# 打印服务开发设计说明

## 1、模板设计

### 1.1 开发环境

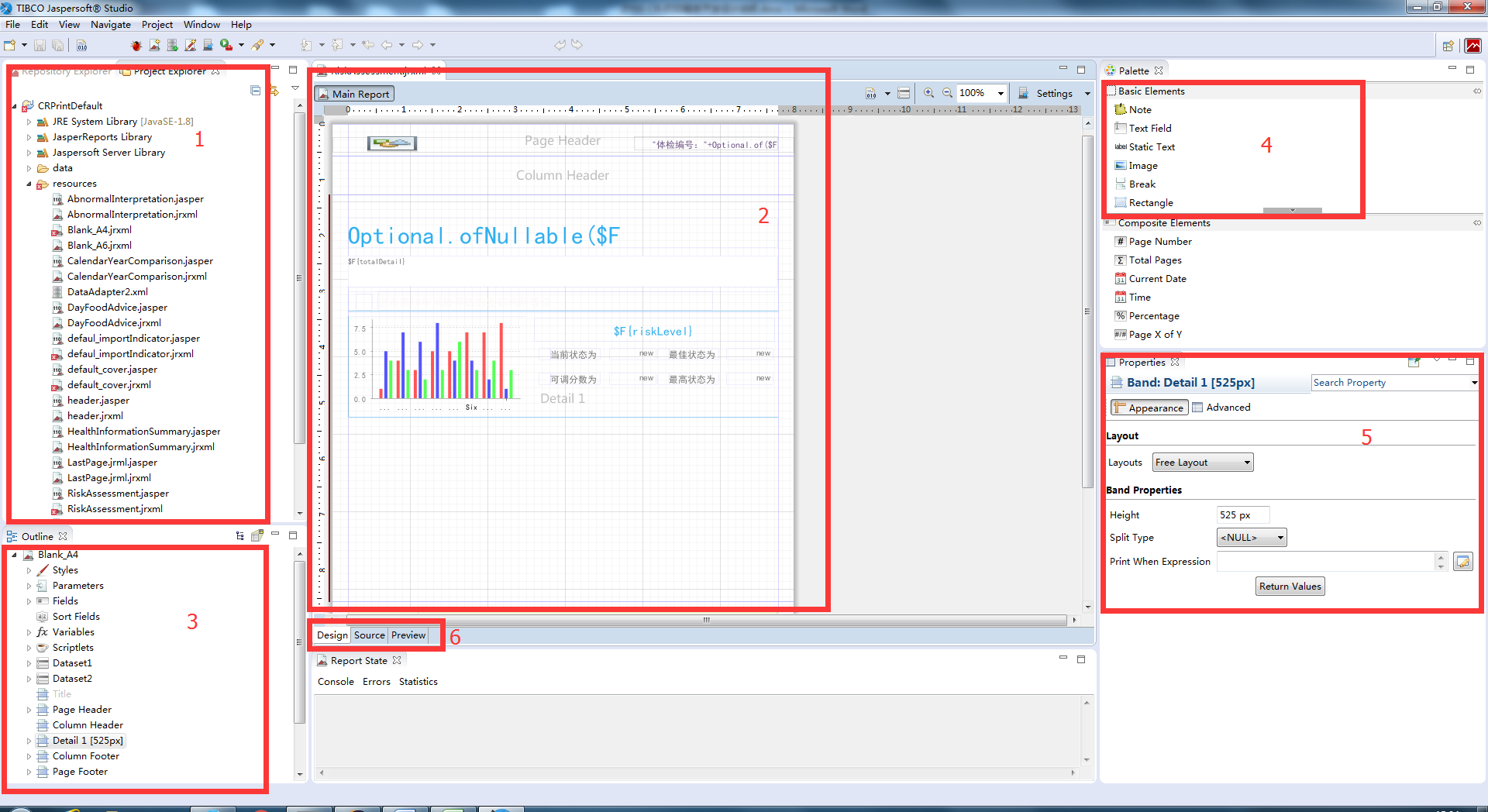
Jaspersoft Studio 6.3.0

Jdk 1.8

### 1.2 Jaspersoft Studio 使用

Jaspersoft Studio是基于eclipase 开发的软件，通用操作都和eclipase操作一致。

#### 1、主界面：

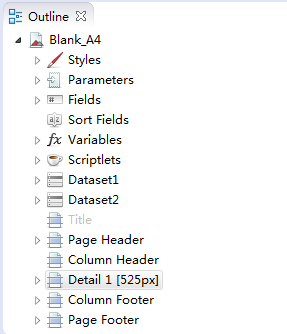


以上为主界面，我将主要使用的内容分为6个区域。

**区域1：**文件结构/工程目录，这个与 eclipase 一致。

**区域2：**报表绘制页面，这里是我们绘制报表的区域，可以图形化操作，也可以以xml代码的形式编辑，更可以预览页面效果。

**区域3：**outline 概述区域，以树型结构展示。根节点是页面，根节点下通常有以下节点：



Styles 样式（通常是表格的单元格样式）；

Parameters 参数（生成时，通过后台传入）；

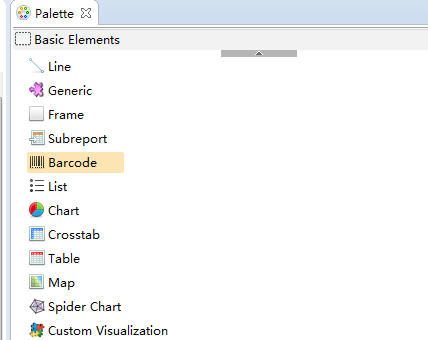
Fields 字段（数据库字段，也可以是某个 java 类的成员变量，json的节点等等）；

