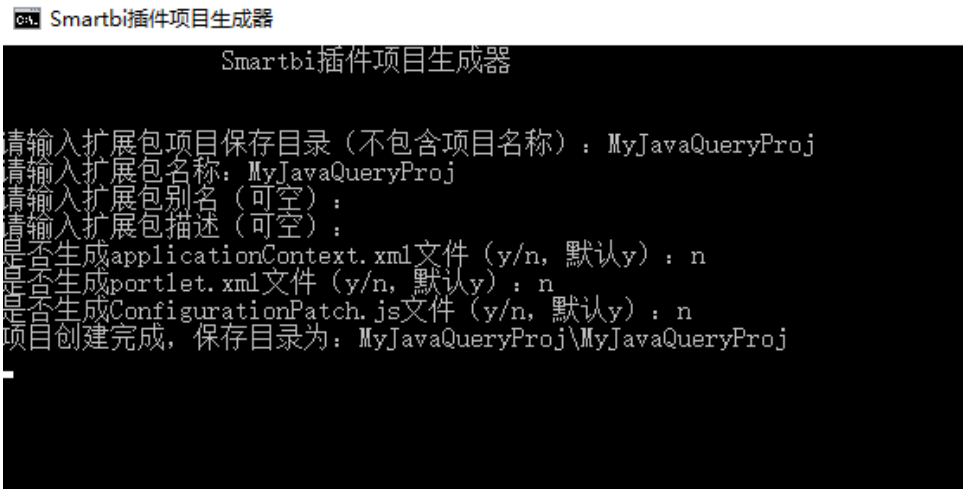
自定义JavaBean示例

此文档说明如何在Smartbi中实现自定义JavaeBean示例。

在项目的使用过程中，有一些数据不能直接从关系或者多维数据库中获取，可能需要从另外一些途径获取又或者需要一些特殊的处理。Smartbi中的查询和多维分析无法完成此功能，因此提供了Java查询二次开发接口允许项目进行定制性的开发。 为了应对这种需求，Smartbi提供了一个可以根据需要扩充的Java查询方式，可以根据实际情况开发来满足您的需求。

