软件开发平台API文档

目录

[配置相关 1](#_Toc516242254)

[新手导入工程说明 1](#_Toc516242255)

[让部署节点支持远程重启应用 2](#_Toc516242256)

[前端相关API 2](#_Toc516242257)

[如何缓存数据对象？ 2](#_Toc516242258)

[如何读取缓存的数据对象？ 3](#_Toc516242259)

[如何移除缓存的数据对象？ 3](#_Toc516242260)

[判断数组是否包含某个值 3](#_Toc516242261)

[如何验证放在TAB容器的所有表单是否合法？ 4](#_Toc516242262)

[获取可编辑表格所勾选的记录数组 4](#_Toc516242263)

[获取可编辑表格所有记录JSON字符串 5](#_Toc516242264)

[获取可编辑表格所有记录数组 5](#_Toc516242265)

[获取JqGrid表格所选择的记录数组 5](#_Toc516242266)

[改变单选框的值 6](#_Toc516242267)

[获取单选框的值 6](#_Toc516242268)

[获取复选框的值 6](#_Toc516242269)

[修改一个控件的权限(隐藏、只读、可写) 7](#_Toc516242270)

[重新加载复选框、单选框、下拉框 7](#_Toc516242271)

[如何改变下拉框的值 8](#_Toc516242272)

[后端相关API 8](#_Toc516242273)

[如何获取后台登录用户信息 8](#_Toc516242274)

[如何用代码跳转到手写的界面 9](#_Toc516242275)

[如何生成有详情内容的日志? 9](#_Toc516242276)

[简易版Lucene全文检索功能集成说明 11](#_Toc516242277)

[与第三方系统单点登录快速集成说明 14](#_Toc516242278)