Variables 变量（页面内使用）；

Dataset 子数据连接（提供给子报表、表格、列表、图表等使用）；

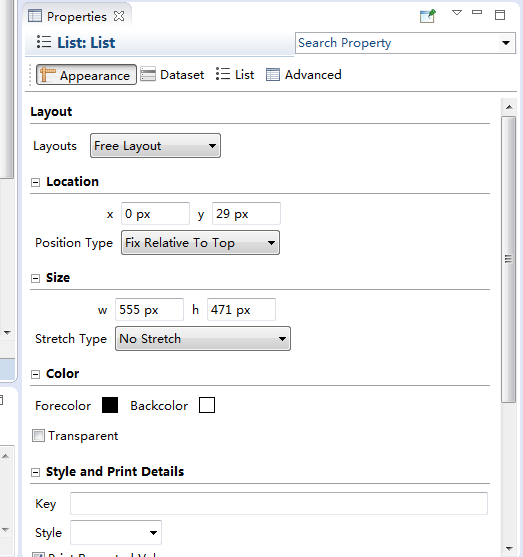
剩下的节点与绘制的页面对应，展示了页面里有什么元素。

**区域4：**



一些常用的控件，比如：图片、表格、子报表、线条、图表、列表等等。点击控件再点击页面就可以添加。

**区域5：**



属性区域，显示页面或者控件的属性，比如Layouts 控件内部布局，Location 位置，Size 控件大小等等。

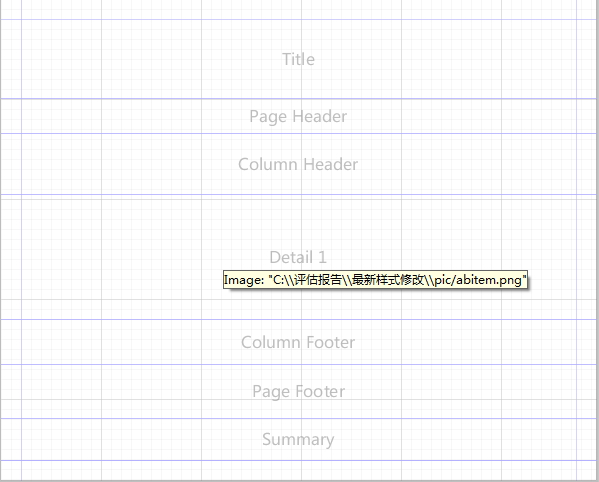
**区域6：**



页面展示切换，Design 图形化设计模式，Source 源码显示模式，预览模式

#### 2、页面结构

页面结构如下图：



页面分为：Tilte、Page Header、Column Header、Detail 、Column Footer 、Page Footer、Summary

Tilte：title绘制的内容只在第一个页面显示；

Page Header：页首，在每一页的页首都显示；

Column Header：列首，在每一页的页首都显示；

Detail：内容，页面要展示的具体内容，该部分会循环；

Column Footer：列尾，在每一页的页尾都显示；

Page Footer：页尾，在每一页的页尾都显示；

Summary： 总结，在最后一页才会显示。

不使用哪个部分可以删除某一个部分。

### 1.3模板设计

#### 1、数据源选择

JasperRepot可以选择各种数据源，比如sql数据库，no-sql数据库，java的FactoryBean/Bean，JSON、XML等等。

几种常用的方式比较：

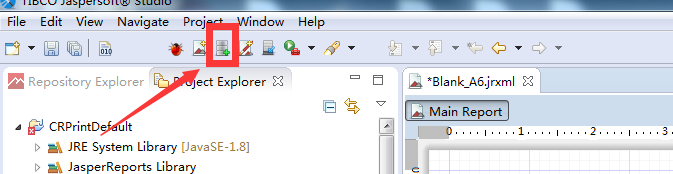
sql数据库：直接写sql语句查询数据，使用上非常简单，但是与数据库耦合非常高，不符合现在项目的结构分层。

