报表导出增加主管授权控制

- 1、需求背景
- 2、实现方案
- 3、部署扩展包
  - 3.1 加载扩展包
  - 3.2 扩展包主要代码
    - 3.2.1 ExportModule类
    - 3.2.2 ExportFilter类
- 4、相关资源（EPPR-8748）

1、需求背景

由于客户有些报表数据敏感，为了不让用户随便导出报表，故提出需求：要求用户在导出报表的时候，需有导出权限的主管来输入用户名密码来授权导出。需要满足：

a、并不是所有报表都需要授权导出，希望提供界面给用户配置

b、哪个主管可以授权导出希望可以配置

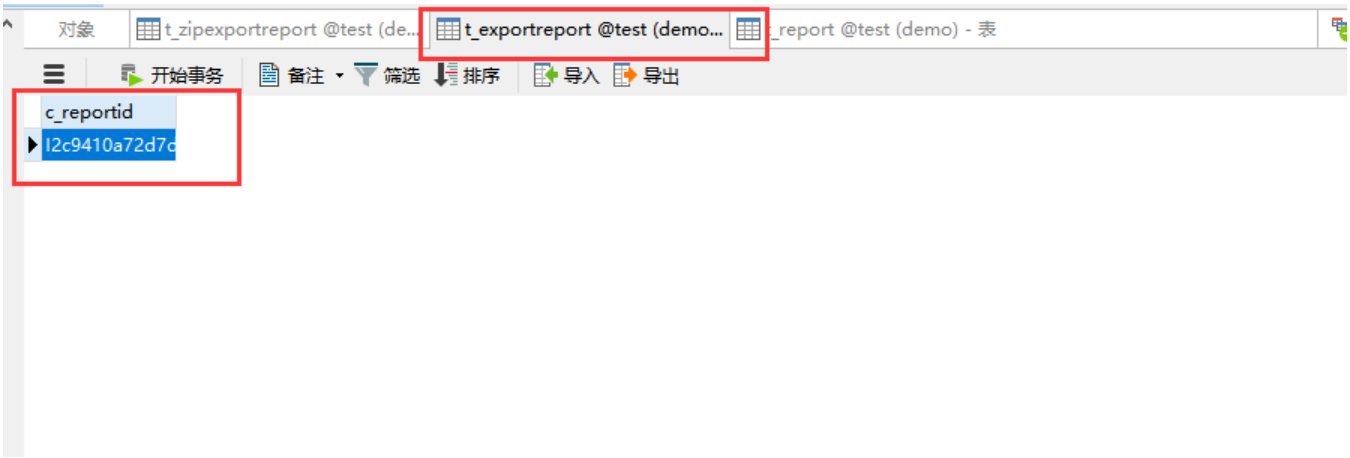
注意：本示例来源于实际项目，原始版本是V6.1，可作为参考性质，也许是不能运行的。