# 配置相关

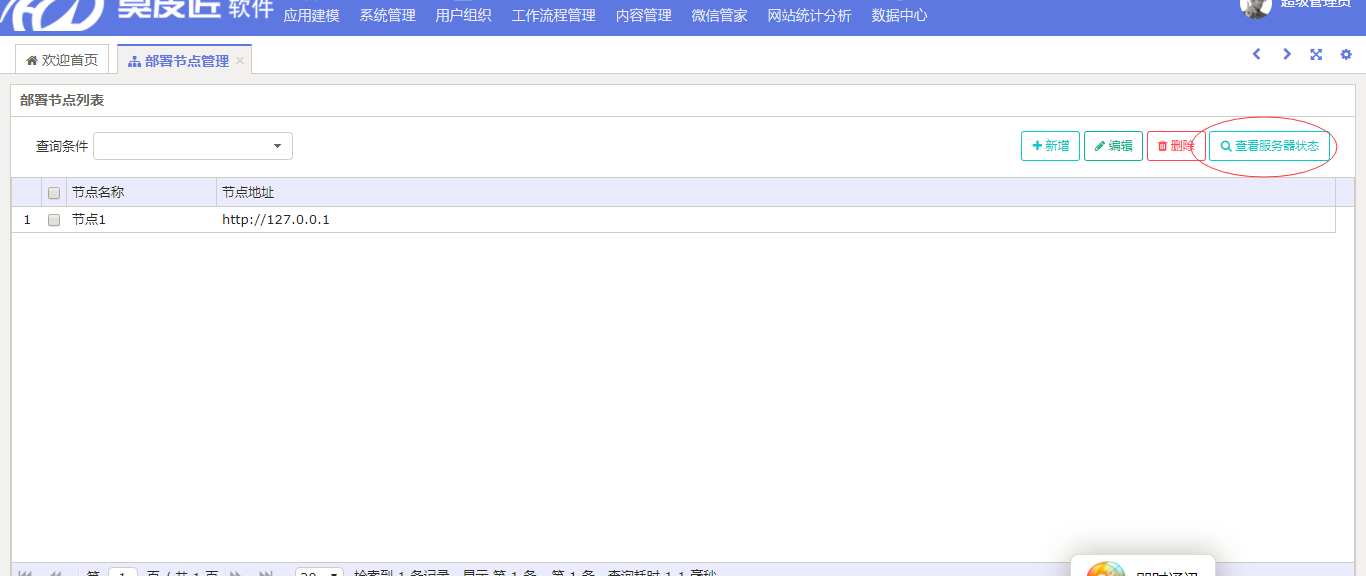
## 新手导入工程说明

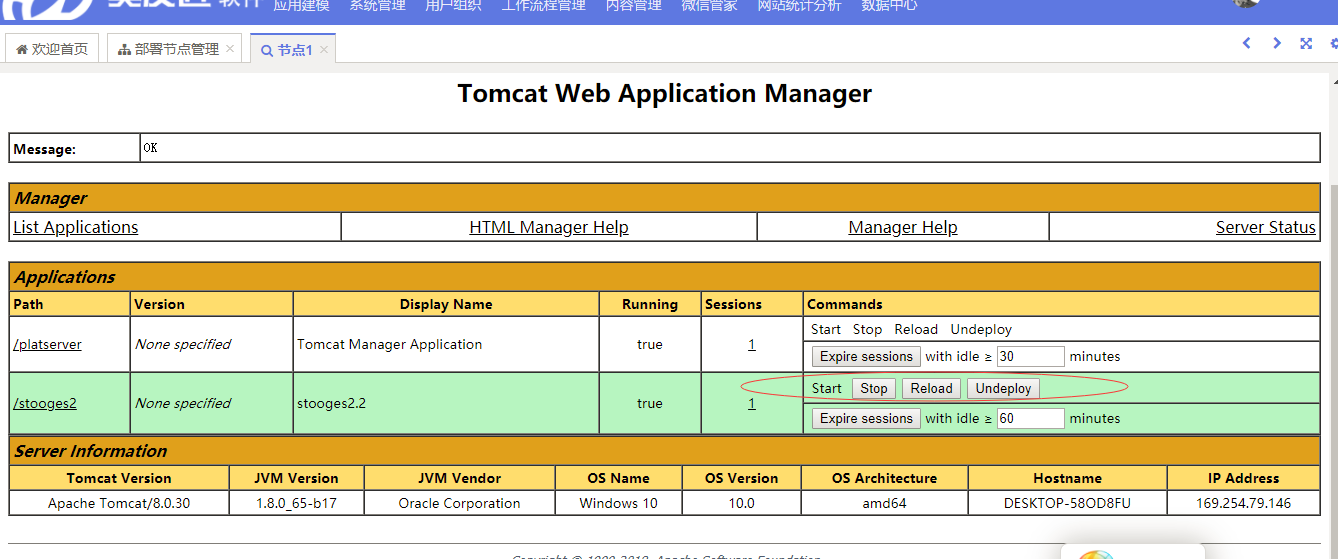
<https://my.oschina.net/u/2356625/blog/1649948>

## 让部署节点支持远程重启应用

如果需要在自动化部署的节点能够管理服务器,那么需要将docs中的platserver一起部署到tomcat下

效果图





# 前端相关API

## 如何缓存数据对象？

**方法名称:**

PlatUtil.setData(key,value);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| key | 缓存对象的KEY |
| Value | 缓存对象的值，可以是JS对象 |

**调用例子:**

var userInfo = {};

userInfo.userName = “张三”;

userInfo.userAge = 13;

PlatUtil.setData(“userinfo”,userInfo);

## 如何读取缓存的数据对象？

**方法名称:**

Var cacheObj = PlatUtil.getData(key);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| key | 缓存对象的KEY |

**调用例子:**

var userInfo = PlatUtil.getData(“userInfo”);

## 如何移除缓存的数据对象？

**方法名称:**

PlatUtil. removeData(key);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| key | 缓存对象的KEY |

**调用例子:**

PlatUtil. removeData (“userInfo”);

## 判断数组是否包含某个值

**方法名称:**

PlatUtil. IsContain(array,targetValue);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| Array | 数组对象 |
| targetValue | 目标值对象 |

**调用例子:**

var myArray = new Array();

myArray.push(“zhangsan”);

myArray.push(“lisi”);

myArray.push(“wangwu”);

boolean result = PlatUtil. IsContain (myArray,”zhangsan”);

## 如何验证放在TAB容器的所有表单是否合法？

**方法名称:**

PlatUtil.validBootstrapTab(boottabId);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| boottabId | TAB对象的ID |

**调用例子:**

Var result= PlatUtil.validBootstrapTab(boottabId)