java的FactoryBean/Bean：与java类依赖较高，需要一些数据就要写一个类和一个工厂类，扩展性非常的弱。

JSON数据源：模板设计只依赖数据（JSON），与后台的模板生成耦合度非常的低。数据改变不会影响后台的模板生成。

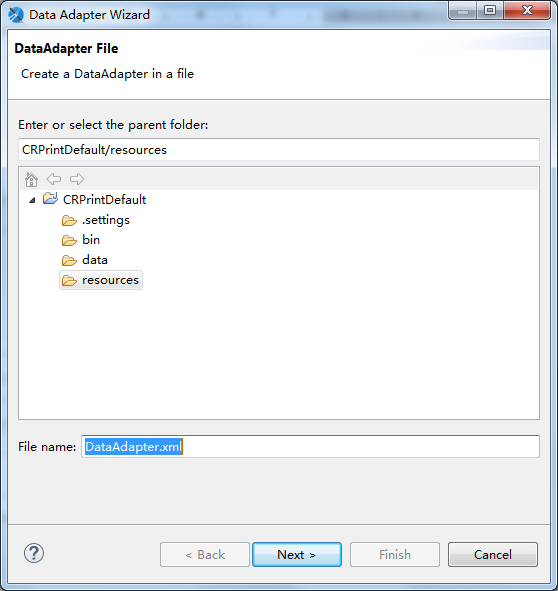
#### 2、数据源模板构建

第一步：点击 new Data Adapter



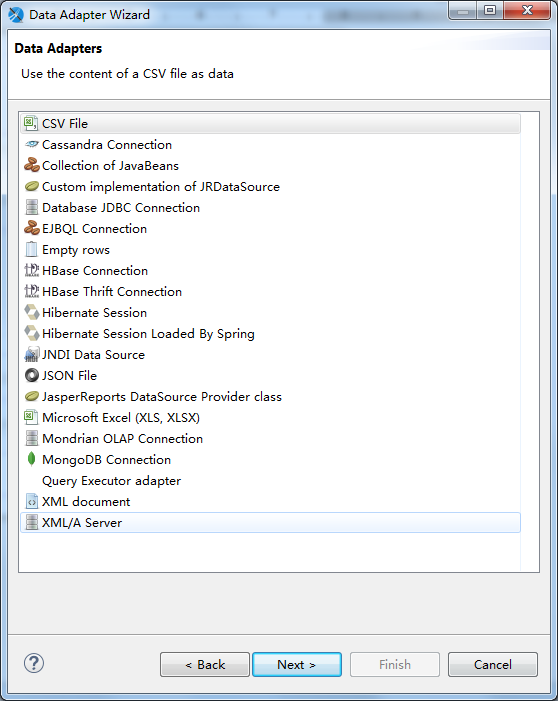
第二步：

设置文件名：



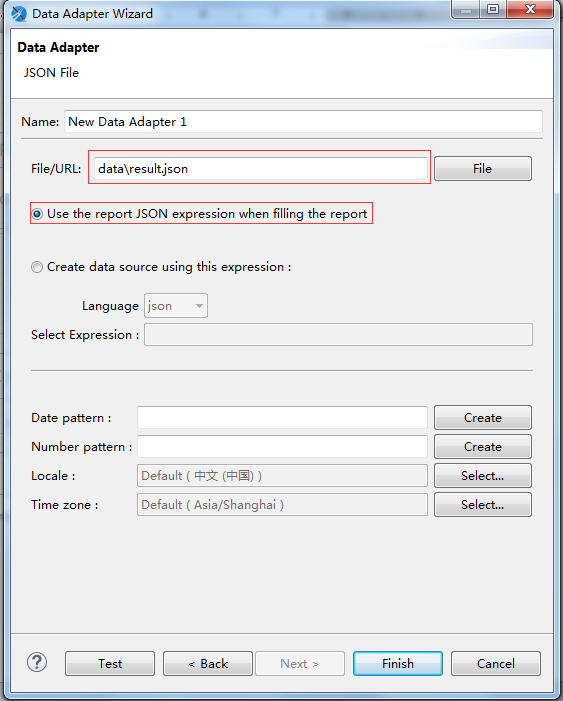
第三步：

选择数据源类型，这里选择JSON File



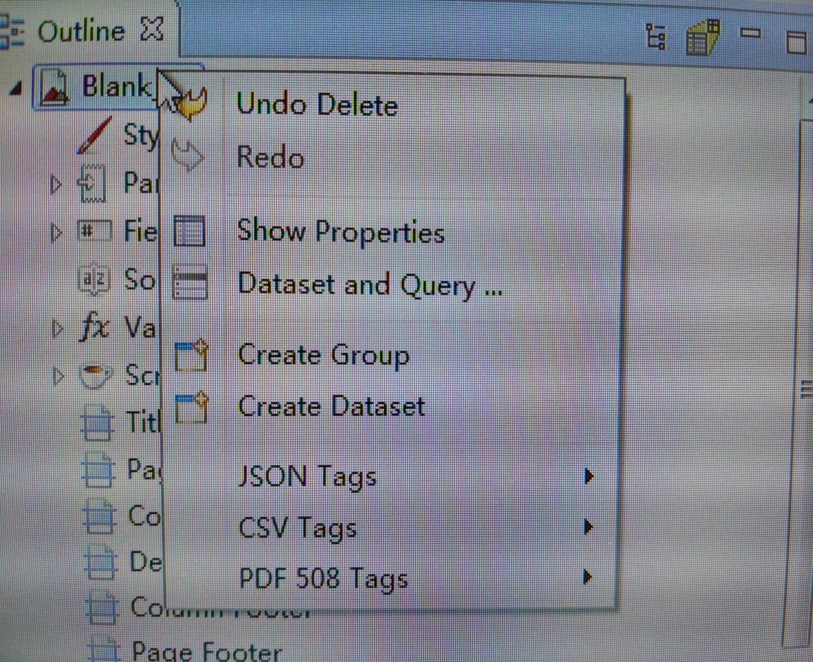
第四步：

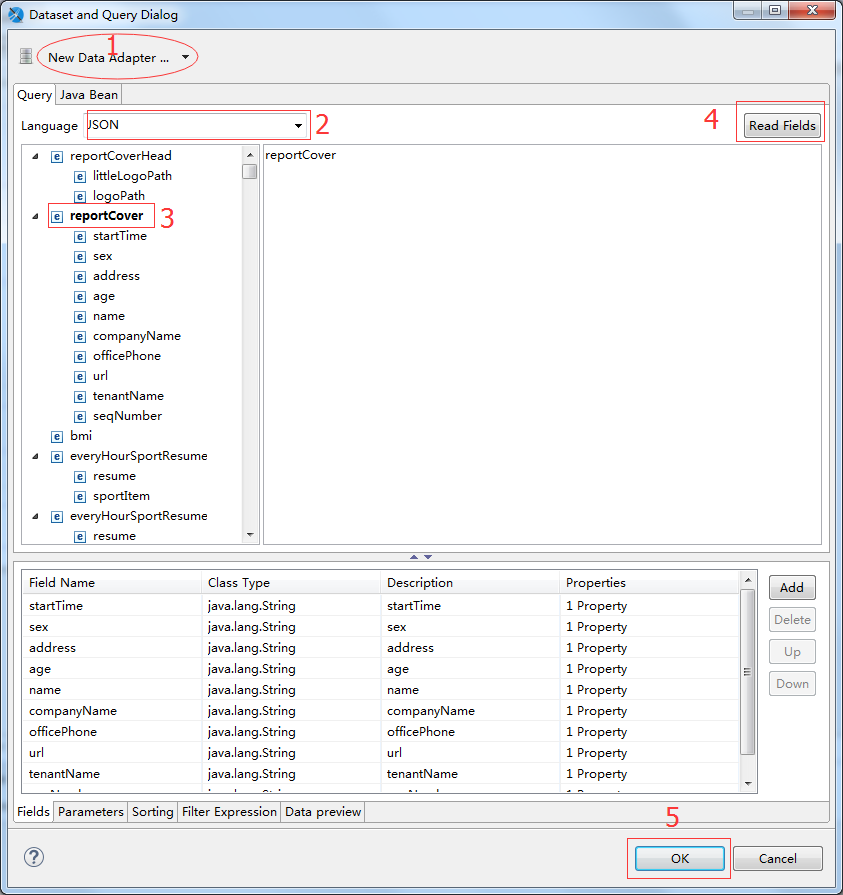
File选择一个json文件，这个文件是你要填充在模板上的数据。这个文件只是在画模板的时候起字段确定和预览作用。另一个选择的是使用的时候使用表达式查询。点击Finish完成。



#### 3、获取数据源中数据

在outline中的根节点，右键，点击Dataset and Query





1：选择之前新建的JSON数据源模板；

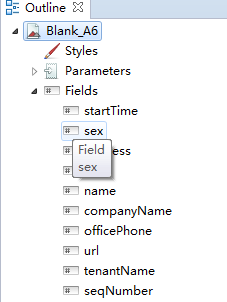
2：查询语言选择 JSON ；

3：选择需要查询的字段，如果一个都不选默认是根节点下的字段；双击某一字段则可获取该字段下的子字段。

4：点击Read Fields，读取字段；

5：点击OK 设置完毕。

查看Outline的Fields可以看见添加的字段：



查看源码来看看这个查询的方式：

点击source，源码如下：



查询的根节点：reportCover

<queryString language="json">

<![CDATA[reportCover]]>

</queryString>

查询的字段：

<field name="startTime" class="java.lang.String">

<property name="net.sf.jasperreports.json.field.expression" value="startTime"/>

<fieldDescription><![CDATA[startTime]]></fieldDescription>

</field>

查询的字段内容都是在reportCover根节点下。



但是我想要更多的字段，不是在reportCover 根节点下的要如何实现？

这样就需要直接对代码修改：

根节点设置为整个JSON的根节点，就是为空

<queryString language="json">

<![CDATA[]]>

</queryString>

然后查询某字段，比如上面的 name:

<field name="name" class="java.lang.String">

<property name="net.sf.jasperreports.json.field.expression" value="reportCover.name"/>

<fieldDescription><![CDATA[reportCover.name]]></fieldDescription>

</field>

可以对比上面的代码，这里用了JSON查询语句，reportCover.name

简单的JSON查询语句：

某一个字段：父字段.目标字段

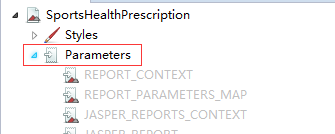
数组中的某个：父字段.目标字段[0]

带条件的：父字段.目标字段(条件)

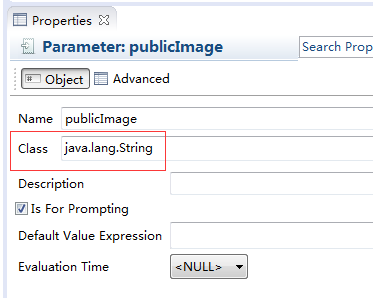
例如： everyHourSportResume(index==3)，index是与everyHourSportResume同级的字段。

可以查看官方例子**http://jasperreports.sourceforge.net/sample.reference/jsondatasource/**

