**Public CMS V1.0**

使用手册

目录

[第一章 PublicCMS介绍 1](#_Toc432410042)

[1. 概述 1](#_Toc432410043)

[2. 获取PublicCMS 1](#_Toc432410044)

[3. 目录结构 1](#_Toc432410045)

[第二章 开发工具集成 3](#_Toc432410046)

[1. 工程导入 3](#_Toc432410047)

[2. 开发工具配置 9](#_Toc432410048)

[3. 工程调试 13](#_Toc432410049)

[4. 工程发布 18](#_Toc432410050)

[第四章 部署 23](#_Toc432410051)

[1. 部署方式 23](#_Toc432410052)

[2. 运行环境 24](#_Toc432410053)

[3. 部署步骤 24](#_Toc432410054)

[第五章 模板制作 26](#_Toc432410055)

[1. 命名规范 26](#_Toc432410056)

[2. 数据模型与相关指令 26](#_Toc432410057)

[第六章 二次开发 37](#_Toc432410058)

[1. 工程代码简介 37](#_Toc432410059)

[3. 获取支持 38](#_Toc432410060)

# PublicCMS介绍

## 概述

PublicCMS是开源JAVA CMS，PublicCMS使用了Servlet 3.0,FreeMarker 2.3.23,SpringMVC 4.2.0,Hibernate 4.3.11,hibernate-search 5.3.0等技术及框架，支持全站静态化，任务计划指令化、天然支持JSON/JSONP接口。

PublicCMS支持建设动态站点，也支持建设静态站点。其中动态站点模板在WebRoot/WEB-INF/web 目录下，静态站点模板在config/properties/other.properties 文件中通过 file.templateLoaderPath 参数指定，以方便集群环境或对模板进行版本管理

PublicCMS开源版本采用最开放的授权协议MIT，是跟jquery一样的开源协议。这意味着：

* 你可以自由使用，复制，修改，可以用于自己的项目。
* 可以免费分发或用来盈利。
* 唯一的限制是必须包含许可声明。

## 获取PublicCMS

代码托管

开源中国：<http://git.oschina.net/sanluan/PublicCMS>

GITHUB：<https://github.com/sanluan/PublicCMS>

如果您没有使用过gradle或者maven，并且不打算使用它们，也可以在<https://github.com/sanluan/PublicCMS-lib> 下载PublicCMS依赖的库

## 目录结构

|  |
| --- |
| data  database  doc  nginx-conf  preview  publiccms-admin-by-gradle  publiccms-admin-by-maven  LICENSE  README.md |

* data目录

data/www/publiccms.com文件夹中包含以下目录

|  |
| --- |
| data  static  template  upload |

其中data目录为数据文件目录，static目录为静态化页面文件存放目录，template为静态化页面模板文件目录，upload为静态资源和附件文件保存目录

* database目录

其中的Database Init-\*\*\*.sql为PublicCMS项目数据库初始化脚本，\*\*\*代表日期，Database Change Log.sql为数据库变更记录，随着第一章 PublicCMS版本的不断更新，Database Change Log文件中将有大量的数据结构变更记录，Database Init则是当前最新版本的初始化数据，旧版本升级到新版本可以选择直接用最新初始化数据覆盖旧数据结构，也可以选择执行Database Change Log中的变更sql更新。

* nginx-conf目录

其中publiccms.com.conf为该文件为本开源项目 http://www.publiccms.com/ 官网使用的配置文件副本，localhost-conf目录则是作者本地开始使用的nginx配置文件

* preview目录为系统后台截图
* publiccms-admin-by-gradle为Gradle管理的工程
* publiccms-admin-by-maven为Maven管理的工程
* LICENSE为PublicCMS授权协议
* README.md为markdown语言编写的简单介绍文档

# 开发工具集成

以下步骤Eclipse为例，下载地址 <https://www.eclipse.org/downloads/>，具体版本信息如下:

Eclipse Java EE IDE for Web Developers.

Version: Luna Service Release 2 (4.4.2)

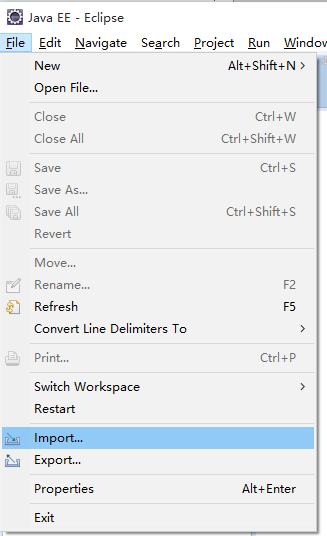
Build id: 20150219-0600

## 工程导入

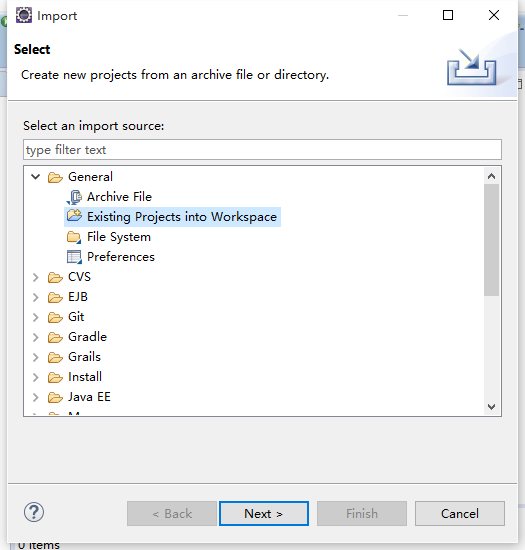
导入到开发工具中后，您可以自由的编译和修改PublicCMS源码，已经进行二次开发等。

### 操作步骤

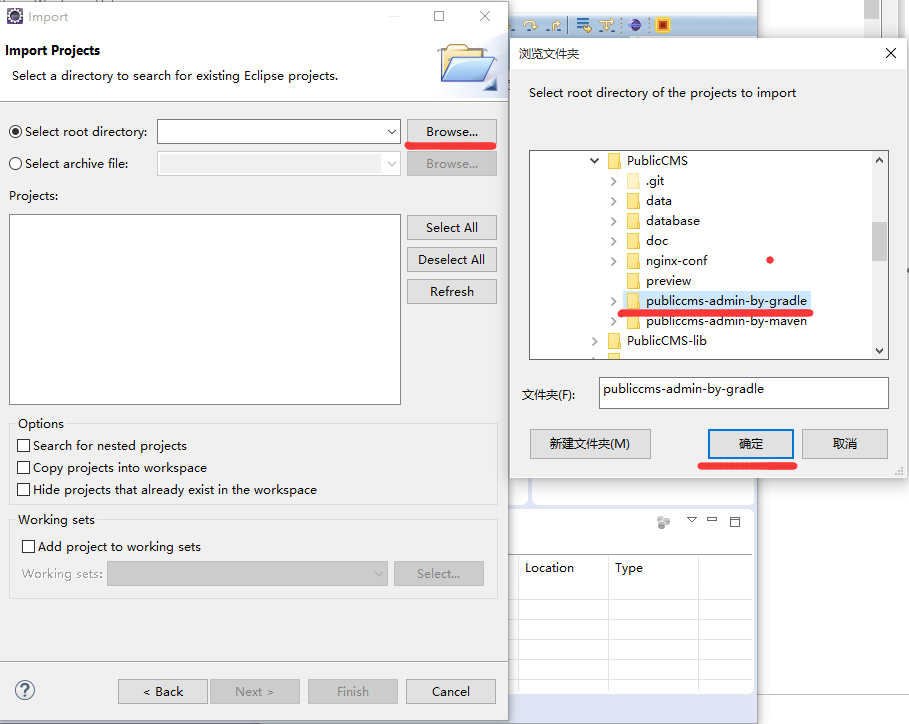
* 首先选择File菜单中的Import(导入)



