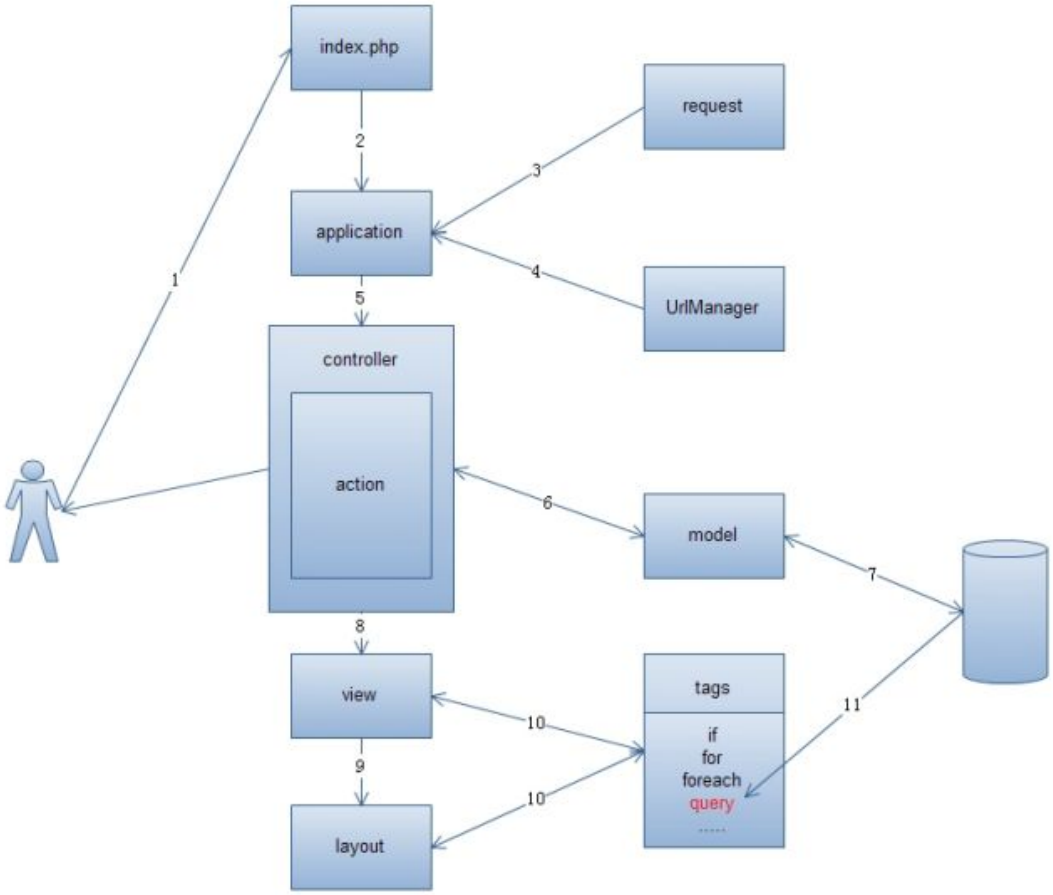
Version V4.7

2017-3-24

目录

iWebcore 处理流程图

首先要了解一下，iWebShop 是建立的 iWebCore 内核的基础之上开发出来的，是一款现代 MVC 模式的产品



1. iWebShop 是单一入口系统，所有的处理都要通过入口文件 index.php 来运行。

2. 系统框架对当前 URL 路径进行分析，找到 controller（控制器）和 action（动作），去调用控制器文件里面的动作方法，URL 路径是调用各个文件的关键！

3. 在 action 动作中可以读取数据，处理业务逻辑，然后把完成的数据渲染到视图，给终端客户呈现出来。

4. iWebShop 不仅支持 WEB 运行方式，同时也支持 php-cli 命令行方式运行（此方式一般用户系统任务等特殊场景）

iWebShop 目录结构

|--backup　数据备份目录.

|--classes　扩展类（自定义）文件目录.

|--config　配制文件目录.

|--controllers　控制器目录

|--docs 软件版本目录

|--install 软件安装目录

|--lib 内核目录

|--plugins　插件目录

|--runtime　编译运行目录

|--upload　上传目录

|--views　视图

|--index.php　统一入口文件

|--.htaccess 伪静态配置文件

对于用户二次开发，重点了解的几个目录是 controllers、views、classes 及 config　目录。

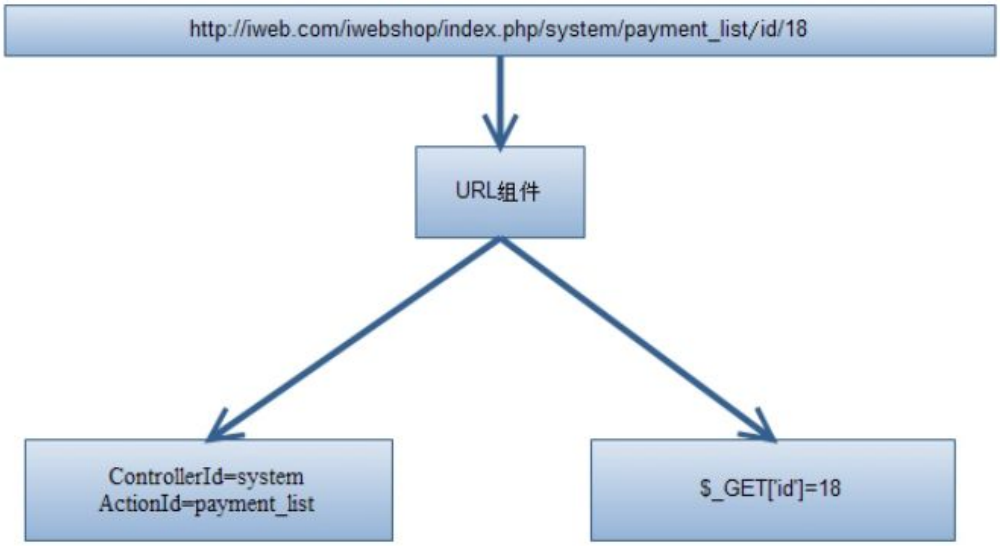
首先了解一下 controllers 目录:

controllers 目录是二次开发必须要熟悉的目录，这里是存放所有控制器类，用户开发自己的控制类一定要继承 IController 类，每个控制器类中都会有很多方法（function），我们把这些方法称为动作（action），也就是说 controller+action 就会运行到某个控制器里面的某个方法了，而方法里面就是具体的 php 代码，就可以运行各种逻辑，处理各种数据和视图呈现，总结：controller 里面包括若干 action，每个 action 就是一个最小的代码单元。

那么写的 action 要如何才能够运行呢？就要靠 URL 解析了。

URL 访问每个动作(action)都是通过 URL 中最重要的 2 个参数！controller 和 action 来进行访问的，controller 参数决定了要引入的控制器文件名，action 参数决定了要调用的 function 名称，这样一个 PHP 的方法代码段就在网页中进行了呈现。

格式:index.php?controller=controllerId&action=actionId



URL 中其余参数都是成对追加在 actionId 后面。

也有一种特殊的情况，当控制器里面的 function 不存的时候，系统会自动调用控制器下的模板进行显示。

配制文件及配制参数说明

❖ [DB]范例：

'DB'=>array(

'type'=>'mysqli',

'tablePre'=>'iwebshop_',

'read'=>array(

array('host'=>'localhost:3306','user'=>'root','passwd'=>'', 'name'=>'iwebshop'),

),

'write'=>array('host'=>'127.0.0.1:3306','user'=>'root','passwd'=>'','name'=>'iwebshop',

),

),

说明：由于内核支持一主多从数据库。所以如果想进行多数据库的支持时，要

配制 read, write。在 read 里面配置多个服务器进行多数据库支持。

❖ [logs]：日志配制信息

path:存放日志的路径。

type:对应的值为 file(文件存储方式),db(数据库存储方式)存放日志的类型。

❖ [viewPath]:视图配制路径，默认为要目录下的 views。

❖ [classes]:存放自定义的类的路径，类型可为 string array。String 时为单一存放路径，可能为数组，来设置多个路径。

❖ [theme]:主题和皮肤配置。支持多客户端自动识别主题。

格式为：‘theme’ => array(‘客户端’ => array(‘主题名称 1’ => ‘主题名称 1——皮肤 1’)...)