## 获取可编辑表格所勾选的记录数组

**方法名称:**

PlatUtil. getEditTableOperMulRecord (tableId);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| tableId | 可编辑表格的ID |

**调用例子:**

var selectDatas = PlatUtil.getEditTableOperMulRecord(tableId);

## 获取可编辑表格所有记录JSON字符串

**方法名称:**

PlatUtil. getEditTableAllRecordJson (tableId);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| tableId | 可编辑表格的ID |

**调用例子:**

var recordJson = PlatUtil.getEditTableAllRecordJson("可编辑表格ID");

## 获取可编辑表格所有记录数组

**方法名称:**

PlatUtil. getEditTableAllRecord (tableId);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| tableId | 可编辑表格的ID |

**调用例子:**

var recordArray = PlatUtil.getEditTableAllRecord("可编辑表格ID");

## 获取JqGrid表格所选择的记录数组

**方法名称:**

PlatUtil. getTableOperMulRecord (tableId);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| tableId | 表格的ID |

**调用例子:**

Var recordArray = PlatUtil. getTableOperMulRecord (tableId);

## 改变单选框的值

**方法名称:**

PlatUtil. setCheckRadio (controlName, value);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| controlName | 控件名称 |
| Value | 改变的值 |

**调用例子:**

PlatUtil. setCheckRadio (“控件的命名”, value);

## 获取单选框的值

**方法名称:**

PlatUtil. getCheckRadioTagValue (controlName,tagName);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| controlName | 控件名称 |
| tagName | 标签名称 |

**调用例子:**

VarTEMPLATE\_CONTYPE= PlatUtil.getCheckRadioTagValue("TEMPLATE\_CONTYPE","VALUE");

## 获取复选框的值

**方法名称:**

PlatUtil. getCheckBoxValues (checkBoxName, parentSelector);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| checkBoxName | 控件名称 |
| parentSelector | 父级选择器，一般为空 |

**调用例子:**

var checkboxValue = PlatUtil.getCheckBoxValues("你的复选框名称")

## 修改一个控件的权限(隐藏、只读、可写)

**方法名称:**

PlatUtil. changeUICompAuth (authValue,compName);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| authValue | 权限值(write,hidden,readonly |
| compName | 控件命名 |

**调用例子:**

PlatUtil. changeUICompAuth (“readonly”,”username”);

## 重新加载复选框、单选框、下拉框

**方法名称:**

PlatUtil. reloadCheckBox (name,config);

PlatUtil. reloadRadio (name,config);

PlatUtil. reloadSelect (name,config);

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| name | 控件命名 |
| config | 查询配置参数 |

**调用例子:**

PlatUtil.reloadSelect("TABLE\_NAME",{

dyna\_param:"1"

});

## 如何改变下拉框的值

**方法名称:**

PlatUtil. changeSelect2Val(selector,newVal)

**参数说明:**

|  |  |
| --- | --- |
| 参数名称 | 参数说明 |
| selector | 获取下拉框Jquery选择器 |
| newVal | 新的值 |

**调用例子:**

PlatUtil. changeSelect2Val(“#username”,”1”);

# 后端相关API

## 如何获取后台登录用户信息

/\*\*

\* 获取后台当前登录用户对象

\* 除了拥有数据库中所设计的字段属性,还拥有以下字段

\* RESCODESET:被授权的资源编码集合,是一个Set

\* RESCODES:被授权的资源编码字符串,是一个字符串

\* GRANTURLS:被授权的访问URL,是一个Set

\* ROLECODESET:被授权的角色编码集合,是一个Set

\* FLOWDEFTYPEIDS:被授权的流程定义和类别ID

\* **@return**

\*/

PlatAppUtil.getBackPlatLoginUser()

