

## 途牛 java 框架二次开发指南

研发部底层框架组

**2/6/2012**

版本 1.0

姓名：王小全

职务：系统架构师

电子邮件：wangxiaoquan@tuniu.com

## 目录

1. 简介 .....	4
2. 环境准备 .....	4
2.1. 详见另一篇跟团 java 开发环境搭建 .....	4
3. 二次开发应用标准 .....	4
3.1. 包名标准 .....	4
3.2. 子包结构(Com.tuniu.gt.complaint) .....	4
3.2.1. Action.[模块名称...] .....	4
3.2.2. DAO .....	4
3.2.3. Entity 存放所有实体 .....	5
3.2.4. service .....	5
3.2.5. Interceptor 存放应用拦截器 .....	5
3.2.6. Webservice 存放应用提供的服务 .....	5
3.3. Html&js 结构 .....	5
3.4. 命名标准 .....	5
3.4.1. 包命名标准 .....	5
3.4.2. 接口命名标准 .....	6
3.4.3. Java 文件命名标准 .....	6
3.4.4. 方法命名标准 .....	6
3.4.5. 注解命名标准 .....	6
3.5. 配置文件标准 .....	7
4. 途牛框架提供的基本功能 .....	7
4.1. ACTION .....	7
4.2. Service .....	8
4.3. jsp 与调用框架静态方法系统 .....	10
4.4. 控件系统 .....	10
4.5. 无限分级系统 .....	10
4.6. 树形菜单系统 .....	11
4.7. 权限系统 .....	11

---

4.8.	配置系统.....	11
4.9.	Webservice 系统.....	11
4.10.	日志系统.....	11
4.11.	用户系统.....	11
5.	开发 Demo .....	12
5.1.	创建新应用 .....	12
5.1.1.	修改 WEB-INF/config/config.properties .....	12
5.1.2.	修改 web.xml .....	12
5.2.	开发准备.....	13
5.3.	进入系统.....	13
5.4.	将刚才的表结构导入系统.....	15
5.5.	设置系统生成界面的 ui.....	15
5.6.	点击提交程序并生成代码.....	16
5.7.	在 url 里访问刚才生成的 example.....	16
6.	复杂开发.....	16
6.1.	联表查询 1-1 .....	16
6.2.	联表查询 1-n .....	17
6.3.	事务处理.....	18
6.4.	回调方法的使用.....	18
6.4.1.	Execute 回调方法 .....	19
7.	测试系统.....	19

## 1. 简介

本文档基于研发部 java 框架 SSI ( struts2.2.3.1 + spring3.1.0+mybait 3.0.6) 开发框架编写, 根据此文档描述过程你可以快速的开发应用

## 2. 环境准备

2.1. 详见另一篇跟团 java 开发环境搭建

## 3. 二次开发应用标准

### 3.1. 包名标准

com.tuniu.系统名称.应用名称

文档以 com.tuniu.gt.complaint 即跟团系统投诉应用为例

### 3.2. 子包结构(Com.tuniu.gt.complaint)

#### 3.2.1. Action.[模块名称...]

#### 3.2.2. DAO

Dao.impl 存放 dao 的实现

Dao.sqlmap 存放 mybatis 的 sqlmap xml 配置文件

Dao.sqlmap.imap 存放 sqlmap xml 配置文件对应的接口文件

### 3.2.3. Entity 存放所有实体

### 3.2.4. service

Service.[模块名称....] 存放所有 service 的接口

Service.[模块名称....].Impl 存放对应接口的实现

### 3.2.5. Interceptor 存放应用拦截器

### 3.2.6. Webservice 存放应用提供的服务

## 3.3. Html&js 结构

Jsp 文件 统一存放在 WebContent/WEB-INF/html/应用名称/[模块名称...]

Js 文件统一存放在 WebContent/res/应用名称/[模块名...]

