**jfinal cms 文档**

目录

[1 jfinal cms简介 3](#_Toc479201656)

[2 jfinal cms部署与配置 4](#_Toc479201657)

[2.1 maven方式部署配置 4](#_Toc479201658)

[2.2 eclipse部署配置 4](#_Toc479201659)

[2.2.1 获取源码地址 4](#_Toc479201660)

[2.2.2 下载git源码 5](#_Toc479201661)

[2.2.3 构建maven项目 7](#_Toc479201662)

[2.2.4 运行项目代码 8](#_Toc479201663)

[2.2.5 项目访问 10](#_Toc479201664)

[2.3 实时编译源码配置 10](#_Toc479201665)

[2.4 其他配置说明 12](#_Toc479201666)

[3 相关技术说明 13](#_Toc479201667)

[3.1 java框架jfinal 13](#_Toc479201668)

[3.2 java模板beetl 13](#_Toc479201669)

[4 jfinal cms技术 15](#_Toc479201670)

[4.1 项目目录结构介绍 15](#_Toc479201671)

[4.2 后台模块开发 16](#_Toc479201672)

[4.2.1 java代码 16](#_Toc479201673)

[4.2.2 页面代码 20](#_Toc479201674)

[4.3 前台模板开发 26](#_Toc479201675)

[4.3.1 java代码 26](#_Toc479201676)

[4.3.2 模板代码 27](#_Toc479201677)

[4.4 前台模板接口 30](#_Toc479201678)

[4.4.1 前台默认入口 30](#_Toc479201679)

[4.4.2 前台接口 31](#_Toc479201680)

[4.5 项目工具类 36](#_Toc479201681)

[5 jfinal cms项目使用说明 38](#_Toc479201682)

[5.1 数据库配置 38](#_Toc479201683)

[5.2 oauth认证配置 38](#_Toc479201684)

[5.3 站点设置 38](#_Toc479201685)

[5.3.1 单站点 39](#_Toc479201686)

[5.3.2 多站点 39](#_Toc479201687)

[5.4 文件备份设置 40](#_Toc479201688)

[5.4.1 不备份（empty） 41](#_Toc479201689)

[5.4.2 文件系统备份（filemanger） 41](#_Toc479201690)

[5.4.3 阿里云OSS备份（alioss） 41](#_Toc479201691)

[5.4.4 自定义模式 41](#_Toc479201692)

[6 jfinal cms项目依赖 42](#_Toc479201693)

[7 联系作者 42](#_Toc479201694)

[8 开源赞助 43](#_Toc479201695)

# jfinal cms简介

jfinal cms，采用了简洁强大的[JFinal](http://www.oschina.net/p/jfinal)作为web框架，模板引擎用使用的是非常方便的[beetl](http://www.oschina.net/p/beetl)，数据库采用mysql，前端[bootstrap](http://www.oschina.net/p/bootstrap)框架。 支持多站点、oauth2认证、帐号注册、密码加密、评论及回复，消息提示，网站访问量统计，文章评论数和浏览量统计，回复管理，支持权限管理。

后台模块包含：栏目管理，栏目公告，栏目滚动图片，文章管理，回复管理，意见反馈，我的相册，相册管理，图片管理，专辑管理、视频管理、缓存更新，友情链接，访问统计，联系人管理，模板管理，组织机构管理，用户管理，角色管理，菜单管理，数据字典管理，参数配置，站点管理。

建站截图：





# jfinal cms部署与配置

## 项目相关地址

### 项目源码

github地址：<https://github.com/jflyfox/jfinal_cms>

oschina码云地址： <https://gitee.com/jflyfox/jfinal_cms>

### API Clinet 项目源码

github地址：<https://github.com/jflyfox/jfinal_cms_api_client>

oschina码云地址：<https://gitee.com/jflyfox/jfinal_cms_api_client>

## maven方式部署配置

maven部署方式的好处在于不依赖于任何IDE，并且跨平台。

1. 下载项目源代码，安装jdk、maven、mysql。
2. 在项目目录下运行mvn install，提示BUILD SUCCESS即可。
3. 创建mysql用户和数据库，运行/jfinal\_cms/sql下对应jfinal\_cms\_v4.sql。
4. 数据库配置文件：/jfinal\_cms/src/main/resources/conf/db.properties，修改相应配置即可。

db\_type=mysql

mysql.jdbcUrl =jdbc:mysql://127.0.0.1:3306/jfinal\_cms2?characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull

mysql.user = root

mysql.password = 123456

mysql.driverClass = com.mysql.jdbc.Driver

1. 运行：mvn tomcat:run。看到“系统启动完成”说明运行成功。
2. 网站效果访问：http://127.0.0.1/jfinal\_cms。
3. 管理地址：http://${ip:port}/${project\_name}/admin，管理账号: admin/admin123
4. 系统默认采用单站点模式，各个站点可以在“其他管理”下“站点管理”菜单方便的切换。
5. 如果使用多站点，可以在“系统管理”下“多站点标示”菜单中，将“多站点标示”项目修改为true。
6. 多站点需要设置各个站点对应的域名，通过域名解析到不同的站点模板。

## eclipse部署配置

因为java开发，用eclipse的比较多~这里再介绍一下eclipse导入，部署和配置。

### 获取源码地址

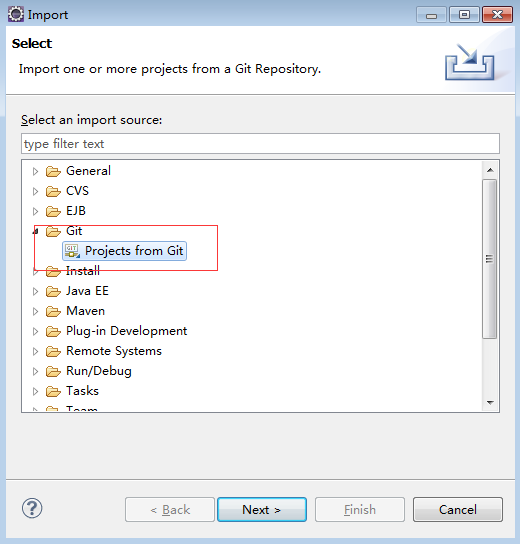
源码：https://gitee.com/jflyfox/jfinal\_cms复制git地址。