如：'theme' => array('pc' => array('default' => ‘red’,’ sysdefault’ => ‘blue’),'mobile' => array('mobile'=>’ default’)), 表示当 pc 客户端访问时 iWebShop 会启用 default 主题及其 red 皮肤，或者 sysdefault 主题及其 blue 皮肤等，而手机客户端访问时则会启用 mobile 主题及其 default 皮肤进行展示。

❖ [timezone]:系统默认时区,系统默认为 Asia/Shanghai。

❖ [debug]:是否为调试模式。

设置 2：所有的错误信息都直接显示到当前页面中，方便调试；

设置 1：仅显示严重的足以让 php 停止的错误；

设置 0：隐藏所有的错误信息，让访问者无感知，应用在正式运营阶段。

- [upload]:文件上传路径。
- [safe]:存储会话变量的方式，session 或者 cookie
- [interceptor]:拦截器，在 iWeb 框架的各个环节可以进行调用绑定的事件函数
- [rewriteRule]:伪静态设置，url：非伪静态；pathinfo：伪静态；
- [configExt]:可扩展配置文件，array(‘配置名称’ => ‘配置文件的路径’)
- [cache]:缓存配置,expire：缓存清除时间；timeout:缓存删除后更新时间；

array(‘server’ => ‘memcache 服务器地址:端口’,’ expire => 2592000,’ timeout’ => 0)；

操作实例：Hello word

WEB 运行方式

(1)动作 action 的方式。

controllers 目录下，然后创建 text.php

```
<?php
 dass Test extends IController
{
        public function hello()
        {
                echo"欢迎使用 Iweb 框架!";
        }
}
?>
```

说明：

这里的文件名要和类名一样，也就是说如果存在一个 text.php 文件，那么这个文件里面必须要有 text 的 class 类！

运行效果如下：

(2)通过视图 Action 来运行，最常用的方式。

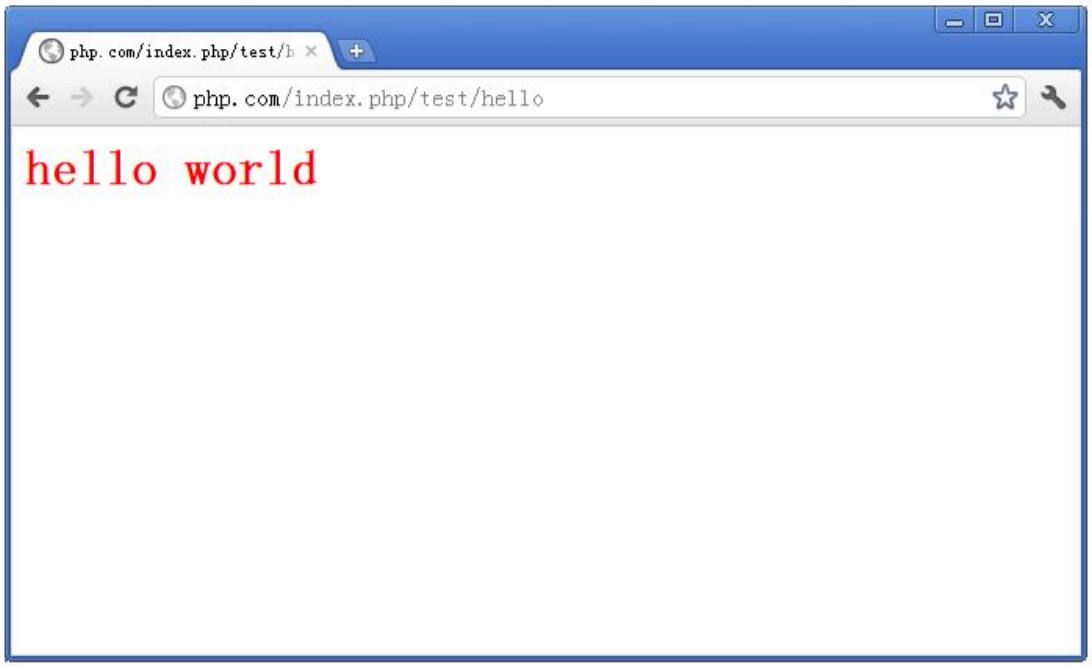
在 views/default/test/hello.html 路径下新建模板文件，我们可以看到如果是给控制器增加视图那么视图目录的名称必须要和控制器的名称相同！也就是说，你要给 test.php 这个控制器增加视图，那么必须要建立相应的 test 目录，把控制器所需要的视图模板文件都放在里面。如图，我们把 hello.html 文件修改成如下内容：

```
<h1 style='color:#F00'>hello world</h1>
```

注释或者删除掉第一种方式。

此时运行 test/hello 就会直接显示 html 模板里面的内容。

运行效果如下：



php-cli 命令行方式

```php
<?php
$iweb = dirname(__FILE__)."/lib/iweb.php";
$config = dirname(__FILE__)."/config/config.php";
require($iweb);

class testCli extends IApplication
{
    public function execRequest()
    {
        echo 'hello word!';
    }
}
IWeb::createApp("testCli",$config)->run();
```

新建 APP 应用类继承 IApplication 基类，重写执行函数：execRequest 即可

最后通过 IWeb 框架调用 createApp->run(); 接口，效果如下：

```
D:\xampp\htdocs\iwebshop\4.8.17040400>php -f a.php
hello word!
```

视图布局 layout

简单说下 iWebShop 的 layout（布局），对于一般网页视图来说，基本的结构形式都是固定好的，比如网页的头部，页脚底部，公共样式，公共 js 类库等等，所有这些公共的部分我们把它抽象出来，做成一个 layout 模板（html 形式的），layout 存在于每个主题模板的 layouts 目录中，可以为不同的控制器（controller），动作（action）指定不同的布局，比如网站的首页和网站的注册页面头部布局就可以不同：

网站首页

网站注册

有了布局，极大的改善了代码的复用（大量公共代码只需要写一份 layout），让各个控制器或者动作都具有可塑性，简单方便容易控制等等，所以说做好 layout 是开发模板的基础！

Layout 的具体开发可以参考本文档最后一章节《主题开发》

介绍一下系统中设置 layout 的几种常用方式：

（1）在控制器（controller）中增加 public $layout 公共属性，表示这个控制器下的所有视图都默认用这个布局

```
<?php
dass Test extends IController
{
        public $layout='site';
}
?>
```

（2）我们也可以在主题目录下的 config.php 文件中配置 layout 布局。

打开 config.php 文件，修改如下内容

```php
<?php
return array(
    'name'    => '步步高',
    'author'  => 'Jooyea',
    'time'    => '2011-01-01',
    'version' => '2.0',
    'thumb'   => 'preview.jpg',
    'info'    => 'Jooyea旗下，IwebShop产品首款默认主题方案，此主题适用于IwebShop 2.0.× 系列产品',
    'layout'  => array('simple' => 'site')
);
```

配置中的 layout 属性书写形式也是很多样性的，比如可以写： layout => array（‘simple’ => ‘site’），这样 simple 控制器也会使用 site 的 layout 布局，如果我们希望 simple 下面有个别的试图用 site_mini 的 layout 布局就应该这么写：

layout => array（‘simple’ => ‘site’，‘simple@reg’ => ‘site_mini’）

小提示：iWebShop 善于用@符号表示包括子关系，比如 site@index 表示的就是 site 控制器下的 index 方法。

（3） 也可以在每个动作（action）方法里面临时设置 layout 比如：

```php
/**
 * @brief 完成或作废订单页面
 **/
public function order_complete()
{
    //去掉左侧菜单和上部导航
    $this->layout='';
    $order_id = IFilter::act(IReq::get('id'),'int');
    $type     = IFilter::act(IReq::get('type'),'int');
    $order_no = IFilter::act(IReq::get('order_no'));
```

注意：如果是模板 layout 增加或者取消的话，必须要清除一下缓存文件，即：iwebshop 根目录下的 runtime 目录。

控制器权限校验

iWebShop4.4 正式版本后权限校验以插件的形式运行，有些时候，我们的控制器里面的方法内容不希望被随意访问到，比如管理后台的操作等。

此时就必须要用到控制器和动作的权限校验，功能强大，易扩展性强，支持角色管理，比如某个身份的管理员可以访问 A，B 方法，而不能访问 C 方法等。

iWebShop 的权限分为 3 大类：

1,admin（后台管理员）继承 adminAuthorization 接口

2,seller（商家管理） 继承 sellerAuthorization 接口

3,user（注册用户） 继承 userAuthorization 接口

使用方法：

（1）权限校验基于控制器，在控制器里面通过 implements 继承接口的方式来确定访问控制器所需要的权限，比如 controllers/goods.php 的控制器是后台商品管理的相关操作我们就让 goods 控制器继承（implements admin）就可以对整个控制器进行 admin 的权限校验了，所有不符合 admin 权限校验的都会拒之门外！其他权限比如 seller 和 user 也是同理，只要是给控制器继承后系统就会自动拦截校验。

（2）权限除了整体拦截之外，还可以对专门的角色进行拦截或者放行，比如后台的 default 默认页面，对于任何管理员都是可见的，但是对于订单管理来说必须只有订单管理权限的这类管理员才能操作，我们可以对管理员进行二次细分，比如商品管理员，订单管理员，系统配置管理员，库存管理员...

