Java查询示例

此文档说明如何在Smartbi中实现Java查询。

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. 解压smartbi.war到临时目录
2. 新建Java项目MyAuthProj，将临时目录中的/WEB-INF/lib/smartbi-FreeQuery.jar及其它依赖包添加为项目的依赖包
3. 新建一个类com.proj.MyJavaQuery，并实现接口IJavaQueryData
4. 编译项目，将项目中生成的.class文件打包成myjava.jar
5. 将myjava.jar复制到WAR包解压目录的/WEB-INF/lib目录
6. 将项目所需要额外的配置文件复制到/WEB-INF目录，将额外需要的依赖包复制到/WEB-INF/lib目录
7. 重新将临时目录打包成smartbi.war，并替换原始的smartbi.war
8. 重启服务器
9. 在【管理】→【系统管理】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）并检测默认配置
11. 点击获取参数与结果集并保存
12. 保存后就可以在【定制】→【数据集定义】的“新建Java查询”中使用

这里是一个CSV文件查询的例子源码。

CSVQuery

```
package smartbi;
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 CSVJavaQuery implements IJavaQueryData.ISimpleData {
 private Map<String, JavaQueryConfig> configs = new LinkedHashMap<String, JavaQueryConfig>();
 private BufferedReader reader;
 private List<JavaQueryOutputField> outputFields;
 private int currentLine;

 public CSVJavaQuery() {
  //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 {
   reader = new BufferedReader(new InputStreamReader(new FileInputStream(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.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) {
    reader.close();
    reader = new BufferedReader(new InputStreamReader(new FileInputStream(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;
 }
}
```