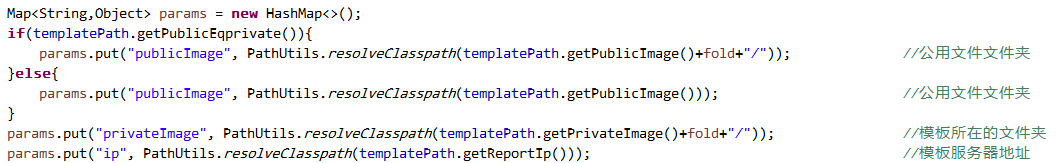
#### 4、参数 Parameters

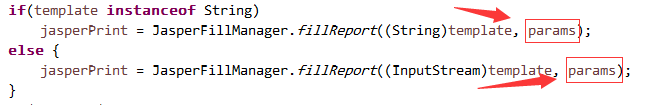


右键新建create Parameter ，在右边属性栏可以设置参数类型：



Parameter是通过后台传入的，后台传入代码：

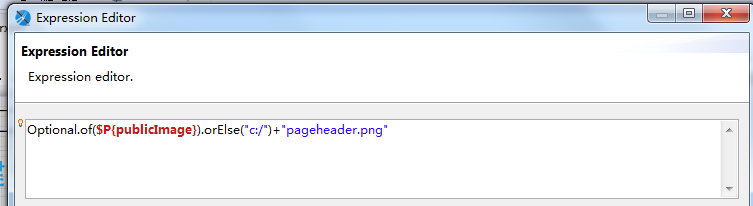




Parameter参数的值是存放在map中的，map中的key对应于页面参数的名字。页面要增加参数，只需要后台map中增加。通常推荐将参数设定为一些通用的、约定的参数/变量，这些参数是不会经常修改，同时支持所有页面。这样的好处是页面的修改与后台分离，后台不需要过多的考虑页面。

**对于页面参数，我作了以下的约定：**

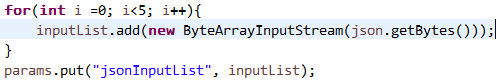
**1、增加String参数publicImage：**页面上的图片可以有两种分类：1、静态的/固定的图片，这种图与模板是一致的，通常在模板设计时就确定，不会轻易修改。所以约定页面增加参数publicImage，这种图片获取路径就以 $P{publicImage}+"pageheader.png"获取。pageheader.png是由设计者自己确定的图片名，publicImage是指公共路径，存放当前模板的所有图片。通常公共图片与当前模板是会存放在一起。



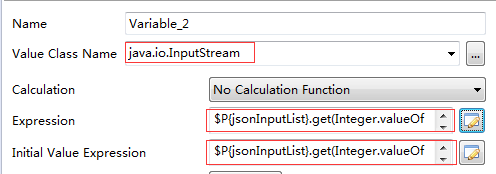
2、**增加String参数privateImage：**另一种图片是可修改的动态的，用户可以随时修改，这个图片就有私人的路径了，这个路径就由运行时传入。当然通常并不太需要这个，这种图片路径可能是直接存放在数据库中的，可以通过Field的数据直接获取。

3、**增加String参数ip：**ip是由于许多图片是以相对路径的形式存放在数据库中，需要提供一个服务器地址。

**4、增加List参数jsonInputList：**list中是输入流，size固定为5（看情况设定适合的大小），内容可以看下面的图片：



这些流是用于向子数据源（Dataset）提供数据的，例如在页面使用table控件，会使用一个dataset，这个dataset使用的数据源就可以设置了。使用方法：设置一个变量Variable

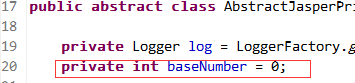


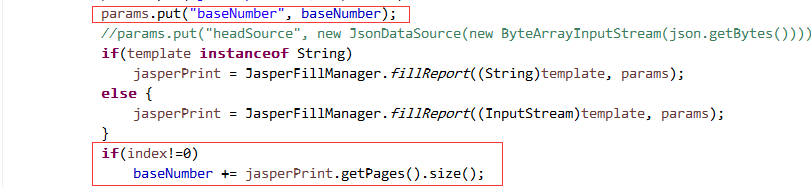
类型为InputStream，初始表达式和表达式为：**$P{jsonInputList}**.get(Integer.valueOf("0"))

具体使用第几个看实际使用。**注意：表达式无法识别数字，想要使用数字必须从字符串转换成数字。**

设置到子数据源详见**1.3.13进阶：给控件的dataset传入数据源**。

**5、增加Integer参数baseNumber：**页面基本页码，由于多个模板才构成一个完整的报告（文件），而且也没有使用子报表进行嵌入，所以页码需要在生成时通过程序控制：



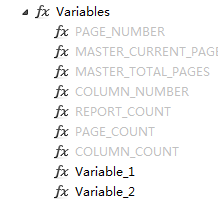


每个模板在生成时会计算自己的页码。但是这是一个模板的，下一个模板又会从头开始计算，比如第一个模板有两页就1，2，然后生成第二个模板有一页就1，总的页码就是1,2,1这样就错误了。通过程序中的基数的累计显示正确的页码。如下：

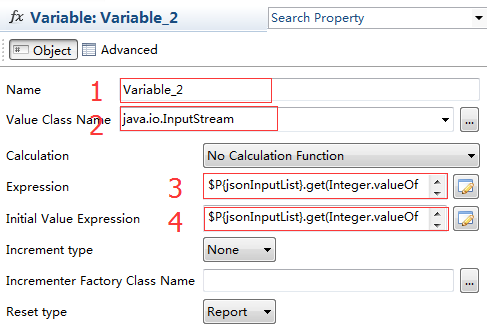
**new** Integer(**$P{baseNumber}**+**$V{PAGE\_NUMBER}**)