本方案是在Smartbi中添加自定义的Java查询类，并使Smartbi正确的展示数据的过程。 下面讲解一下具体的实现步骤：

1. 按照使用smartbi提供工具创建扩展包的说明，新建扩展包项目MyJavaQueryProj，如果是使用cmd脚本创建，配置如下图：

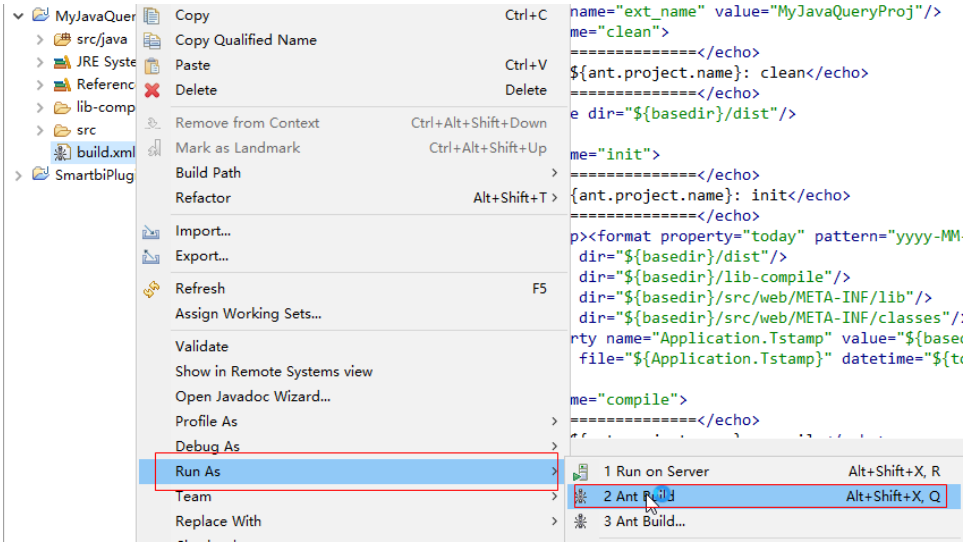


2. 解压smartbi.war到临时目录

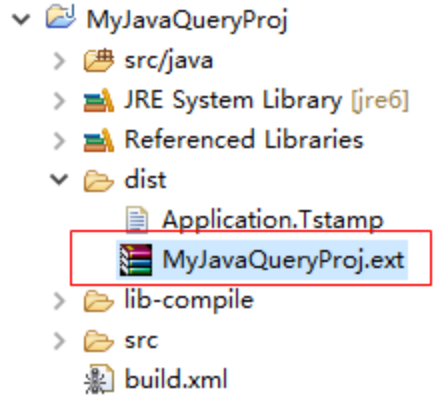
3. 将临时目录中的/WEB-INF/lib/smartbi-FreeQuery.jar、smartbi-Common.jar等smartbi依赖包复制到扩展包项目MyJavaQueryProj的lib-compile目录，并添加该目录下所有jar为项目的依赖包（如果要添加的jar不是smartbi.war中的jar，但是自定义javabean需要使用，则需要将jar包复制到MyJavaQueryProj的src/web/META-INF/lib目录，该目录下的jar在打包扩展包时将一同打包）

4. 在MyJavaQueryProj项目中新建一个类com.proj.MyJavaQuery，并实现接口IJavaQueryData.ISimpleData(Smartbi)
5. 编译项目，运行/MyJavaQueryProj/build.xml将项目打包为扩展包MyJavaQueryProj.ext，如下图

