如何扩展报表的保存方法

在集成环境中，经常需要在smartbi创建报表后做一些扩展，有以下2种方式实现。

1、创建扩展包

请参考“插件开发框架”，创建自己的扩展包，扩展包写好后，编译打包并参考“扩展包部署”文档部署该扩展包。通过后端扩展或前端文件扩展修改，电子表格资源只能通过后端文件扩展。

后端扩展方法：

java api 请参考：javaapi

1、实现：smartbi.catalogtree.ICatalogTreeListener 接口

示例：

```
package bof.audit.service;

import smartbi.catalogtree.ICatalogElement;
import smartbi.catalogtree.ICatalogTreeListener;

public class ReportChangePostHandler implements ICatalogTreeListener {

        public void onCatalogElementCreated(ICatalogElement element) {
                String type = element.getType();
                // type
                /*
        1.:BUSINESS_VIEW
        2.SQL:TEXT_BUSINESS_VIEW
        3.:PROC_BUSINESS_VIEW
        4.SQLRAWSQL_BUSINESS_VIEW
        5.:SIMPLE_REPORT
        6.COMBINED_QUERY
        7.INSIGHT
        8.: Dashboard
        9.: DashboardMap
        10.SPREADSHEET_REPORT
        11.OLAP_REPORT
        12.portal: PAGE
        13.WEBURL
                14.SMARTBIX_DATASET
                15.SMARTBIX_PAGE
        */
                //TODO
        }

        public void onCatalogElementDeleted(ICatalogElement element) {
                // TODO Auto-generated method stub
                //TODO
        }

        public void onCatalogElementMoved(String nodeId, String originalParentId, String destParentId) {
                // TODO Auto-generated method stub
        }

        public void onCatalogElementDeleting(ICatalogElement element) {
                //TODO
        }

        public void onCatalogElementUpdated(ICatalogElement element) {
                // TODO Auto-generated method stub

        }
}
```

2、注册Listener实现。

module类，在服务器启动的时候注册，activate方法会在服务器启动的时候执行；module类的写法和配置可参考 第六课：高级应用 之六：**自定义Module** 章节。

```
package bof.audit.service;
import java.util.List;
import org.h2.util.New;
import bof.audit.service.ReportChangePostHandler;
import smartbi.catalogtree.CatalogTreeModule;
import smartbi.catalogtree.ICatalogElement;
import smartbi.framework.IFrameworkListener;
import smartbi.framework.IModule;
import smartbi.sdk.service.catalog.CatalogService;
import smartbi.usermanager.local.LocalClientConnector;
public class AuditModule implements IModule, IFrameworkListener {
        private static AuditModule instance;
        private CatalogTreeModule catalogTreeModule;
        public static AuditModule getInstance() {
                if (instance == null)
                        instance = new AuditModule();
                return instance;
        }
        public CatalogTreeModule getCatalogTreeModule() {
                return catalogTreeModule;
        }
        public void setCatalogTreeModule(CatalogTreeModule catalogTreeModule) {
                this.catalogTreeModule = catalogTreeModule;
        }
        private static LocalClientConnector conn;
        protected AuditModule() {
                conn = new LocalClientConnector();
        }
        public void activate() {
                catalogTreeModule.addCatalogTreeListener(new ReportChangePostHandler());
        }
        @Override
        public void afterStartup() {
                // TODO Auto-generated method stub
        }
}
```

```
<?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="framework" class="smartbi.framework.Framework" factory-method="getInstance">
        <property name="modules">
                <map>
                                <entry><key><value>AuditModule</value></key><ref bean="AuditModule" /></entry>
                </map>
        </property>
  </bean>

  <bean id="rmi" class="smartbi.framework.rmi.RMIModule" factory-method="getInstance">
                <property name="modules">
                        <map>
                                <entry><key><value>AuditModule</value></key><ref bean="AuditModule" /></entry>
                        </map>
                </property>
 </bean>
  <bean id="AuditModule" class="bof.audit.service.AuditModule" factory-method="getInstance">
   <property name="catalogTreeModule" ref="catalogtree"/>

  </bean>
</beans>
```

以下说明前端扩展方法：

1、找到报表对象的前端处理文件

- 灵活分析：queryView.js
- 仪表、地图分析：dashboard.js
- 多维分析：OlapQueryView.js
- 透视分析：Insight.js
- 组合分析：CombinedQuery.js
- 可视化查询、sql查询等所有查询：BusinessView.js
- 门户定制：PageWizardAction.js
- 业务主题：BusinessThemeView.js

2、找到报表对象的前端保存方法

- 灵活分析： doSave方法
- 仪表、地图分析： save方法
- 多维分析： saveOlapQuery方法
- 透视分析： saveQuery方法
- 组合分析： saveAsCallback方法，组合分析是基于灵活分析创建的。
- 可视化查询、sql查询等所有查询：save方法
- 门户定制：savePage方法
- 业务主题：elemBtnSave_click_handler方法

3、扩展报表对象的前端保存方法

请参考“第四课：如何修改Smartbi JS文件”，扩展报表保存对象的方法，示例：

```
QueryView.prototype.saveQueryCallback_new = QueryView.prototype.saveQueryCallback;
QueryView.prototype.saveQueryCallback = function(ret, dialog) {
        this.saveQueryCallback_new(ret, dialog);
        var folderId = ret.folderId
        var ret = util.remoteInvokeEx("TestService", "saveAs", [folderId]);
}
```

4、实现报表保存新的后台逻辑

请参考“第六课：高级应用”之六：“**自定义Module**”创建自己的 java 文件，并在其中写好保存扩展的方法。示例：

```
 public class TestModule implements IModule{

        private static final Logger LOG = Logger.getLogger(TestModule.class);
        private static TestModule instance;

        public static TestModule getInstance() {
                if (instance == null)
                        instance = new TestModule();
                return instance;
        }

        public void activate() {
        }

        public void saveAs(String folderId){
                // .....
        }
}
```

5、在步骤4中调用步骤4实现的方法

```
JavaScript
```

```
QueryView.prototype.saveQueryCallback_new = QueryView.prototype.saveQueryCallback;
QueryView.prototype.saveQueryCallback = function(ret, dialog) {
        this.saveQueryCallback_new(ret, dialog);
        var folderId = ret.folderId
        var ret = util.remoteInvokeEx("TestModule", "saveAs", [folderId]);
        if (!ret.succeeded) {
                modalWindow.showServerError(ret);
        }
}
```

示例工程源码

saveQueryCallback_new_前端拓展方法.rar

ExtendedPreservationMethod_后端拓展方法.rar