Jsp & js 文件

## 3.4. 命名标准

### 3.4.1. 包命名标准

包名全部以小写字母命名，不得含有大写，下划线或其它特殊字符

### 3.4.2. 接口命名标准

接口声明文件一律以大写 I 开头

### 3.4.3. Java 文件命名标准

文件统一下大写字母开头的驼峰规则命名

action/ entity/service 包下的文件一定以\$(各自首字母大写).java 结尾

接口实现文件以 xxxImpl.java 结尾

注 Dao.impl 下的文件以 Dao.java 结尾

### 3.4.4. 方法命名标准

方法首字母小写以驼峰规则命名, 别一个方法的后续方法以 do 前一方法名 如 addForm (展示表单页), 点提效按钮的处理方法则为 doAddFrom

### 3.4.5. 注解命名标准

注解名为 应用名称\_+下划线包名+-下划线文件名

下划线包名: 指将包名中的.号换成下划线

下划线文件名:指将文件名中的大写换成\_小写,并去掉后缀, 如

Entity,Action,ServiceImpl 等

如包 `com.tuniu.gt.frm.service.privilege.impl` 中的

`MenuPrivilegeServiceImpl.java` 文件

注解名为 `frm_service_privilege_impl-menu_privilege`

### 3.5. 配置文件标准

Action 配置

统一存放在 `WEB-INF/config/应用名称/package.xml`

Ajax 配置

统一存放在 `WEB-INF/config/应用名称/ajax-package.xml`

webservice 配置

统一存放在 `WEB-INF/config/应用名称/webservice-package.xml`

## 4. 途牛框架提供的基本功能

### 4.1. ACTION

自动获取列表，并自动分页,支持搜索

自动获取当前登录用户 `UserEntity user`

支持添加，编辑功能

支持回调功能

Action 所有操作都和 service 交互，不直接与 dao 交互

## 4.2. Service

```
    /**
     * 无条件返回实体列表
     * @return
     */
    public List<?> fetchList();

    /**
     * 根据指定条件返回实体列表
     */
    public List<?> fetchList(Map<String, Object> paramMap);

    /**
     * 无条件返回map列表
     * @return
     */
    public List<?> fetchListMap();

    /**
     * 根据指定条件返回map列表
     * @param paramMap
     * @return
     */
    public List<?> fetchListMap(Map<String, Object> paramMap);

    /**
     * 根据指定条件 返回以指定的field 为key的map
     * @param paramMap
     * @param field
     * @return
     */
    public Map<String, Object> fetchMapMap(Map<String, Object> paramMap, String
    field);

    /**
     * 根据id返回一个实体
     * @param id
     * @return
     */
    public Object fetchOne(Integer id);
```



```
/**
 * 根据指定条件反回一个实体
 * @param paramMap
 * @return
 */
public Object fetchOne(Map<String, Object> paramMap);

/**
 * 根据指定条件反回一个实体或map
 * @param paramMap
 * @param getEntity
 * @return
 */

public Object fetchOne(Map<String, Object> paramMap, Boolean getEntity);

/**
 * 根据实体或map插入一条数据
 * @param e
 * @return
 */
public int insert(Object e);

/**
 * 根据map更新一条数据
 * @param e
 */
public void update(Object e);

/**
 * 根据id反回一个实体
 * @param id
 * @return
 */
public Object get(Object id);

/**
 * 根据指定条件反回一个实体或map
 * @param paramMap
 * @param getEntity
 * @return
 */
```

```
*/  
public Object get(Map<String,Object> paramMap);
```

```
/**  
 * 删除一条数据  
 * @param id  
 * @return  
 */  
public int delete(Object id);
```