**PAGE\_NUMBER是模板自带的变量，不用赋值直接使用。PAGE\_NUMBER会自动计算页码（同一模板下），而baseNumber会在程序中累计页码，作为一个基本值。**

#### 5、变量Variable

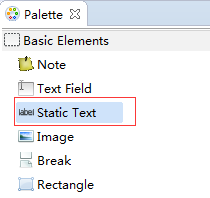


变量是在模板中使用的，右键点击create就可以设置，在右边的属性栏设置参数：



1. 变量名
2. 变量类型
3. 表达式
4. 初始值表达式

#### 6、静态文本Static Text

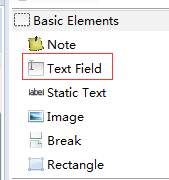


点Static Text 然后点击要放置的页面位置，就可以了

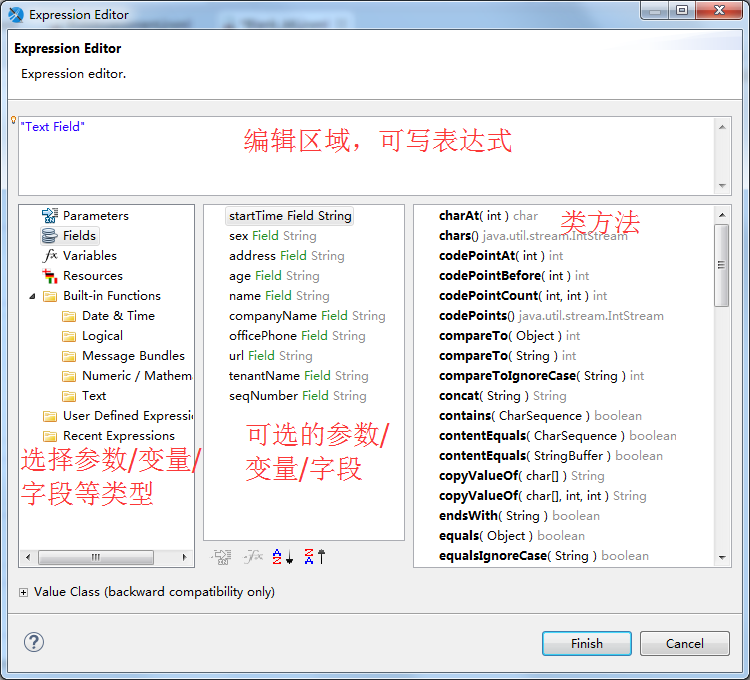


双击就可以编辑，这里的文字在显示时是固定不变的。注意中文要支持中文的字体才能显示，推荐统一使用黑体，后台代码也必须添加对应的字体支持。

#### 7、动态文本Text Field

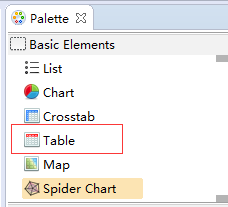


双击添加的Text Field，出现编辑界面



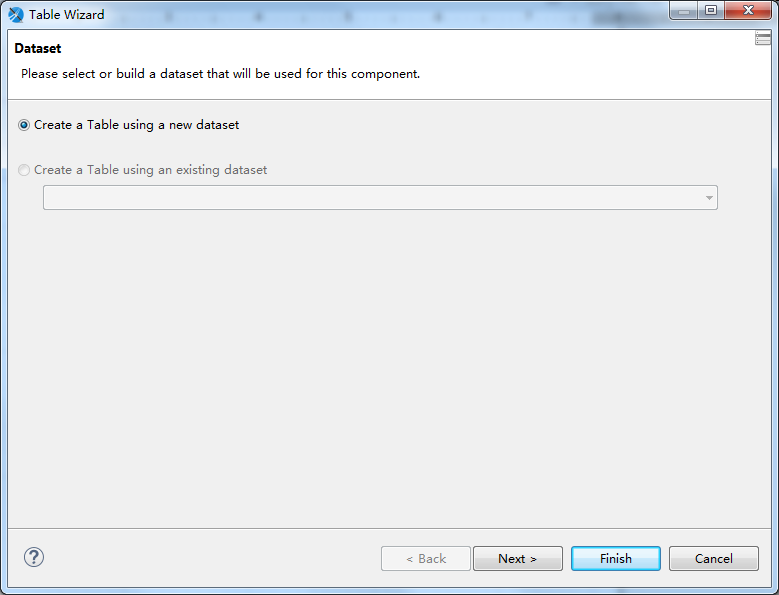
表达式最终会输出字符串，可以认为是在System.out.println()中写表达式。

