Public CMS 2016

PublicCMS笔记整理

作者：Be OK

目录

[第一章 网站配置 3](#_Toc462329282)

[第二章 字段的使用 4](#_Toc462329283)

[1. 分类 4](#_Toc462329284)

[2. 内容 5](#_Toc462329285)

[3. 其他 6](#_Toc462329286)

[第三章 后台管理 7](#_Toc462329287)

[1. 模型 7](#_Toc462329288)

[2. 页面和模板 7](#_Toc462329289)

[3. 分类和内容 8](#_Toc462329290)

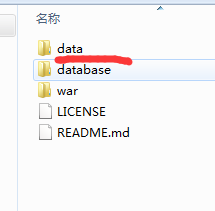
[4. 代码 9](#_Toc462329291)

[第四章 数据库的连接 13](#_Toc462329292)

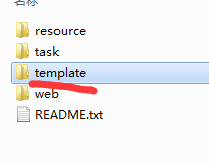
# 网站配置

使用PublicCMS-war-master开发（只用于非静态化的操作）：

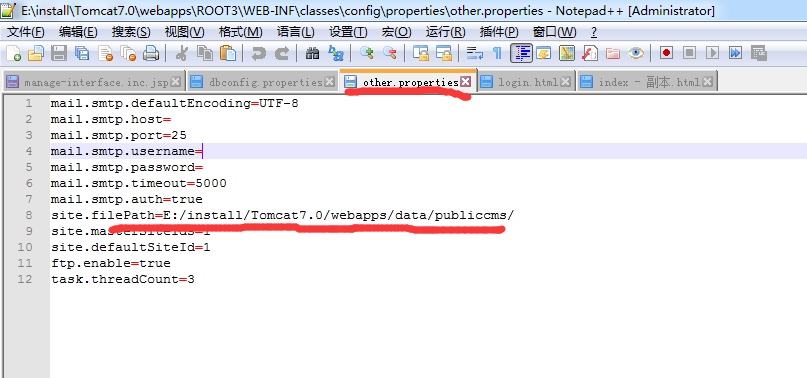
data文件夹存放前台的东西



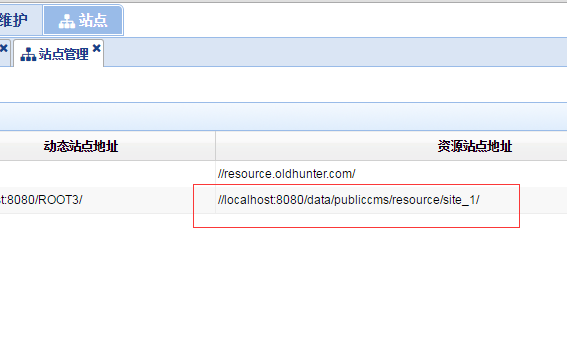
Template存放的是前台模板



模板存放位置，如果没有模板则通过后台增加而增加。如果已经存在模板，则可把路径指向模板（指向上诉的Template文件夹上一级位置）



资源站点位置设置，包括css样式，js脚本和上次资源路径等。如果没有则会根据位置自动生成。如果有上诉的data文件夹，拷贝在Tomcat7.0\webapps\的文件夹下，然后路径设置为data文件夹里面网址文件夹的位置。如下图所示：



# 字段的使用

## 分类

<@\_categoryList>

<#list page.list as a>

<li>

<a href="${a.url!}">${a.name}</a>

<#if a.childIds?has\_content>

<ul>

<@\_categoryList parentId=a.id>

<#list page.list as a>

<li><a href="${a.url!}">${a.name}</a></li>

</#list>

</@\_categoryList>

</ul>

</#if>

</li>

</#list>

</@\_categoryList>

## 内容

<@\_contentList modelId=1 orderField='clicks' count=10>

<#list page.list as a>

<li><a href="${a.url!}"

target="\_blank">${a}</a><span>${a.clicks}</span></li>

</#list>

</@\_contentList>

<@\_content id=id >

<#assign content=object/>

</@\_content>

<html>

<head>

<title>${content.title}</title>

</head>

</html>

获取@标签的集合对象

1.<#list page.list as a>

<li><a href="${site.dynamicPath}place/redirect.do?id=${a.id!}"

