扩展包开发规范

扩展包最终是作为产品代码的一部分运行的，所以对质量同样需要有高度的要求。扩展包需要遵循基本规范的作用主要有以下几点：

1. 尽量减少对产品原有功能的影响，以尽量避免产品在新功能开发或BUG修复后无需修改扩展包代码即可生效。
2. 保持代码风格统一，充分利用产品基本框架所带来的便利和稳定，同时方便后续代码维护和调整。
3. 规避以往经验中较常出现问题的开发方法和习惯。

扩展包开发规范，主要包括基本的代码规范（遵循通用的代码规范，如命名、代码格式等），以及针对扩展包的规范。基本代码规范遵循常用的代码规范，本文不作赘述，重点列举扩展包的规范。以下规范需严格遵守，否则可能造成严重事故。

1 多语言支持原则

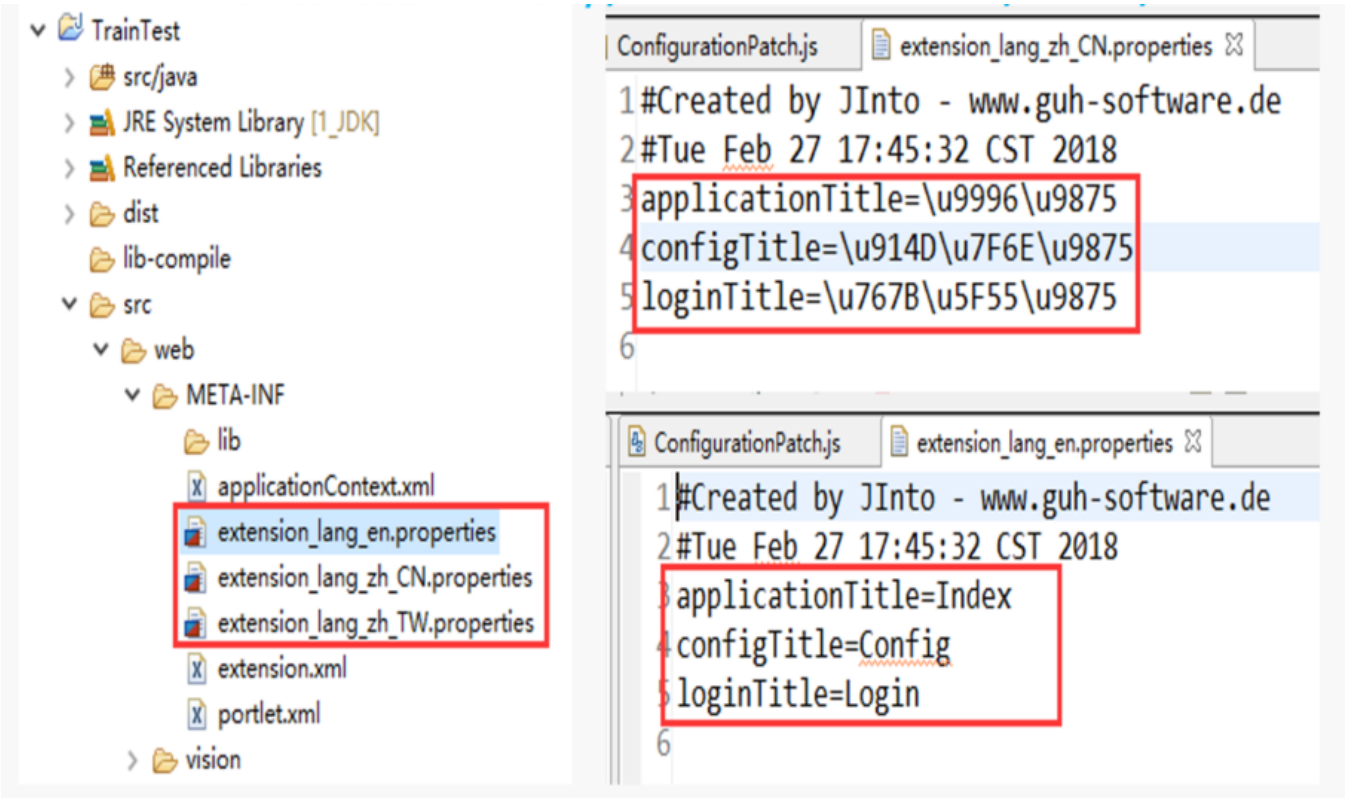
代码中出现中文的地方，需要采用多语言机制处理。

```
if (typeof fn == 'function') {
    var ret = result || modalWindow.ID_UNKNOWN;
    if (ret && ret.succeeded) {
        if (ret.result === true) {
            fn.call(this, isPass);
        } else {
            alert("操作失败，请重试。");
            console.error(ret);
        }
    } else if (ret != "cancel") {
        console.error("unknow return value : " + ret);
    }
```

