

Java查询介绍

1 概述

Smartbi 默认提供可视化查询、SQL查询、存储过程查询等给客户使用，但这些查询均依赖于系统连接的关系数据源。有些项目存在一些特殊的数据源，例如文本文件或非结构化的数据，“Java查询”即是 Smartbi 产品提供给客户实现自定义数据结构的一种扩展方法。

- 1 概述
- 2 示例说明

Java 数据源不同于关系数据源和多维数据源之处为：它没有一个物理的数据库存储其字段和数据；但它可以通过Java类将任意一个含有数据的文件或报表通过解析后，在Smartbi中展现分析。

为实现自定义数据结构的查询要求，系统提供接口类 **IJavaQueryData** 供实现Java查询扩展开发。需实现的接口方法说明如下：

IJavaQueryData接口说明

```
package com.freequery.metadata;

import java.util.List;
import java.util.Map;
import com.freequery.expression.Expression;
import com.freequery.querydata.GridData;
import com.freequery.report.OrderBy;

/**
 * Java
 */
public interface IJavaQueryData {
    /**
     *
     */
    public interface ICalculateFieldSupport {
        public void addCalcField(String name, Expression exp);
    }

    /**
     *
     */
    public interface IOrderSupport {
        public void setOrderBy(List<OrderBy> orderBy);
    }

    /**
     *
     * @param configStr
     */
    public void loadConfigs(String configs);

    /**
     *
     * @return
     */
    public String saveConfigs();

    /**
     * Java
     */
    public List<JavaQueryConfig> getConfigs();

    /**
     *
     * @param key
     * @param value
     */
}
```

```

    public void setConfigValue(String key, String value);

    /**
     *
     */
    public void setConfigValues(Map<String, String> configValues);

    /**
     * Java
     */
    public void init();

    /**
     *
     */
    public List<JavaQueryParameter> getParameters();

    /**
     * Java
     */
    public List<JavaQueryOutputField> getOutputFields();

    /**
     *
     */
    public void setParameterValue(String id, String value, String displayValue);

    /**
     * Integer.MAX_VALUE
     */
    public int getRowCount();

    /**
     *
     */
    public GridData getGridData(int from, int count);

    /**
     * Java
     */
    public void close();
}

```

2 示例说明

以下部分说明如何在项目中进行Java查询的二次开发。

1、打开服务器部署文件smartbi.war，解压后将smartbi.war\WEB-INF\lib\目录下的 **smartbi-FreeQuery.jar**、**smartbi-Common.jar** 包加入到扩展包项目的classpath中去。

2、撰写Java查询对象类JavaQueryDemo.java，实现接口 **IJavaQueryData**。

Java查询对象类

```

package bof.ext.JavaQuery;
import java.io.*;
import java.util.*;
import java.util.Map.Entry;
import smartbi.net.sf.json.JSONObject;
import smartbi.util.StringUtil;
import smartbi.util.ValueType;
import smartbi.freequery.metadata.*;
import smartbi.freequery.querydata.CellData;
import smartbi.freequery.querydata.GridData;
/**
 * Java
 */
public class JavaQueryDemo implements IJavaQueryData {
    private Map<String, JavaQueryConfig> configs = new LinkedHashMap<String, JavaQueryConfig>();
}

```

```

private BufferedReader reader;
private List<JavaQueryOutputField> outputFields;
private int currentLine;
public JavaQueryDemo() {
    // 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 {
            ClassLoader c1=this.getClass().getClassLoader();
            reader = new BufferedReader(new InputStreamReader(c1.getResourceAsStream(configs.get(
                "FileName").getValue()), configs.get("Encoding").getValue()));
            String titleLine = reader.readLine();
            String[] fields = titleLine.split(",");
            outputFields = new ArrayList<JavaQueryOutputField>();
            for (String str : fields) {
                JavaQueryOutputField f = new JavaQueryOutputField();
                f.setId(str);
                f.setName(str);
                f.setAlias(str);
                f.setDataTypes(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) {