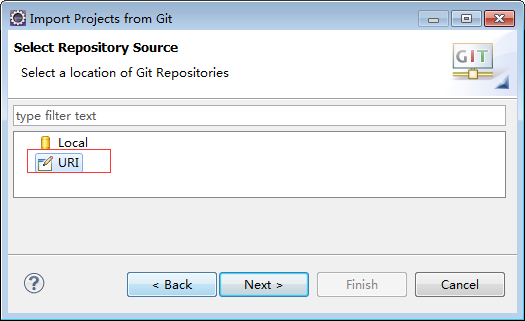
### 下载git源码

然后我们打开eclipse，导入git项目。（这里也可以用git工具先将源码下载下来，或者直接下载zip包）

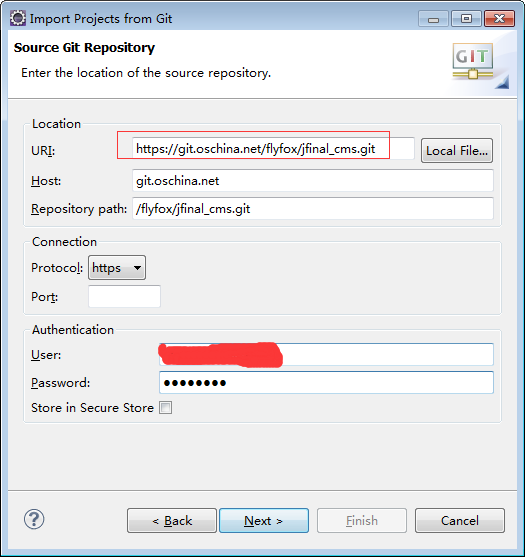
选择git导入：



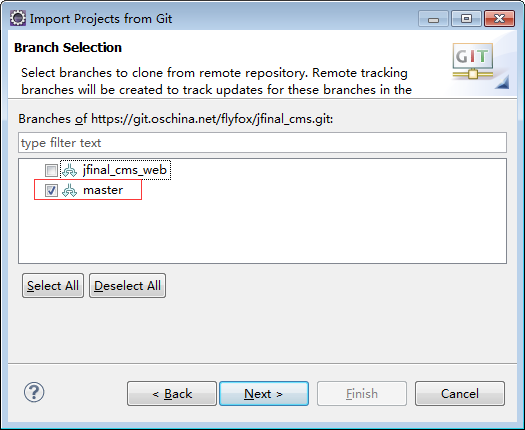
点击URI导入方式：



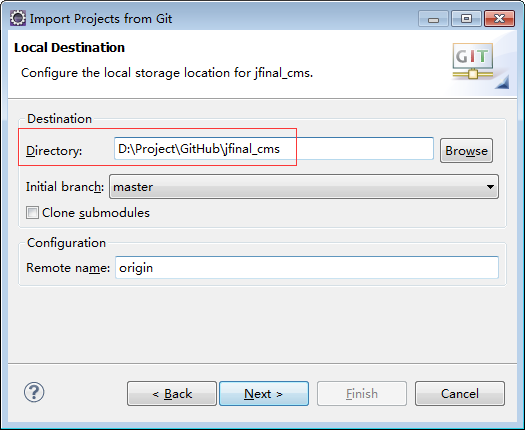
输入复制的git地址：



选择下载master主分支：



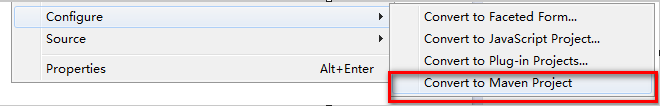
选择导入到本地的目录：



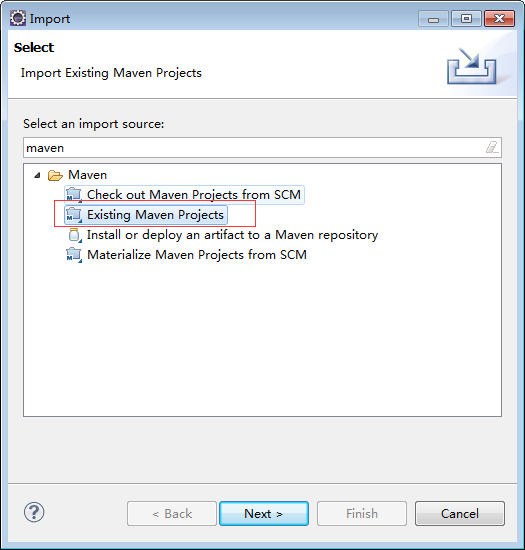
点击下一步获取git源代码。

### 构建maven项目

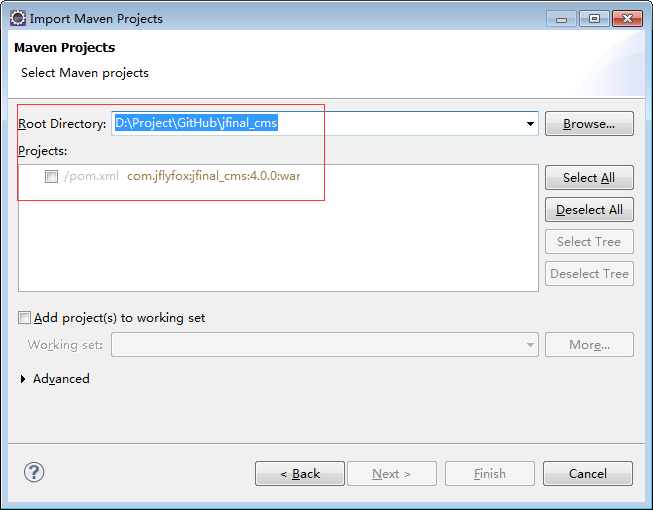
如果git获取后，我们选择了作为一个普通项目导入，那么我们需要将其转换为maven项目，项目上面邮件选择Configure->Convert to Maven Project：



如果我们通过其他git工具或者下载zip，那么我们需要导入maven项目



选择项目路径即可：

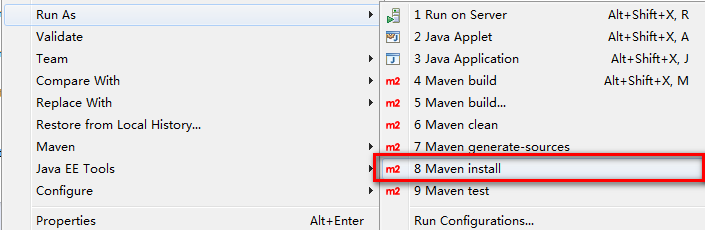


这样我们项目就导入完成了。

### 运行项目代码

到这里我们需要安装好mysql，运行完初始化脚本，并且配置好数据库连接串以及账号密码。这些上面maven方式部署有介绍，就不再重复了。

现在我们就可以进行maven项目构建了，运行maven install，安装，并下载依赖包：



如下提示说明构建成功

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

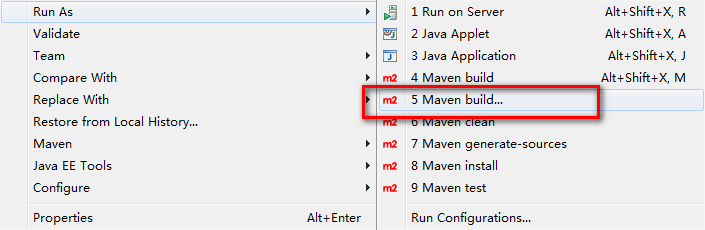
[INFO] Total time: 22.565s

[INFO] Finished at: Sat Jun 11 07:41:04 GMT 2016

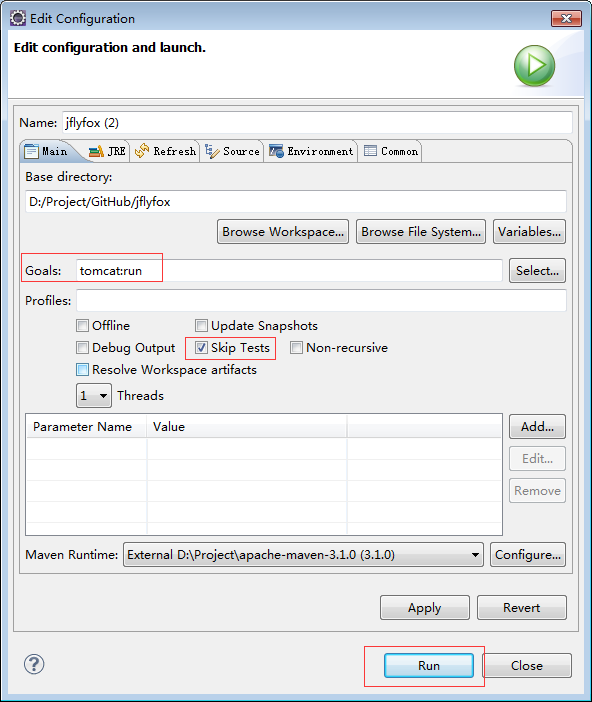
[INFO] Final Memory: 6M/12M

[INFO] ------------------------------------------------------------------------

然后我们运行项目，项目右键，Run As-> Maven build...