* 在弹出的导入对话框中选择Existing Projects into Workspace(导入已有的工程)



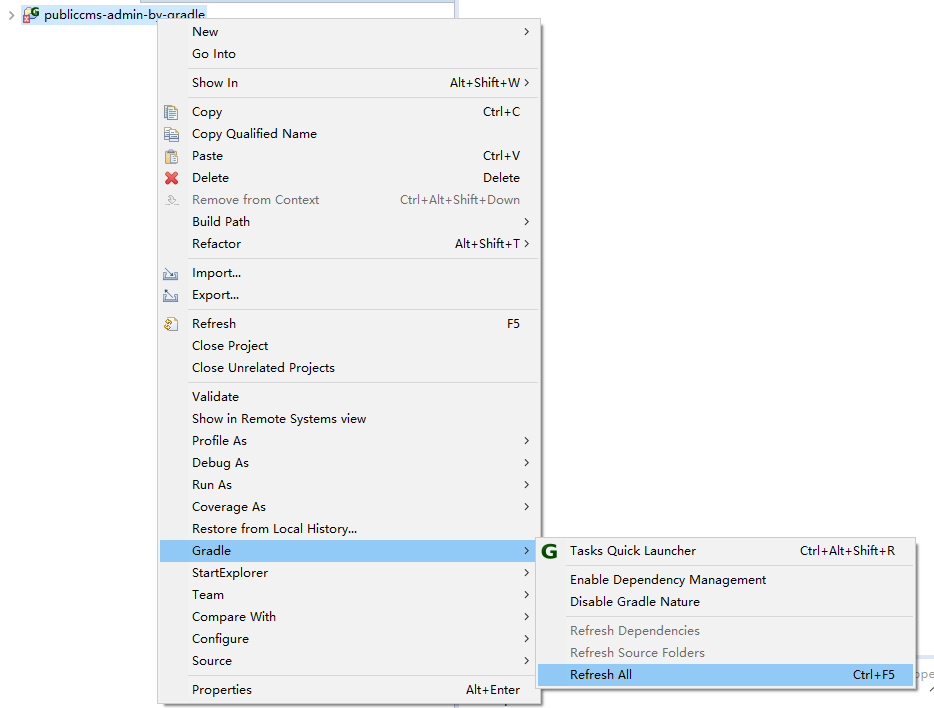
* 使用Select root directory(选择跟目录)的方式浏览选择PublicCMS目录或某个工程目录



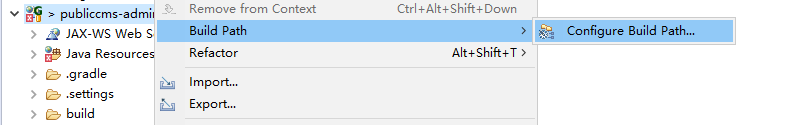
* 在根据需要选择gradle工程或者maven工程，建议勾选Copy projects into workspace(复制到工作空间)

### 注意事项

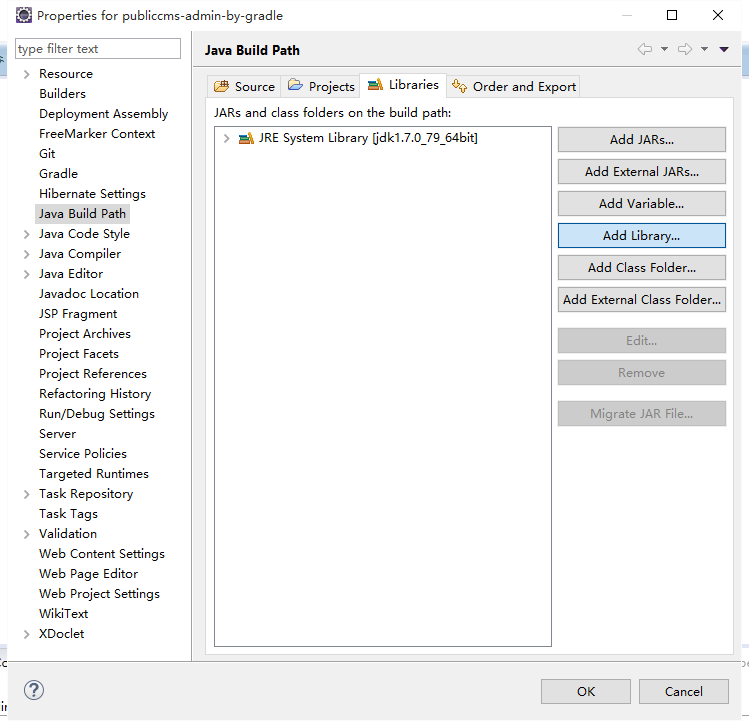
1. 如果是gradle工程，需要执行Gradle->Refresh All来解决一些导入后的报错信息



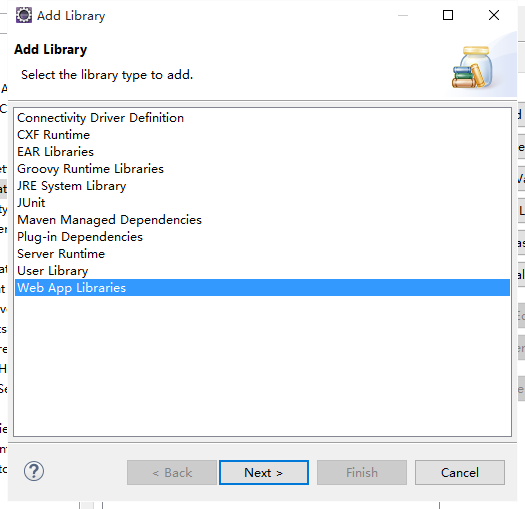
* 如果您没有使用gradle或者maven，请使用导入gradle工程到开发工具中，然后在工程WebContent/WEB-INF/目录下创建lib文件夹，并将PublicCMS-lib目录下所有jar包复制到lib文件夹下，并修改build path(编译路径)



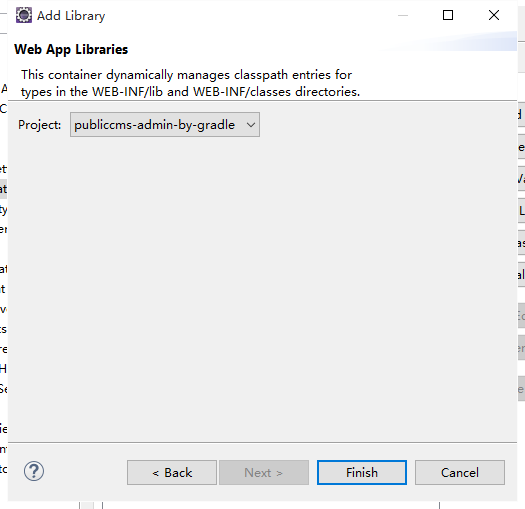
点击右侧Add Library按钮



选择Web App Libraries(Web应用类库)，然后点击next >(下一步)