注：Eclipse等IDE编译用的JDK必须与Smartbi服务器运行的JDK使用同一个版本



6. 执行后刷新MyJavaQueryProj项目，可以看到新增一个dist目录，目录下MyJavaQueryProj.ext文件就是生成的扩展包

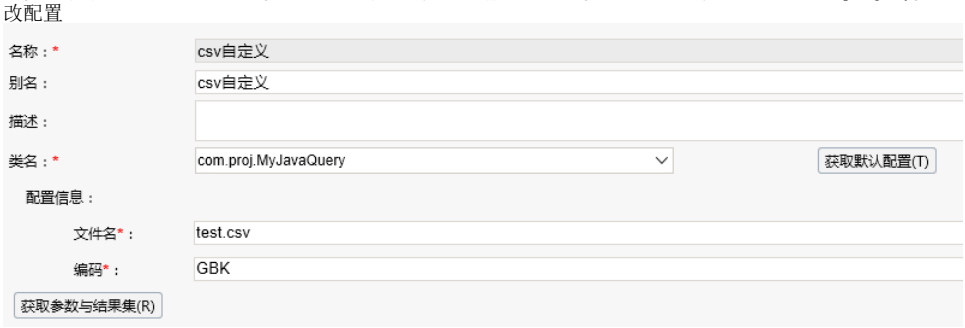


7. 按照扩展包部署的方法，部署MyJavaQueryProj.ext到smartbi

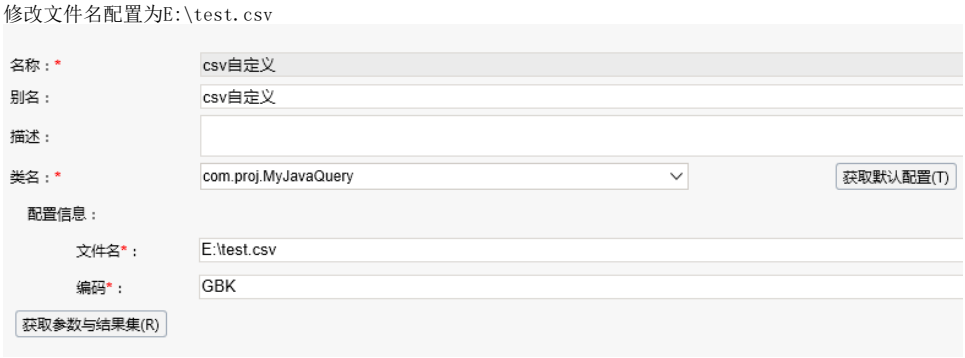
8. 重启smartbi应用服务器

9. 在【定制管理】→【数据管理】的“数据源”节点右键新建Java数据源

10. “Java数据源”上右键“新建Java查询对象”在类名中输入正确的Java查询实现类全名（如com.proj.MyJavaQuery）并获取默认配置，可自定义修改配置