然后运行tomcat:run，点击运行按钮：



看到如下提示，并且没有报错，说明启动成功。

##################################

############系统启动完成##########

##################################

### 项目访问

网站效果访问：http://127.0.0.1/jfinal\_cms/

管理地址：http://127.0.0.1/jfinal\_cms/admin，管理账号: admin/admin123

到这里jfinal cms部署就大功告成了~！~

## 实时编译源码配置

上面部署方式有一个小问题就是，每次修改都需要进行maven install后，再maven tomcat:run才能看到修改后的效果。可以进行如下配置解决这个问题。

修改pom，删除warSourceDirectory配置。

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<version>1.1</version>

<configuration>

<port>80</port>

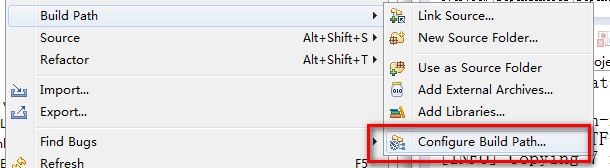
<path>/${project.artifactId}</path>

~~<warSourceDirectory>${basedir}/target/${artifactId}</warSourceDirectory>~~

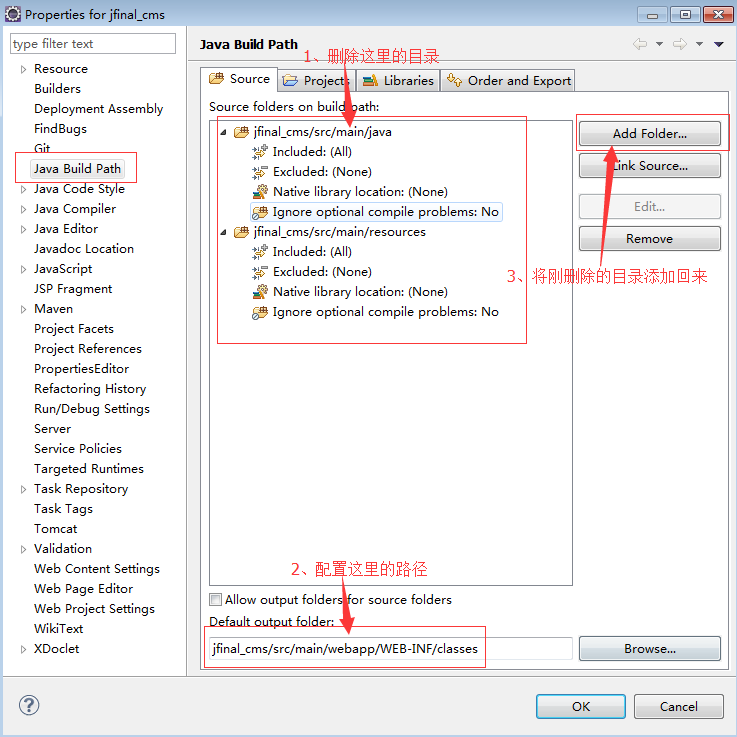
</configuration>

</plugin>

项目右键，选择构建路径（Build Path->Configure Build Path...）：



删除源码路径，重新配置，再修改默认输出路径（Default output folder）：



确定后会有一个提示，点击确认；这是再运行就可以进行实时编辑查看了。

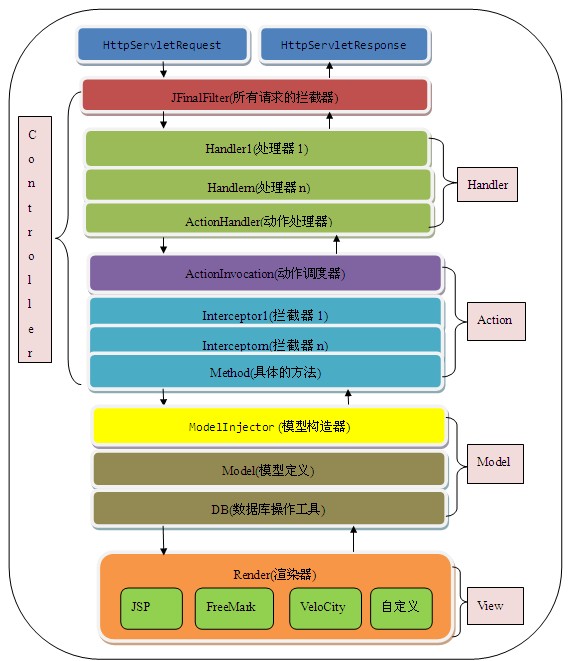
**注意：这里的配置一定要把运行的tomcat先进行停止；如果没有看到效果，说明编译失败，再重新配置一下。**

# 相关技术说明

## java框架jfinal

jfinal官网：<http://www.jfinal.com/>

下面我在网上截了一张jfinal架构图，个人觉得梳理的很清楚。jfina官网上面的文档在使用方面介绍的简单明了，上手很容易，个人感觉没有必要在进行其他介绍了。

[](http://cms.csdnimg.cn/article/201312/06/52a18247e6897.jpg)

## java模板beetl

beetl文档：<http://ibeetl.com/guide/beetl.html>

beetl的有点和好处可以看上面文档第一章节，作者介绍的已经很清楚了。详细介绍大家可以看上面的文档，我下面制作一些简单介绍。

* 变量引用

变量引用与其他模板相同，可以通过${name}调用。

这里要特殊说明的是beetl支持获取map属性,即调用get(Object key)。如果对象既有具体属性getName()，又有get(Object key)，则以具体属性优先级高。如${user.name}先进行调用getName()方法如果没有再调用get(Object key)方法。

* list循环

代码片段如下：

总共 ${list.~size}

<%

**for**(user in list){

%>

hello,${user.name};

<%}%>

当前页${pageMap['page']},总共${pageMap["total"]}

* 函数调用

可以直接调用registerFunctionPackage(namespace,yourJavaObject),这时候yourJavaObject里的所有public方法都将注册为Beetl方法，方法名是namespace+"."+方法名

jfinal cms中有许多方法和前台接口都是用了这种方式，所以这里也别说明一下。后续会进行详细介绍。

* 其他

剩下的大家就参考项目代码进行理解就可以了~如果不明白可以看beetl官方文档。

# jfinal cms技术

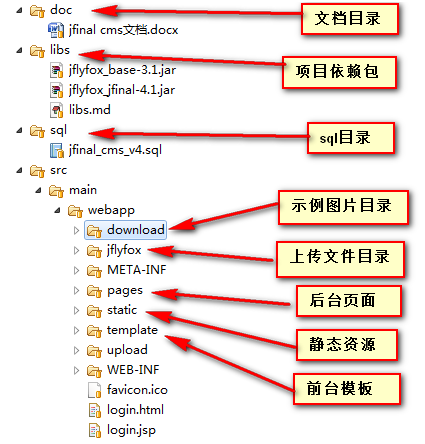
源码依赖于作者的其他两个开源项目jflyfox\_base和jflyfox\_jfinal两个项目，相关介绍在jfinal cms项目依赖章节有所介绍。

## 项目目录结构介绍

首先我们先来熟悉一下项目的目录结构，以下为该项目目录结构说明，java代码：



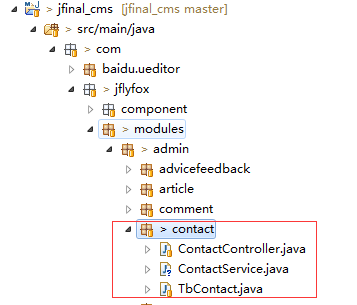
项目文件目录：



## 后台模块开发

### java代码

首先后台代码主要分为Controller，Model和Service。Controller负责参数传递和页面跳转；Model负责数据承载；Service负责复杂业务处理。



