

高级应用之六：自定义Module

1、简介

Smartbi提供了一个特定的接口smartbi.framework.IModule，实现这个接口，并在扩展包的配置文件applicationContext.xml中将module注册到Framework和RMIModule（见[扩展插件开发基础（内含目录及配置文件介绍）](#)），这样smartbi在系统启动时就会调用自定义module的activate方法执行一些初始化操作，并且这个module中的共有方法都可以在前端异步调用。通常有以下几个主要场景需要自定义module：

1，前端需要调用后端方法执行某个特定的逻辑，就是前后端有异步交互需求时，需要自定义module，在module里实现方法，前端使用util.remoteInvoke(className, methodName, paramArray, callback, that, noLookup)调用module的方法，详细见下面示例：

2，需要编写[升级类](#)（或者系统启动成功后执行的[升级类](#)）时也需要先写个module；

3，需要在扩展包中执行一些系统启动初始化的操作；

需要知道的特性：可以在module中直接引用公共组件，譬如dao（知识库操作）、state（会话状态）、catalogtree（资源树操作）、usermanager（用户管理），常用组件对应的接口类型见[扩展插件开发基础（内含目录及配置文件介绍）](#)里的说明，如下面示例就有引用catalogtree组件（自定义module里面定义了catalogTreeModule属性）。

```
package smartbi.framework;

/**
 *
 * ;
 *
 * 1
 * 2
 */
public interface IModule {
    void activate();//
}
```

文档目录：

- [1、简介](#)
- [2、前端调用module方式](#)
- [3、Module的前后端交互原理](#)
- [4、示例说明](#)
- [5、示例代码下载](#)

2、前端调用module方式

前端js使用util.remoteInvokeEx/remoteInvoke（className, methodName, paramArray, callback, that, headers）方法调用，其中remoteInvokeEx如果同步请求出现异常会自动弹窗提示，参数说明：

className：配置再applicationContext.xml中注册到rmi中的名称，譬如下面示例中就是ExtSample8Service

methodName：要请求module中的哪个方法

paramArray：上面方法接收的参数数组，数组中的第一个对应方法的第一个参数，依次类推

callback：回调函数，请求返回执行，如果不传递此参数代表同步请求

that：callback里的this对象

headers：请求头信息，譬如：json对象，譬如{If-Modified-Since:0}

可执行示例请见[宏代码中执行sql语句](#)。

module调用示例

```
//
var util = jsloader.resolve("freequery.common.util");
//
var ret = util.remoteInvoke("DashboardService", "getParamValueFromDashboard", [this.clientId, paramId]);
if(ret.succeeded) {
    return ret.result;
} else {
    modalWindow.showServerError(ret);
}

//
var ret = util.remoteInvoke("DashboardService", "getParamValueFromDashboard", [this.clientId, paramId],
function(ret){
    if(ret.succeeded){
        var result = ret.result; //getParamValueFromDashboardjson
    }
}, this);
```

3、Module的前后端交互原理

前面介绍了Module的编写、注册及前端如何调用，这里简单介绍里面的原理：

1)、util.remoteInvoke/remoteInvokeEx实际是封装了对RMIServlet的请求，其接收三个主要参数：className、methodName、params，这三个参数就是对应remoteInvoke方法的参数

2)、RMIServlet接收到className等三个参数为什么会调用对应的module，是因为RMIServlet引用了RMIModule中的modules属性，通过提供的className，找到真实对应的module，这也是为什么编写好的module需要在applicationContext.xml注册到RMIModule的原因之一，配置示例见下文。

4、示例说明

打开“http://localhost:8080/smartbi/vision/test/extsample8.html”并点击页面中的按钮，将会看到类似下面的结果：

```
Get Demo Info

===== demonstrate getRootElementInfo =====
All root elements:
1: alias=工具宏,type=MACRO_PACKAGE,id=MACRO_LIBRARIES
2: alias=资源包,type=RESOURCE_PACKS,id=RESOURCE_PACKS
3: alias=安全管理,type=SECURITY_MANAGER,id=SECURITY_MANAGER
4: alias=数据源,type=DATASOURCES,id=DATASOURCES
5: alias=业务主题,type=BUSINESS_THEMES,id=BUSINESS_THEMES
6: alias=函数,type=FUNCTIONS,id=FUNCTIONS
7: alias=公共设置,type=ADVANCED_SECURITY,id=ADVANCED_SECURITY
8: alias=根目录,type=DEFAULT_TREENODE,id=DEFAULT_TREENODE
9: alias=个人目录,type=SELF_TREENODE,id=SELF_ADMIN
10: alias=用户属性,type=USER_PROPERTIES,id=USER_PROPERTIES
11: alias=数据权限,type=ROW_PERMISSION,id=ROW_PERMISSION
12: alias=计划任务,type=SCHEDULETASK,id=SCHEDULETASK
13: alias=公共页面,type=PUBLIC_PAGES,id=PUBLIC_PAGES
14: alias=个人页面,type=SELF_PAGES,id=SELF_PAGES_ADMIN
15: alias=系统选项,type=OPTIONS,id=OPTIONS
Total size: 15.

===== demonstrate getCatalogElementById =====
alias['DEFAULT_TREENODE']=根目录
```

修改Spring声明文件applicationContext.xml 将自定义Module对象extSample8Service配置到扩展插件的spring声明文件中。通过自定义Module引用系统内部模块，实现系统内部方法的调用。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<beans>
    <bean id="rmi" class="smartbi.framework.rmi.RMIModule" factory-method="getInstance">
        <property name="modules">
            <map>
                <entry><key><value>ExtSample8Service</value></key><ref bean="
extSample8Service" /></entry>
            </map>
        </property>
    </bean>
    <bean id="extSample8Service" class="smartbi.ext.sample8.ExtSample8Service" factory-method="
getInstance">
    </bean>
</beans>
```

5、示例代码下载

示例代码下载: [Sample8.rar](#)