（3）角色控制也是通过在控制器里面设置【$checkRight】属性来决定哪个动作（action）需要特殊的身份才能运行。

具体写法：

(1) $checkRight = 'all' 表示所有动作都需要校验

(2) $checkRight = array(

'check(需要校验)' => array('动作 ID'), 'uncheck(不需要校验)' => array('动作 ID'),

)

在对应的 Controller 类中添加如下代码：

```
protected $checkRight  = array('actionId');

actionId:指对应的动作 Action 和视图 Action，如果对该控制器下的所有动作都进行控制，则可以写成 "all",

写成: protected $checkRight  = 'all';
```

再次访问页面,得到如下提示：

用户自定义类

用户自定义类是一个很容易的事情，对于 iWebShop 来说，几乎所有的客户自定义类都是惰性加载（随用随引入），我们只要在 classes 目录下创建自己的类文件就可以了，并且在用户定义的类中可以直接使用任何一个框架系统的类，且不用引入。下面我们来举一个简单的例子。在 classes 目录下创建文件 shoptest.php,代码如下：

```
<?php
class ShopTest
{
        public function show()
        {
                return "shop 自定义类的测试！";
        }
}
?>
```

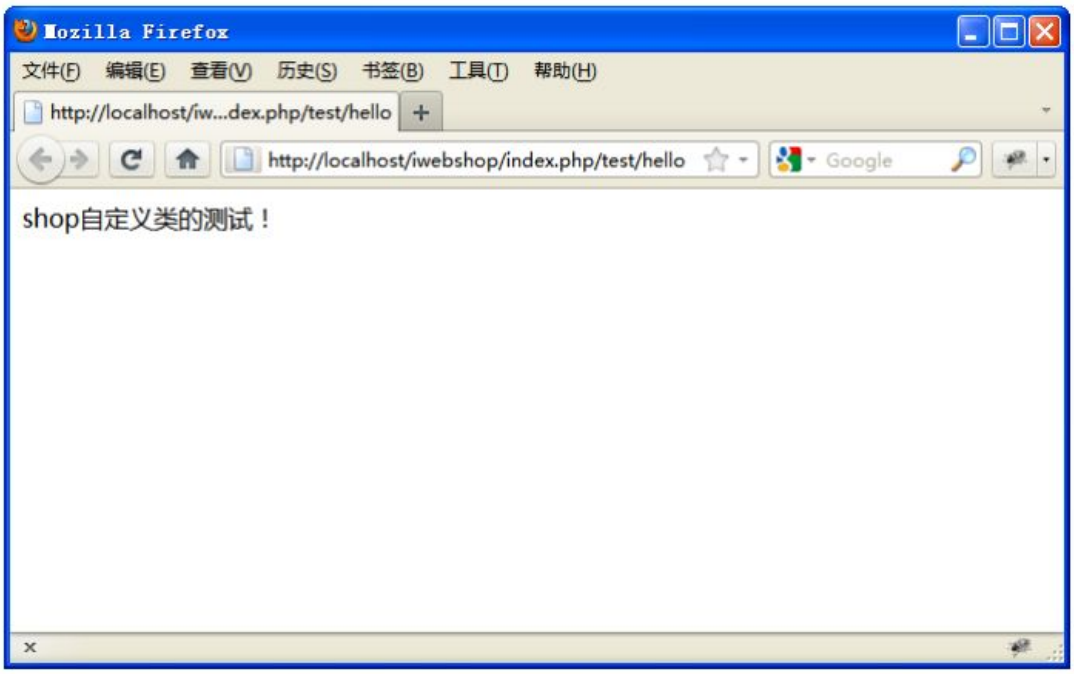
修改 test 控制器类，代码如下：

```
<?php
dass Test extends IController
```

```
{
        public $layout='site';

        public function hello()
        {
                $sp = new ShopTest();
                echo $sp->show();
        }
}
?>
```

运行时效果如下：



如何给模板渲染数据

答：一共有 2 种方法可以把控制器里面的数据传递给模板，可以通过【$this 类属性】和【$this->setRenderData】

```
    function hello()
    {
        $this->aaaaaa = "1234567";
        $this->setRenderData(array("bbb" => "wahaha"));
        $this->redirect("hello");
    }
```

一共 3 行代码：

第一行：展示了第一种传值方式，通过类属性赋值操作，把字符串“1234567”赋值给了 $this->aaaaaa；【推荐】

第二行 :展示了第二种传值方式，通过控制器自身方法 setRenderData 把参数数组里面的键变成模板里面的变量。

第三行：把视图引入进来！ 引入 test 控制器下的 hello.html 视图，注意：如果要给视图渲染数据那么要引入的视图名字必须和 action 方法名字相同，即类方法名字叫 hello，要显示的视图也必须叫 hello。

当然如果方法名称和视图名称不一致却仍想渲染数据，需要在最后的 redirect 方法中增加第二个参数——false 比如要把$this->aaaaaa 传递给 test1.html 视图，那么第三行就要写成： $this->redirect('test1',false); 此时就可以实现显示 test1.html，并且带着 hello 方法里面的数据了。

模板文件可以直接通过以下方式来显示数据

```
<h2>{$this->aaaaaa}</h2>
<h3>{$bbbbbb}</h3>
```

很简单的就把控制器里面的数据传递给了模板！

明白了产品的工作原理，下面来重点说说标签的使用和模板的开发

标签的使用

标签是模板开发中的重要组成部分，他是显示数据，读取数据的工具，标签的介绍我们将从简单到复杂的方式给大家介绍。

标签的书写格式： {标签名:属性},解释一下，这个格式的选择。

下面是一些主要程序对标签的处理：

.net <asp:标签名 属性>补充:vs2010 mvc3.0 作了改进

举例： <asp:Button ID="btnSumit" runat="server" Text="复制" onclick="btnSumit_Click" />

Java struts2<s:标签名 属性>

举例：

<s:if test="#bir>=180 || #bir<0"> <s:/if>

以上的标签存在着一个很大缺点就是，如果美工在开发与调试页面的时候，都会受到标签的干扰，使美工不易于开发，再一个对程序人员来说，对于每一个属性也没必要去写那些没有必要的’”’ 引号去，鉴于以上情况 标签格式定义如下：

我们的 {标签名:属性}

输出类标签

{$name}

直接输出变量 name 的值，这是最直接的，也是大多数主流语言所采用。

{echo:content}

输出 content 内容,content 也可以是方法调用，也可以是常量与变量。

举例：

输出变量 $age {$age}

输出字符串 IWEB {echo:”IWEB”}

地址，路径类标签

url 标签

{url:path} 通过 path 转换为系统统一的路径。

这是最常用的连接方式写法，比如你要访问 site.php 控制器下面的 index.html，就可以在模板里面写成：

首页

path 字符的前 2 个默认为系统的 控制器（controller）和动作（action） ，如果需要传递 url 参数，可以直接以 / 分隔符，按照 /变量名/变量值 这种键值对的形式追加到后面。

另外{url:http://www.aircheng.com} 也支持绝对 URL 路径，总之在 iWebShop 模板里面写网页地址的时候，就必须写成这种形式，保证路径可以正确显示和支持伪静态等。