这里我们用联系人模块进行简单讲解。

下面是controller示例代码，首先需要继承BaseProjectController类，这样方便项目统一扩展，现在里面已经有一些公共方法了，请自行查看源码。

页面跳转主要包含list（列表页）、add（添加页）、view（查看页）、delete（删除数据）、edit（编辑页）和save（保存方法）。这几个是基本操作，常规增删改查均以此规范进行编写。Controller遵循jfinal设置规则，并加入了注解功能。以下列表访问地址就是/admin/contact/list；编辑页面地址就是/admin/contact/edit/${id}。其他地址不再列举。

**以下黄色背景字体需要特别注意。**

**package** com.jflyfox.modules.admin.contact;

**import** com.jfinal.plugin.activerecord.Page;

**import** com.jflyfox.component.base.BaseProjectController;

**import** com.jflyfox.jfinal.component.annotation.ControllerBind;

**import** com.jflyfox.jfinal.component.db.SQLUtils;

**import** com.jflyfox.util.StrUtils;

/\*\*

\* 联系人管理

\*

\* **@author** flyfox 2014-2-11

\*/

@ControllerBind(controllerKey = "/admin/contact")

**public** **class** ContactController **extends** BaseProjectController {

**private** **static** **final** String *path* = "/pages/admin/contact/contact\_";

**public** **void** list() {

TbContact model = getModelByAttr(TbContact.**class**);

**// 这里是拼接查询sql并且获取前台传递的参数**

SQLUtils sql = **new** SQLUtils(" from tb\_contact t where 1=1 ");

**if** (model.getAttrValues().length != 0) {

sql.setAlias("t");

sql.whereLike("name", model.getStr("name"));

sql.whereEquals("type", model.getStr("type"));

}

**// 这里是统一做了排序功能**

String orderBy = getBaseForm().getOrderBy();

**if** (StrUtils.*isEmpty*(orderBy)) {

sql.append(" order by id desc ");

} **else** {

sql.append(" order by ").append(orderBy);

}

Page<TbContact> page = TbContact.*dao*.paginate(getPaginator(), "select t.\* ", //

sql.toString().toString());

// 下拉框

setAttr("page", page);

setAttr("attr", model);

render(*path* + "list.html");

}

**public** **void** add() {

render(*path* + "add.html");

}

**public** **void** view() {

TbContact model = TbContact.*dao*.findById(getParaToInt());

setAttr("model", model);

render(*path* + "view.html");

}

**public** **void** delete() {

TbContact model = **new** TbContact();

Integer userid= getSessionUser().getUserID();

String now = getNow();

model.put("update\_id", userid);

model.put("update\_time", now);

model.deleteById(getParaToInt());

list();

}

**public** **void** edit() {

TbContact model = TbContact.*dao*.findById(getParaToInt());

setAttr("model", model);

render(*path* + "edit.html");

}

**public** **void** save() {

Integer pid = getParaToInt();

TbContact model = getModel(TbContact.**class**);

Integer userid= getSessionUser().getUserID();

String now = getNow();

**// 这两个字段设置为了日志收集**

model.put("update\_id", userid);

model.put("update\_time", now);

**if** (pid != **null** && pid > 0) { // 更新

model.update();

} **else** { // 新增

model.remove("id");

model.put("create\_id", userid);

model.put("create\_time", now);

model.save();

}

renderMessage("保存成功");

}

}

下面Model示例代码，Model需要继承BaseProjectModel，方便业务扩展。里面已经加入了缓存、日志等扩展功能，请自行查看源码。jfinal Model可以不添加get、set方法，这里添加主要是为了代码规范。Model遵循jfinal设置规则，并加入了注解功能。通过ModelBind可以直接设置对应的表和主键，主键默认是ID。

get、set方法可以通过作者的AutoCreate组件自动生成。这里不再做多介绍。

**package** com.jflyfox.modules.admin.contact;

**import** com.jflyfox.component.base.BaseProjectModel;

**import** com.jflyfox.jfinal.component.annotation.ModelBind;

@ModelBind(table = "tb\_contact")

**public** **class** TbContact **extends** BaseProjectModel<TbContact> {

**private** **static** **final** **long** *serialVersionUID* = 1L;

**public** **static** **final** TbContact *dao* = **new** TbContact();

// columns START

**private** String ID = "id"; // 主键

**private** String NAME = "name"; // 姓名

**public** TbContact setId(java.lang.Integer value) {

set(ID, value);

**return** **this**;

}

**public** java.lang.Integer getId() {

**return** get(ID);

}

**public** TbContact setName(java.lang.String value) {

set(NAME, value);

**return** **this**;

}

**public** java.lang.String getName() {

**return** get(NAME);

}

**……………………………………**

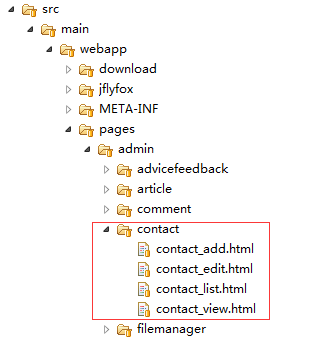
}

Service集成BaseService，主要是Controller中复杂业务处理，将Controller中大量代码或者重复代码进行封装，统一调用。由于联系人模块相对简单，这里就没有Service代码了。

### 页面代码

后台模板主要是增删改查，jfinal cms使用beetl模板，这里就不多介绍了。

页面主要以${module}\_名开头，包含add（添加）、edit（编辑）、list（列表）、view（查看）几个页面。各个页面页格式相对统一，可以进行参考。



以下为add页面代码，添加页面可以与edit页面公用~只是传递参数均为空。

<%

include("contact\_edit.html"){}

%>

以下为edit页面代码，首先edit页面采用了beetl安全模式，保证参数为空不报异常，这样能够让add很好的引用。其次后端页面都采用了公共的头部标签head.html，列表和查看页面也是如此，不在赘述。

在编辑页面对js验证进行了封装，只需要添加valid和validname属性就可以实现js验证功能，提交时调用 validForm()方法即可。

<%

DIRECTIVE SAFE\_OUTPUT\_OPEN;

var headContent = {

include("/pages/template/head.html"){}

%>

<script type=*"text/javascript"*>

**var** oper = {

save:**function**(id){

**if**(!validForm()) {

**return** **false**;

}

id = id || '0';

**var** url = 'admin/contact/save/'+id;

form1.action = url;

form1.submit();

**return** **true**;

}

};

</script>

<%

};

var bodyContent = {

%>

<form name=*"form1"* action=*""* method=*"post"* class=*"form-horizontal"* role=*"form"*>

<input type=*"hidden"* name=*"model.id"* value=*"${model.id}"* />

<table class=*"table"*>

<% // 列表头部 %>

<tr>

<td>姓名</td>

<td>

<input class=*"form-control"* type=*"text"* name=*"model.name"* value=*"${model.name}"*

valid=*"vtext"* validname=*"名称"* />

</td>

</tr>

………………………………

<tr>

<td>说明</td>

<td>

<textarea class=*"form-control"* rows=*"3"* cols=*"30"* name=*"model.remark"*>${model.remark}</textarea>

</td>

</tr>

</table>

<div style="height: *50px*;clear: *both*;">&nbsp;</div>

<nav class=*"navbar navbar-default navbar-fixed-bottom"*>

<div class=*"container"* style="padding: *5px 0px 5px 0px*;text-align: *center*;">

<button class=*"btn btn-primary"* onclick="return oper.save(${model.id!'0'});">保 存</button>

<button class=*"btn btn-default"* onclick="closeIframe();return false;">关 闭</button>