不规范写法，必须采用多语言处理中文

如何使用多语言机制，参照如下步骤：

步骤一：创建多语言文件（固定文件名），采用key=value的方式设置中英文显示内容。（建议安装多语言Propertys编辑工具，Unicode在线转码：http://tool.chinaz.com/tools/unicode.aspx）

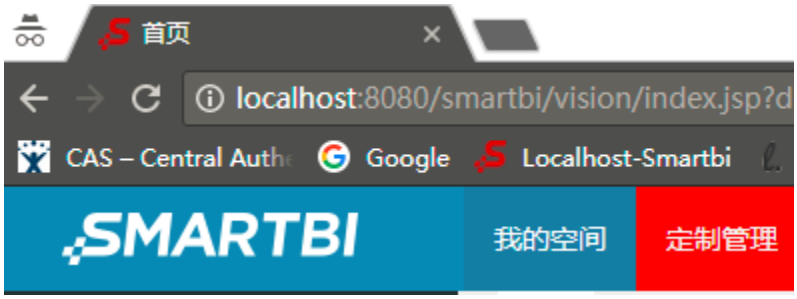


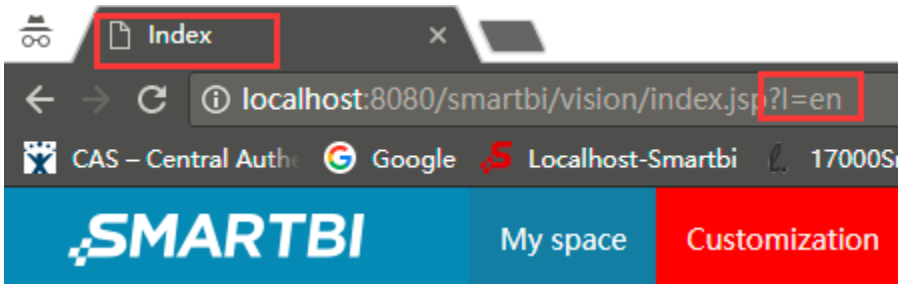
步骤二：

1）前端：在需要动态显示中英文的地方输入多语言key值，效果如下

ConfigurationPatch.js

```
var ConfigurationPatch = {
        loginTitle : "${loginTitle}",//登录页标题
        applicationTitle : "${applicationTitle}",//首页标题
        configTitle : "${configTitle}",//配置页面标题
        extensionPoints : {
            css : [ 'vision/css/patch.css' ],
            SystemConfig : {
                configItem : [ {
```