#### 8、表格table

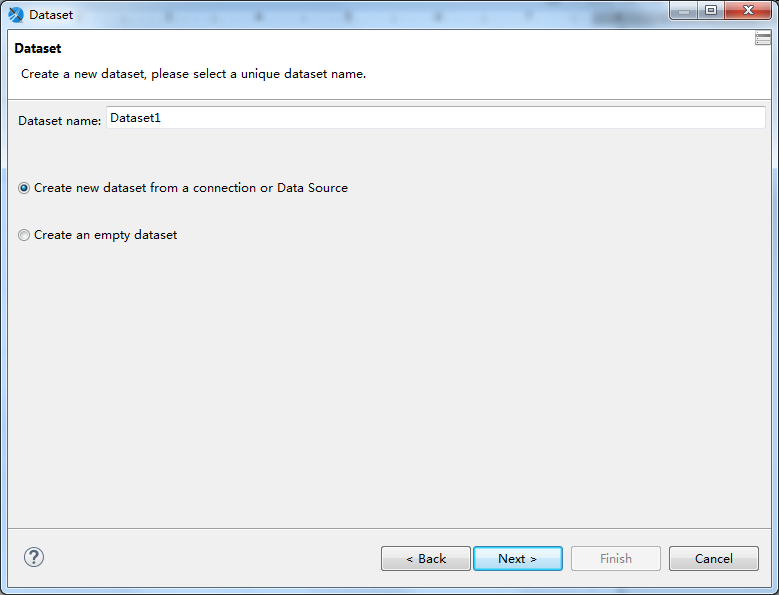


添加步骤：

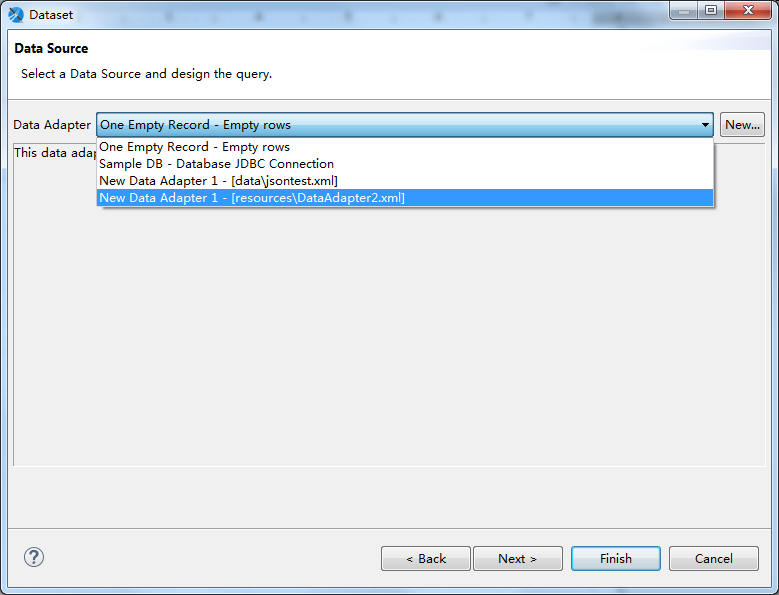
1、首先需要一个数据源，如果已经有了可以选择已有的：



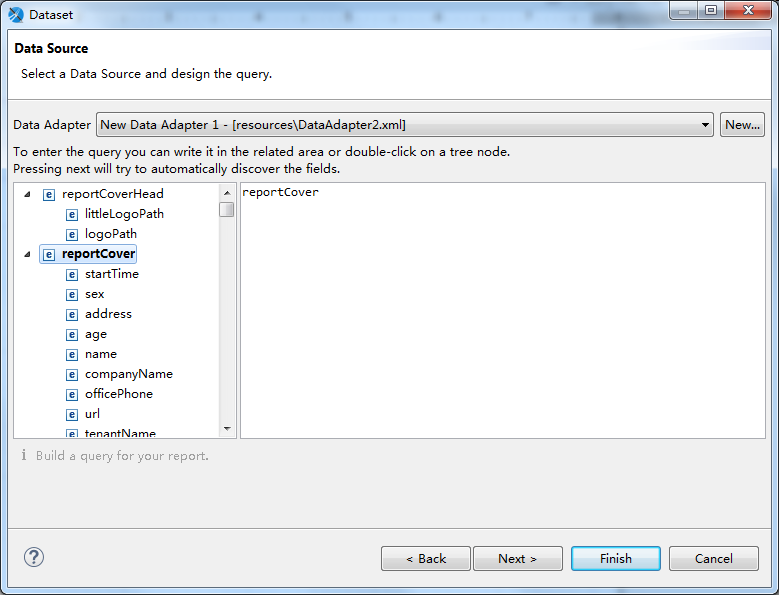
2、起个名字，选择不为空



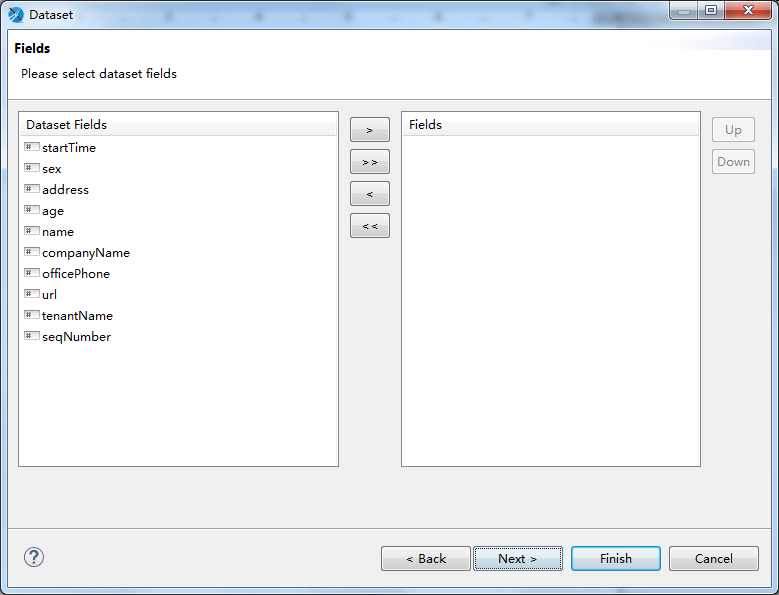
3、选择一个之前建立的数据源模板xml，参考1.3.1，1.3.2



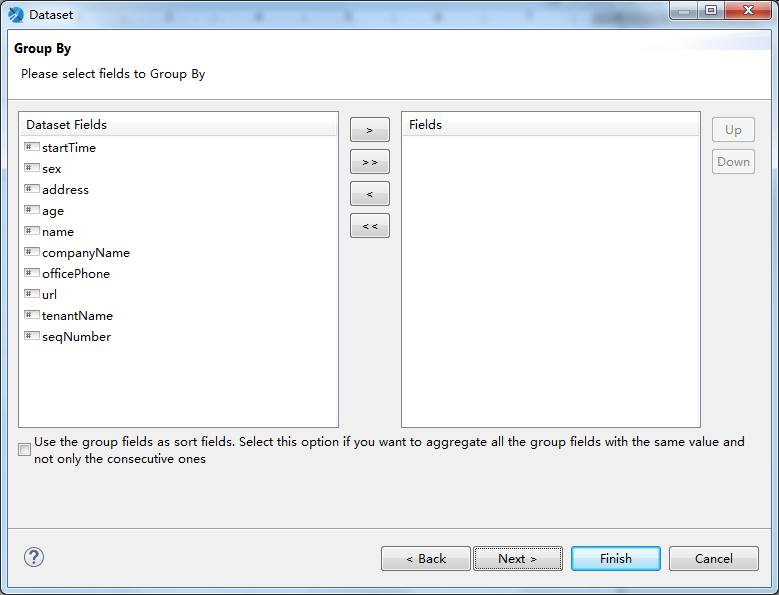
读取需要的根字段：



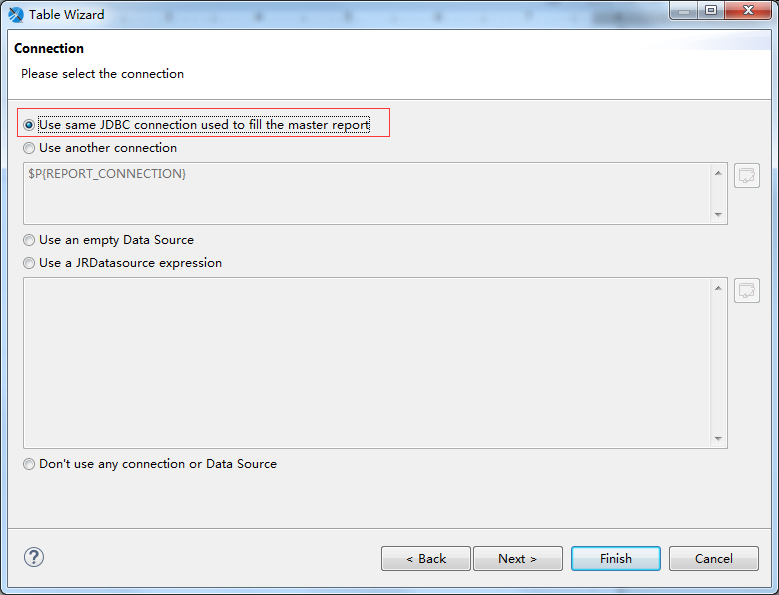
4、根字段下的字段会被读取，选择添加



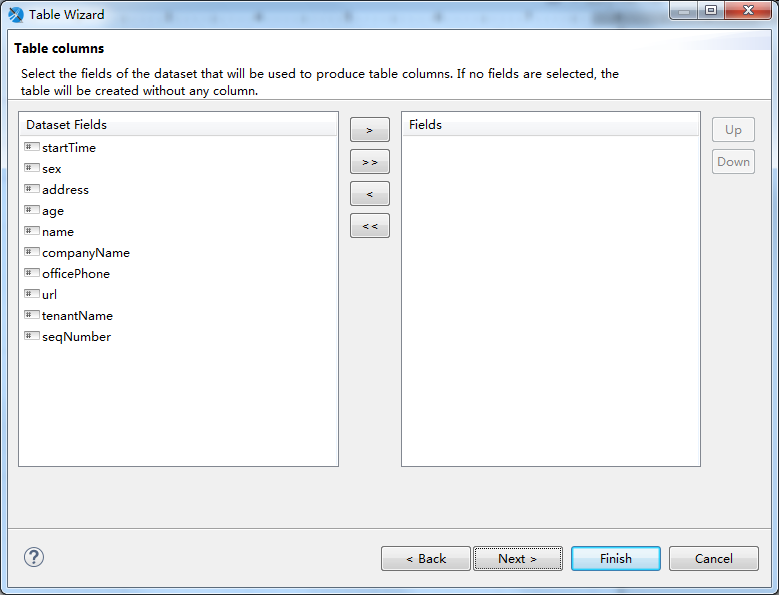
5、选择分组，看需求，表格如果不需要分组显示可以不设置