```

```

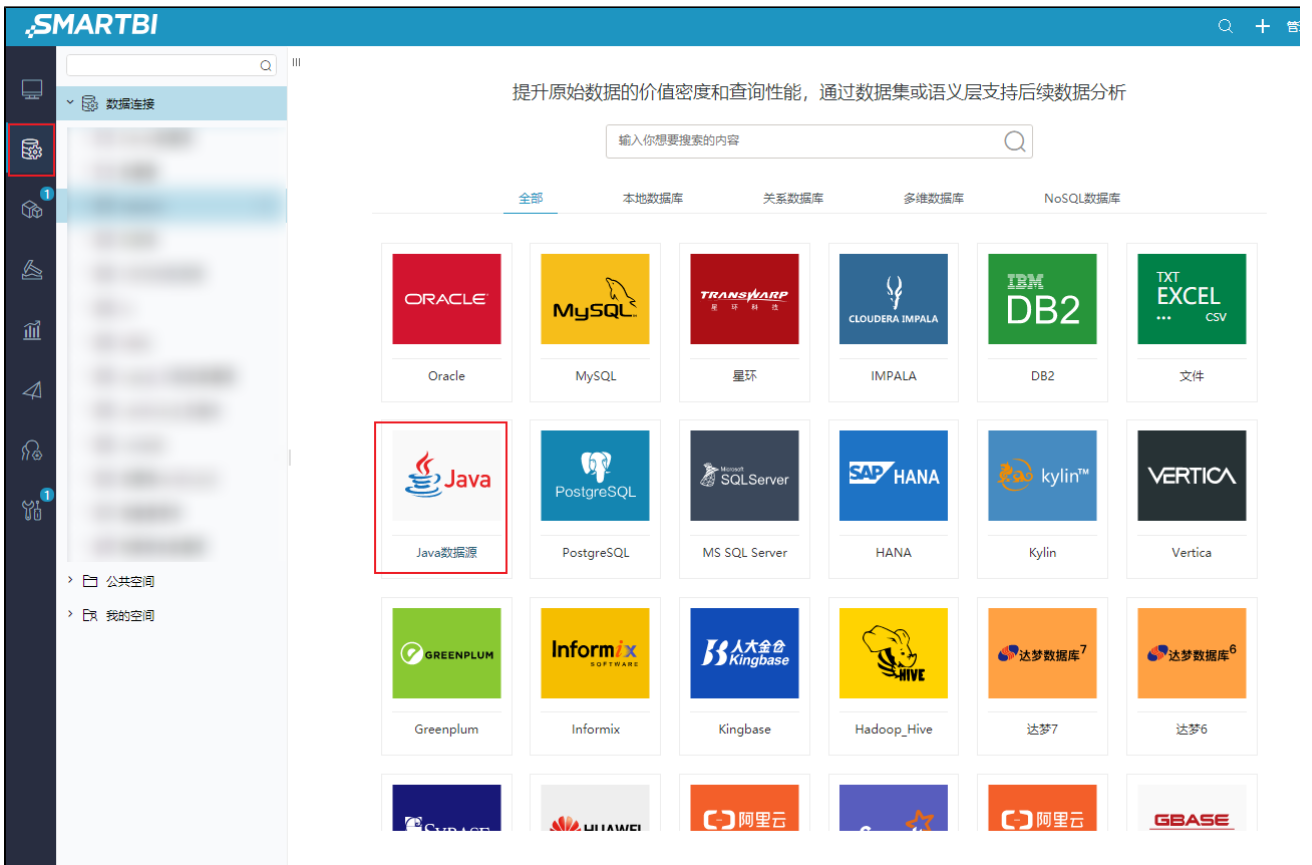
        try {
            if (currentLine > from) {
                reader.close();
                ClassLoader c1=this.getClass().getClassLoader();
                reader = new BufferedReader(new InputStreamReader(c1.getResourceAsStream
(configs.get(
                                "FileName").getValue()), configs.get("Encoding").
getValue()));

                reader.readLine();
                currentLine = 0;
            }
            while (currentLine < from) {
                reader.readLine();
                currentLine++;
            }
            List<List<CellData>> datas = new ArrayList<List<CellData>>();
            for (int i = 0; i < count; i++) {
                String line = 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;
    }
}