选择publiccms-admin工程，点击Finish



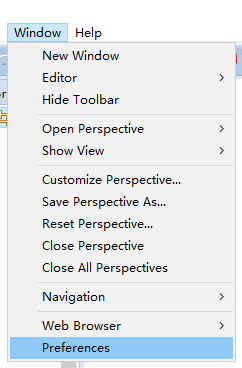
回到Build Path对话框后点击Ok保存配置

## 开发工具配置

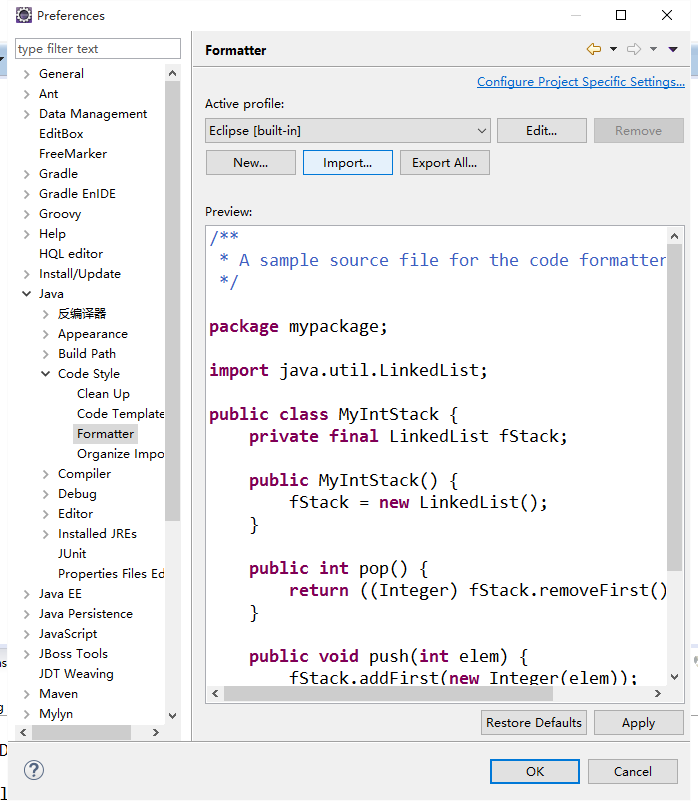
开发工具配置选项是为了您更好的对PublicCMS进行开发，并保持代码风格的一致性

### 导入格式化文件

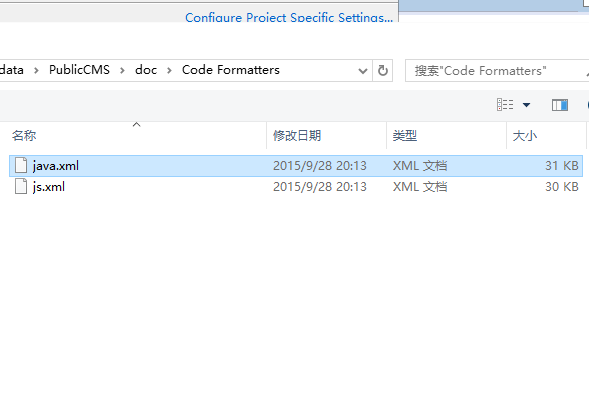
选择Window菜单下的Preferences菜单项



展开到Java->Code Style->Formatter选项卡点击Import按钮



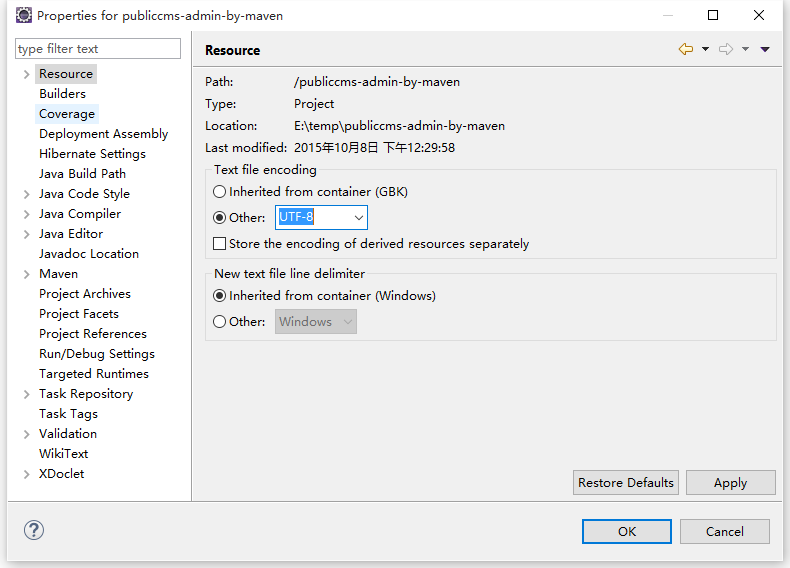
选择java.xml文件



使用相似的步骤导入js.xml到JavaScript->Code Style->Formatter中

### 字符编码设置

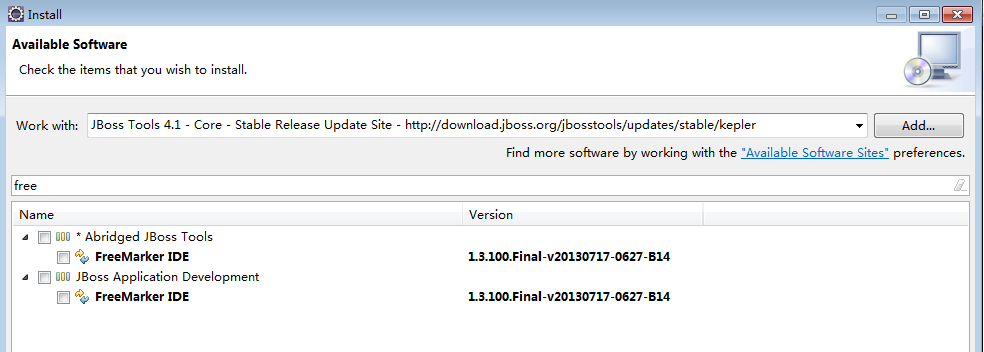
工程的编码都是UTF-8格式



### 开发工具插件

FreeMarker编辑器Eclipse插件被包含在jboss tools中

<http://download.jboss.org/jbosstools/updates/stable/> (需要选择IDE版本)



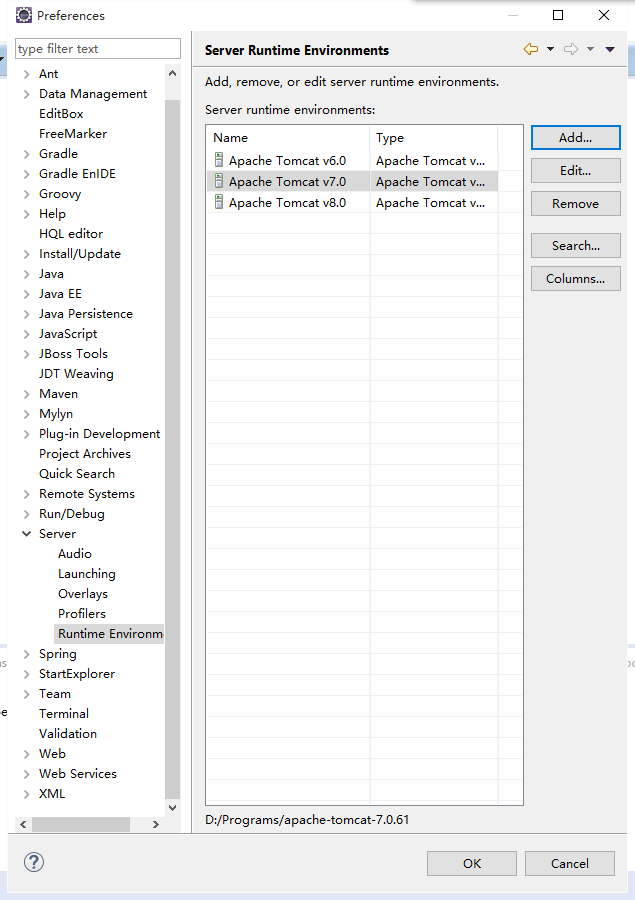
通过过滤选择FreeMarker IDE插件安装

需要注意的是该插件更新较慢，而且不是非常稳定，对于FreeMarker比较熟悉的人员来说，它并不能加快开发速度，因此只建议FreeMarker的初学者或对FreeMarker指令不熟悉的开发人员使用