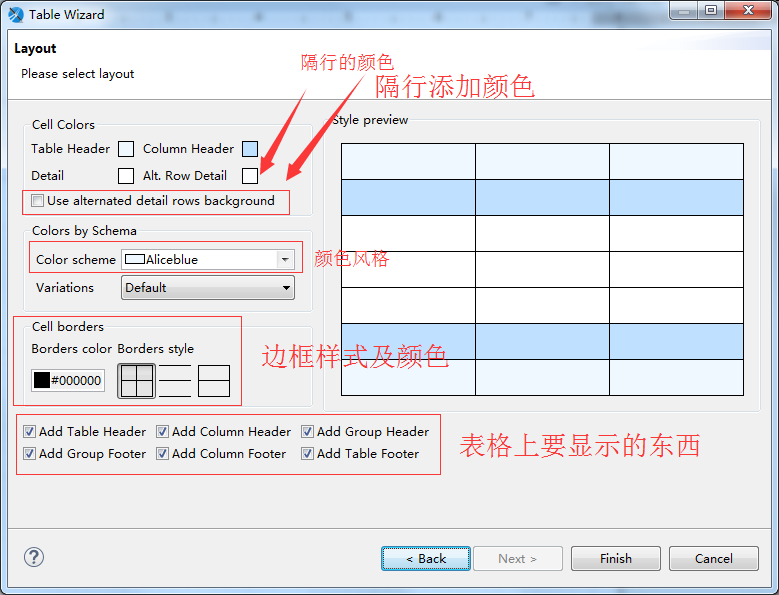
1. 选择数据源的数据来自哪里，选择来自主表



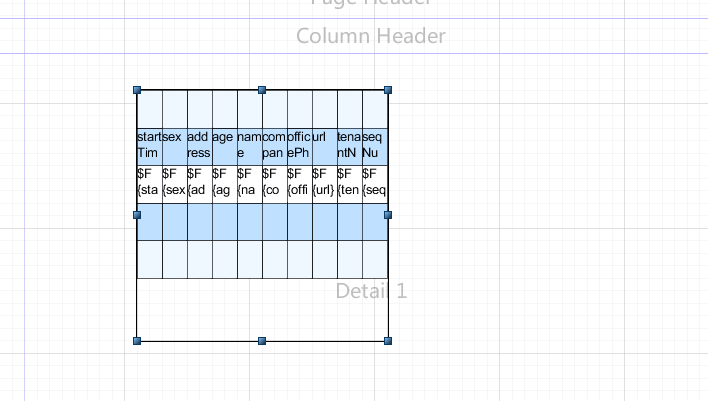
1. 选择table上显示的字段



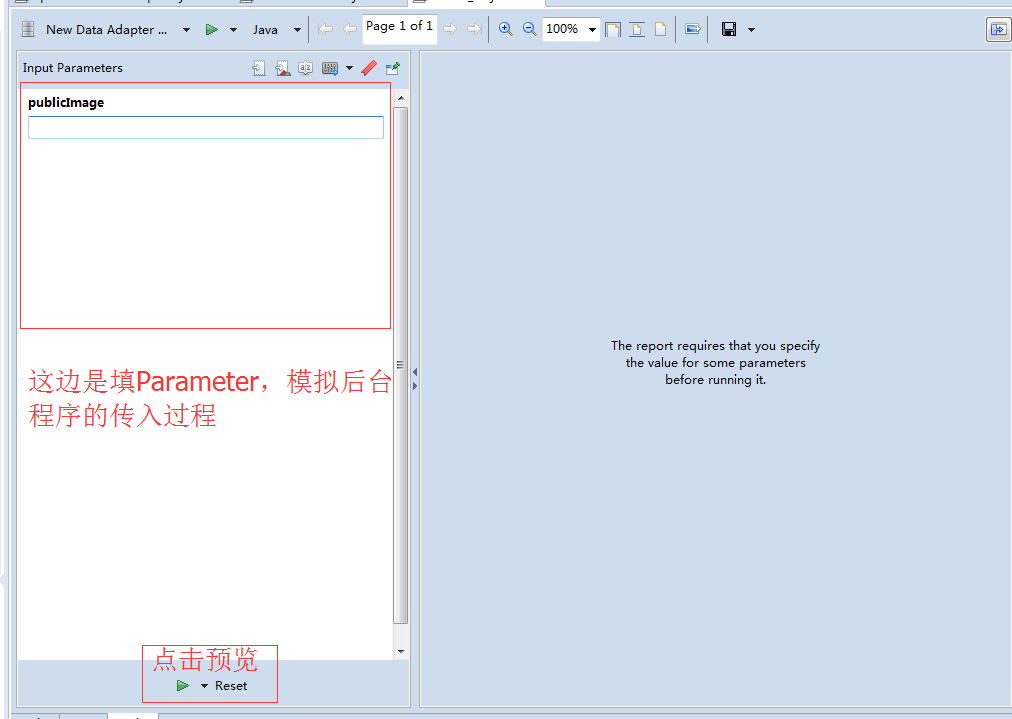
7、表格的布局样式



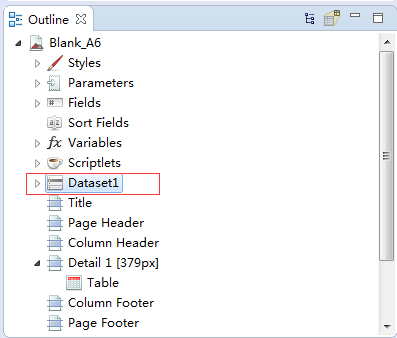
1. 添加完成，可以看到以下的样子



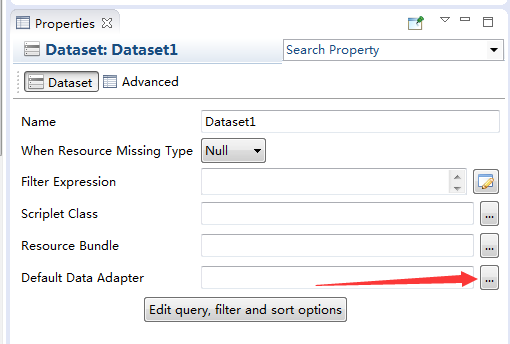
先尝试预览以下效果，点击perview 



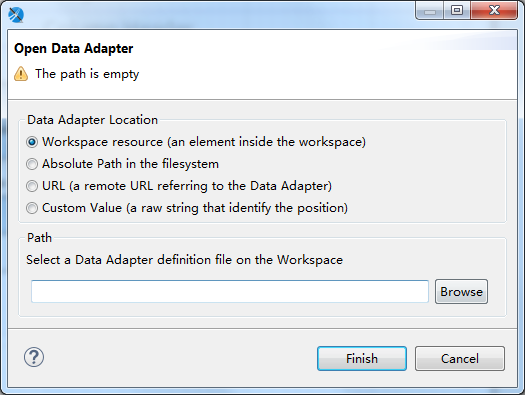
点击绿色小箭头，运行你会发现什么都没有，这个时候你要设置一个可以预览的数据，返回Design界面，查看Outline多出了一个Dataset1，这个是添加table时设置的子数据源点击Dataset1可以在属性界面看到以下信息：



