# 定制数据集（定制查询）

## 1. 定制数据集（定制查询）的使用

用 java 实现的数据源采集方式，获取数据的逻辑需要自己实现，需要用到的 jar 包是产品安装目录下 YH/Yonghong/product 的 product.jar 和 thirds.jar

创建 SampleQuery.java:

```
import g5.DataGrid;
import g5.dc.impl.DCUtil;
import g5.grid.impl.GridUtil;
import g5.meta.*;
import g5.meta.t.*;
import g5.qry.*;
import g5.qry.sql.SQLQuery;
import g5.gqry.*;
import g5.qry.impl.DynamicObjectSeg;
import g5.qry.impl.QGrid;
import g5.rep.AssetRef;
import g5.rep.info.LocalQueryFolder;
import g5.secure.GPrincipal;
import g5.sv.db.impl.DBServiceImpl;
import g5.thread.ThreadMgr;
import g5.util.InfoRegister;
import g5.util.KeyUtil;
import java.util.*;

/**
 * SampleQuery for customer.
 */
public class SampleQuery extends GCustomQuery {
    /**
     * Constructor.
     */
    public SampleQuery() {
        super();
    }
```

```
    @Override
    public Query createRuntime(QContext context) {
        return this;
    }

    // 实现方法(在执行查询时会调用该方法)

    public DataGrid exec(QCol select[], QContext context) {
        QGrid qgrid = create(context);

    // 自己的逻辑写在这里，往 qgrid 里面添加数据。

PreparedStatement pstatm = conn.prepareStatement(sql);
ResultSet rs = pstatm.executeQuery();


//添加多行数据

while(rs.next()) {
for(int i = 0; i < select.length; i++) {
Object object = rs.getObject(select[i].getID());

qGrid.add(i, object); // 向第 i 列追加一个数据

}
}
        qgrid.complete();
        return qgrid;
    }

    @Override
    public Parameter[] findParams(LocalQueryFolder pro) throws Exception {
        return new Parameter[0];
    }

    /**

     * Create a new grid，这个可以不动

     */
    private QGrid create(QContext context) {
        QCol[] cols = metaCols(context);
        String[] harr = new String[cols.length];
        byte[] sarr = new byte[cols.length];

        for(int i = 0; i < cols.length; i++) {
            harr[i] = cols[i].getView();
            sarr[i] = DynamicObjectSeg.getPreferredSeg(cols[i].getDType(), false);
        }
```

```
        return (QGrid) QGrid.create(harr, sarr, cols);
    }

    // 需要自己定义查询返回的字段名

    public QCol[] metaCols(QContext context) {

        // BCol 参数 1 列名，参数 2 数据类型，参数 3 是否是维度

        /**

            DType 支持类型

            STRING BOOLEAN FLOAT DOUBLE DECIMAL CHAR BYTE SHORT INTEGER LONG
DATE_TIME DATE TIME
        **/
        return new QCol[] {new BCol("col1", DType.STRING, true),
                                    new BCol("col2", DType.STRING, true),
                                //    new BCol("col3", DType.DOUBLE)
        };
    }
}
```

# 使用方式：

（1） 首先要修改 SampleQuery.java，按照里面的注释修改成自己的逻辑，这个需要引用永洪产品的 product.jar 和 thirds.jar 包（产品默认放置位置为 YH\Yonghong\product 下，不需要手动配置）才能运行。

（2） 修改完毕后，需要编译 SampleQuery（建议使用 jdk1.7 编译），会生成 SampleQuery.class。

**方法 1：**把这个类放到**\YH\tomcat\webapps\bi\WEB-INF\classes** 里面（YH\tomcat\bin 下面的 setclasspath 文件，7.0 以后的版本都已默认配置该路径，linux 为 setclasspath.sh 文件，win 为 setclasspath.bat 文件，请检查文件中路径是否存在，如不存在请手工配置。）

setclasspath.bat

```
set "JAVA_ENDORSED_DIRS=%CATALINA_HOME%\endorsed"
:gotEndorseddir

setlocal EnableDelayedExpansion

set tmpclasspath=
for %%i in (..\..\Yonghong\lib, ..\..\example, ..\..\Yonghong\product) do (
    for /F "delims=" %%s in ('dir %%i\*.jar /s/b/a-d') do (
        set tmpclasspath=!tmpclasspath!;%%s
    )
)

ENDLOCAL & set tmpclasspath=%tmpclasspath%

set CLASSPATH=%CLASSPATH%;%JAVA_HOME%
\lib\tools.jar;..\lib\servlet-api.jar;..\..\Yonghong\lib\jdbcDriver\kylin-jdbc-1.6.0\kylin-jdbc-1
.6.0.jar;..\webapps\bi\WEB-INF\classes;%tmpclasspath%;F:\Yonghong7.5\Yonghong\lib\jdbcDriver\

```

**方法 2：**将 class 文件打成 jar 包，在 YH\Yonghong\lib 下面新建文件夹后，放置 jar 包。(该目录下所有的 jar 包都会自动引用，不需要配置 setclasspath 文件)

（3） 重启 tomcat

（4） 打开永洪，进入创建数据集模块，新建定制数据集，输入类型，刷新元数据，就可以看到列了。