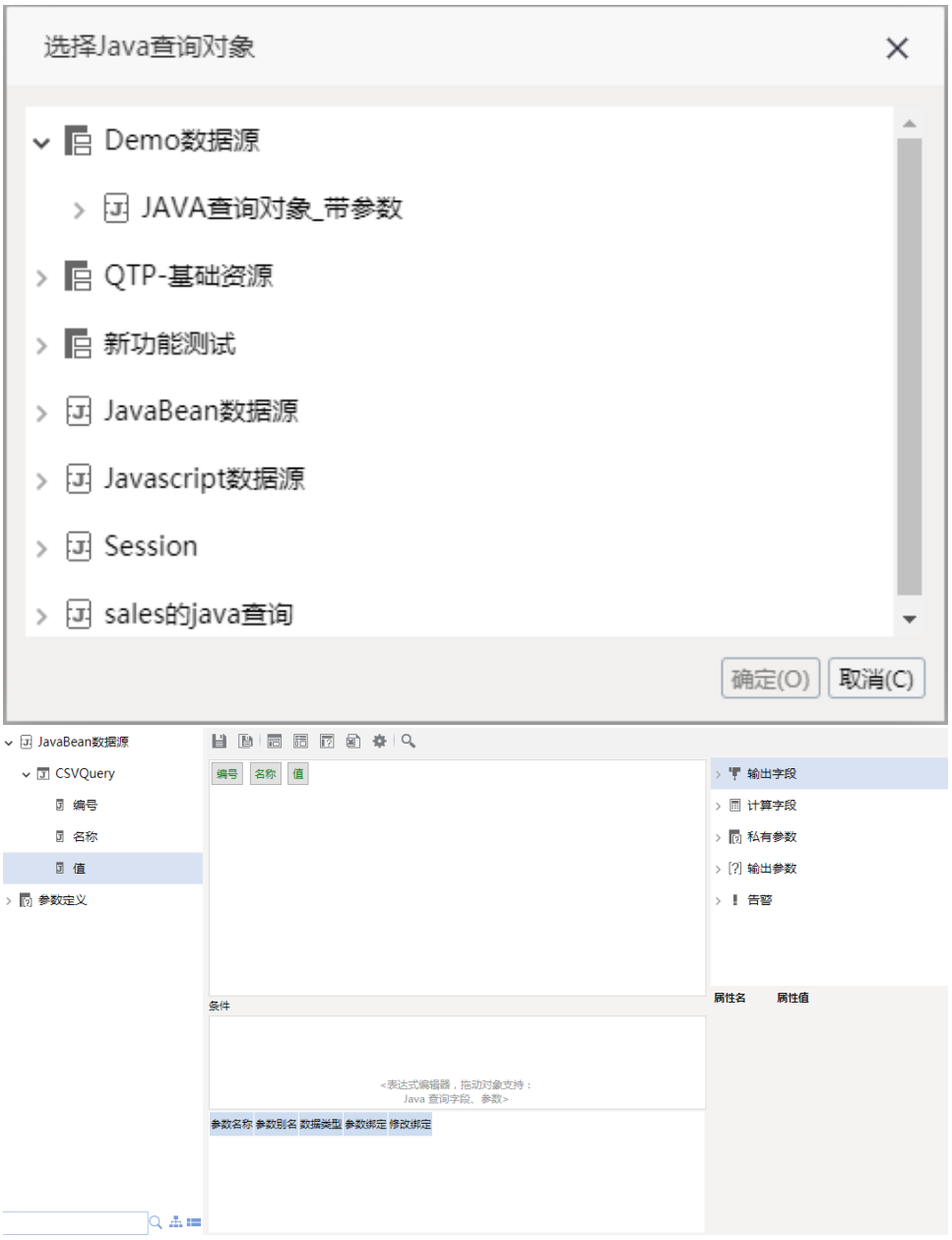


修改文件名配置为E:\test.csv

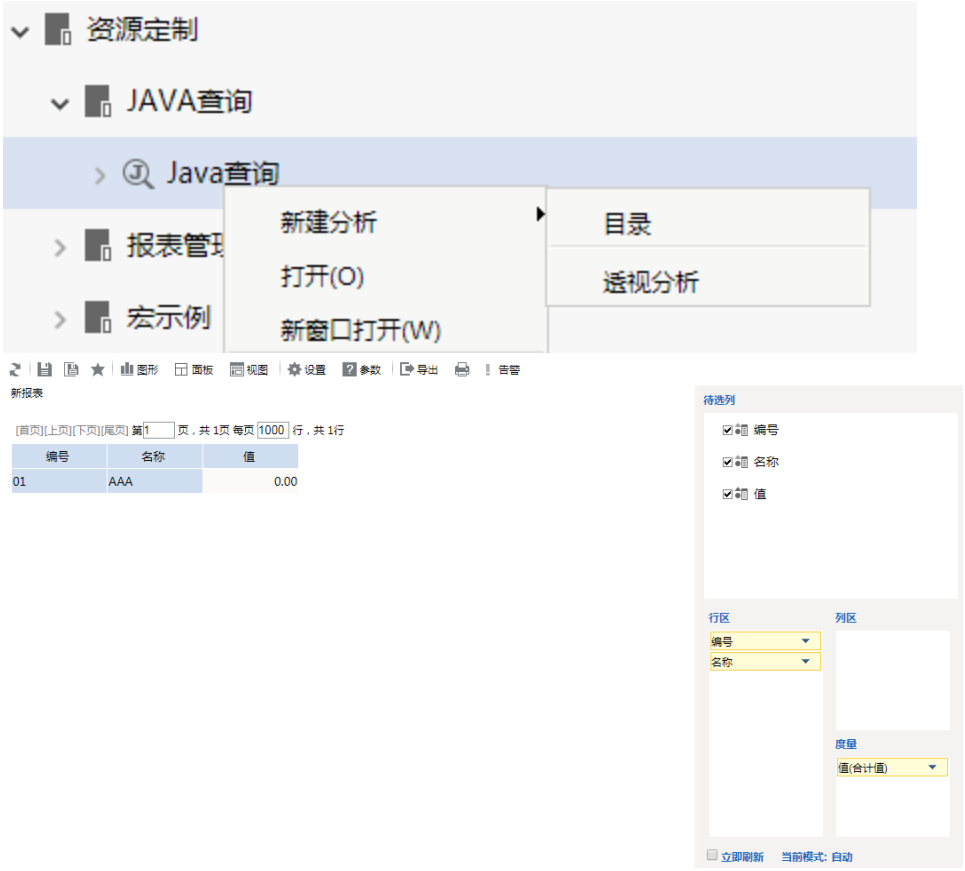


11. 点击获取参数与结果集并保存

12. 保存后就可以在【定制管理】→【数据集定义】的“Java查询”中使用对应的Java查询对象新建Java查询，拖动Java查询字段到表达式编辑器，然后保存Java查询