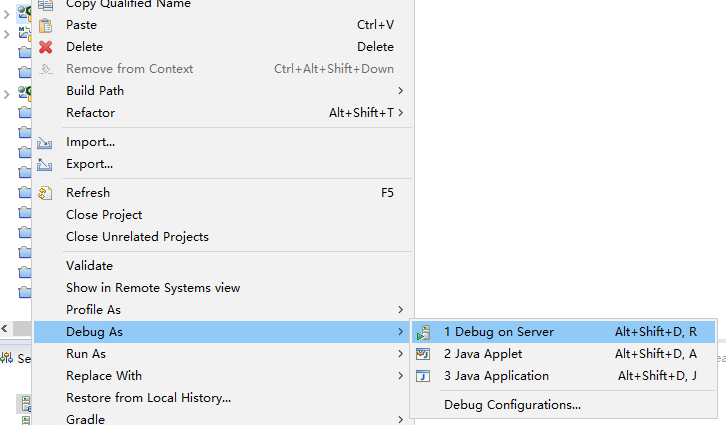
## 工程调试

在开发工具中调试时您可以选择编写测试代码进行单元测试，也可以直接在容器中运行工程，进行在线调试。模板的调试建议您选择在线进行。

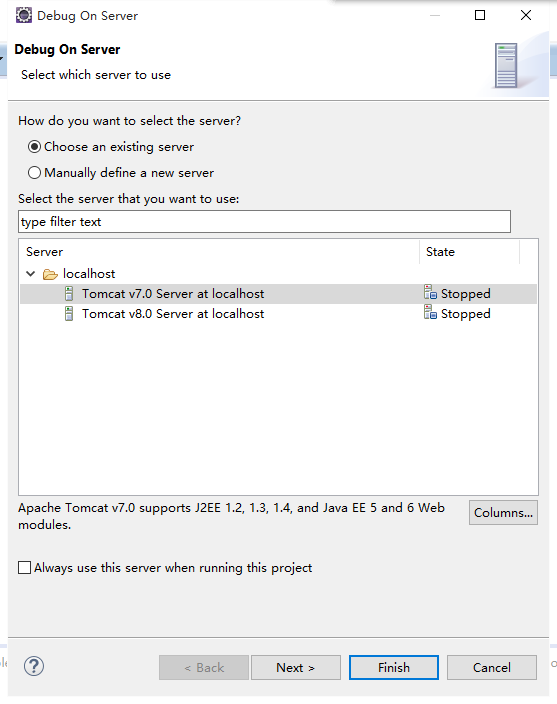
选择Window菜单下的Preferences菜单项, 展开到Server->Runtime Environments选项卡，新增一个Tomcat7.0以上版本的容器



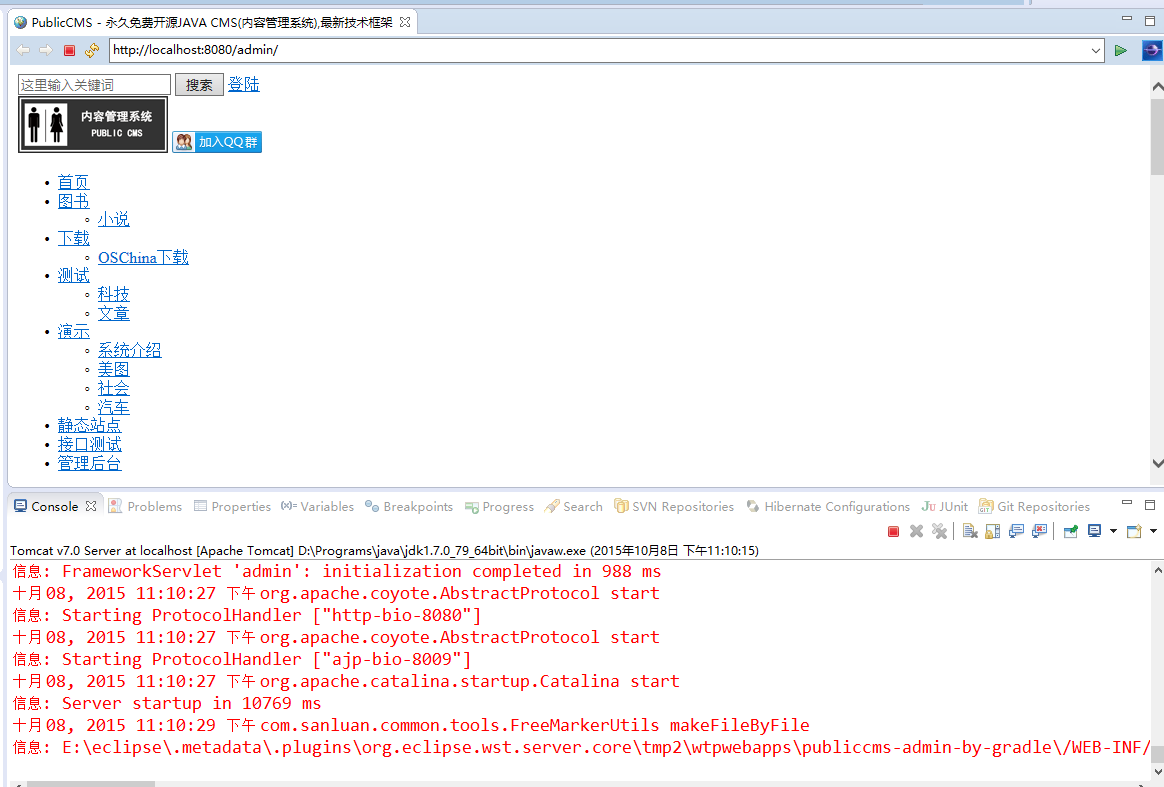
在工程上右键，选择Debug As菜单的Debug on Server子菜单项或者Run As菜单的Run on Server子菜单项