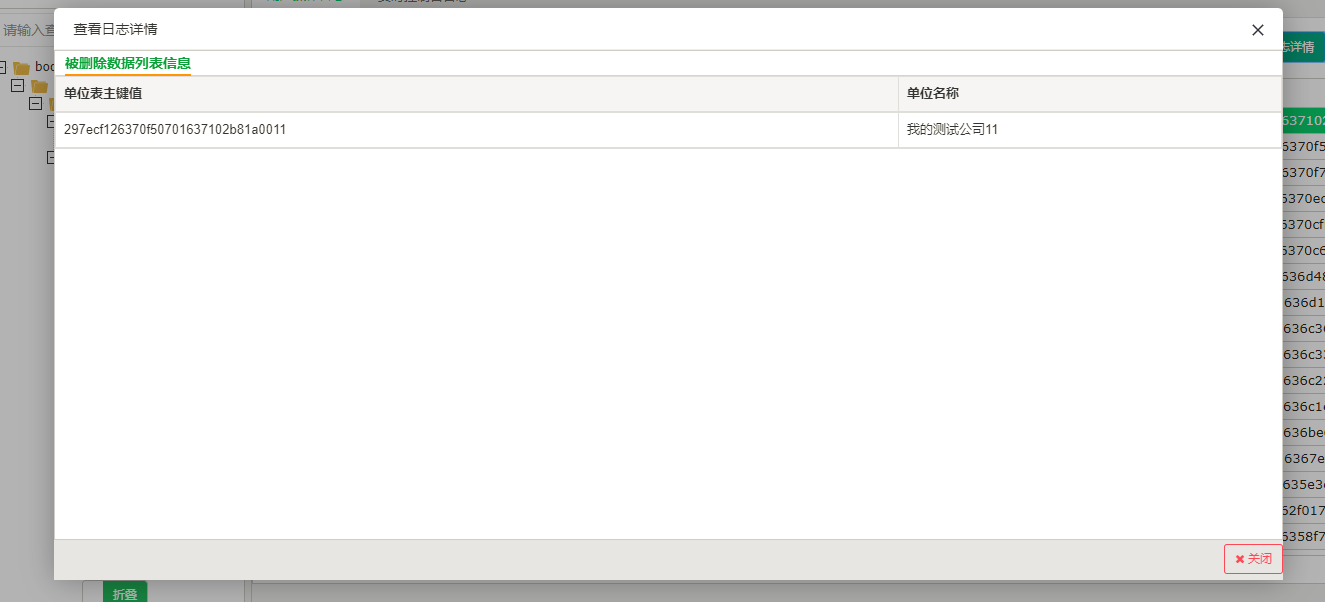
## 如何用代码跳转到手写的界面

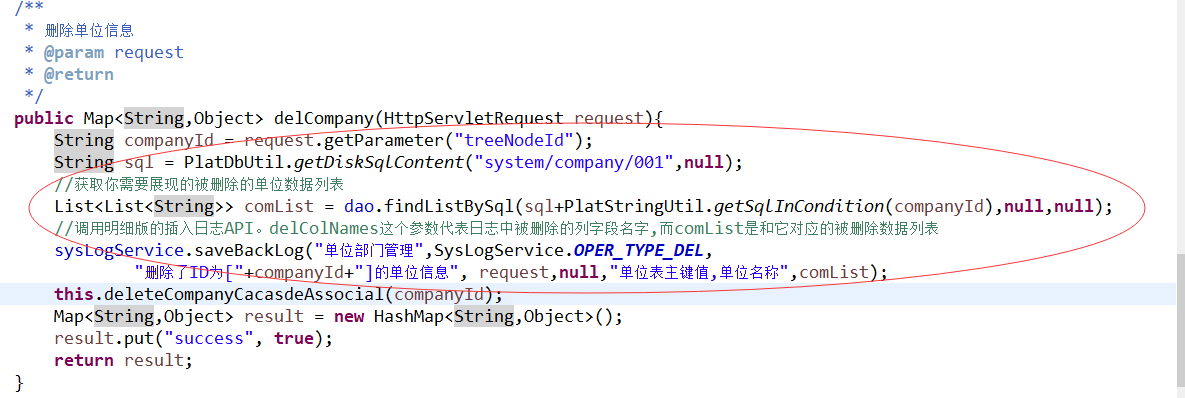
https://my.oschina.net/u/2356625/blog/1648988