### 4.3.jsp 与调用框架静态方法系统

frm:md5

ui:makeCheckbox

ui:makeRadio

ui:makeSelect

### 4.4.控件系统

管理系统 ui 显示控件 如 input,checkbox,html 编辑器, 时间, 日  
因等

### 4.5.无限分级系统

系统提供统一的无限分级处理机制

## 4.6. 树形菜单系统

系统提供统一的树形菜单方法

## 4.7. 权限系统

系统提供统一的权限验证，管理方法

## 4.8. 配置系统

系统提供统一的配置管理

## 4.9. Webservice 系统

系统提供统一的调用(http,soap,xmlrpc,webservice)方法

另提供统一 service 文档描述生成器

## 4.10. 日志系统

系统提供统一日志处理机制

## 4.11. 用户系统

系统提供与 tuniu uc 同步用户/部门/岗位/职位 等方法

## 5. 开发 Demo

### 5.1. 创建新应用

如果应用不存在,我们需建立一个新应用.

在已有应用下进行模块开发,跳至 5.2 节

#### 5.1.1. 修改 WEB-INF/config/config.properties

app\_tbl\_pre=complaint:ct\_supply:spl\_uc:uc\_应用英文标识:应用表前缀

#### 5.1.2. 修改 web.xml

```
<filter>
    <filter-name>struts2</filter-name>
    <!--
<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class> -->

    <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/frm/action/*</url-pattern>
    <url-pattern>/frm/ajax/*</url-pattern>
    <url-pattern>/frm/webservice/*</url-pattern>
    <url-pattern>/complaint/action/*</url-pattern>
    <url-pattern>/complaint/ajax/*</url-pattern>
    <url-pattern>/complaint/webservice/*</url-pattern>

    <url-pattern>/应用的英文标识/action/*</url-pattern>
```

</filter-mapping>

## 5.2. 开发准备

创建表结构 frm\_test

```
CREATE TABLE IF NOT EXISTS `frm_test` (  
    `id` int(10) NOT NULL AUTO_INCREMENT,  
    `title` varchar(100) NOT NULL COMMENT '标题',  
    `content` text NOT NULL COMMENT '内容',  
    `add_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00'  
    COMMENT '添加时间',  
    `update_time` datetime NOT NULL DEFAULT '1970-01-01 00:00:00'  
    COMMENT '删除时间',  
    `del_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '删除标识',  
    PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

## 5.3. 进入系统

## 跟团游 java版框架系统测试

用户名

wangxiaoquan

密码

●●●●●●●●●●

☐ 记住我的登录信息

登录 »

[忘记密码?](#)

[← 返回 BOSS](#)

boss系统的用户名/密码

管理

系统功能

- 表管理
- 导入字段
- 菜单管理
- 参数管理
- 操作日志
- 开发帮助
- 重建程序

表管理

新增

id	所属应用	表名
1	frm	table
2	frm	log
3	frm	control
4	frm	table_field
5	frm	menu
6	frm	user
7	frm	menu_privilege

添加新表

所属应用前缀 *	frm_ 	→ 在config/config.properties里配置
表名 *	<input type="text"/>	→ 不在前缀的应用表名
表是否存在 *	<input checked="" type="radio"/> 是 <input type="radio"/> 否	
程序生成目录 *	<input type="text"/>	→ 程序所属模块
生成action *	<input type="checkbox"/> 是	→ 如不先择则系统不生成html/action/js文件
<input type="button" value="提交"/>		

struts启动时间:30 ms

action执行时间:0 ms

c标签解析时间1882 ms

## 5.4. 将刚才的表结构导入系统

系统功能

- 表管理
- 导入字段**
- 菜单管理
- 参数管理

操作成功

struts启动时间:154 ms

action执行时间:317 ms

c标签解析时间43001 ms

## 5.5. 设置系统生成界面的 ui

点击表管理 在刚才添加的那条记录选择结构

设置列表/表单 是否显示，以及其显示样式

表单	表单名	表单样式	必填	搜索	搜索样式	缓
<input type="checkbox"/> 显示	id	input	<input type="checkbox"/> 是	<input type="checkbox"/> 是	input	<input type="checkbox"/>
<input checked="" type="checkbox"/> 显示	标题	input	<input checked="" type="checkbox"/> 是	<input checked="" type="checkbox"/> 是	input	<input type="checkbox"/>
<input checked="" type="checkbox"/> 显示	内容	textarea	<input checked="" type="checkbox"/> 是	<input type="checkbox"/> 是	input	<input type="checkbox"/>
<input type="checkbox"/> 显示	添加时间	input	<input type="checkbox"/> 是	<input type="checkbox"/> 是	input	<input type="checkbox"/>

## 5.6. 点击提交程序并生成代码

系统将自动生成相应的

Action/dao/service/sqlmap.xml/isqlmap/list/form/js 等

## 5.7. 在 url 里访问刚才生成的 example

<http://localhost:8080/ssi/frm/action/test/test> 即可看到生成的代码

# 6. 复杂开发

## 6.1. 联表查询 1-1

本框架是基于 mybatis 持久层的，该层不强制要求实体与表结构



### 1-1 对应

假设有 a,b 两张表 a 与 b 1-1 对应关系, 且设 a 为主表

则在 a 实体里加上 `private BEntity be;` 和 `get,set` 方法