2）后端：中英文处理 StringUtil.getLanguageValue（可指定需要获取的语言类型，也可不指定，则默认获取当前的语言类型）

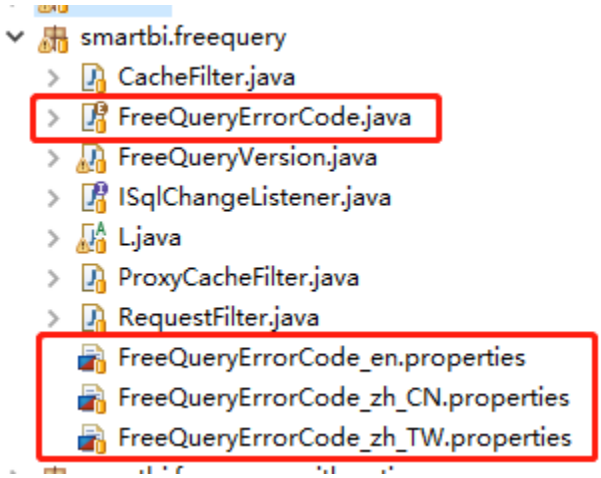
```
StringUtil.getLanguageValue("TEST_MSG");

StringUtil.getLanguageValue("TEST_MSG", CommonConfiguration.getInstance().getLocale());
```

其他：后端错误类型，后端代码中在抛出异常时也要兼容中英文环境，常常会使用枚举类与多语言文件配合实现。

- 创建枚举类，枚举类中的枚举项为中英文文件 中对应的key值（新增或修改枚举项的时候请同步更新与类名同名的properties文件）

```java
/*
 * 新增或修改枚举项的时候请同步更新与类名同名的properties文件
 */
public enum FreeQueryErrorCode {
    BUSINESSVIEW_CLIENT_ERROR,
    BUSINESSVIEW_DATASOURCE_DELETED,
    BUSINESSVIEW_EXCEL_TEMPLATE_NOT_FOUND,
    BUSINESSVIEW_FIELD_IS_NOT_EXIST,
    BUSINESSVIEW_REFERENCE_RECURSIVELY,
    BUSINESSVIEW_RUNTIME_CREATE_FIELD_ERROR,
    BUSINESSVIEW_RUNTIME_CYCLE,
    BUSINESSVIEW_RUNTIME_FIELD_EXSIST,
    BUSINESSVIEW_RUNTIME_FIELD_NOT_EXSIST,
    BUSINESSVIEW_SET_PROP_ERROR,
```

- 创建多语言文件（与枚举类同名+固定后缀）， 多语言文件内容与步骤一所述相同

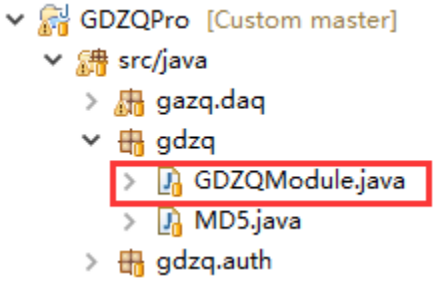


- 使用时直接调用枚举项即可

```java
throw new SmartbiException(
        FreeQueryErrorCode.META_DATASOURCE_NOT_EXIST);
```