## 如何生成有详情内容的日志?

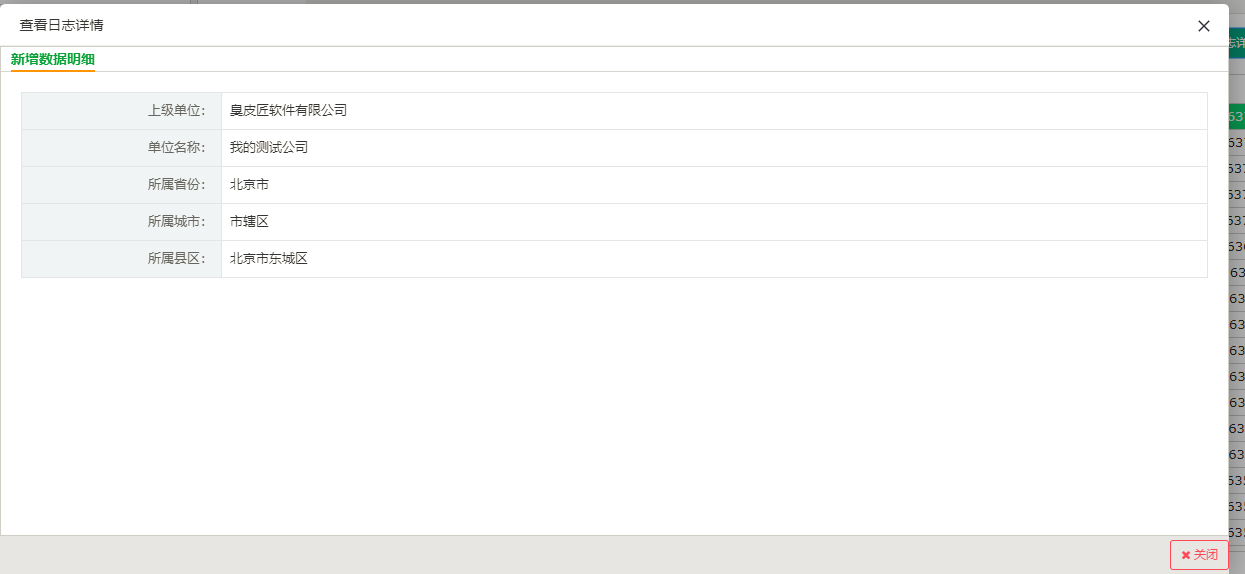
系统生成的代码，调用的是简易版的日志插入API,无法查看明细的日志内容，如果我们想要知道用户具体变更了哪些数据字段，需要调用明细版的日志API，以单位的增删改操作为例，说明下调用过程。

删除单位的时候，我们想要生成以下的明细日志内容:

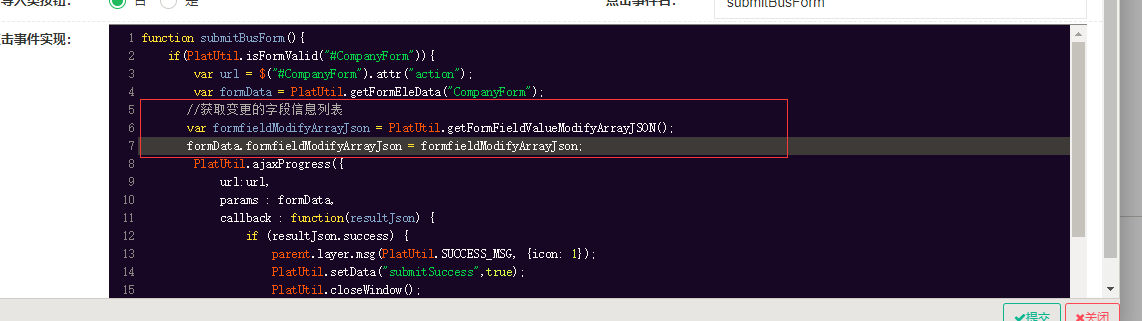




新增单位的时候，我们响应看到以下的日志明细内容:



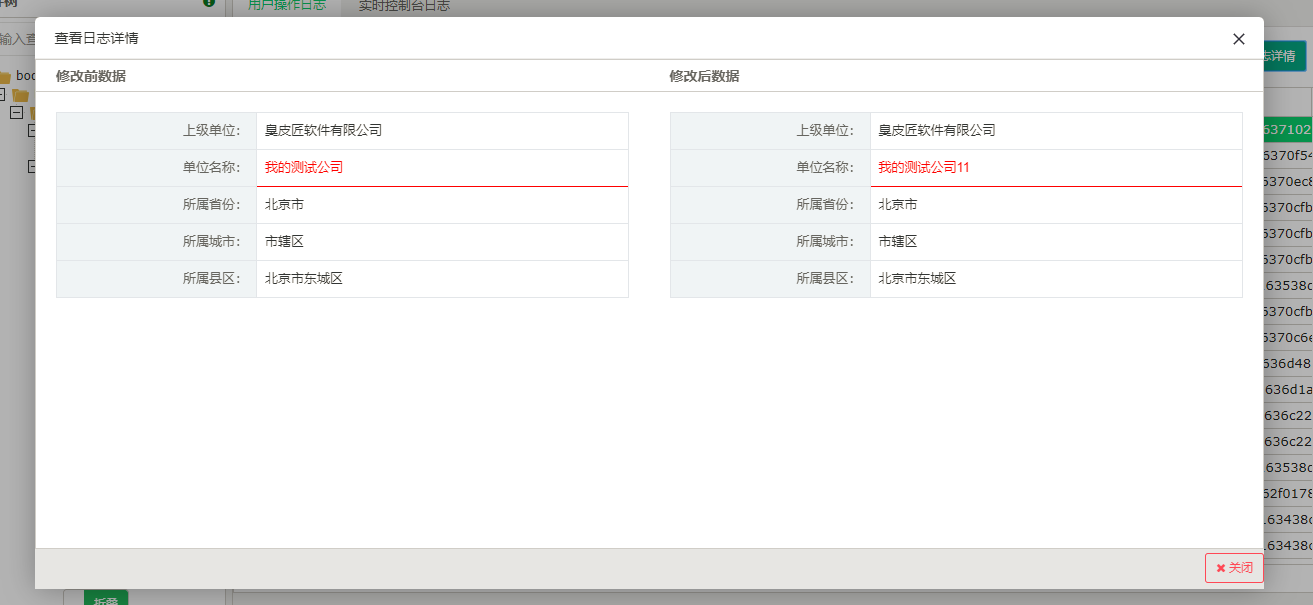
第一步先要在前端调用获取字段变更的JSON字符串



第二步在后端，解析该字符串，插入字段变更明细的API

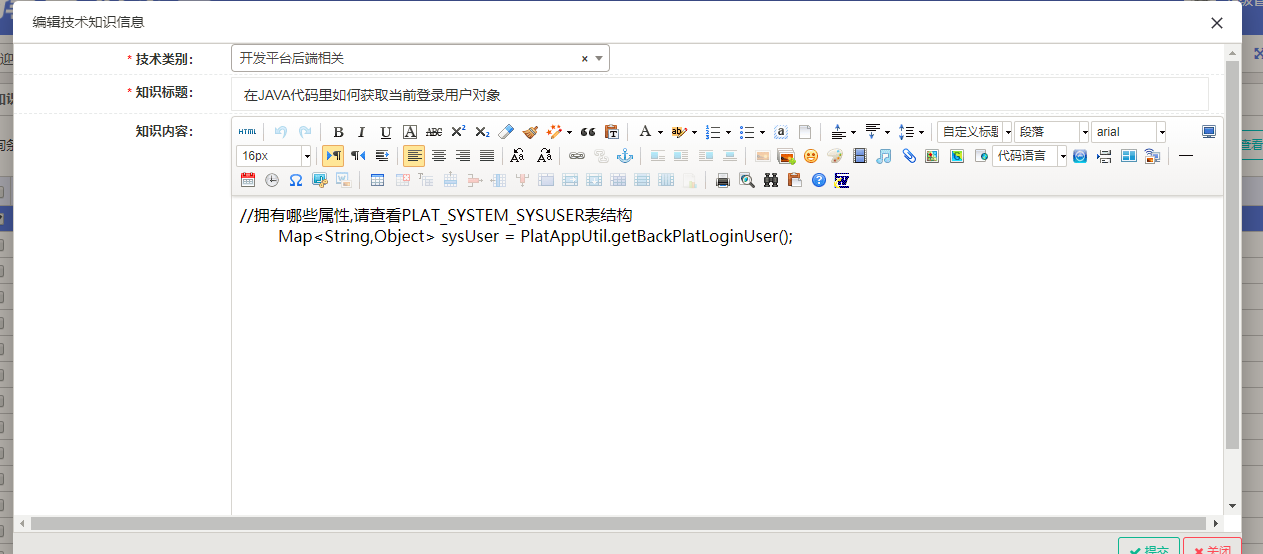