本案例同样适用于v7。

测试的时候请自行将系统知识库中的t_exportreport和t_report两张表进行id关联。具体关联请看后面实现方案

2、实现方案

a、通过升级类增加一张维表t_exportreport，字段c_reportid，请见知识库升级。并关联两张表

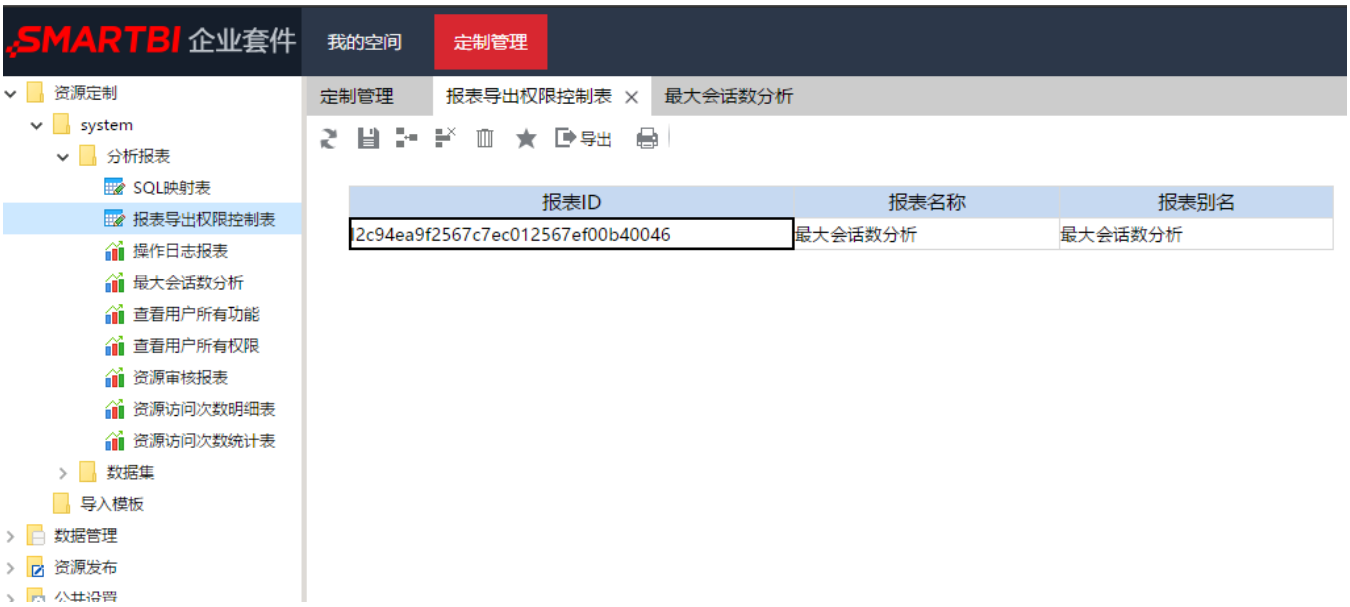


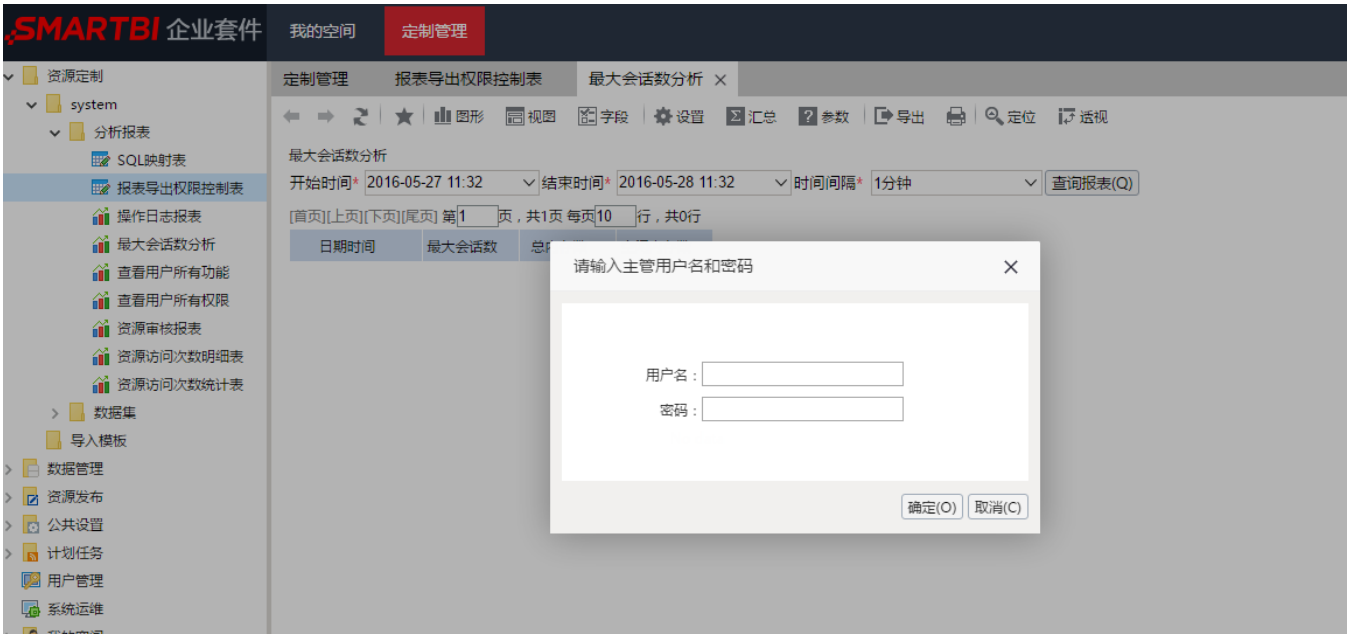
b、通过升级类自动导入一个电子表格回写表"报表导出权限控制表"，用来添加哪些需要导出权限控制的报表，请见自动导入资源