```
<association      property="author"      column="blog_author_id"
javaType=" B">

<id property="id" column="author_id"/>

<result property="username" column="author_username"/>

</association>
```

## 6.2. 联表查询 1-n

在 a 实体里加的为 `private List< BEntity> bel` 和 `get,set` 方法

```
<collection property="posts" ofType=" BEntity ">

<id property="id" column="post_id"/>

<result property="subject" column="post_subject"/>

<result property="body" column="post_body"/>

</collection>
```

集合元素的作用几乎和关联是相同的。实际上, 它们也很相似, 文档的异同是多余的。所以我们更多关注于它们的不同。

我们来继续上面的示例, 一个博客只有一个作者。但是博客有很多文章。在博客类中, 这可以由下面这样的写法来表示:

要映射嵌套结果集到 `List` 中, 我们使用集合元素。就像关联元素一样, 我们可以从连接中使用嵌套查询, 或者嵌套结果。

### 6.3. 事务处理

本框架用的编程式事务，自己手动在系统中写 `START commit`  
`rollback`

### 6.4. 回调方法的使用

当系统中提供的默认方法只有部份满足要求，如，我们需要对参数进行前置处理，或系统方法执行完后，我们需要后续处理时

系统默认提供两个保护的成员变量来提供回调功能

`callbackMap` //设置回调方法

`callbackParam` //设置回调的参数

系统默认自动调用 以 `doAdd` 为例

`_pre+方法名首字母大写` (`_preDoAdd`)

`_aft++方法名首字母大写` (`_aftDoAdd`)

如不想调用默认的自己可以在代码中写上

`callbackMap("pre","xxx");`

`callbackMap("aft","yyy");`

则系统会自动调用 `xxx,yyy` 方法

注意自动调用的方法声明，必须与参数一致

### 6.4.1. Execute 回调方法

系统默认自动调用如下两个方法

`_preExecute` 执行前预处理

`_aftExecute` 执行后处理

如果需要对系统默认查询的数据进行处理

则在 `execute` 方法中设置

```
callbackParam.put("dataList","");
```

同时声明 `_aftExecute(List<?> dataList)`

## 7. 测试系统

测试系统是基于 `junit test` 的

本系统所有测试代码保存到 `src` 平级的 `test` 目录下,目录结构自动按照 `src` 下的目录结构保存

因框架本身动态获取运行时 `tomcat` 因素,故对 `junit test` 进行了一层封装

所有开发代码测试时需 `extends TestCaseExtend`

下面是一个可运行的 简单 demo

```
public class LogActionTest extends TestCaseExtend {

    @Test
    public final void testSetService() {
        LogAction la = (LogAction) Common.getBean("frm_action_log-log");
    }

    public final void testExecute() {

    }

}
```