2 一个Module类原则

一个扩展包中，原则上只能有一个Module类。



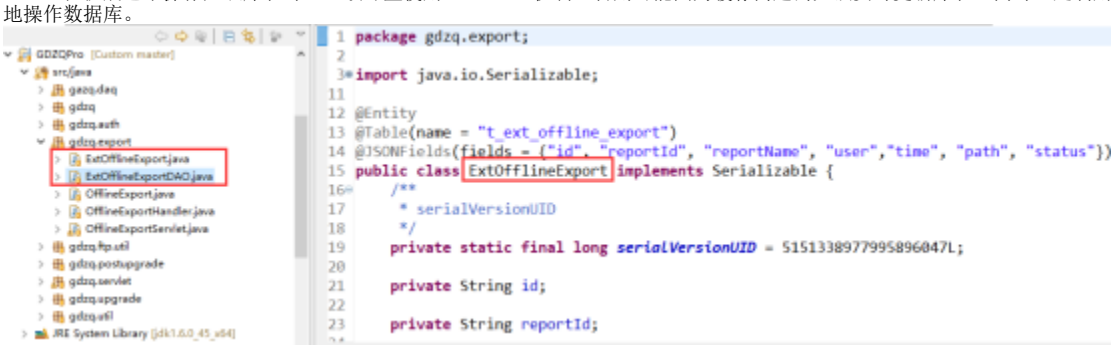
3 升级类检查机制

升级类的逻辑必须非常注意，保证其能在各种场景下正常执行（不同数据库的SQL语法）。尤其是建表及插入数据操作，必须先检查表或记录是否已经存在。否则可能导致服务器无法启动。

```
public class UpgradeTask_0_0_10 extends UpgradeTask {

    static Logger Log = Logger.getLogger(UpgradeTask_0_0_10.class);

    @Override
    public boolean doUpgrade(Connection conn, DBType type) {
        Statement stat = null;
        String tableName = "t_ext_offline_export";
        try {
            if (UpgradeHelper.isTableExists(conn, tableName)) { // 备份/恢复知识库时该表可能已存在
                return true;
            }                                    先检查表是否已经存在
            // 创建"Excel离线下载"表
            //记录ID，报表ID，报表名称，下载用户，下载时间，下载路径
            String sql = UpgradeHelper.getCreateSQL(type, tableName,
                    new String[] { "c_id","c_reportid", "c_reportname", "c_reporttype", "c
```

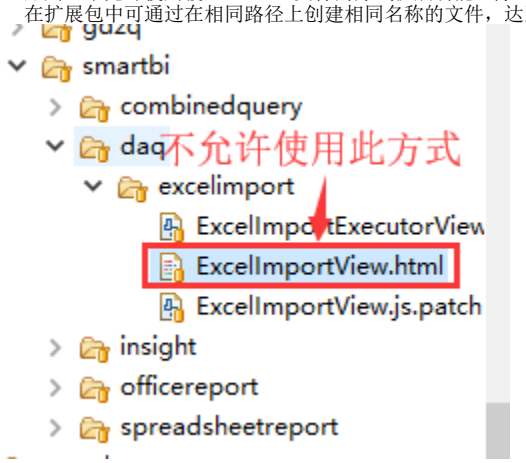
4 知识库实体类原则

在扩展包中操作知识库表时，必须尽量使用hibernate机制。否则可能因为缓存问题而无法实时更新库表。同时也是利用hibernate的优势方便、安全地操作数据库。



5 禁止覆盖Smartbi文件

原则上不允许使用覆盖Smartbi文件的方式扩展功能（除img图片资源）。
在扩展包中可通过在相同路径上创建相同名称的文件，达到覆盖Smartbi的war包文件的目的，在扩展包开发过程中，原则上不允许使用。



在实际开发中，需要尽可能想办法以不覆盖文件的方式达到扩展功能的目的。如需要在html文件中增加元素，可以通过在js中扩展方法，动态创建所需元素。
示例：在导出界面增加离线导出按钮


```
ExportTxtDialog.prototype.init_old180614 = ExportTxtDialog.prototype.init;
ExportTxtDialog.prototype.init = function(parent,data,fn,obj){
    this.init_old180614(parent, data, fn, obj);  先执行原方法加载界面
    //新增离线导出按钮
    var dialogButtonArea = domutils.findElementByClassName(this.btnButtonArea,"dialog-but
    this.offlineExportBtn = document.createElement("input");  创建自定义的离线导出按钮
    this.offlineExportBtn.setAttribute("type", "button");
    this.offlineExportBtn.setAttribute("class", "barbtn");
    this.offlineExportBtn.value = "${OfflineExportButtonValue}(F)";
    this.offlineExportBtn.setAttribute("accesskey", "F");
    dialogButtonArea.insertBefore(this.offlineExportBtn, this.btnOK);
    this.btnOK.value = "${DirectExportButtonValue}(O)";   添加自定义按钮到界面上
    // 获取系统选项中配置的直接导出最大行数限制
    this.overflowTips = "超出最大导出行数!";
    var result = util.remoteInvoke("ConfigClientService", "getSystemConfig", ["EXPORT_ROW
    if (result && result.succeeded && result.result) {
        var config = result.result;
        this.directExportRows = parseInt(config.value.split(',')[0]);
        this.offlineExportRows = parseInt(config.value.split(',')[1]);
        this.overflowTips = config.longValue;
    }
    this.addListener(this.offlineExportBtn, "click", this.offlineExport, this);
```