选择Java查询对象
Demo数据源
JAVA查询对象_带参数
QTP-基础资源
新功能测试
JavaBean数据源
Javascript数据源
Session
sales的java查询
确定(O)
取消(C)
JavaBean数据源
CSVQuery
编号
名称
值
参数定义
编号
名称
值
输出字段
计算字段
私有参数
输出参数
告警
条件
属性名
属性值
<表达式编辑器，拖动对象支持：
Java 查询字段、参数>
参数名称
参数别名
数据类型
参数绑定
修改绑定

13. 保存Java查询到指定目录后，可以基于java查询新建分析



这里是一个CSV文件查询的例子源码。

CSVQuery

```
package com.proj;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import smartbi.freequery.metadata.IJavaQueryData;
import smartbi.freequery.metadata.JavaQueryConfig;
import smartbi.freequery.metadata.JavaQueryOutputField;
import smartbi.freequery.metadata.JavaQueryParameter;
import smartbi.freequery.querydata.CellData;
import smartbi.freequery.querydata.GridData;
import smartbi.net.sf.json.JSONObject;
import smartbi.util.StringUtil;
import smartbi.util.ValueType;
public class MyJavaQuery implements IJavaQueryData.ISimpleData {
        private Map<String, JavaQueryConfig> configs = new LinkedHashMap<String, JavaQueryConfig>();
        private BufferedReader reader;
        private List<JavaQueryOutputField> outputFields;
        private int currentLine;
        public MyJavaQuery() {
                // FileName
```

```
        addConfig("FileName", "", "", "test.csv", true);
        addConfig("Encoding", "", "", "GBK", true);
}
/**
 * Java
 */
public List<JavaQueryConfig> getConfigs() {
        return new ArrayList<JavaQueryConfig>(configs.values());
}
/**
 *
 *
 * @param name
 *
 * @param alias
 *
 * @param desc
 *
 * @param defaultValue
 *
 * @param notNull
 *
 */
private void addConfig(String name, String alias, String desc,
                String defaultValue, boolean notNull) {
        JavaQueryConfig p = new JavaQueryConfig();
        p.setName(name);
        p.setAlias(alias);
        p.setDesc(desc);
        p.setValue(defaultValue);
        p.setNotNull(notNull);
        configs.put(name, p);
}
/**
 *
 *
 * @param configStr
 *
 */
public void loadConfigs(String configStr) {
        if (StringUtil.isNullOrEmpty(configStr))
                return;
        JSONObject obj = JSONObject.fromString(configStr);
        configs.get("FileName").setValue(
                        obj.has("FileName") ? obj.getString("FileName") : null);
        configs.get("Encoding").setValue(
                        obj.has("Encoding") ? obj.getString("Encoding") : null);
}
/**
 *
 *
 * @return
 */
public String saveConfigs() {
        JSONObject json = new JSONObject();
        for (JavaQueryConfig config : configs.values())
                json.put(config.getName(), config.getValue());
        return json.toString();
}
/**
 *
 *
 * @param key
 *
 * @param value
 *
 */
public void setConfigValue(String key, String value) {
        configs.get(key).setValue(value);
}
/**
```

```
         *
         */
        public void setConfigValues(Map<String, String> configValues) {
                for (Entry<String, String> config : configValues.entrySet())
                        configs.get(config.getKey()).setValue(config.getValue());
        }
        /**
         * Java
         */
        public void init() {
                try {
                        outputFields = new ArrayList<JavaQueryOutputField>();
                        File file = new File(configs.get("FileName").getValue());
                        if (file.exists()) {
                                FileInputStream fis = new FileInputStream(file);
                                if (configs.get("Encoding").getValue().toLowerCase().equals("utf-8")) {
                                        byte[] headData = new byte[3];
                                        fis.read(headData, 0, 3);
                                        if (headData[0] != (byte) 0xef || headData[1] != (byte) 0xbb || headData
[2] != (byte) 0xbf) {
                                                fis.close();
                                                fis = new FileInputStream(file);
                                        }
                                }
                                reader = new BufferedReader(new InputStreamReader(fis, configs.get("Encoding").
getValue()));
                                String titleLine = reader.readLine();
                                String[] fields = titleLine == null ? "".split(",") : titleLine.split(",");
                                for (String str : fields) {
                                        JavaQueryOutputField f = new JavaQueryOutputField();
                                        f.setId(str);
                                        f.setName(str);
                                        f.setAlias(str);
                                        if (str.equals(StringUtil.getLanguageValue("Value1"))) {
                                                f.setDataType(ValueType.DOUBLE);
                                        } else {
                                                f.setDataType(ValueType.STRING);
                                        }