{webroot:file}表示从 iWebShop 根目录下的路径

如：要引用根目录下的 image 目录下的 logo.png 文件，那么可以使用

此标签就是专门引入资源文件时候使用的。

{theme:file}表示从当前主题目录下的路径

iWebShop 系统是一个支持多主题，多皮肤的商城系统，我们在 views 下面可以放无限个主题方案，默认的比如 default,还有 jd，bubugao 等等，模版库可以参考： http://www.aircheng.com/theme

所以这就使得在开发一套模板的过程中，里面会用到一些文件和资源，需要加载，比如 css，图片，js 等。比如，在 default 主题下的系统首页，views/default/site/index.html 里面需要有个幻灯片的 jquery 插件，书写路径的时候，我们就不用写冗长繁琐的路径了，直接通过{theme:.....}就可以了，比如：

```
<link rel="stylesheet" type="text/css" href="{theme:javascript/jquery.bxSlider/jquery.bxslider.css}" />
<script type="text/javascript" src="{theme:javascript/jquery.bxSlider/jquery.bxSlider.min.js}"></script>
```

这里的{theme:}就表示了 iwebshop/views/default/ 这个路径了。

{skin:file}表示从当前皮肤目录下的路径

原理同上，和主体路径{theme:}基本一致，就是多了在主题目录下的 skin/皮肤目录

如:引用本主体的一个当前皮肤下的一个 css 目录一下的 style.css

文件，则可写成：<link rel="stylesheet" href="{skin:css/style.css}" />

可以直接引用到 views/当前主题方案/skin/当前皮肤方案/css/style.css

{js:name}表示 iWebShop 系统内置 JS

iWebShop 系统默认提供了大量的优秀 JS 工具和插件，比如日历，jquery，artDialog 弹出框 UI，artTemplate 模板引擎等等...所有系统内置的 JS 都在\lib\web\js\jspackage_class.php 有兴趣的用户可以自己扩展一些常用工具，在模板里面引用更为简单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>{echo:$siteConfig->name}</title>
    <link type="image/x-icon" href="favicon.ico" rel="icon">
    <link rel="stylesheet" href="{skin:css/index.css}" />
    {js:jquery}
    {js:form}
    {js:validate}
    {js:dialog}
    {js:artTemplate}
```

如:引用 jquery 则可写成{js:jquery}

大部分工具都可以从百度找到使用教程和 api 手册 ,下面我们主要是介绍 iWebShop 自己开发的一些重要 JS 类库。

❖ 下面说一下，Validate (from 表单校验) 插件的使用：

首先要通过 js 标签引入此插件,然后只要有 form 表单中的 input 标

签中 type 为 text,password,select-one,textarea 中添属性

Pattern（校验规则）和 alt（提示信息） 属性系统将会自动添加验证功能：

如：对 email 的验证:

在 hello 的视图文件里编写如下代码：

{js:validate}

<form>

<input name='email' pattern='email' alt='请输入正确的 email' >

</form>

然后清空 test 控制器里的内容，使其为一空类。

<?php

class Test extends IController

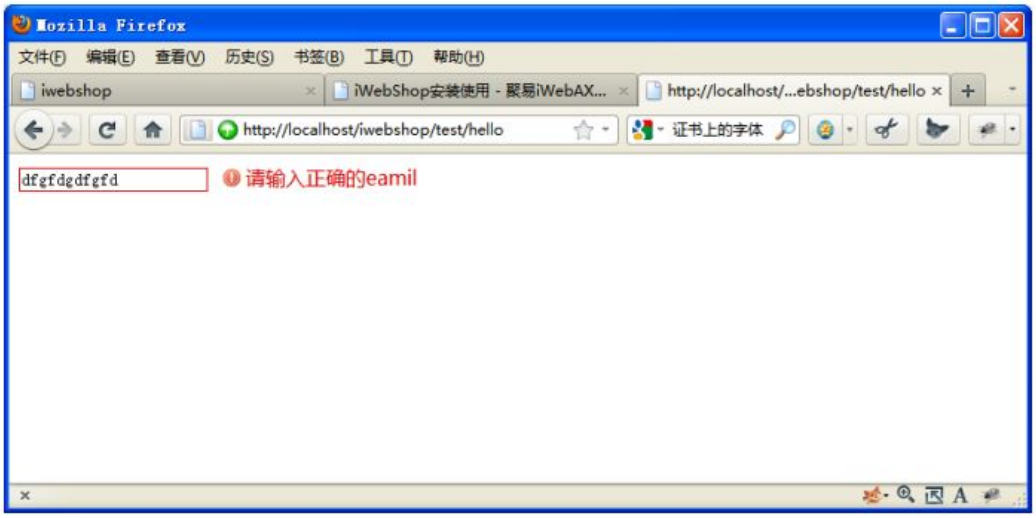
{

}

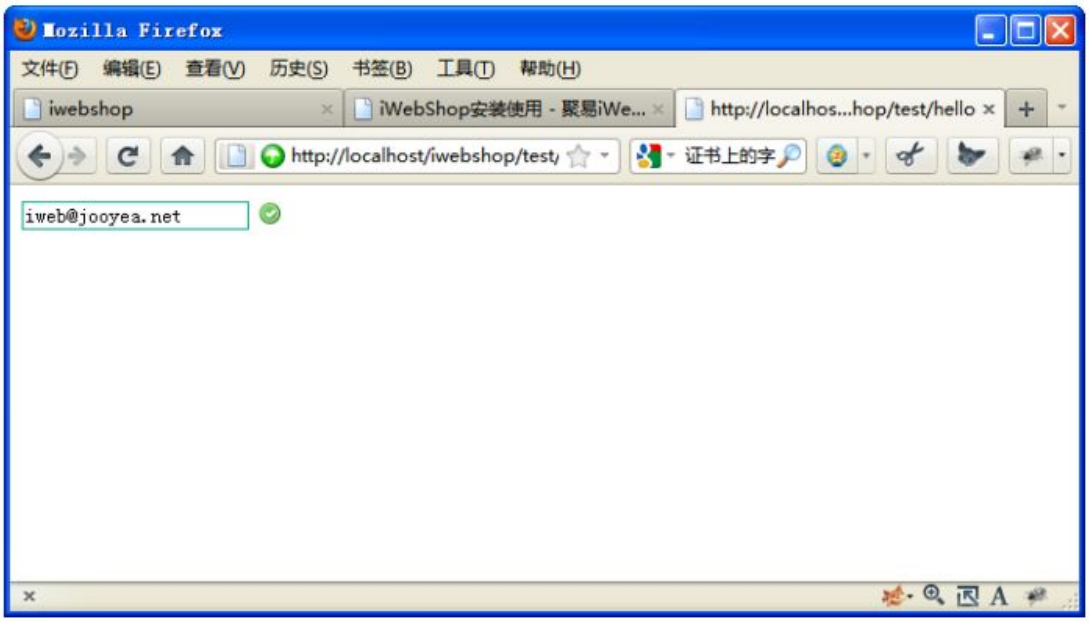
?>清空 runtime 目录

运行效果如下：

当你输入不正确的格式时，效果如下图所示:



当你输入正确的格式时，如下图所示:



对于 pattern 的正则系统对常用的正则都作了一些封装如 :required ,email ,qq ,id ,ip ,zip ,phone ,mobi ,

url ，date，datetime ，int ，float 等等，

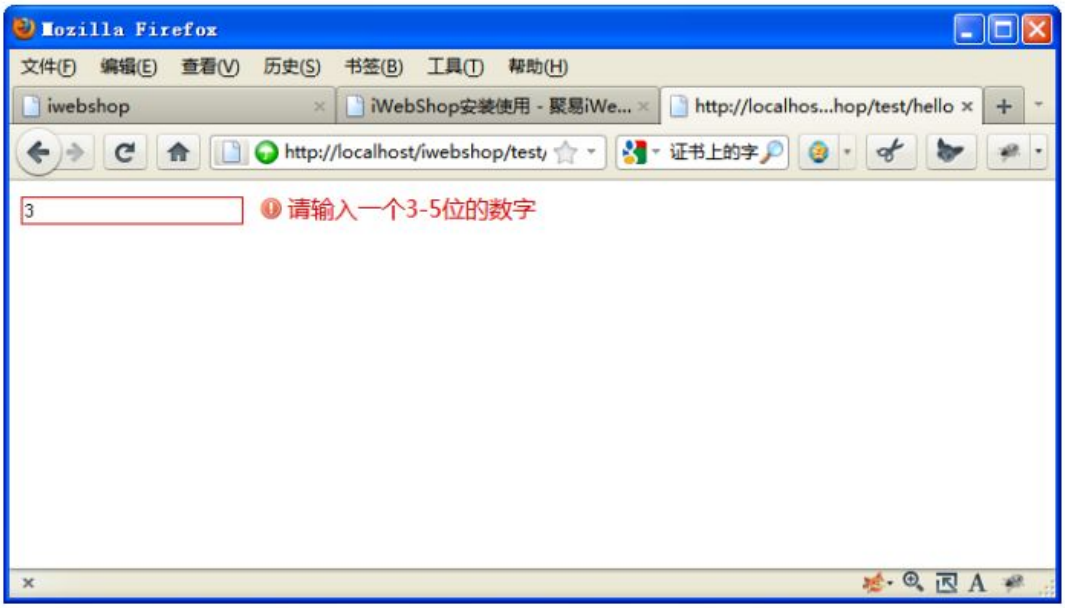
如果以上还没能满足你的要求，则用户可以自己写正则,如，我想让用户输入一

个 3-5 位的数字：则修改代码<input name='email' pattern='^\d{3,5}' alt='请

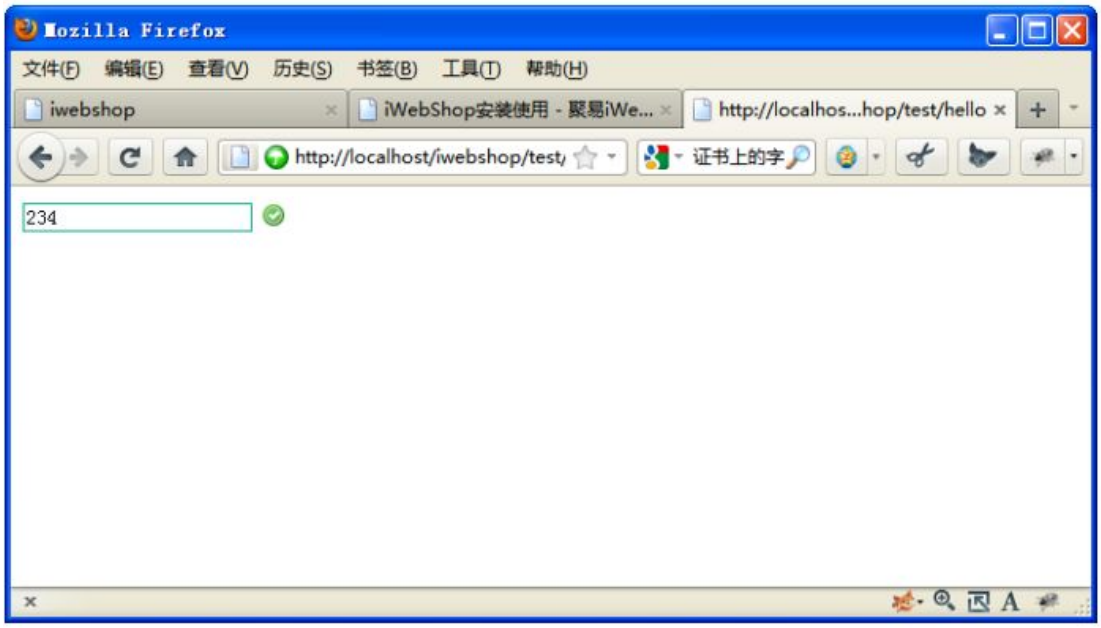
输入一个 3-5 位的数字'>

另外要注意，要验证时，隐藏字段会失效，不进行验证处理！

运行效果:



填写正确:



就是这么简单，那么用户如果想在验证完后再运行自己的 js 函数怎么办？

修改 hello.html 文件

{js:validate}

<form callback='test("回调一下")'>

<input name='email' pattern='^\d{3,5}' alt='请输入一个 3-5 位的数字'>