</div>

</nav>

</form>

<%}; %>

<% layout("/pages/template/\_layout.html",{head:headContent,body:bodyContent}){ %>

DIRECTIVE SAFE\_OUTPUT\_CLOSE;

<%} %>

下面是列表页代码， 方法调用以oper对象调用形式封装。增加、修改、查看页面以弹框形式展示，这样可以很好的保存列表页查询条件。查看页面关闭不会刷新列表，添加和修改页面关闭后均会刷新列表，均已封装为Iframe方法。

paginator属性为列表分页查询调用方法。

showMenu方法为菜单选中展现，具体参数在菜单管理进行配置。

menu.html为菜单栏，各页面直接引用即可。paginator.html为分页栏，固定获取page参数进行处理。

列表头部设置设置class=*"sorting"*为支持排序功能，name字段为传到后台的参数值。

<%

var headContent = {

include("/pages/template/head.html"){}

%>

<script type=*"text/javascript"*>

**var** oper;

jQuery(**function**($) {

// 页面方法

oper = {

width : 400,

height : 430,

form : document.form1,

list : **function**() {

**var** url = 'admin/contact/list';

**this**.form.action = url;

**this**.form.submit();

},

view : **function**(id) {

**var** url = 'admin/contact/view/'+id;

**var** title = '查看联系人';

Iframe(url, **this**.width, **this**.height, title, **false**, **false**, **false**, EmptyFunc);

},

add : **function**() {

**var** url = 'admin/contact/add';

**var** title = '添加联系人';

Iframe(url, **this**.width, **this**.height, title);

},

edit : **function**(id) {

**var** url = 'admin/contact/edit/'+id;

**var** title = '修改联系人';

Iframe(url, **this**.width, **this**.height, title);

},

del : **function**(id) {

**var** url = 'admin/contact/delete/'+id;

**var** title = '确认要删除该联系人信息？';

Confirm(title, **function**() {

form1.action = url;

form1.submit();

});

}

};

//显示Menu索引

showMenu('page\_contact');

});

//分页

**var** paginator = **function**(page) {

oper.list();

};

</script>

<%

};

var bodyContent = {

%>

<form name=*"form1"* action=*""* method=*"post"* class=*"form-inline"* role=*"form"*>

<!-- 菜单 -->

<% include("/pages/template/menu.html"){} %>

<div class=*"tableSearch"*>

<% //查询列表 %>

<div class=*"form-group"*>

<input class=*"form-control"* type=*"text"* name=*"attr.name"* value=*"${attr.name!''}"*

placeholder=*"请输入姓名"* required=*'required'* />

</div>

<button type=*"button"* class=*"btn btn-default"* onclick="oper.list();" name=*"search"*>

<span class=*"glyphicon glyphicon-search"*></span> 查 询

</button>

<button type=*"button"* class=*"btn btn-default"* onclick="resetForm();">

<span class=*"glyphicon glyphicon-refresh"*></span> 重 置

</button>

<button type=*"button"* class=*"btn btn-default"* onclick="oper.add();">

<span class=*"glyphicon glyphicon-plus"*></span> 新 增

</button>

</div>

<!-- 数据列表 -->

<table class=*"table table-striped table-bordered table-hover"*>

<thead>

<tr>

<th>序号</th>

<% // 列表头部 %>

<th name=*"name"* class=*"sorting"*>姓名</th>

<th name=*"phone"* class=*"sorting"*>手机号</th>

<th name=*"email"* class=*"sorting"*>Email</th>

<th>说明</th>

<th>操作</th>

</tr>

</thead>

<tbody>

<% for(item in page.list){ %>

<tr>

<td>${itemLP.index }</td>

<% // 列表内容 %>

<td>${item.name}</td>

<td>${item.phone}</td>

<td>${item.email}</td>

<td title=*"${item.remark}"*>

<% if (strutil.length(item.remark) > 6) { %>

${strutil.subStringTo(item.remark, 0, 6)}...

<% } else { %>

${item.remark}

<% } %>

</td>

<td>

<a href=*"javascript:void(0);"* class=*"btn btn-sm btn-success"* onclick="oper.view(${item.id});">查看</a>

<a href=*"javascript:void(0);"* class=*"btn btn-sm btn-primary"* onclick="oper.edit(${item.id});">修改</a>

<a href=*"javascript:void(0);"* class=*"btn btn-sm btn-danger"* onclick="oper.del(${item.id});">删除</a>

</td>

</tr>

<% } %>

</tbody>

</table>

<%include("/pages/includes/paginator.html"){} %>

</form>

<%}; %>

<% layout("/pages/template/\_layout.html",{head:headContent,body:bodyContent}){ %>

<%} %>

view页面相对简单，只是一个数据展示，这里就不多做介绍了。

如果你看到了这里，有了后台代码和前台页面，联系人模块就完成了，配置一下菜单就能看到效果了~！~

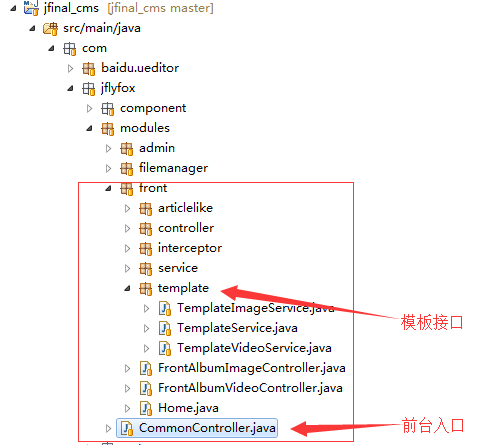
*温馨提示：列表分页、菜单栏、js验证和列表排序功能封装实现这里先不做说明，有兴趣的可以查看源码。*

## 前台模板开发

### java代码

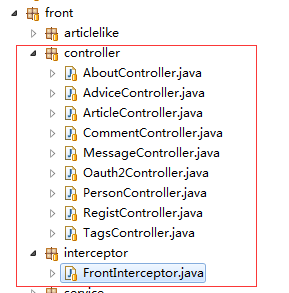
前台CommonController为整个项目默认入口，项目根目录、登陆、登出等。

TemplateService为前台模板接口，这里的TemplateService通过beetl方法注册实现接口调用。再config中进行注册，代码如下groupTemplate.registerFunctionPackage("temp", TemplateService.class);这个不明白可以看下beetl文档，这样就实现了页面调用temp.article(1)返回文章对象。



后台service封装实现具体方法，template进行service调用提供接口。这样前台页面直接访问template接口，很好的实现了代码前后端代码分离。当然前台的java还进行了公共Controller的封装以及公共参数的传递。

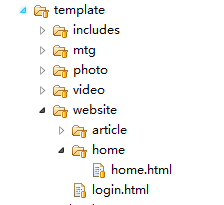
如文章、评论、消息、个人信息、注册、标签等公共Controller以及统一前台参数传递拦截器FrontInterceptor。截图如下：



有一些比较个性化的东西，我这里就没有封装接口，直接创建了一个Controller。

### 模板代码

前台模板现在有几个了~这里我们以最简单的website单页滚动网站为例。



这里主要得模板页面就是home.html。项目栏目通过后端管理进行配置，默认是home目录下的${urlkey}.html文件，website配置的是home。当然也可以直接通过栏目管理配置模板路径。

这里我们主要看顶部的接口调用，temp.articlePage返回当前站点的前20条文章列表，进行列表数据展示。temp.article返回当前站点设置的默认文章，进行一些属性展示。

<%

// 获取前20个文章

var articlePage = temp.articlePage(1, 20, session.site.model.site\_folder\_id);

var homeArticle = temp.article(session.site.model.site\_article\_id);