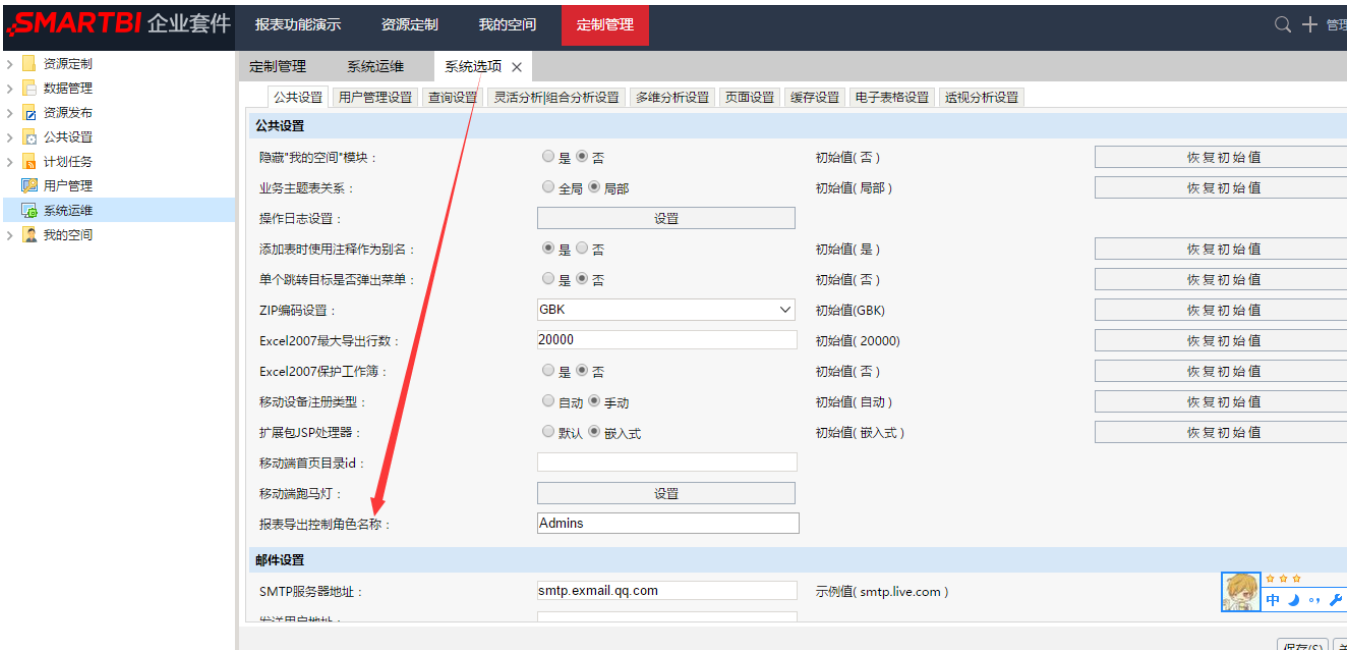
c、添加主管授权时的输入用户名密码界面

d、在系统选项配置导出权限的角色，请见新增系统选项

e、使用过滤器拦截系统的导出，给需要导出授权的报表增加主管授权的逻辑，授权通过才继续导出，过滤器配置请见扩展插件开发基础（内含目录及配置文件介绍）中的插件声明文件extension.xml。


SMARTBI 企业套件
我的空间
定制管理
资源定制
system
分析报表
SQL映射表
报表导出权限控制表
操作日志报表
最大会话数分析
查看用户所有功能
查看用户所有权限
资源审核报表
资源访问次数明细表
资源访问次数统计表
数据集
导入模板
数据管理
资源发布
定制管理
报表导出权限控制表
最大会话数分析
导出
报表ID
报表名称
报表别名
2c94ea9f2567c7ec012567ef00b40046
最大会话数分析
最大会话数分析


SMARTBI 企业套件
我的空间
定制管理
资源定制
system
分析报表
SQL映射表
报表导出权限控制表
操作日志报表
最大会话数分析
查看用户所有功能
查看用户所有权限
资源审核报表
资源访问次数明细表
资源访问次数统计表
数据集
导入模板
数据管理
资源发布
公共设置
计划任务
用户管理
系统运维
定制管理
报表导出权限控制表
最大会话数分析
图形
视图
字段
设置
汇总
参数
导出
定位
透视
最大会话数分析
开始时间*
2016-05-27 11:32
结束时间*
2016-05-28 11:32
时间间隔*
1分钟
查询报表(Q)
[首页][上页][下页][尾页] 第1页，共1页 每页10行，共0行
日期时间
最大会话数
请输入主管用户名和密码
用户名：
密码：
确定(O)
取消(C)