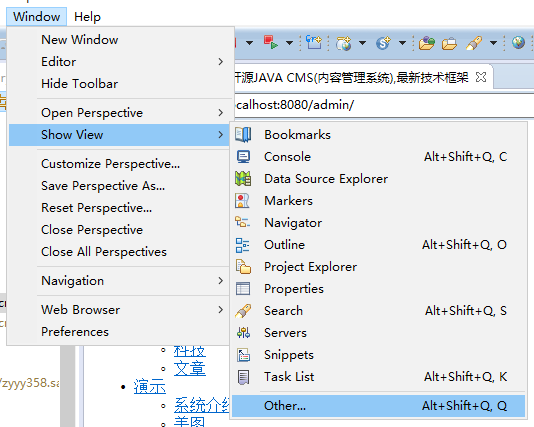
选择刚才添加的容器，然后点击Finish按钮



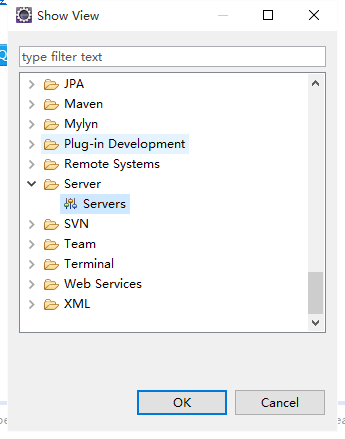
即可在工程中调试或运行工程



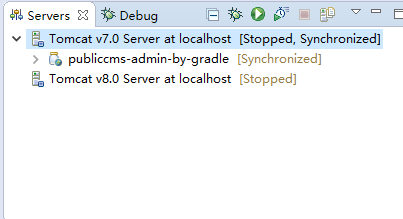
还可以在Window菜单中的Show View子菜单选择Other菜单项



打开Servers视图



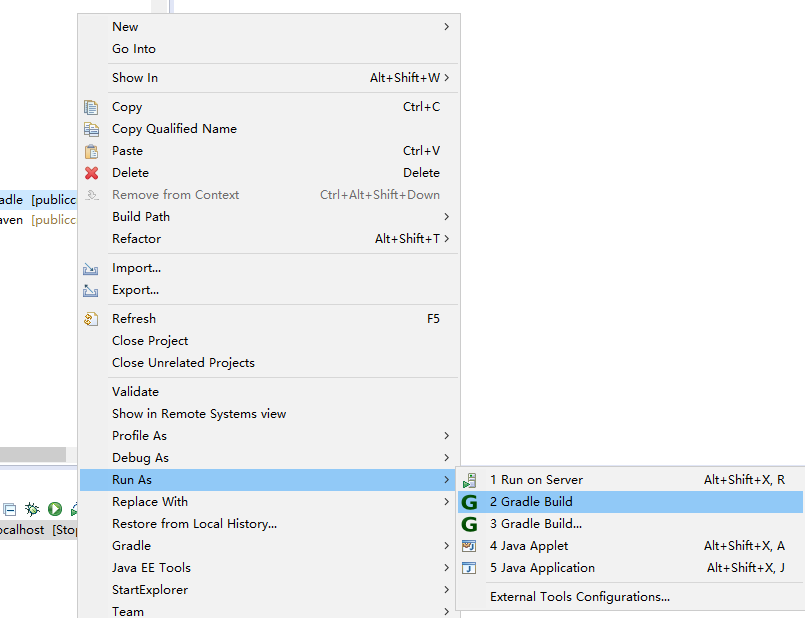
对所有容器和容器中运行或调试的工程进行管理



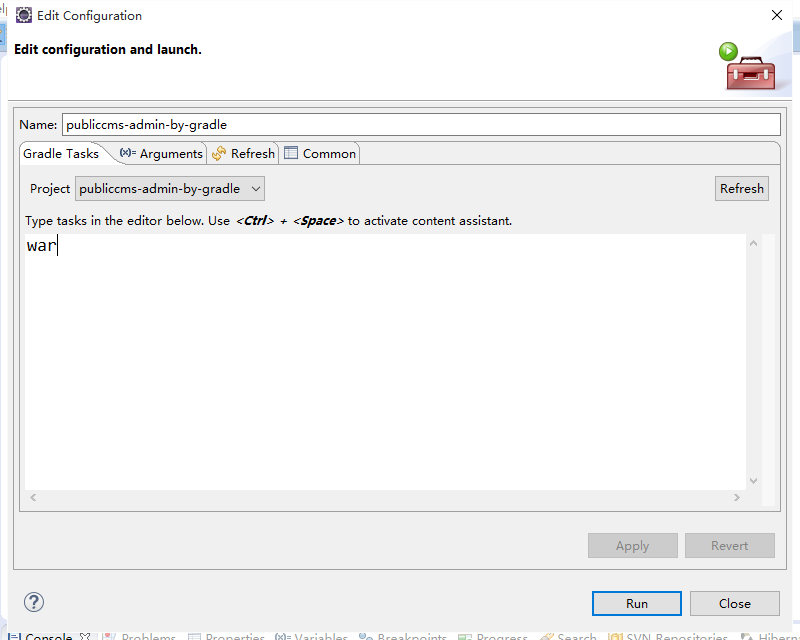
## 工程发布

### gradle工程发布

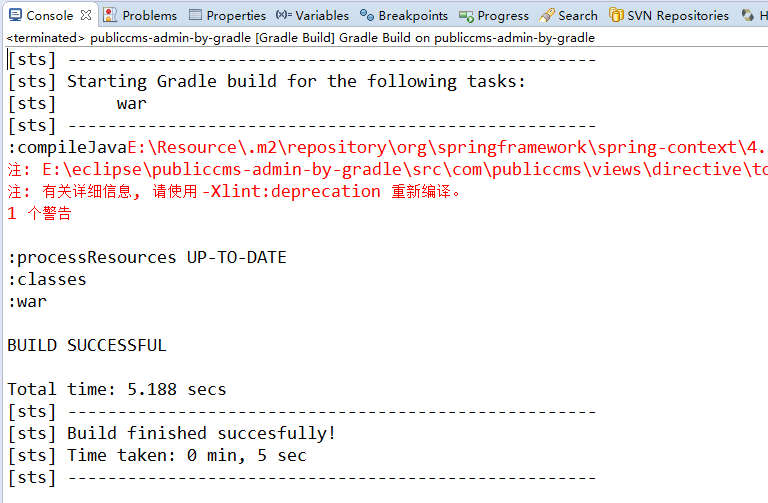
* 选择工程，右键弹出菜单中选择"Run As",在子菜单中选择"Gradle build"



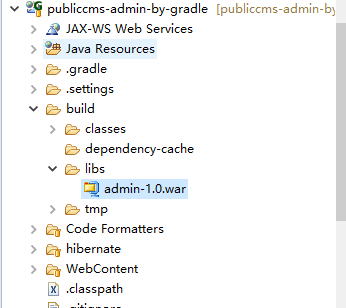
* 在弹出的对话框中输入"war",点击按钮"Run"