```
<input type='submit'/>

</form>

<script>

function test(txt)

{

alert(txt);

return false;

}

</script>
```

然后点击提交按钮:

这里要解释一下回调函数的 return true 和 false

如果返回的是 true,表单通过验证后，表单将提交，不通过将不提交。

如果返回的是 false，则无论表单是否通过表单都将不会被提交。

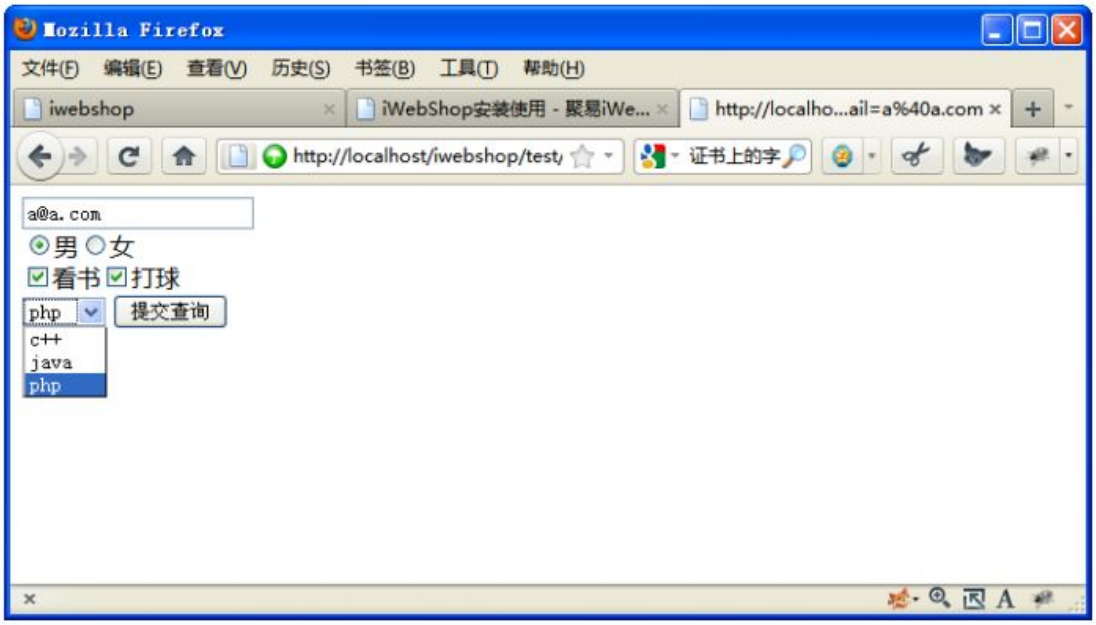
❖ 下面再说一下 form.js（form 表单自动填写） 文件，这个主要是完成表单回写功能的，比方说，你想让表单里

的项初始化一些值：

修改 hello.html 文件：

```
{js:form}
<form name='test'>
<input name='email' ></br>
<input type="radio" name="sex" value='1'>男<input type="radio" name="sex"
value='0'>女</br>
<input type="checkbox" name="live" value='a'>看书<input type="checkbox"
name="live" value='b'>打球</br>
<select name="code">
<option value="c++">c++</option>
<option value="java">java</option>
<option value="php">php</option>
</select>
<input type='submit'/>
</form>
<script>
var form = new Form('test');
form.init({'email':'a@a.com','sex':'1','live':'a;b','code':'php'});
</script>
```

运行效果如下：



同样也可以用 getItems 取得表单里所有数据的对象。也可以用 setValue(name,value)

的方式只给一个设置对应的值，也可以通过 getValue(name)得到对应项的值。

要注意的两点：

1、多选的时候是用;号来分开的。

2、此 form 插件不能对 file 字段进行处理。

此外 iWebShop 还又用到了很多 JS 组件，比如：artDialog，artTemplate，kindEditor 等等，这些公开的类库都

可以通过百度来寻找使用方法，这里不在赘述。

自定义 PHP 代码标签

{set: expression}

此标签是一个简单的标签，主要是自定义 PHP 代码举例：

{set:$name = “iWebShop”;}

{set:$url = Url::createUrl(‘ddd/ddd’);}

{set:break;}

判断类标签

{if: condition} expression {elseif:condition} expression {else:} expression {/if}

类似 php 的 if else 的选择性判断。

举例：

{set:$num=90}

{if:$num>=90}

优秀

{elseif:$num>=70}

良好

{elseif:$num>=60}

及格

{else:}

不及格

{/if}

while 循环标签

{while:condition}expression{/while}

While 标签也是一个很简单的标签，举例:

{set:$num=100;}

{while:$num-->0}

{$num}

{/while}

for 循环标签

{for:attribute}{/for}

关于 for 标签的属性说明：

From: 可选 默认 1 从那一值。

Upto: 可选 默认 10 上升到那一个值。

Downto: 可选 默认 upto 下升到那一个值。

Step: 可选 默认 1 步幅

Item: 可选 默认为 i 输出的变量名。

举例说明：

修改 hello.html 文件，内容如下：

{for:}

{$i}

{/for}

运行效果如下：

修改如下：

{for:from=12 upto=20}

{$i}

{/for}

运行效果：

修改代码如下：

{for:from=10 downto=0 step =-2 item=$num}

{$num}

{/for}

运行效果如下：

foreach 循环标签

{foreach:attribute}{/foreach}

属性：

items:必选 所有遍历的数组

key:可选 默认 key 键值

item:可选 默认 item 每一项

举例说明实用：

修改代码如下：

{set:$numbers = array(1,2,6,7,3,9,34,54)}

{foreach:items = $numbers}

{$item}

{/foreach}

运行效果如下：

修改代码如下：