                                        outputFields.add(f);
                                }
                        }
                        currentLine = 0;
                } catch (UnsupportedEncodingException e) {
                        throw new IllegalArgumentException(e);
                } catch (FileNotFoundException e) {
                        throw new IllegalArgumentException(e);
                } catch (IOException e) {
                        throw new IllegalArgumentException(e);
                }
        }
        /**
         * Java
         */
        public void close() {
                try {
                        if (reader != null) {
                                reader.close();
                                reader = null;
                        }
                } catch (IOException e) {
                        throw new IllegalArgumentException(e);
                }
        }
        /**
         *
         */
        public List<JavaQueryParameter> getParameters() {
                return new ArrayList<JavaQueryParameter>();
        }
        /**
         *
```

```
         */
        public void setParameterValue(String id, String value, String displayValue) {
        }
        /**
         * Java
         */
        public List<JavaQueryOutputField> getOutputFields() {
                return outputFields;
        }
        /**
         *
         */
        public GridData getGridData(int from, int count) {
                try {
                        if (currentLine > from) {
                                if (reader != null) {
                                        reader.close();
                                        FileInputStream fis = new FileInputStream(configs.get("FileName").
getValue());
                                        if (configs.get("Encoding").getValue().toLowerCase().equals("utf-8")) {
                                                byte[] headData = new byte[3];
                                                fis.read(headData, 0, 3);
                                                if (headData[0] != (byte) 0xef || headData[1] != (byte) 0xbb ||
headData[2] != (byte) 0xbf) {
                                                        fis.close();
                                                        fis = new FileInputStream(configs.get("FileName").
getValue());
                                                }
                                        }
                                        reader = new BufferedReader(new InputStreamReader(fis, configs.get
("Encoding").getValue()));
                                        reader.readLine();
                                }
                                currentLine = 0;
                        }
                        if (reader != null) {
                                while (currentLine < from) {
                                        reader.readLine();
                                        currentLine++;
                                }
                        }
                        List<List<CellData>> datas = new ArrayList<List<CellData>>();
                        for (int i = 0; i < count; i++) {
                                String line = reader == null ? null : reader.readLine();
                                if (line == null)
                                        break;
                                currentLine++;
                                String[] fs = line.split(",");
                                List<CellData> row = new ArrayList<CellData>();
                                for (int j = 0; j < fs.length; j++) {
                                        CellData c = new CellData();
                                        c.setStringValue(fs[j]);
                                        row.add(c);
                                }
                                datas.add(row);
                        }
                        GridData d = new GridData();
                        List<String> headers = new ArrayList<String>();
                        for (JavaQueryOutputField f : outputFields)
                                headers.add(f.getName());
                        d.setStringHeaders(headers);
                        d.setData(datas);
                        return d;
                } catch (UnsupportedEncodingException e) {
                        throw new IllegalArgumentException(e);
                } catch (FileNotFoundException e) {
                        throw new IllegalArgumentException(e);
                } catch (IOException e) {
                        throw new IllegalArgumentException(e);
                }
        }
```

```
        /**
         * Integer.MAX_VALUE
         */
        public int getRowCount() {
                return Integer.MAX_VALUE;
        }
}
```

附件

源代码：MyJavaQueryProj.rar，扩展包：MyJavaQueryProj.ext，测试csv文件：test.csv