%>

<!DOCTYPE html>

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<title>${HEAD\_TITLE }</title>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<link rel=*"icon"* href=*"${BASE\_PATH }favicon.ico"* />

<link rel=*"shortcut icon"* href=*"${BASE\_PATH }favicon.ico"* />

<meta http-equiv=*"pragma"* content=*"no-cache"*>

<meta http-equiv=*"cache-control"* content=*"no-cache"*>

<meta http-equiv=*"expires"* content=*"0"*>

<meta name=*"keywords"* content=*"${HEAD\_KEYWORDS }"*>

<meta name=*"description"* content=*"${HEAD\_DESCRIPTION }"*>

<!-- IE兼容处理 -->

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<link rel=*"stylesheet"* type=*"text/css"* href=*"http://cdn.bootcss.com/fullPage.js/2.7.6/jquery.fullPage.min.css"* />

<script src=*"http://cdn.bootcss.com/jquery/1.8.3/jquery.min.js"*></script>

<script type=*"text/javascript"* src=*"http://cdn.bootcss.com/fullPage.js/2.7.6/jquery.fullPage.min.js"*></script>

<link href=*"http://cdn.bootcss.com/bootstrap/3.2.0/css/bootstrap.min.css"* rel=*"stylesheet"*>

<style>

……………………

</style>

<script type=*"text/javascript"*>

**var** anchorsArray = [];

**var** navigationTooltipsArray = [];

<% **for**(item **in** articlePage.list){ %>

anchorsArray.push('j${item.id}');

navigationTooltipsArray.push('${item.title}');

<% } %>

$(document).ready(

**function**() {

$('#fullpage').fullpage({

anchors : anchorsArray, //[ 'j0','j1', 'j2','j3','j4','j5','j6' ],

navigation : **true**,

navigationPosition : 'right',

navigationTooltips : navigationTooltipsArray // [ '英雄联盟','德玛西亚皇子', '德邦总管','黑暗之女', '皎月女神','寒冰射手','联系我们']

,afterLoad: **function**(anchorLink, index){

$('ul.nav li').removeClass('active');

$('ul.nav li#nav\_'+anchorLink).addClass('active');

}

});

});

</script>

</head>

<body>

<nav class=*"navbar navbar-default navbar-fixed-top"*>

<div class=*"container-fluid"*>

<div class=*"navbar-header"*>

<a class=*"navbar-brand"* href=*"#"*> <img alt=*"logo"* src=*"${flyfox.getImage(homeArticle)}"* width=*"24"* height=*"24"*>

${homeArticle.title}

</a>

</div>

<div class=*"container"*>

<ul class=*"nav navbar-nav navbar-right"*>

<% for(item in articlePage.list){ %>

<li id=*"nav\_j${item.id}"*><a href=*"#j${item.id}"*>${item.title}</a></li>

<% } %>

</ul>

</div>

</div>

</nav>

<div id=*"fullpage"*>

<% for(item in articlePage.list){ %>

<div class=*"section"* id=*"section${itemLP.index}"* style="background-image: *url(*${*flyfox.getImage*(*item)*});">

${item.content!}

</div>

<% } %>

</div>

</body>

</html>

到这里简单的前台模板功能就介绍完了。

## 前台模板接口

### 前台默认入口

首先上一章节提到了默认入口，这里我要详细说明下默认访问路径规则。FrontInterceptor前台拦截器会传递友情链接、关于我们、推荐文章和栏目信息数据。然后根目录会默认访问当前站点设置的栏目，如果未设置那么展示TbFolder.*ROOT栏目*，如果url传递了目录Id或者目录urlKey那么会访问对应的栏目。跳转规则是优先处理跳转jumpUrl，其次是配置的模板路径，如果都没有按照默认home/${urlKey}.html的规则跳转。从这里可以看出，简单的栏目规则就是每个站点下面栏目是和urlKey一一对应的。

如我们添加一条栏目，id：101 urlKey：jflyfox那么我们可以通过“项目地址/101.html”访问，也可以通过“项目地址/jflyfox.html”访问。但是默认的栏目菜单生成规则是优先使用urlKey。具体可以看模板下includes/ header\_menu.html实现。

@ControllerBind(controllerKey = "/")

**public** **class** CommonController **extends** BaseProjectController {

**…………………………**

/\*\*

\* 首页，菜单

\*

\* 2015年5月25日 下午11:00:28 flyfox 330627517@qq.com

\*/

@Before(FrontInterceptor.**class**)

**public** **void** index() {

// new FrontService().menu(this);

**int** folderRoot = TbFolder.*ROOT*;

SessionSite site = getSessionSite();

Integer siteFolderId = site.getModel().getSiteFolderId();

**if** (siteFolderId != **null** && siteFolderId > 0) {

folderRoot = siteFolderId;

}

String folderStr = getPara();

Integer folderId = folderRoot;

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

**if** (NumberUtils.*parseInt*(folderStr) > 0) {

folderId = NumberUtils.*parseInt*(folderStr);

} **else** {

folderId = NumberUtils.*parseInt*(FolderService.*getMenu*(folderStr));

}

}

**if** (folderId == **null** || folderId <= 0) {

folderId = folderRoot;

}

// 活动目录

setAttr("folders\_selected", folderId);

TbFolder folder = **new** FolderService().getFolder(folderId);

setAttr("folder", folder);

setAttr("paginator", getPaginator());

// seo：title优化

String folderName = (folder == **null** ? "" : folder.getStr("name") + " - ");

setAttr(JFlyFoxUtils.*TITLE\_ATTR*, folderName + getAttr(JFlyFoxUtils.*TITLE\_ATTR*));

// 栏目跳转规则

String jumpUrl = folder.getJumpUrl();

String path = folder.getPath();

String urlKey = folder.getKey();

**if** (StrUtils.*isNotEmpty*(jumpUrl)) {

redirect(jumpUrl);

} **else** **if** (StrUtils.*isNotEmpty*(path)) {

renderAuto(path);

} **else** {

renderAuto(Home.*PATH* + urlKey + ".html");

}

}

……………………………………

}

其次是FrontAlbumImageController和FrontAlbumVideoController分别是图片和视频模块统一入口，现在功能相对简单，这里就不详细介绍了。

### 前台接口

前台接口主要是TemplateService这个类，提供方式是利用beetl函数注册方式，如groupTemplate.registerFunctionPackage("temp", TemplateService.class);代码，这样我们就可以通过temp.${method}进行调用了。

下面是接口代码，例如通过temp.countView(100),我们就能获取到文章id为100的浏览量，temp. articlePage(23,1,20)我们就能够获取到栏目id为23下面的前20条文章。

接口主要提供了文章列表、文章内容、浏览量、评论量，栏目标签，栏目、滚屏图片列表和提醒列表等信息。

**package** com.jflyfox.modules.front.template;

**……………………**

/\*\*

\* 模板方法接口

\*

\* 2016年1月18日 下午6:05:54 flyfox 330627517@qq.com

\*/

**public** **class** TemplateService **extends** BaseService {

**private** **final** **static** FrontCacheService *service* = **new** FrontCacheService();

**private** **final** **static** ArticleLikeCache *articleLikeservice* = **new** ArticleLikeCache();

/\*\*

\* 获取浏览数

\*

\* 2015年6月2日 下午6:30:56 flyfox 330627517@qq.com

\*

\* **@param** articleId

\* **@return**

\*/

**public** **static** **int** countView(**int** articleId) {

TbArticle article = *service*.getArticleCount(articleId);

**return** article == **null** ? 0 : article.getCountView();

}

/\*\*

\* 获取评论数

\*

\* 2015年6月2日 下午6:30:56 flyfox 330627517@qq.com

\*

\* **@param** articleId

\* **@return**

\*/

**public** **static** **int** countComment(**int** articleId) {

TbArticle article = *service*.getArticleCount(articleId);