* 在Console中可以看到编译过程日志

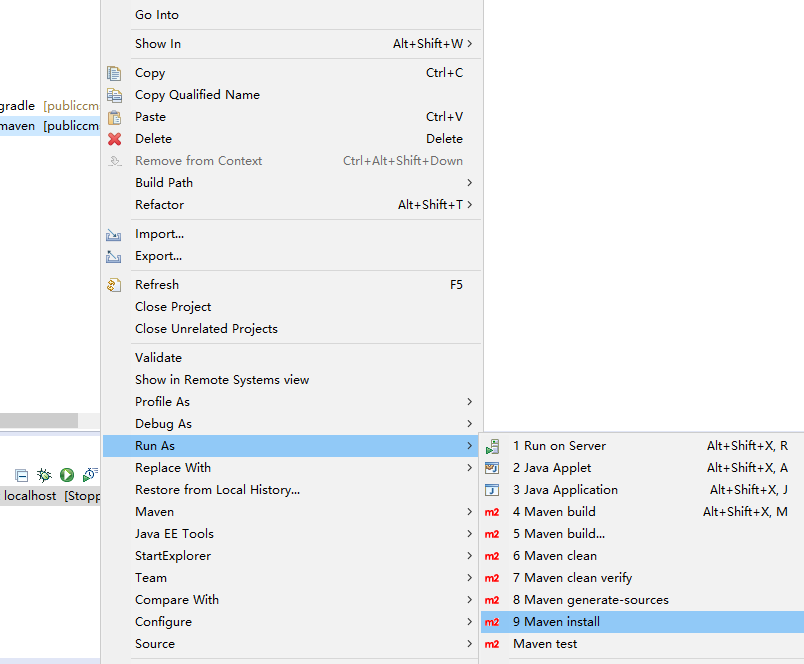


* 如果编译成功，您可以在工程下面 build/libs 目录下找到编译完成的 war包



### maven工程发布

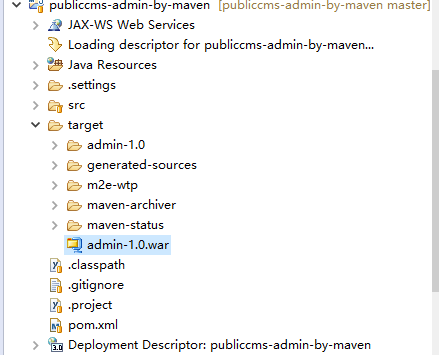
* 选择工程，右键弹出菜单中选择"Run As",在子菜单中选择"Maven install"



* 在Console中可以看到编译过程日志



* 如果编译成功，您可以在工程下面 target 目录下找到编译完成的 war包

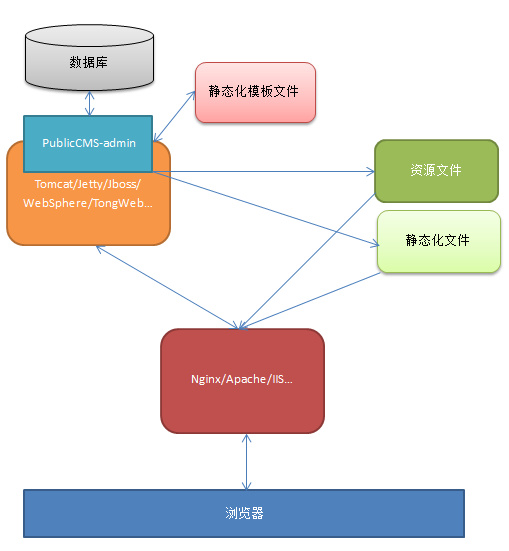


# 部署

## 部署方式

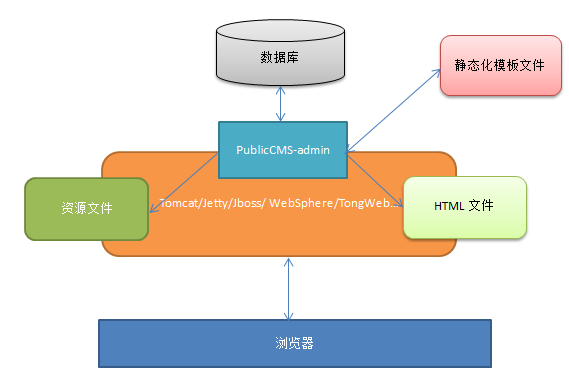
PublicCMS根据不同需要，有很多种不同的部署方式，其中部署方式1为[www.publiccms.com](http://www.publiccms.com)采用的部署方式，部署方式2中使用tomcat等动态web容器受理静态资源请求会造成大量服务器资源的浪费，也会对并发性能造成很大限制。

### 部署方式1



PublicCMS运行在Tomcat等支持Servlet3.0的容器中，静态化模板、资源文件、静态化文件独立于PublicCMS工程存在于服务器磁盘中。其中静态化模板由PublicCMS管理后台进行在线编辑及生成静态化文件时访问；资源文件、静态化文件通过HTTP服务程序发布为基础HTTP服务，对外提供页面访问，图片等资源文件访问。PublicCMS运行的容器通过HTTP服务程序反向代理对外提供HTTP服务,对外提供后台管理界面访问，搜索、标签、登陆等功能页面访问。

### 部署方式2



PublicCMS运行在Tomcat等支持Servlet3.0的容器中, 对外提供后台管理界面访问，搜索、标签、登陆等功能页面访问。静态化模板独立于PublicCMS工程存在于服务器磁盘中，由PublicCMS管理后台进行在线编辑及生成静态化文件时访问。资源文件、静态化文件作为独立的应用或者放在PublicCMS-admin/resource/目录下等方式，对外提供页面访问，图片等资源文件访问。

## 运行环境

PublicCMS运行部署需要

jdk1.7+ (<http://www.oracle.com/technetwork/java/javaee/downloads/index.html>)

tomcat7.0+ (<http://tomcat.apache.org/>)/jetty8.0+ (<http://www.eclipse.org/jetty/downloads.php>)

nginx(<http://nginx.org/en/download.html>)/apache (<http://httpd.apache.org/download.cgi>)/IIS

Mysql5.0+ (<http://dev.mysql.com/downloads/mysql>)

## 部署步骤

以下步骤以部署方式1为例

### 初始化数据

将database/Database Init.sql数据库初始化脚本和Database Change Log.sql变更脚本导入数据库

### 配置Nginx

安装Nginx后，可以参考nginx-conf下的配置文件对Nginx进行配置

* 如果是在Windows环境，可以参考nginx-conf/localhost-conf/目录下的配置文件
* 如果是在Linux环境，可以参考nginx-conf/publiccms.com.conf文件

需要配置：

* 一个tomcat的反向代理:需要支持SSI，并且/include/路径需要映射到为(data/www/publiccms.com/static/include/)静态化页面文件存放目录的include子目录
* 一个静态页面的站点或地址：本地路径为(data/www/publiccms.com/static)静态化页面文件存放目录，需要支持SSI
* 一个资源文件的站点或地址：本地路径为(data/www/publiccms.com/upload)静态资源和附件文件保存目录