6 禁止覆盖JS方法

原则上不允许直接覆盖js文件中的方法。
示例：在透视分析中字段菜单中新增一个自定义计算函数
原始做法：直接重写整个菜单初始化方法，在同期值菜单下增加子菜单项：交易日环比。

```
FieldsSpace.prototype.initFuncMenu = function() {
    var PopupMenu = jsloader.resolve("freequery.menu.PopupMenu");
    this.funcPopupMenu = new PopupMenu(document.body, this);
    this.measureCalculationMenu = this.funcPopupMenu.createMenuItem("${TimeCalculation}");
    var corrMenu = this.measureCalculationMenu.createMenuItem("${CorrespondingPeriod}");
    ... ...
    //EPPR-14950 begin edit by qiushitong 2017-12-11
    //
    corrMenu.createMenuItem("", "PreValueDealCompare", null);
    //EPPR-14950 end edit by qiushitong 2017-12-11
    ... ...
    var removeFieldMenu = this.funcPopupMenu.createMenuItem("${Delete}", "removeField", null);
}
```

直接重写整个方法的方式原则不允许使用，因此需要认真理解整个方法的代码逻辑，考虑在方法执行之前或执行之后添加代码达到自己的目的。
优化过的实现方式：

```
FieldsSpace.prototype.initFuncMenu_old180720 = FieldsSpace.prototype.initFuncMenu;
FieldsSpace.prototype.initFuncMenu = function() {
    this.initFuncMenu_old180720();
    if (this.measureCalculationMenu && this.measureCalculationMenu.menuItems) {
        var preMenu = this.measureCalculationMenu.menuItems.menuItems[1]; // get previous menu
        if (preMenu && preMenu.createMenuItem) {
            preMenu.createMenuItem("", "PreValueDealCompare", null);
        }
    }
}
```

7 避免导入资源覆盖原有资源

在使用PostUpgrade升级类导入资源时，需要仔细思考名称及路径的命名，避免出现覆盖目标系统资源或无法导入的情况。另外，不允许在升级类中导入知识库。

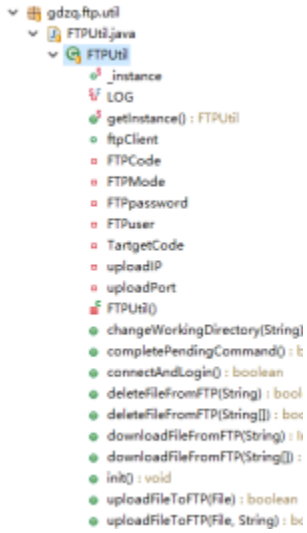
8 灵活配置原则

代码中不应该存在写死的配置内容。如果扩展包代码中用到第三方的配置信息或者约定的一些固定内容、操作方式，均需要采用系统选项的方式支持客户进行灵活配置。比如WebService的URL链接，第三方系统的登录信息，提示框内容，系统编码格式，某个功能是否启用等等。

```
ISystemConfigService config = FreeQueryModule.getInstance().getSystemConfigService();
// 系统选项设置的FTP地址
uploadIP = config.getValue("SYSTEM_CONFIG_UploadIP");
uploadPort = config.getValue("SYSTEM_CONFIG_UploadPort");
FTPuser = config.getValue("SYSTEM_CONFIG_UploadUser");
FTPpassword = config.getValue("SYSTEM_CONFIG_UploadPassword");
// 用户设置的FTP服务器模式
FTPMode = config.getValue("SYSTEM_CONFIG_FTPMode");
FTPCode = config.getValue("SYSTEM_CONFIG_FTPCode");
if (StringUtil.isNullOrEmpty(FTPCode)) {
    FTPCode = System.getProperty("sun.jnu.encoding");
    LOG.warn("Currently not set FTP code, then using system default code: " + FTPCode);
}
```