**return** article == **null** ? 0 : article.getCountComment();

}

/\*\*

\* 获取浏览数

\*

\* 2015年6月2日 下午6:30:56 flyfox 330627517@qq.com

\*

\* **@param** articleId

\* **@return**

\*/

**public** **static** **boolean** isLike(**int** userId, **int** articleId) {

**return** *articleLikeservice*.isLike(userId, articleId);

}

/\*\*

\* 获取标签信息

\*

\* 2015年5月25日 下午11:49:58 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folderId

\* 目录

\* **@return**

\*/

**public** List<TbTags> tagsListByArticle(**int** articleId) {

**return** *service*.getTagsByArticle(articleId);

}

/\*\*

\* 获取标签信息

\*

\* 2015年5月25日 下午11:49:58 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folderId

\* 目录

\* **@return**

\*/

**public** Page<TbTags> tagsPageByFolder(**int** pageNo, **int** pageSize, **int** folderId) {

**return** *service*.getTagsByFolder(**new** Paginator(pageNo, pageSize), folderId);

}

/\*\*

\* 获取标签信息

\*

\* 2015年5月25日 下午11:49:03 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@return**

\*/

**public** Page<TbTags> tagsPage(**int** pageNo, **int** pageSize, **int** siteId) {

**return** *service*.getTags(**new** Paginator(pageNo, pageSize), siteId);

}

/\*\*

\* 查询文章，展示的和类型为11,12的推荐文件,前10个

\*

\* 2015年4月29日 下午4:48:24 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folder\_id

\* **@return**

\*/

**public** Page<TbArticle> articlePageRecommend(**int** pageNo, **int** pageSize, **int** siteId) {

**return** *service*.getRecommendArticle(**new** Paginator(pageNo, pageSize), siteId);

}

/\*\*

\* 返回最新文章

\*

\* 2015年5月24日 下午10:52:05 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@return**

\*/

**public** Page<TbArticle> articlePageTop(**int** pageNo, **int** pageSize, **int** siteId) {

**return** *service*.getNewArticle(**new** Paginator(pageNo, pageSize), siteId);

}

/\*\*

\* 返回文章列表

\*

\* 2015年5月24日 下午10:52:05 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folderId

\* **@return**

\*/

**public** Page<TbArticle> articlePageSite(**int** pageNo, **int** pageSize, **int** siteId) {

**return** *service*.getArticleBySiteId(**new** Paginator(pageNo, pageSize), siteId);

}

/\*\*

\* 返回文章列表

\*

\* 2015年5月24日 下午10:52:05 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folderId

\* **@return**

\*/

**public** Page<TbArticle> articlePage(**int** pageNo, **int** pageSize, **int** folderId) {

**return** *service*.getArticle(**new** Paginator(pageNo, pageSize), folderId);

}

/\*\*

\* 返回文章列表不走缓存

\*

\* 2015年5月24日 下午10:52:05 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folderId

\* **@return**

\*/

**public** Page<TbArticle> articlePageNoCache(**int** pageNo, **int** pageSize, **int** folderId) {

**return** *service*.getArticleByNoCache(**new** Paginator(pageNo, pageSize), folderId);

}

/\*\*

\* 返回对应文章

\*

\* 2015年5月24日 下午10:52:05 flyfox 330627517@qq.com

\*

\* **@param** paginator

\* **@param** folderId

\* **@return**

\*/

**public** TbArticle article(**int** articleId) {

**return** *service*.getArticle(articleId);

}

/\*\*

\* 获取栏目滚动图片

\*

\* 2016年1月28日 下午5:28:25 flyfox 330627517@qq.com

\*

\* **@param** folderId

\* **@return**

\*/

**public** List<TbFolderRollPicture> rollPicture(**int** folderId) {

**return** *service*.getRollPicture(folderId);

}

/\*\*

\* 获取公告信息

\*

\* 2016年1月28日 下午5:29:47 flyfox 330627517@qq.com

\*

\* **@param** folderId

\* **@return**

\*/

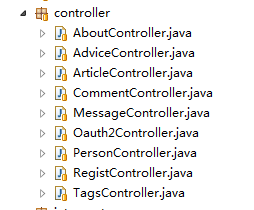
**public** List<TbFolderNotice> notice(**int** folderId) {

**return** *service*.getNotice(folderId);

}

}

除了默认出口和上述接口外，我们还提供了一些公共Controller和个性Controller，如下图所示：



这些Controller大致实现了关于我们、建议、文章查看、评论、消息、Oauth2登陆、个人信息、注册信息和标签功能。

## 项目工具类

base下包含了Model和Controller的父类：Model实现了增删改统一日志，自定义缓存查询（类似jfinal自带缓存查询）以及查询方法扩展，部分实现在BaseModel中。Controller实现了获取和设置用户session、站点session获取、自动识别移动和PC端跳转、带提示跳转，获取分页、获取form等功能，部分实现在BaseController中。

beetl下包含了beetl公共方法封装：数据字典获取、图片路径转换、菜单地址获取以及常规字符串处理方法。

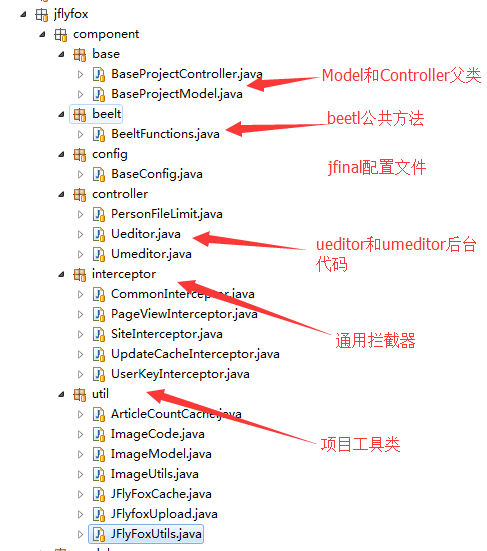
config下是jfinal配置文件入口，里面有大量的jfinal初始化配置，这里就不一一说明了。

controller下是ueditor和umeditor的后台实现代码。

interceptor是通用拦截器：包含SEO属性设置，页面访问量统计，站点Session处理、缓存更新以及用户Session唯一键设置。在jflyfox\_jfinal中还包含登陆session拦截、Session属性设置、form获取统一处理。

util下是一些公共配合及方法：包含文章访问量和评论量缓存，验证码生成、图片属性获取、缓存管理统一入口、上传组件封装以及常量类等。

项目内部工具类大致如下图所示：



# jfinal cms项目使用说明

## 数据库配置

数据库配置文件路径：src/main/resources/conf/db.properties，修改相应配置即可。

db\_type=mysql

mysql.jdbcUrl =jdbc:mysql://127.0.0.1:3306/jfinal\_cms2?characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull

mysql.user = root

mysql.password = 123456

mysql.driverClass = com.mysql.jdbc.Driver

## oauth认证配置

配置文件路径：src/main/resources/conf/db.properties

这里以qq oauth2认证为例：默认使用openid\_qq，openkey\_qq，redirect\_qq。但是如果使用了多站点，由于各个站点域名的不同回调地址也会不同，这里我们就加入了站点名称为前缀，区分不同的站点配置参数，配置如下：