如果想要完成下图修改单位的日志展现,请使用和新增一样的做法。

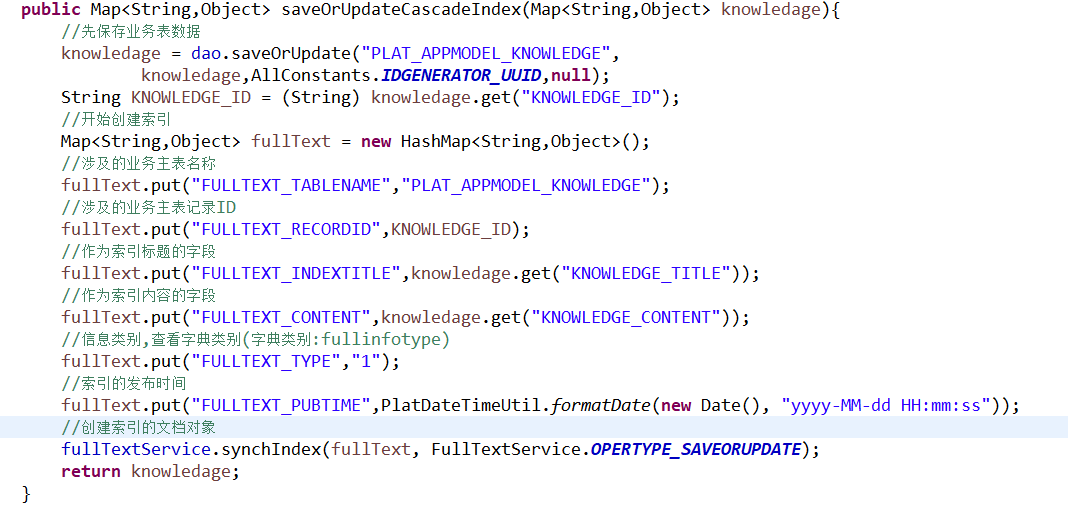


## 简易版Lucene全文检索功能集成说明

下面以知识库管理集成全文检索功能为例:



该模块知识库的新增或者修改会调用lucene的创建索引功能

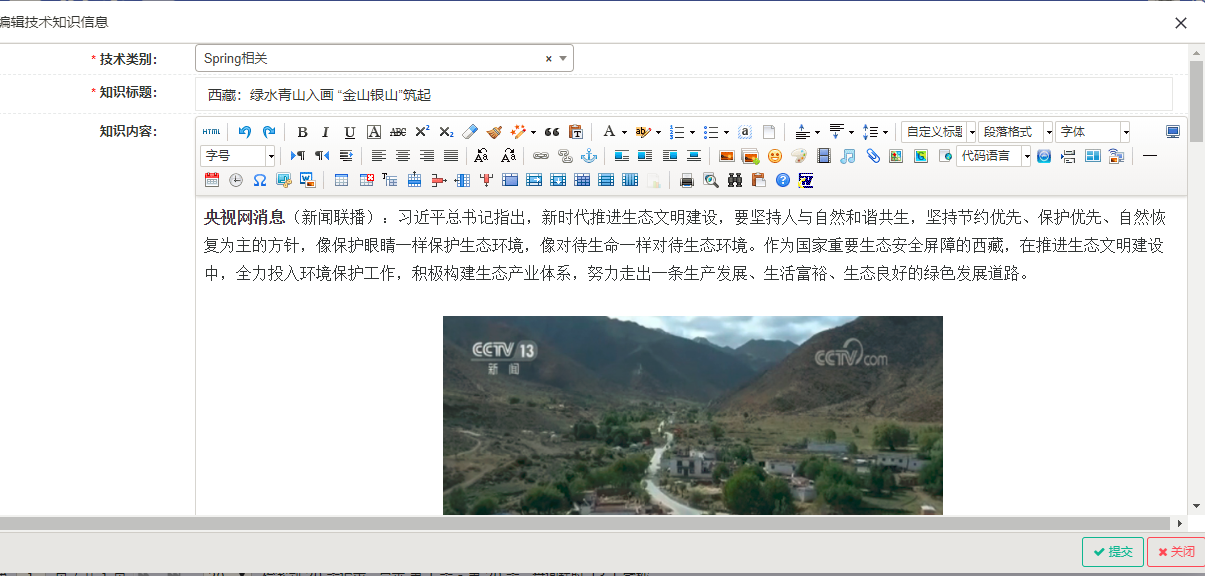


其中信息类别字段配置在字典当中，



创建新的信息类别的数据，记得需要配置文章被搜索后的访问地址

我们发布如下的文章到知识库，并且同步到全文索引库。

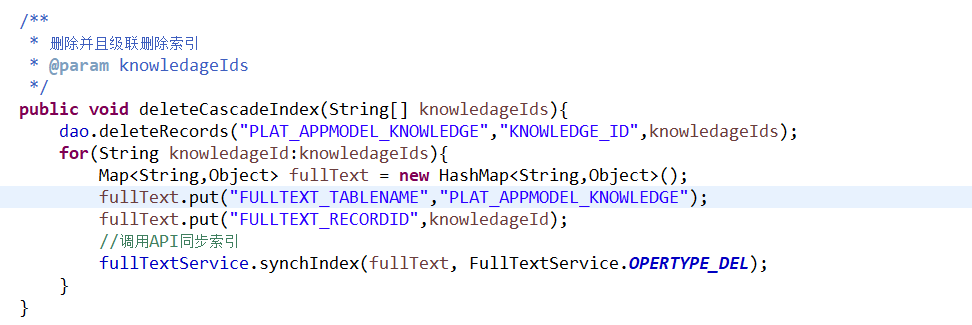




如下是测试搜索结果



如果删除知识库的时候，也会调用全文检索API删除索引



## 与第三方系统单点登录快速集成说明

第一步：先拷贝docs目录下的platsso-1.0.jar文件到第三方系统的lib下。

第二步：配置第三方系统的web.xml，加入以下单点登录的过滤器，注意该过滤器的优先级需要是最高的。

<!-- 开始集成单点登录插件 -->

<filter>

<filter-name>ssoClientFilter</filter-name>

<filter-class>com.stooges.platform.sso.SSOClientFilter</filter-class>

<!-- 定义不被拦截的URL,多个用分号分割,如果不拦截根访问路径,那么设置blank -->

<init-param>

<param-name>anonurls</param-name>

<param-value>blank;webSiteController;userInfoController;</param-value>

</init-param>

<!-- 定义SSO服务器地址 -->

<init-param>

<param-name>ssoserverurl</param-name>

<param-value>http://localhost:80/</param-value>

</init-param>

**<!-- 定义SSO登录成功后需要回调的接口 -->**

<init-param>

<param-name>invokeservice</param-name>

<param-value>net.evecom.platform.system.service.impl.SysUserServiceImpl.addUserToSession</param-value>

</init-param>

</filter>

<filter-mapping>

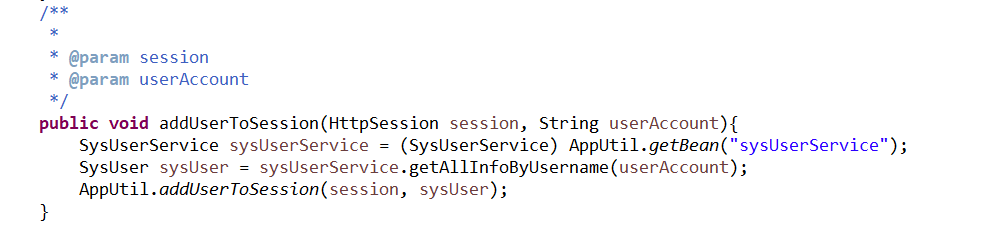
<filter-name>ssoClientFilter</filter-name>

<url-pattern>\*.do</url-pattern>

</filter-mapping>

<!-- 结束集成单点登录插件 -->

第三步:实现单点登录成功后的回调接口，因为单点登录服务器登录成功后只会往session写入当前登录用户的账号，第三方业务系统有可能需要写入其它数据，以下为参考实现



第四步:第三方业务系统，需要改写登出的URL实现，以便实现和单点登录服务器共同销毁会话。