```
{set:$numbers = array(1,2,6,7,3,9,34,54)}

{foreach:items = $numbers key=$k item=$v}

key:{$k}---value:{$v}<br/>

{/foreach}
```

对于二维的数组，修改代码如下：

```
{set:$numbers = array('a'=>array('k'=>'a'),'b'=>array('k'=>'b'))}

{foreach:items = $numbers}

{$item['k']}<br/>
```

{/foreach}

Include 包含类标签

{include:路径}

可以在模板里面直接包含另外一个模板文件，比如在 site/products.html 模板中包含 site/_products_time.html 和 site/_products_groupon.html 模板，我们就可以这么写：

```
<!--抢购活动,引入 "_products_time"模板-->
{if:$promo == 'time' && isset($time)}
{include:_products_time}
{/if}

<!--团购活动,引入 "_products_groupon"模板-->
{if:$promo == 'groupon' && isset($groupon)}
{include:_products_groupon}
{/if}
```

此时，模板就被引入进来了，上面是在一个控制器下的引入，如果是跨控制器的模板调用我们就用路径的形式来写，比如在 site/products.html 中引入 simple/seller.html 视图，我们就写：

{include:/simple/seller}

另外值得注意的是，模板引入后，需要清空 runtime 缓存才能生效。

query 查询类标签

{query:attribute}{/query}

分页显示：{$query->getPageBar()}

是一个十分重要的标签，属性无任何先后顺序，一定要掌握好,先说说它的属性：

属性名	数据类型	说明
name	String 必选	表名(不带表前缀)
fields	String 可选 默认*	读取表字段
where	String 可选 默认无	查询条件
join	String 可选 默认无	表关联
group	String 可选 默认无	根据字段分组显示
having	String 可选 默认无	分组筛选 设置了 group 属性有意义
order	String 可选 默认无	数据排序
limit	Int 可选 默认 20 条	显示数据条数
page	Int 可选 默认无	分页，设置后通过 Query 对象实例调用 getPageBar() 显示分页
pagesize	Int 可选 默认 20 条	每页显示数据量
pagelength	Int 可选 默认 10 页	分页页码数量
item	Mixed 可选 默认$item	循环数据变量，当循环嵌套时候设置 item=$item1
id	Mixed 可选 默认 $query	Query 对象实例，当模板里面有多个 query 的时候避免冲突，可以设置 id=$query2 赋值给另外一个变量
key	Mixed 可选 默认$key	循环键值变量，当循环嵌套时候设置 key=$key1

cache	String 缓存类型 file,memcache	缓存结果数据
debug	Int 可选 0:关闭；1：开启	是否输出原生态 SQL 语句

在属性中如果遇到下面的符号，一定要注意转换，这一点十分重要，也是初学者容易

犯的错误

原符号	转换符号
=	eq
>	g
<	l
>=	ge
<=	le
!=	neq

注：转换后的符号前后都有空格。

举例查询表的内容：

在数据库里任意创建一张表，或者直接使用 iWebShop 里的表,我这里就使用系统里的

给大家举例了。

举例：

修改代码如下：

```
{query:name=goods}
{$item['name']}<br/>
{/query}
```

运行效果：

下面我们来限制一下条数：

{query:name=goods limit=5}

{$item['name']}

{/query}

同时再进行排序：

{query:name=goods limit=5 order=id desc}

{$item['name']}

{/query}

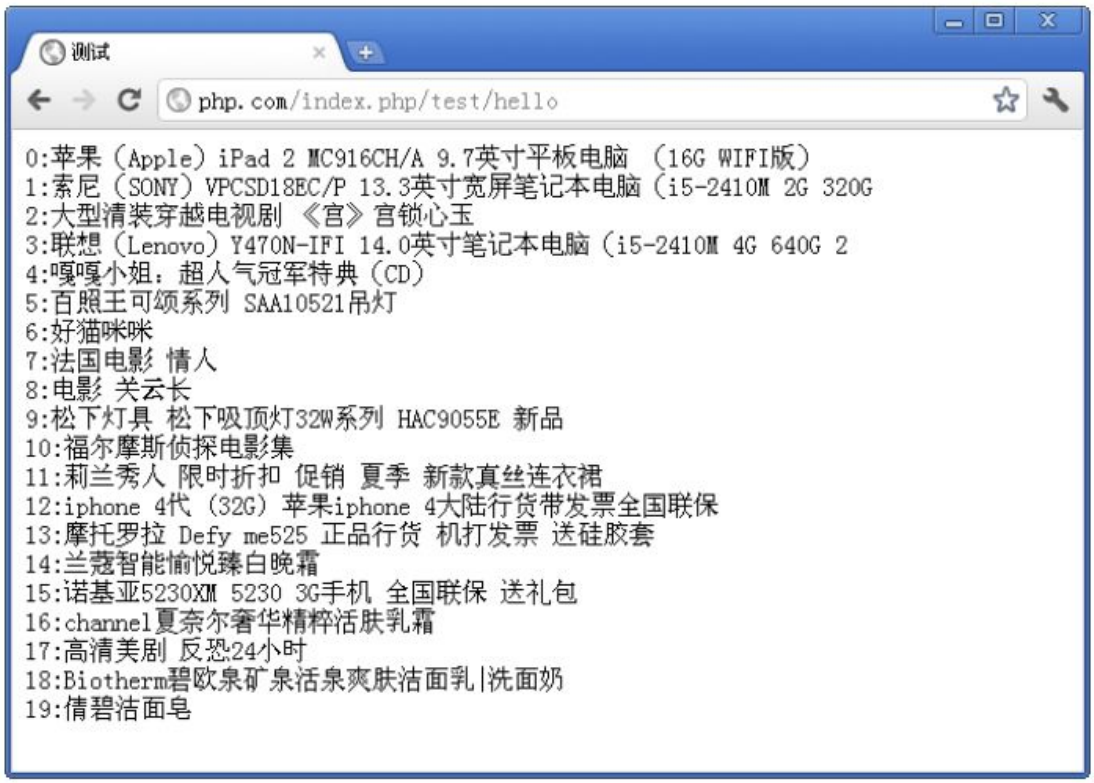
由于所有的属性和 sql 中的完全对应这里就不一 一对应的举例了。

下面说一下分页:

{query:name=goods page=1}

{$key}:{$item['name']}

{/query}



修改代码如下，实现分页：

```
{set:$page=IReq::get('page')==null?1:IReq::get('page');}

{query:name=goods page=$page pagesize=5}

{$key}:{$item['name']}<br/>

{/query}

{$query->getPageBar()}
```

如果想修改分页的样式，查看一下源码，就很容易通过 css 来改变样式了。

也可以做多表关联，比如把 user 表与 member 表进行关联

```
{query:name=user as u join=left join member as m on m.user_id eq u.id}{$item[ ‘user_id’ ]}{/query}
```

数据库读取和写入

（1）读取操作用 IQuery 类

此 IQuery 类就是上文 query 标签中封装的，用法和属性大同小异。

IQuery 提供了很丰富的属性和功能，通过简单的属性配置，就可以实现很多的 SQL 查询功能，比如我们要查询商品表（goods）里面的 id=10 的商品，则：