## qq openid

openid\_qq =

openkey\_qq =

redirect\_qq = http://www.jflyfox.com/mtg/oauth2/qq\_callback

blog.openid\_qq =

blog.openkey\_qq =

blog.redirect\_qq = http://blog.jflyfox.com/oauth2/qq\_callback

## 站点设置

jfinal cms支持单站点和多站点。单站点就是一套前台展示对应一套后台管理系统；多站点就是多套前台展示（不同域名区分）多应一套后台管理系统。

首先登陆后台管理系统，在“系统管理--》参数配置”中找到站点相关参数；通过配置“SITE.MULTI.FLAG”参数来切换单站点和多站点，默认false为单站点模式。配置参数如下图：



### 单站点

单站点模式只需要在“其他管理--》站点管理”设置默认模板，就是前台要展示的模板了。如下图：



### 多站点

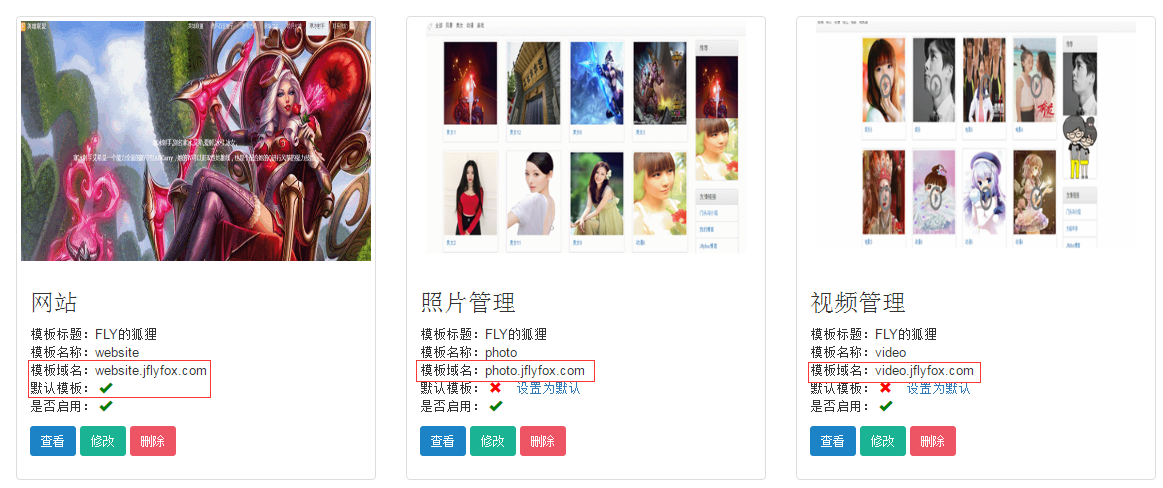
多站点模式也需要在“其他管理--》站点管理”进行设置，首先通过模板域名进行匹配展示；如果访问的域名没有配置，那么就展示默认模板。如下图：

如果访问<http://website.jflyfox.com>这个地址，前台展示“网站”模板

如果访问<http://photo.jflyfox.com>这个地址，前台展示“图片管理”模板

如果访问<http://video.jflyfox.com>这个地址，前台展示“视频管理”模板

如果访问[http://mtg.jflyfox.com](http://mtg.jflyfox.com/)这个地址，由于没有匹配，前台展示“网站”模板



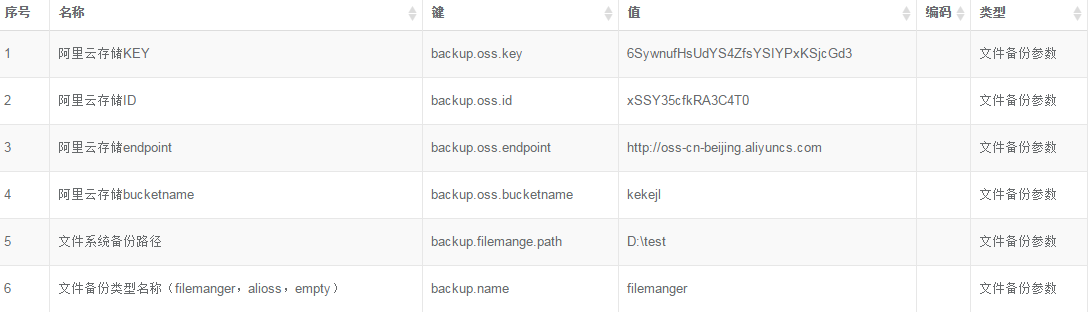
## 文件备份设置

文件备份是指网站相关图片、视频、文件等附件上传后，通过本地文件，阿里云OSS等其他存储形式进行备份。由于附件上传后均存储到本地项目目录下，为了安全起见，需要进行附件备份。

1. 管理员访问服务器，上传图片，备份到文件系统或者阿里云OSS等存储；
2. 用户访问服务器，浏览前台网页图片，从服务器进行获取；如果服务器不存在，会从文件系统或者阿里云OSS等存储进行恢复，恢复后再展示给用户。



首先登陆后台管理系统，在“系统管理--》参数配置”中找到文件备份相关参数；暂时支持文件系统备份、阿里云OSS备份、不备份三种模式；通过配置**backup.name**参数进行切换，配置分别为filemanger，alioss，empty；



### 不备份（empty）

这个模式就不用介绍了~！~

### 文件系统备份（filemanger）

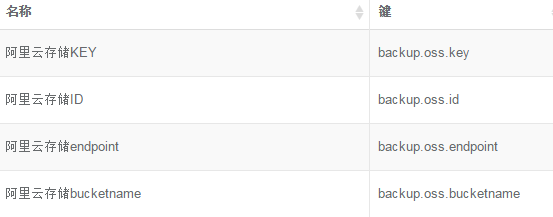
这种模式就是将项目中的附件目录，再在项目以外的文件系统目录中进行一份备份；由于目录结构完全一直维护、管理和手动整体恢复十分方便。

这里只需要配置文件系统的备份根目录即可，参数为backup.filemange.path。

### 阿里云OSS备份（alioss）

这种模式是通过阿里云的对象存储OSS进行备份，阿里云OSS可以很方便的进行CDN支持；由于是阿里云第三方文件系统，维护、管理和整体恢复稍微有点麻烦；但是阿里云提供了OSS管理工具，还是很不错的。

这里需要配置四个参数，如下图所示：



### 自定义备份

如果三种备份模式均不能满足你的需求；也可以实现自己的备份策略哦。

只需要实现com.jflyfox.system.file.service.IFileBackup接口的备份和恢复方法,并在com.jflyfox.system.file.service.FileUploadService的getFileBackup方法中配置你的类即可。

# jfinal cms项目依赖

jfinal cms采用maven结构，依赖与作者的其他两个项目。

* [jflyfox\_base](https://github.com/jflyfox/jflyfox_base)项目

jfinal\_base提供了java的工具类，如StrUtils,NumberUtils,DateUtils 等

github地址：<https://github.com/jflyfox/jflyfox_base>

oschina码云地址： <https://gitee.com/jflyfox/jflyfox_base>

* jflyfox\_jfinal项目

jflyfox\_jfinal是对Jfinal等组件的封装。

包含controller，model，form，service基础类封装。

对分页进行了后台和前台的实现。

加入了自动扫描model和controller以及注解支持。

实现了ueditor后台Controller代码。

加入了页面增删改查代码自动生成功能，可通过beetl模板进行配置。

实现了SessionAttrInterceptor、页面和手机设备判断拦截器以及BasePathHandler。

github地址：<https://github.com/jflyfox/jflyfox_jfinal>

oschina码云地址：<http://gitee.com/flyfox/jflyfox_jfinal>

# 联系作者

作者：FLY的狐狸

jfinal cms交流群：568909653

QQ：369191470

邮件：[369191470@qq.com](mailto:xiandafu@126.com)

源码地址：<http://git.oschina.net/flyfox/jfinal_cms>

# 开源赞助

支付宝和微信支付：项目的发展，离不开大家得支持~！~