9 功能/方法提炼

出现较多重复代码或重复逻辑时，需要考虑提炼工具方法或工具类。（注：代码重复率也是衡量一个开发人员代码质量的一个重要标准）比如：代码中多个地方使用到FTP文件上传下载的代码逻辑，就需要将FTP文件上传下载的逻辑进行封装，提炼成工具类使用。



10 切记关闭连接池

当使用到数据库连接时，尤其注意要关闭连接池，否则极易导致客户系统连接池溢出的严重问题。产品提供了smartbi.util.DbUtil工具类，可以简便关闭连接池。

```
public String select(String id) {
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    try {
        con = ConnectionPool.getInstance().getConnection("DS.SYSTEM知识库");
        String selectsql = "select * from t_ext_exportrule where e_id=?";
        pstmt = con.prepareStatement(selectsql);
        pstmt.setString(1, id);
        rs = pstmt.executeQuery();
        while (rs.next()) {
            return  rs.getString("e_setting");
        }
    } catch (Exception e) {
        Log.error("寻找导出规则出错" + e.getMessage());
    } finally {
        DbUtil.closeDBObject(rs, pstmt, con);
    }
    return "";
```

11 升级类建表需要添加知识库对象

新增加的扩展插件中如果需要往知识库中增加库表，必须添加对应的知识库对象，这样**备份知识库**时候才能将新建的表和表信息备份。

具体建立升级类和知识库对象请看创建知识库对象和知识库升级。

当使用空库恢复知识库时候，项目启动过程是。

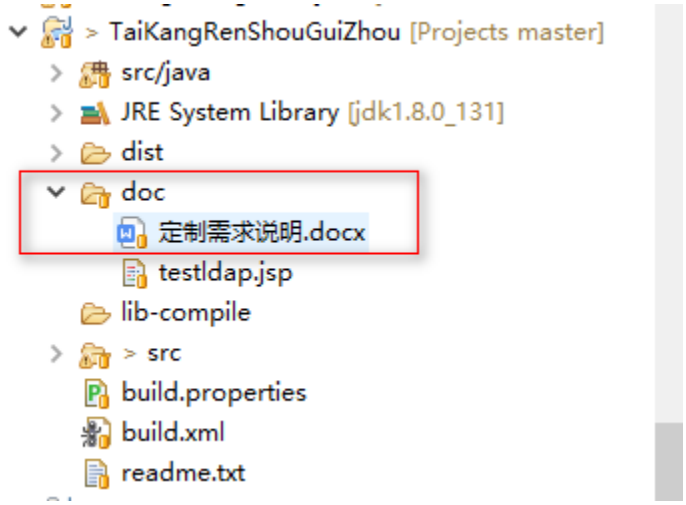
1、先启动项目，然后加载扩展包。这时候扩展包里面用于建表的升级类会在数据库建立一张空表。

2、当项目启动完全时候，点击恢复知识库，这时候项目会先删除掉有**注册知识库对象的表**，再新建这张表，然后将备份的数据恢复。

注意：没有注册知识库对象的表里面的数据是不会被备份的。所以建表升级类必须注册知识库对象。切记。

12 扩展包记录新增定制的功能

每做一项新的定制需求，需要在对应的扩展包下的word文档里记录该需求的需求说明、回复给客户的定制说明、以及定制的开发人员，如果没有word文档，开发人员在扩展包下新建即可。这份文档需要迭代更新到git上面，以保证它最新的动态。如图所示：