```
$goodsDB = new IQuery( ‘goods’ );
$goodsDB->where = “id = 10” ;
$goodsData = $goodsDB->find();
```

- find()就是执行查询最终数据的一个接口。
- 构造函数的参数是表名，支持多个逗号，比如：goods,user 等

创建好 IQuery 对象以后，就可以设置各种类属性来组合查询数据了。

主要属性列表如下：

属性名称	数据类型	说明
fields	String 可选 默认*	读取表的字段数据
where	String 可选 默认无	查询条件
join	String 可选 默认无	表连接操作，比如： $db = new IQuery(‘goods as g’); $db->join=” left join products as p on g.id = p.goods_id”; $db->find();
group	String 可选 默认无	表分组操作，比如： $db = new IQuery(‘goods as g’); $db->group = “price”; $db->find();
having	String 可选 默认无	表分组结果的筛选，设置 group 属性才有意义
order	String 可选 默认无	表排序字段
limit	Int 可选 默认 20 条	读取指定数量的条数
page	Int 可选 默认无	设置分页。设置后 IQuery 类库增加 paging 属性（分页类）
pagesize	Int 可选 默认 20 条	每个分页显示的数据量，设置 page 属性才有意义
pagelength	Int 可选 默认 10 页	显示多少页数，设置 page 属性才有意义
cache	String 可选 memcache， file	缓存查询结果提高效率，降低数据库压力。填写缓存的模式，前提是您的系统必须支持 iWebShop 缓存技术
debug	Int 可选 默认 0 关闭 ;1 开启	调试 SQL 语句，系统会自动输出完整的 SQL 原生态语句

（2）写入操作用 IModel 类

数据库写入一般用 IModel 类，目前支持 update（更新），add（添加），del（删除）,dropTable（卸载表）,createTable(创建表)。

一般使用都是通过创建 new IModel（表名）对象来创建的数据库实例，然后调用 setData（）接口进行数据设置。

比如要更新 goods 表的 price 字段，那么就是：

```
$goodsDB = new IModel('goods');
$goodsDB->setData(array('price' => 1000));
$goodsDB->update('id = 2');
```

方法名字	数据类型	说明
update($where, $except=array())	$where：string 更新条件 $except :array 特殊表达式字段（非字符串类型）	更新记录 通过 $this->setData($array); $array（字段=>更新数据）表更新字段对应数据关系
add()		添加记录 通过 $this->setData($array); $array（字段=>添加数据）表添加记录对应数据关系
del($where)	$where：string 删除条件	删除记录 把满足$where 条件的记录都删除
dropTable()		卸载表 把当前 IModel 实例对应的表删除
createTable()		创建表

		创建当前 IModel 实例对应的表 通过 $this->setData($array); 设置创建表元素（多维数组） $array = array("column" => 字段配置 array("type" => 数据类型,"default" => 默认值,"comment" => 字段注释,"auto_increment" => 数值自增长) "comment"=> 表注释 "index" => 表索引 array("索引类型:PRIMARY,KEY,INDEX,UNIQUE" => "字段名称 或 Array(字段名称 1,字段名称 2...)")

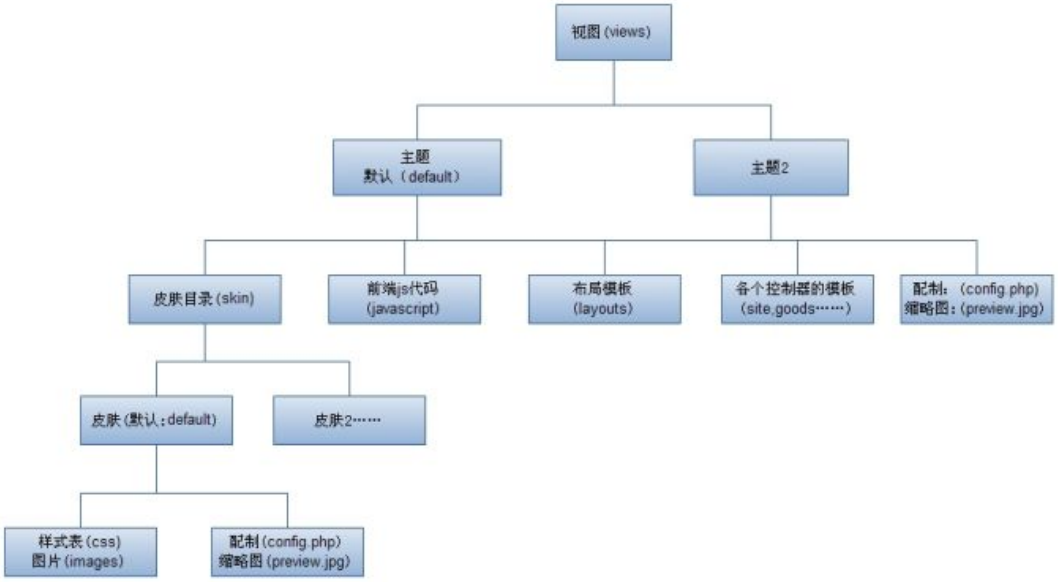
（3）原生态的 SQL 可以用 IDBFactory::getDB()->query($sql);此时可以直接写原生态的 SQL 语句，但是要注意 SQL 语句中把表前缀增加上，因为 IDBFactory 不会对$sql 有任何的修改，直接 100%原样输送给 mysql，当要执行一些特殊的，复杂的 SQL 可以使用此方法。

模板的开发

iWebShop 前端文件的目录结构图及说明

欢迎选购 iwebshop 产品，iWebShop 前端文件

统一放置在程序根目录下的 views 目录下，层次模型图(图 1)：



(图 1)

实际层次结构图(图 2)所示：

Views 目录下的每个目录就是一个模板主题，要启用哪个主题都在 config/config.php 里面的 theme 配置中，包括后台主题和商家主题等。

shop 系统中默认前台主题是 default；默认后台主题是 sysdefault；默认商家主题是 sysseller

主题的组成：布局模板，普通模板，主题相关的 javascript，皮肤。

皮肤的组成：CSS，图片，皮肤相关的 javascript。

如果要新建主题或皮肤只需要在指定的目录位置中新建文件夹即可（文件夹名称就是方案的名称），系统允许存放多个主题，且每个主题又可以拥有多套皮肤，在网站运行时可以从中选择一种主题和一种皮肤进行使用。

iwebshop 的模板分为两种类型：布局模板（layouts）和普通模板。可以简单的理解为共性化的公共模板和个性化的局部模板，对于一个网站来说，很多页面的 head（网站头部，包括导航，广告，引入的 js，css 等）和 foot（网站底部，包括备案信息）往往都是相同的，不同的是页面的主体部分，针对这种情况，shop 的页面在绝大部分情况下采取布局模板和普通模板拼接组合而成（有些特殊情况可以没有布局）。布局模板的创建有利于模板代码共用，更便于模板的升级与维护，所以合理的使用布局是很重要的。

主题的开发

每个主题 （比如：/views/default）下的目录名称（site，simple，ucenter...）都是与控制器（controllers）目录里面的.php 文件名对应的，你要开发哪个控制器下的模板，就去相应的主题目录下新建与控制器名称相同的目录和与视图 action 相同的 .html 文件，比如 controllers/site.php 这个前台控制器文件，如果你要二次开发主题，就要在你自定义的主题目录里面，比如： my_theme 目录下，新建 site 目录，然后把 controllers/site.php 里面所需要的视图都在/views/my_theme/site/目录下新建.html 视图。

在对主题和皮肤进行二次开发时，我们强烈建议不要在原有的 default 方案下进行直接修改，您完全可以在 views 目录下新建一个属于自己的主题目录，如 test，然后把系统默认的 views /default 目录下的所有文件都复制到 test 目录中，最后进入后台[网站管理后台->系统->网站管理->主题设置]，启用 test 方案即可(更改主题的具体步骤可以参考 6.5 章节)，此后所有的开发都基于此目录。

主题的 config.php 配置信息说明：

打开 views / default 目录下的 config.php 文件，其格式一个 php 数组

如图所示：