3、部署扩展包

3.1 加载扩展包

exportreportcontrol.ext，扩展包部署见扩展包部署。

3.2 扩展包主要代码

3.2.1 ExportModule类

增加主管授权的验证方法，判断指定报表是否需要授权才能导出。

ExportModule类

```
package cn.com.smartbi;


import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;


import org.apache.log4j.Logger;


import smartbi.SmartbiException;
import smartbi.config.ISystemConfig;
import smartbi.connectionpool.ConnectionPool;
import smartbi.framework.IModule;
import smartbi.freequery.FreeQueryErrorCode;
import smartbi.freequery.client.config.ConfigClientService;
import smartbi.olap.OlapErrorCode;
import smartbi.olap.OlapQueryService;
import smartbi.olap.query.OlapQueryBO;
import smartbi.state.IStateModule;
import smartbi.usermanager.IUserManagerModule;
import smartbi.usermanager.Role;
```

```
import smartbi.usermanager.User;
import smartbi.usermanager.UserBO;


public class ExportModule implements IModule{
 private static final Logger LOG = Logger.getLogger(ExportModule.class);
 private static ExportModule instance;

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


 /**
  *
  */
 private IUserManagerModule userManagerModule;

 public IUserManagerModule getUserManagerModule() {
  return userManagerModule;
 }


 public void setUserManagerModule(IUserManagerModule userManagerModule) {
  this.userManagerModule = userManagerModule;
 }

 /**
  *
  */
 private IStateModule stateModule;

 public IStateModule getStateModule() {
  return stateModule;
 }


 public void setStateModule(IStateModule stateModule) {
  this.stateModule = stateModule;
 }

 public boolean isContainRole(String username, String password){
  User user = (User) userManagerModule.getUserByName(username);
  if(user==null){
   throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR).setDetail("");
  }else{
   String currentUsername = userManagerModule.getCurrentUser().getName();
   if(currentUsername.equals(user.getName())){
    throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR).setDetail("");
   }
   String pwd = user.getPassword();
   UserBO ub = new UserBO(user);
   ub.setPasswordInner(password);
   String MD5Password = ub.getPassword();
   if(pwd.equals(MD5Password)){
    try{
     //String roleName = getValue("exportRoleName");
     String roleName = "";  //
     ConfigClientService cs = ConfigClientService.getInstance();
     ISystemConfig scf = cs.getSystemConfig("ExportRoleName");
     if(scf != null){
      roleName = scf.getValue();
     }
     Role role = (Role) userManagerModule.getRoleByName(roleName);
     if(role==null){
      throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR).setDetail("");
     }else{
      List<Role> roleList = user.getAssignedRoles();
      for(Role rl : roleList){
```

```
      if (rl.getId().equals(role.getId()))
       return true;
      }
      return false;
     }
    }catch(Exception e){
     throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR,e);
    }

   }else{
    throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR).setDetail("");
   }
  }
 }

 //
 public boolean isInReport(String resid){
  String sql = "select c_reportid from t_exportreport";
  ArrayList<String> list = new ArrayList<String>();
  Connection conn = null;
  PreparedStatement prep = null;
  ResultSet rs = null;
  try {
   conn = ConnectionPool.getInstance().getConnection("DS.SYSTEM");
   prep = conn.prepareStatement(sql);
   rs = prep.executeQuery();
   while (rs.next()){
    String reportid = rs.getString("c_reportid");
    list.add(reportid);
   }
   rs.close();
   if(list.contains(resid)){
    return true;
   }
   return false;
  } catch (Exception e) {
   throw new SmartbiException(FreeQueryErrorCode.SQL_ERROR, e);
  } finally {
   try {
    prep.close();
    conn.close();
   } catch (SQLException e) {
    LOG.error(e.getMessage(), e);
   }
  }
 }

 public boolean olapIsInReport(String clientId){
  OlapQueryBO report = (OlapQueryBO) OlapQueryService.getInstance()
    .getStateModule().getSessionAttribute(clientId);
  if (report == null) {
   throw new SmartbiException(
     OlapErrorCode.EXPORT_REPORT_NOT_FOUND);
  }
  return isInReport(report.getId());
 }

 @Override
 public void activate() {
 }
}
```