### 部署PublicCMS程序

* 修改src/config/properties/other.properties文件

file.templateLoaderPath为(data/www/publiccms.com/template)静态化页面模板文件目录的绝对路径

file.dataFilePath为(data/www/publiccms.com/data)数据文件目录的绝对路径

file.staticFileDirectory为(data/www/publiccms.com/static)静态化页面文件存放目录

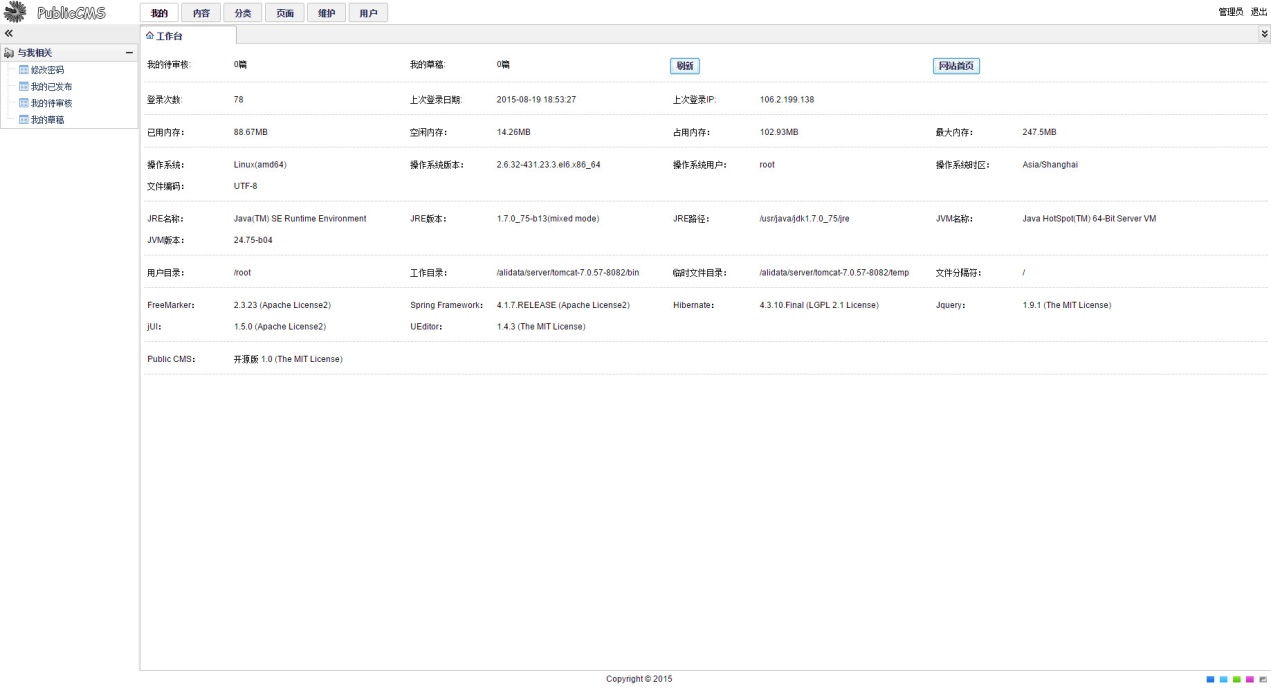
file.uploadFilePath为(data/www/publiccms.com/upload)静态资源和附件文件保存目录

* 将PublicCMS工程编译为war包，放置到容器中或者容器指定的位置，然后启动容器

### 访问PublicCMS

访问nginx配置的静态页面地址，此时可能为预置的PublicCMS欢迎信息提示，需要访问后台，在维护->任务计划->任务计划管理中执行重新生成所有页面的任务计划。此时访问，应该看到的是演示的静态化站点。

访问nginx配置的tomcat方向代理站点即可访问动态站点，页面有管理后台的链接，后台账号默认为admin密码admin，可以登录后台进行管理操作。



# 模板制作

## 命名规范

模板后缀约定为为html，宏定义文件后缀约定为为ftl

在动态站点模板中，模板文件名即为前台访问的文件名，请求参数可以在模板中直接使用${参数名}输出

## 数据模型与相关指令

### 分类

字段列表

name:名称

parentId:父分类

englishName:英文名

templatePath:模板路径

path:路径

url:地址

contentPath:内容路径

chapterPath:章节路径

pageSize:每页数据

disabled:是否删除

contents:内容数

extend1:扩展字段1

extend2:扩展字段2

extend3:扩展字段3

extend4:扩展字段4

自定义指令

分页列表查询

<@\_categoryList parentId=queryParentId extend1=queryExtend1

name=queryName extend3=queryExtend3 extend2=queryExtend2

extend4=queryExtend4 disabled=queryDisabled

pageIndex=pageNum count=numPerPage></@\_categoryList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsCategory>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询

<@\_category id=id ids=ids></@\_category>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 分类模型映射

* 字段列表

categoryId:分类

modelId:模型

templatePath:模板路径

* 自定义指令
* 分页列表查询

<@\_categoryModelList modelId=queryModelId categoryId=queryCategoryId

pageIndex=pageNum count=numPerPage></@\_categoryModelList>

结果

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsCategoryModel>对象 使用<#list t\_list as a></#list> 遍历数据

### 内容

* 字段列表

title:标题

userId:发布用户

deptId:部门

categoryId:分类

modelId:模型

parentId:父内容

isCopied:是否转载

source:来源

sourceUrl:来源地址

author:作者

editor:编辑

url:地址

description:描述

cover:封面

childs:章节数

scores:分数

comments:评论数

clicks:点击数

publishDate:发布日期

createDate:创建日期

status:状态

disabled:已删除

extend1:扩展字段1

extend2:扩展字段2

extend3:扩展字段3

extend4:扩展字段4

modelExtend1:扩展数字字段1

modelExtend2:扩展数字字段2

modelExtend3:扩展数字字段3

modelExtend4:扩展数字字段4

* 自定义指令
* 分页列表查询

<@\_contentList status=queryStatus categoryId=queryCategoryId containChild=queryContainChild

isDisable=queryDisabled modelId=queryModelId parentId=queryParentId title=queryTitle

userId=queryUserId extend1=queryExtend1 extend2=queryExtend2 extend3=queryExtend3 extend4=queryExtend4

modelExtend1=queryModelExtend1 modelExtend2=queryModelExtend2 modelExtend3=queryModelExtend3 modelExtend4=queryModelExtend4

startPublishDate=queryStartPublishDate endPublishDate=queryEndPublishDate

orderField=orderField orderType=orderType pageIndex=pageNum count=numPerPage></@\_contentList>

条件:

containChild:是否包含子栏目内容

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsContent>对象 使用<#list t\_list as a></#list> 遍历数据

* 单条记录查询

<@\_content id=id ids=ids></@\_content>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 扩展

* 字段列表

itemType:项目类型

itemId:项目ID

extendType:扩展类型

isCustom:自定义字段

isRequired:必填

name:名称

description:描述

code:字段

inputType:表单类型

defaultValue:默认值

* 自定义指令
* 分页列表查询

<@\_extendList isRequired=queryIsRequired extendType=queryExtendType

itemType=queryItemType itemId=queryItemId isCustom=queryIsCustom

pageIndex=pageNum count=numPerPage></@\_extendList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsExtend>对象 使用<#list t\_list as a></#list> 遍历数据

### 表单

* 字段列表

title:标题

data:数据

createDate:标题

disabled:禁用

menu项:

<li><a href="cmsForm/list.html?navTabId=cmsForm" target="navTab" rel="cmsForm">管理</a></li>

* 自定义指令
* 分页列表查询：

<@\_formList title=queryTitle startCreateDate=queryStartCreateDate endCreateDate=queryEndCreateDate

disabled=queryDisabled

orderField=orderField orderType=orderType pageIndex=pageNum count=numPerPage></@\_formList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsForm>对象 使用<#list t\_list as a></#list> 遍历数据

* 单条记录查询：

<@\_form id=id ids=ids></@\_form>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 模型

* 字段列表

parentId:父模型

name:名称

chapterName:章节模型名称

templatePath:模板路径

chapterTemplatePath:章节模板路径

hasChild:有章节

isUrl:是链接

isImages:是图集

disabled:已删除

* 自定义指令

分页列表查询：

<@\_modelList parentId=queryParentId hasChild=queryHasChild isImages=queryIsImages

disabled=queryDisabled isUrl=queryIsUrl

pageIndex=pageNum count=numPerPage></@\_modelList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsModel>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_model id=id ids=ids></@\_model>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 用户统计

* 字段列表

contentCount:文章数

commentCount:评论数

* 自定义指令

分页列表查询：

<@\_userList pageIndex=pageNum count=numPerPage></@\_userList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<CmsUser>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_user id=id ids=ids></@\_user>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 邮件验证日志