target="\_blank">${a.title}</a></li>

</#list>

2.<#list page.list>

<ul>

<#items as a>

<li><a href="${site.dynamicPath}place/redirect.do?id=${a.id!}"

target="\_blank">${a.title}</a></li>

</#items>

</ul>

</#list>

模型

<@\_modelList>

<#list page.list?reverse as model>

<@\_contentList modelId=model.id count=3>

<#if page.totalCount gt 0>

<h3>最新${model.name!}</h3>

<#list page.list as a>

<@include path="/ftl/data-list.ftl"/>

</#list>

</#if>

</@\_contentList>

</#list>

</@\_modelList>

## 其他

1.推荐位标签 <@\_includePlace/>

2.模板元素数据的显示 ${(metadata.extendData.XXXXX?html)!}

3.@标签对应集合对象获取 <#list page.list as a> </#list>

a.cover封面路径

a.title数据标题

a.url 网址

a.childIds 子元素

4.网站根路径 ${site.dynamicPath}

5.资源文件路径 ${site.resourcePath!} 在站点管理中设置

6. 首页地址 ‘#’ ${site.sitePath}

7. <@\_categoryList></@\_categoryList>

8. <#list page.li<li><a href="${a.url!}">${a.name}</a></li>

9. getContentAttribute(id).xxx 调用自定义字段内容，需要在



可接受参数中填写id，才是使用自定字段的内容

10. <script src="${site.resourcePath!}js/jquery-1.9.1.min.js"></script>

<script src="${site.resourcePath!}js/jquery.cookie.js"></script>

<!-- 焦点图组件 -->

<link href="${site.resourcePath!}owl-carousel/owl.carousel.css" rel="stylesheet"/>

<link href="${site.resourcePath!}owl-carousel/owl.theme.css" rel="stylesheet"/>

11.具体类的意义在data\template\cmsTemplate中有个help.html介绍，@标签的对象对应的属性在代码的中的com.publiccms.entities.sys包中可以看到。

# 后台管理

内容、分类、页面、维护、站点。

内容和分类是联合使用的。

## 模型

添加模型：维护->系统维护->内容模型管理

选中链接后添加的模型字段才会被保存。

## 页面和模板

创建模板和更改模板的html代码：维护->模板管理->页面模板

模板元数据：在界面上可以显示自定义的一些字段名称和字段属性。通过

${(metadata.extendData.XXXXX?html)!}调用。字段的内容在页面->页面管理中设置





推荐位管理：推荐位其实就是在一个界面中组成部分的一部分代码，这个代码可以背具体的修改和重用，例如页首和页尾可以当推荐位使用。一个界面可以有多个推荐位，而且推荐位都可以被其他页面调用。

添加推荐位：

推荐位元数据操作：页面->页面管理 选择推荐位进行内容设置。

推荐位数据，访问方式：

1.<#list page.list as a>

<li><a href="${site.dynamicPath}place/redirect.do?id=${a.id!}"

target="\_blank">${a.title}</a></li>

</#list>

2.<#list page.list>

<ul>

<#items as a>

<li><a href="${site.dynamicPath}place/redirect.do?id=${a.id!}"