```

3、在Smartbi中新建Java查询定义

- (1) 登录Smartbi，在【数据连接】选择“Java数据源”、新建Java数据源

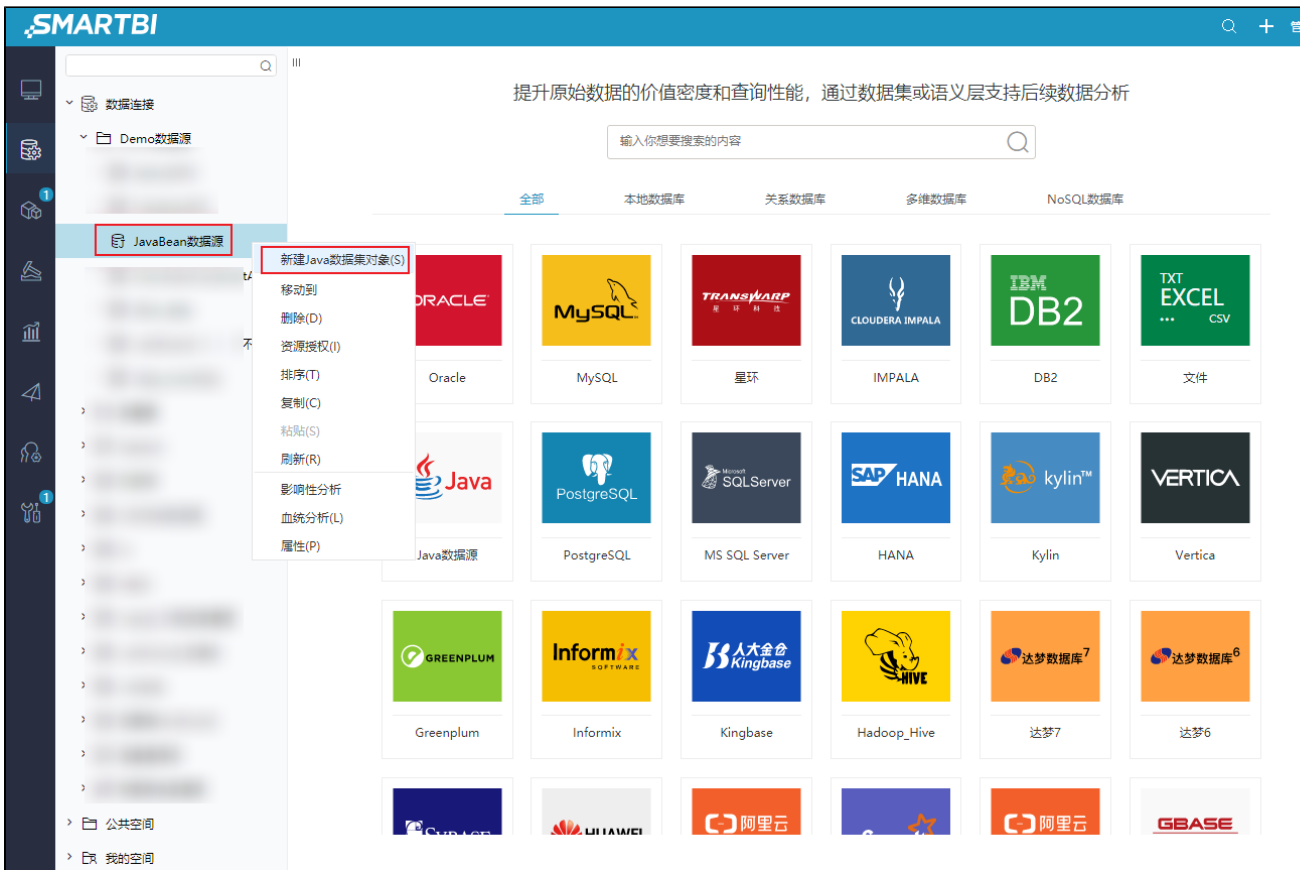


新建Java数据源

名称*	JavaBean数据源
别名	JavaBean数据源
描述	

保存(S) 关闭(C)

(2) 在刚才新建的“Java数据源”上右键“新建Java数据集对象”，在类名中输入正确的Java查询实现类全名（如bof.ext.JavaQuery.JavaQueryDemo）并检测默认配置；



3、再点击获取参数与结果集并保存；

导航 | JAVA_Object x

描述:

类名: * bof.ext.JavaQuery.JavaQueryDemo 获取默认配置(T)

配置信息:

文件名*: test.csv

编码*: GBK

获取参数与结果集(R)

参数:

	名称	别名	类型
1	编号	编号	STRING
2	名称	名称	STRING
3	值	值	STRING

结果集设置:

	名称	别名	类型
1	编号	编号	STRING
2	名称	名称	STRING
3	值	值	STRING

保存(S) 关闭

4、“保存”后就可以在“新建Java数据集”中使用。

5、示例代码下载: [JavaQuery.rar](#)