* 字段列表

userId:用户

email:邮件地址

code:验证码

createDate:创建日期

checked:验证通过

* 自定义指令

分页列表查询：

<@\_logEmailCheckList userId=queryUserId startCreateDate=queryStartCreateDate endCreateDate=queryEndCreateDate

checked=queryChecked

orderField=orderField orderType=orderType pageIndex=pageNum count=numPerPage></@\_logEmailCheckList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<LogEmailCheck>对象 使用<#list t\_list as a></#list> 遍历数据

### 登陆日志

* 字段列表

name:用户名

userId:用户

ip:IP

result:登录结果

createDate:登录日期

errorPassword:错误密码

* 自定义指令

分页列表查询：

<@\_logLoginList result=queryResult userId=queryUserId

name=queryName startCreateDate=queryStartCreateDate endCreateDate=queryEndCreateDate ip=queryIp

orderField=orderField orderType=orderType pageIndex=pageNum count=numPerPage></@\_logLoginList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<LogLogin>对象 使用<#list t\_list as a></#list> 遍历数据

### 操作日志

* 字段列表

userId:用户

operate:操作

ip:IP

createDate:操作日期

content:内容

* 自定义指令

分页列表查询：

<@\_logOperateList content=queryContent operate=queryOperate ip=queryIp

userId=queryUserId startCreateDate=queryStartCreateDate endCreateDate=queryEndCreateDate

orderField=orderField orderType=orderType pageIndex=pageNum count=numPerPage></@\_logOperateList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<LogOperate>对象 使用<#list t\_list as a></#list> 遍历数据

### 任务计划日志

* 字段列表

taskId:任务ID

begintime:开始时间

endtime:结束时间

result:执行结果

* 自定义指令

分页列表查询：

<@d\_logTaskList status=queryStatus

pageIndex=pageNum count=numPerPage></@d\_logTaskList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<LogTask>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@d\_logTask id=id ids=ids></@d\_logTask>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 部门

* 字段列表

name:名称

parentId:父部门

description:描述

userId:负责人

* 自定义指令

分页列表查询：

<@\_systemDeptList parentId=queryParentId userId=queryUserId name=queryName

pageIndex=pageNum count=numPerPage></@\_systemDeptList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemDept>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_systemDept id=id ids=ids></@\_systemDept>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 模块

* 字段列表

name:名称

url:地址

parentId:父模块

* 自定义指令

分页列表查询：

<@\_systemMoudleList parentId=queryParentId url=queryUrl

pageIndex=pageNum count=numPerPage></@\_systemMoudleList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemMoudle>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_systemMoudle id=id ids=ids></@\_systemMoudle>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 角色

* 字段列表

name:名称

deptId:部门

ownsAllRight:拥有全部权限

* 自定义指令

分页列表查询：

<@\_systemRoleList pageIndex=pageNum count=numPerPage></@\_systemRoleList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemRole>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_systemRole id=id ids=ids></@\_systemRole>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 角色模块映射

* 字段列表

roleId:角色

moudleId:模块

* 自定义指令

分页列表查询：

<@\_systemRoleMoudleList roleId=queryRoleId moudleId=queryMoudleId

pageIndex=pageNum count=numPerPage></@\_systemRoleMoudleList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemRoleMoudle>对象 使用<#list t\_list as a></#list> 遍历数据

### 角色用户映射

* 字段列表

menu项:

<li><a href="systemRoleUser/list.html?navTabId=systemRoleUser" target="navTab" rel="systemRoleUser">管理</a></li>

* 自定义指令

分页列表查询：

<@\_systemRoleUserList

pageIndex=pageNum count=numPerPage></@\_systemRoleUserList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemRoleUser>对象 使用<#list t\_list as a></#list> 遍历数据

### 任务计划

* 字段列表

name:任务名称

status:状态

cronExpression:表达式

description:描述

content:任务内容

* 自定义指令

分页列表查询：

<@\_systemTaskList status=queryStatus

pageIndex=pageNum count=numPerPage></@\_systemTaskList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemTask>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_systemTask id=id ids=ids></@\_systemTask>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

### 系统用户

* 字段列表

name:用户名

password:密码

nickName:用户昵称

email:邮箱

emailChecked:已验证邮箱

superuserAccess:是否管理员

disabled:已禁用

lastLoginDate:上次登录日期

lastLoginIp:上次登录IP

loginCount:登录次数

dateRegistered:注册日期

* 自定义指令

分页列表查询：

<@\_systemUserList startDateRegistered=queryStartDateRegistered endDateRegistered=queryEndDateRegistered startLastLoginDate=queryStartLastLoginDate endLastLoginDate=queryEndLastLoginDate

superuserAccess=querySuperuserAccess emailChecked=queryEmailChecked name=queryName

disabled=queryDisabled

orderField=orderField orderType=orderType pageIndex=pageNum count=numPerPage></@\_systemUserList>

结果：

page:分页信息，totalCount：总条数，totalPage：总页数，pageSize每页数据条数，pageIndex当前页数

page.list: List<SystemUser>对象 使用<#list t\_list as a></#list> 遍历数据

单条记录查询：

<@\_systemUser id=id ids=ids></@\_systemUser>

结果：

id不为空时：object:属性参考字段列表

id为空且ids不为空时：t\_map:{key:id,value:bean}其中bean属性参考字段列表

# 二次开发

## 工程代码简介

### 架构

1. 软件基础架构

软件使用了FreeMarker 2.3.23, SpringMVC 4.2.0.RELEASE, Hibernate 4.3.11.Final, hibernate-search5.3.0.Final, Mysql jdbc5.1.35, c3p0 0.9.1.2, dom4j 1.6.1, log4j+bogback, Apache commons及其依赖的包,包依赖关系,工程编译等工作由Gradle处理

### 命名规范

config.language 语言文件

config.properties 配置文件

config 项目配置项

.common 工具类、解决方案类等同用类包

.base 基础类包

.constants 常量定义类包

.dialect 数据库方言类包

.handler 分页处理，查询处理，FreeMarker异常等处理器包

.servlet Servlet包

.source 源码工具包

.tools 工具类包

.entities 实体类包

.logic 逻辑类包

.component 组件包

.dao 数据库操作类包

.service 事务与逻辑封装类包

.tasks 任务计划包

.views 视图类包

.controller 控制器类包

Index.java 同用分发Controller类

.directive freemaker自定义指令类包

.method freemaker自定义方法类包

### 模式

Controller主要用于表单处理等逻辑，FreeMaker用于展现部分，以html为基础，通过使用FreeMaker模板标签等实现大部分功能类需求。

展现中的字段关联行为尽量使用模板自定义指令嵌套查询，将大部分的关联查询分解为简单查询。必要时使用FreeMaker生成页面功能将部分页面静态化以混存耗时较大的数据

### 开发流程

1. 在开发工具中使用hibernate工具生成实体类，修改数据库配置文件
2. 对实体类属性使用com.sanluan.common.source.entity.MyColumn注解类标记字段类型【condition：是否用于查询条件，order：是否用于排序，like：查询条件是否为like匹配，or：查询条件是否为or逻辑，name：字段名，当or为true时，相同name 的字段为同一组or条件，title：字段真实含义】
3. 使用com.sanluan.common.source.Maker工具类 生成自定义指令，Controller，Service，Dao，html
4. 根据实际业务调整代码中的逻辑

## 获取支持

在线调试FreeMarker <http://www.sanluan.com/freemarker_test.html>