```php
<?php
return array(
    'name'    => '锦绣前程',
    'author'  => 'aircheng',
    'time'    => '2014/12/29 10:03:05',
    'version' => '4.5',
    'thumb'   => 'preview.jpg',
    'info'    => 'aircheng旗下，IwebShop产品首款默认主题方案，此主题适用于IwebShop4.5+系列产品',
    'type'    => '商城前台-PC',
    'ad'      => array(
        "页面顶部通栏广告条960*70(default)",
        "首页中部通栏960*70(default)",
        "首页右上方198*104(default)",
        "文章-公告内容页左册198*120(default)",
        "商品搜索结果页左侧198*120(default)",
    ),
);
```

从简单的英文单词中很容易看出各个项所要求填写的信息：

name	名称
author	作者
time	开发时间
version	主题版本号
thumb	缩略图名称
info	简介
ad	广告位
api	模板特有的 API 数据接口方法

这些信息会被后台 《主题管理》页面所提取展示，以供网站管理员开启。

6.3 Layout 布局模板和普通模板

此部分是前端模板设计的核心部分，layout 的主要作用是呈现页面的整体轮廓

和共享 head，foot 等公共部分。

先来看一个 shop 中简单的布局模板，打开 views / default / layouts

/site_mini.html ，代码如图所示：

```
<script type='text/javascript' src="{theme:javascript/common.js}"></script>
</head>
<body class="second" >
<div class="brand_list container_2">
    <div class="header">
        <h1 class="logo"><a title="{echo:$siteConfig->name}" style="background:url({echo:IUrl::cre
        <ul class="shortcut">
            <li class="first"><a href="{url:/ucenter}">我的账户</a></li>
            <li><a href="{url:/ucenter/order}">我的订单</a></li>
            <li class='last'><a href="{url:/site/help_list}">使用帮助</a></li>
        </ul>
        {set:$user = $this->user}
        <p class="loginfo">{if:(isset($user['user_id']) && $user['user_id']!='')}{$user['username'
        {set:$callback = IReq::get('callback')}
        {if:$callback==""}
        <a class="reg" href="{url:/simple/reg}">免费注册</a>
        {else:}
        <a class="reg" href="{url:/simple/reg?callback=$callback}">免费注册</a>
        {/if}
        ]{/if}</p>
    </div>

    {viewcontent}        ▶ 普通模板的替换标签

    {echo:$siteConfig->site_footer_code}

</div>
<script type='text/javascript' src='{theme:javascript/site.js}'></script>
</body>
</html>
```

(图 4)

可以看到 html 和模板标签，代码的组织结构与普通模板是完全相同的，唯一特殊的是在布局模板中有一个特殊的标签，即 {viewcontent} ，此标签是其他非布局模板所没有的，这就是普通模板的替换标签，前面讲过一个页面是由布局模板和普通模板拼接组合而成的，这里的拼接组合就是通过 {viewcontent} 这个标签来实现的，因为系统在运行时会自动把此标签替换为浏览器要访问的普通模板内容，所以你可以把这个标签理解为是一个普通模板，他被嵌入在布局模板之中。 布局是被定义在各个控制器文件中的$layout 属性中，控制器的具体位置是在根目录下的 controllers 文件夹内，如图所示：

(图)

这里各个控制器的文件名与主题方案（default）目录下的各个模板文件夹的名称是一一对应的，也就是说如果在某个控制器中设置了布局模板（$layout 属性值）则该控制器下的所有模板默认都会调用这个布局模板。比如，在 simple 控制器中设置布局模板，打开 simple.php 这个控制器并查看源码，如图所示：

```
class Simple extends IController
{
    public $layout='site_mini';
```

$layout 的值就是布局模板的名称了。

我们通过 URL 来访问<用户注册>页面，如图：

<用户注册>页面的模板代码在 views/default/simple/reg.html 如图：

```
<div class="wrapper clearfix">
    <div class="wrap_box">
        <h3 class="notice">用户注册</h3>
        <p class="tips"><span class="gray f_r">已有iWebShop帐号？请点<a class="orange bold
        <div class="box clearfix">
            <form action='{url:/simple/reg_act}' method='post'>
            {set:$callback = IReq::get('callback')}
                    {if:$callback!==null && $callback!=''}<input type="hidden" name='callb
                    {else:}
                    <input type="hidden" name='callback' value="{echo:IUrl::getRefRoute()}
                    {/if}
                <table class="form_table f_l">
                    <col width="260px" />
                    <col />
                    <tr><th>邮箱：</th><td><input class="gray" type="text" name='email' va
                    <tr><th>用户名：</th><td><input class="gray" name='username' value='{$
                    <tr><th>设置密码：</th><td><input class="gray" type="password" name='p
                    <tr><th>确认密码：</th><td><input class="gray" type="password" name='r
                    <tr><th>验证码：</th><td><input type='text' class='gray_s' name='captc
                    <tr class="low"><th></th><td><img src='{url:/simple/getCaptcha}' id='c
                    <tr><td></td><td><input class="submit_reg" type="submit" value="同意一
                </table>
            </form>
```

可以看到 reg.html 只是<用户注册>页面的一个片段，这个片段替换掉了

site_mini.html 布局中的{viewcontent}标签，所以最终显示的页面代码就是

site_mini.html 和 reg.html 的组合。

皮肤的开发

皮肤的 config.php 配置信息说明：

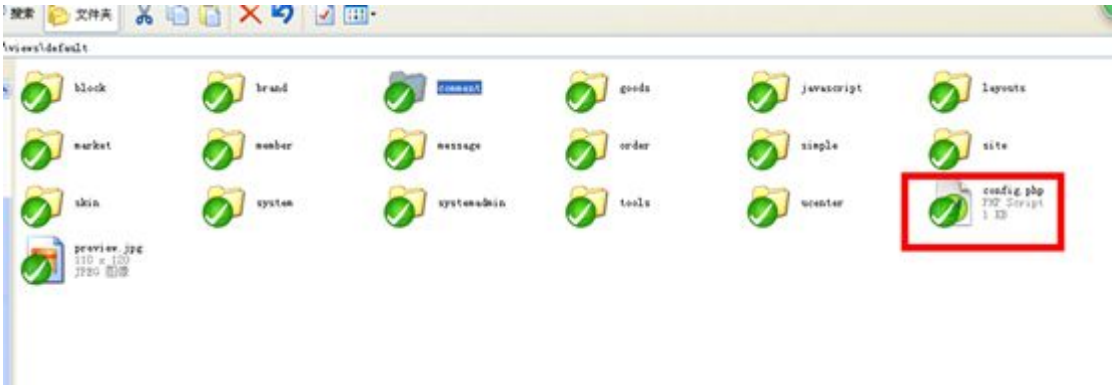
皮肤的配置信息在对应皮肤方案的目录下，默认

views/default/skin/default/config.php，因为其配置方式与数据结构和主题信息的配置方法完全相同，所以可以

直接参考主题的 config.php 配置方法(图 3)及(表 1)。

如果控制器不需要布局模板则直接把$layout 设置为空即可，此时该控制器下的

每个模板都必须是一个完整的 HTML 文档。




```php
<?php
return array(
    'name'    => '锦绣前程',
    'author'  => 'Jooyea',
    'time'    => '2011-01-01',
    'version' => '2.0',
    'thumb'   => 'preview.jpg',
    'info'    => 'Jooyea出下，IwebShop产品首款默认主题方案，此主题适用于IwebShop 2.0.x 系列产品',
);
?>
```

主题与皮肤的切换

主题和皮肤都可以在《网站管理后台->系统->网站管理->主题设置》自由切换。

左侧菜单包括：网站前台，后台管理，商家管理 一共是 3 类主题模板，开发者可以根据控制器目录不同而分别开发不同场景用到的主题！

插件机制

插件是一个很神奇的东西！他可以在不改变系统运行的情况下动态增加功能，增加拦截，增加封装等等，最新版本的 iWebShop 很多重要的地方都用到了插件，比如后台菜单和个人中心菜单，商家和管理员的权限校验，用户的注册登录等等，甚至插件还可以定于全局的 JS 方法和常量，开发者可以对插件进行修改和开发，具体开发步骤和流程可以参考《iWebShop 插件开发手册》，目前常用的系统插件可以参考《iWebShop 产品使用手册》中的插件部分。