3.2.2 ExportFilter类

使用过滤器拦截系统的导出，给需要导出授权的报表增加主管授权的逻辑，授权通过才能继续导出。

ExportFilter类

```
package cn.com.smartbi;
```

```
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import org.apache.log4j.Logger;
import smartbi.SmartbiException;
import smartbi.decisionpanel.DecisionPanelModule;
import smartbi.decisionpanel.dashboard.DashboardBO;
import smartbi.freequery.FreeQueryErrorCode;
import smartbi.olap.OlapErrorCode;
import smartbi.olap.OlapQueryService;
import smartbi.olap.query.OlapQueryBO;
import smartbi.oltp.FreeQueryModule;
import smartbi.state.IStateModule;
public class ExportFilter implements Filter{
 private Logger log = Logger.getLogger(ExportFilter.class);
 @Override
 public void destroy() {
  // TODO Auto-generated method stub

 }
 @Override
 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain){
  try{
   request.setCharacterEncoding("UTF-8");
   HttpServletRequest req = (HttpServletRequest)request;
   if(req.getRequestURI().indexOf("ExportServlet") != -1){
    String actionType = request.getParameter("actionType");
                //2servletactionType"download"
    if ("download".equals(actionType)) {
     chain.doFilter(request, response);
     return;
    }
    ExportModule em = ExportModule.getInstance();
    String clientId = request.getParameter("clientId");
    String resid = "";
                //
    if(req.getRequestURI().indexOf("/ExportServlet") != -1){
     resid = request.getParameter("queryId");
    }else if(req.getRequestURI().indexOf("/InsightExportServlet") != -1){ //
     /*HttpSession session = req.getSession();
     InsightBO report = (InsightBO)  session.getAttribute(clientId);
     if (report == null)
      throw new SmartbiException(FreeQueryErrorCode.EXPORT_REPORT_NOT_FOUND);*/
     resid = request.getParameter("insightId");
    }else if(req.getRequestURI().indexOf("/DpExportServlet") != -1){  //
     IStateModule stateModule = DecisionPanelModule.getInstance().getStateModule();
     DashboardBO report = (DashboardBO) stateModule.getSessionAttribute(clientId);
     if (report == null)
      throw new SmartbiException(FreeQueryErrorCode.REPORT_CLIENT_ERROR);
     resid = report.getId();
    }else if(req.getRequestURI().indexOf("/OlapExportServlet") != -1){  //
     OlapQueryBO report = (OlapQueryBO) OlapQueryService.getInstance()
       .getStateModule().getSessionAttribute(clientId);
     if (report == null) {
      throw new SmartbiException(
        OlapErrorCode.EXPORT_REPORT_NOT_FOUND);
     }
     resid = report.getId();
    }else{
     chain.doFilter(request, response);
     return;
    }

    if(em.isInReport(resid)){
     String username = request.getParameter("uname");
     String password = request.getParameter("upsw");
     boolean rtn = ExportModule.getInstance().isContainRole(username, password);
```

```
     if(rtn){
      chain.doFilter(request, response);
     }else{
      throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR).setDetail("");
     }
    }else{
     chain.doFilter(request, response);
    }
   }else if(req.getRequestURI().indexOf("/ssreportServlet") != -1){  //
    String refreshType = request.getParameter("refreshType");
    if("refresh".equals(refreshType)){   //servlet
     chain.doFilter(request, response);

    }else{
     ExportModule em = ExportModule.getInstance();
     String resid = request.getParameter("resid");
     if(em.isInReport(resid)){
      String username = request.getParameter("uname");
      String password = request.getParameter("upsw");
      boolean rtn = ExportModule.getInstance().isContainRole(username, password);
      if(rtn){
       chain.doFilter(request, response);
      }else{
       throw new SmartbiException(FreeQueryErrorCode.EXPORT_ERROR).setDetail("");
      }
     }else{
      chain.doFilter(request, response);
     }
    }
   }else{
    chain.doFilter(request, response);
   }

  }catch(Exception e){
   log.error(e.getMessage(), e);
  }

 }

 protected IStateModule getStateModule() {
  return FreeQueryModule.getInstance().getStateModule();
 }
 @Override
 public void init(FilterConfig arg0) throws ServletException {

 }
}
```

4、相关资源（EPPR-8748）

ExportReportControl.rar

exportreportcontrol.ext