target="\_blank">${a.title}</a></li>

</#items>

</ul>

</#list>

## 分类和内容

分类：相当于栏目管理

内容：对栏目对应的网页进行编辑

Demonstrate

单独调用内容界面内容：

<@\_content id=id >

<#assign content=object/>

</@\_content>

<html>

<head>

<title>${content.title}</title> <!—调用内容界面固有填写的内容-->

${getContentAttribute(id).text} <!—调用内容界面使用模型的扩展字段-->

</head>

</html>

关于模板网页的导入：

首先把几个分页全部丢在分类中，然后让其的category或者content去绑定页面

## 代码

1.结构

Publiccms

Common

Entities

Logic

Component 组件

Dao 数据库的操作

Service 提供需要的数据库的数据的内容请求

Views

Controller spring拦截器，对数据进行加工

Directive freemarker的设置，与网页数据内容对接，与html界面传值和取值

Sanluan

2.freemark

Macro 和类 来进行定义，然后使用@来调用

1）<#macro name param1 param2 ... paramN ></#macro>

2）宏定义：

<#macro test>

Test text

</#macro>

调用方式：

<@test/>

输出：

Test text

3）<#macro do\_twice>

1. <#nested>

2. <#nested>

</#macro>

调用方式：

<@do\_twice>something</@do\_twice>

输出：

1. something

2. something

4）java代码

XXXXDirective类里面execute方法设置传取数据

例：handler.put("page", page).render();传

handler.getInteger("txt",1)取

<@\_XXXX txt =111>

<#list page.list as a>

${a.text\_view}

</#list>

</@\_XXXX>

原：

public class MyDirective implements TemplateDirectiveModel {

public void execute(Environment env, Map params, TemplateModel[] loopVars, TemplateDirectiveBody body) throws TemplateException, IOException{

…

}

}

env.getOut() 输出

params 获取前端设置的内容

loopVars 循环变量

env.getOut().write(String);输出String

body.render(env.getOut());输出标签里面的东西

root.put("repeat", new com.example.MyDirective());

<@repeat count=3 hr=true>

Test

</@repeat>

例：

Writer out = env.getOut();

**if** (body != **null**) {

**for** (**int** i = 0; i < countParam; i++) {

// 输出 <hr> 如果 参数hr 设置为true

**if** (hrParam && i != 0) {

out.write("<hr>");

}

// 设置循环变量

**if** (loopVars.length > 0) {

loopVars[0] = **new** SimpleNumber(i + 1);

}

// 执行标签内容(same as <#nested> in FTL).

body.render(env.getOut());

}

}

1. <#assign x = 1>
3. 一个参数：
4. <@repeat count=4>
5. Test ${x}
6. <#assign x = x + 1>
7. </@repeat>
9. 二个参数：
10. <@repeat count=3 hr=**true**>
11. Test
12. </@repeat>
14. 循环变量：
15. <@repeat count=3; cnt>
16. ${cnt}. Test
17. </@repeat>

Java代码

1. 一个参数：
2. Test 1
3. Test 2
4. Test 3
5. Test 4
7. 二个参数：
8. Test
9. <hr>  Test
10. <hr>  Test
12. 循环变量：
13. 1. Test
14. 2. Test
15. 3. Test

5）item是一个循环变量. item\_index代表索引号，是数值；item\_has\_next 用于判断当前项之后是否还存在，是布尔型

3.CmsContentDirective类结构

CmsContentDirective

AbstractTemplateDirective

BaseTemplateDirective

BaseDirective

Base

Directive

TemplateDirectiveModel

HttpDirective

BASE:对空和非空的操作

Directive：接口 里面有execute方法 单个参数

Basedirective：空的

TemplateDirectiveModel：包里面的接口

BasetemplateDirective： execute方法 多个参数

HttpDirective： execute方法 多个参数

RenderHandler渲染和打印的接口

Base 非空判断的类

SysSite 系统网站的实体类

CmsContent 内容页面实体类

TemplateDirectiveHandler 前台获取的内容全部丢进来了

4.dwz.js

用来指定标签跳转的：

dwz.min.js

navMenu 上的导航

leftside 左边的菜单

navTab 菜单对应的链接内容

Menus 显示左边的菜单树结构，使用的是menus.html 里面定义的东西。根据prentid来确定显示什么东西

Main 显示主体内容，工作台的页面是根据main.html内容显示，而其他点击后的栏目内容是根据cmspage、cmscontent和cmscategory来确定的。

url页面本身的地址，里面有值时显示内容就按照值里面的来设定。如果没有值就会找到其子类。

authorizedurl 页面里面需要的url地址



5.Merge 变量的赋值操作

6. 对于\_sysMoudleList在index界面中把a.id 赋值给parentid 通过？的形式传给menus里面的\_sysMoudleList了

# 数据库的连接

通过这几个包结构实现，具体部署到项目中了：