3、EPPR-26651【泰康-长护】移动端报表禁止缩放

3-1、需求说明：

缩放功能关闭，目前 H5 的页面可以随便放大和缩小，移动端缩放的话效果太难看了，所以希望可以锁住页面不可以缩放。

3-2、定制说明

新建宏资源包，在里面选中需要控制缩放的报表。

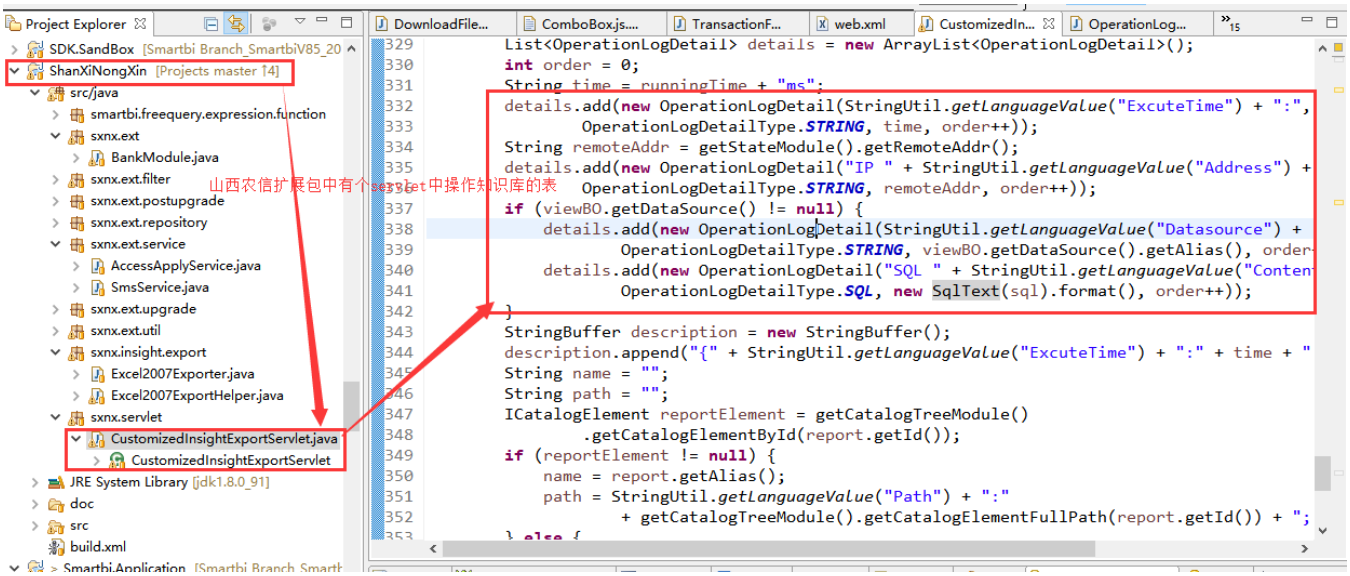
3-3、开发人员

祝宇森

13 扩展包自定义Servlet有操作知识库表的必须过TransactionFilter

若扩展包开发过程中使用到Servlet，若Servlet里面有使用注册的知识库操作类的（例如：使用知识库表的DAO类操作知识库表），则该自定义的Servlet一定要过TransactionFilter这个过滤器把事务管理起来，否则会导致知识库的连接池满，从而使得系统卡死、服务挂起、宕机等严重的问题。

之前出现过的一个例子，山西农信项目，如下图：（因为下面这个自定义的Servlet有操作知识库，但是又没有经过TransactionFilter把事务管理起来，导致服务器没运行一段时间就宕机，系统慢或者卡死）



修改的方法：在extension.xml配置文件中加上Servlet的过滤器配置即可（如下图）

extension.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE extension SYSTEM "extension.dtd">
<extension name="ShanXiNongXin" alias="ShanXiNongXin" desc="" priority="100" version="1.0" depe

    <enable-jsp-processor>1</enable-jsp-processor>
    <filter>
        <filter-name>TransactionFilter</filter-name>
        <filter-class>smartbi.framework.rmi.TransactionFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>TransactionFilter</filter-name>
        <url-pattern>/vision/CustomizedInsightExportServlet</url-pattern>
    </filter-mapping>

    <servlet>
        <servlet-name>CustomizedInsightExportServlet</servlet-name>
        <servlet-class>sxnx.servlet.CustomizedInsightExportServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CustomizedInsightExportServlet</servlet-name>
        <url-pattern>/vision/CustomizedInsightExportServlet</url-pattern>
    </servlet-mapping>
```