点击上图的按钮



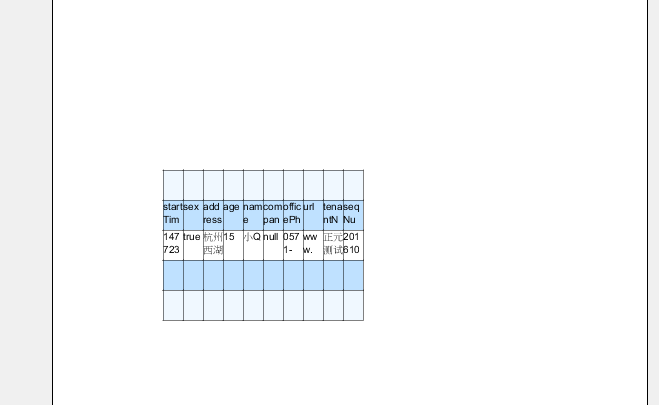
这里是要我们选择一个Data Adapter其实就是之前的数据源模板（1.2.1）



选择好：

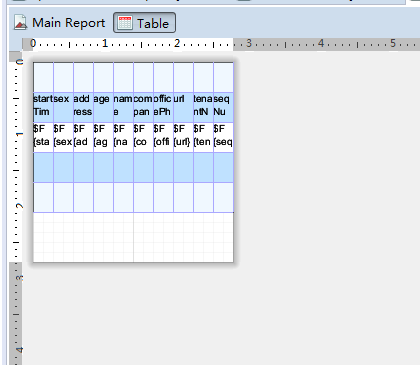


此时再预览效果：

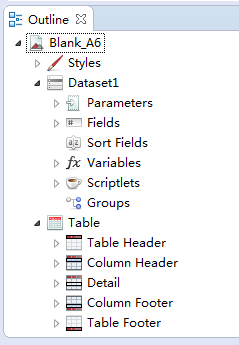


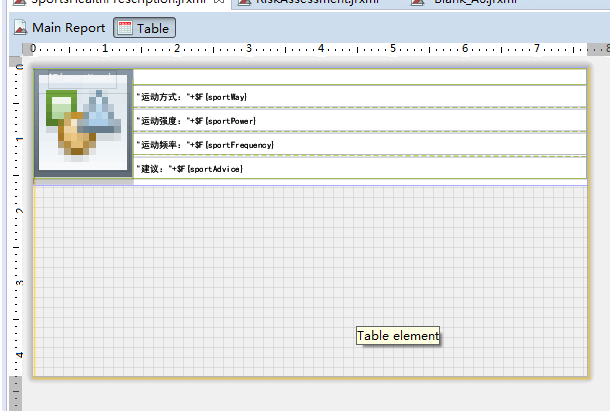
数据就显示出来了。

在Design页面双击table，进行编辑，会进入一个子页面



要设置table的大小，要在Main Report页面进行设置大小，table的设置与主页的设置基本相同。可以在某个单元格中增加文字、图片等各种控件。Table中一样可以设置parameter、field、variable等。不过这些东西都是在Dataset中的。

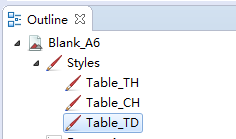




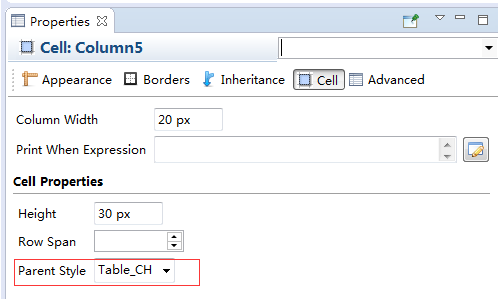
#### 9、表格高级进阶

##### 单元格样式修改：

查看outline中有一个Styles，这些是单元格样式：



可以看某个单元格使用什么样式，选择某个单元格，看parent style：



接下来要修改单元格的样式:

推荐使用修改代码的方式来修改样式，因为图形界面修改可能会发生软件卡死的问题。点击source查看代码：



核心就有几个属性：

**Backcolor :** 单元格背景色

**lineWidth:** 边框宽度

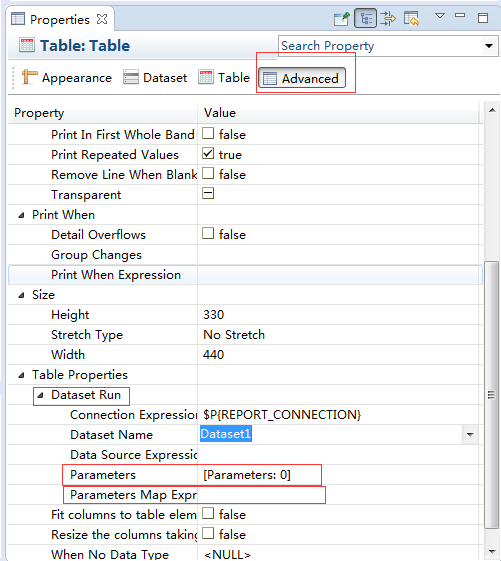
**lineColor:** 边框颜色

边框可以设置上下左右各条边都不一样，设置好代码保持，如果点击design页面为空白，就重新打开模板文件即可。

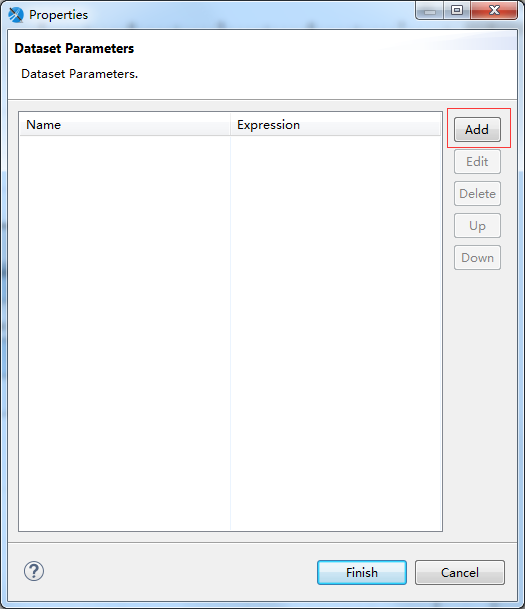
##### 向dataset传递参数：

Dataset的parameter新建一个参数，但是会发现无法将值传递给它。

点击table的属性，点击Advanced，找到Dataset Run，里面有Parameters和Parameter Map Expression



点击Parameters：

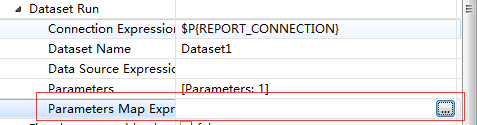


添加参数：

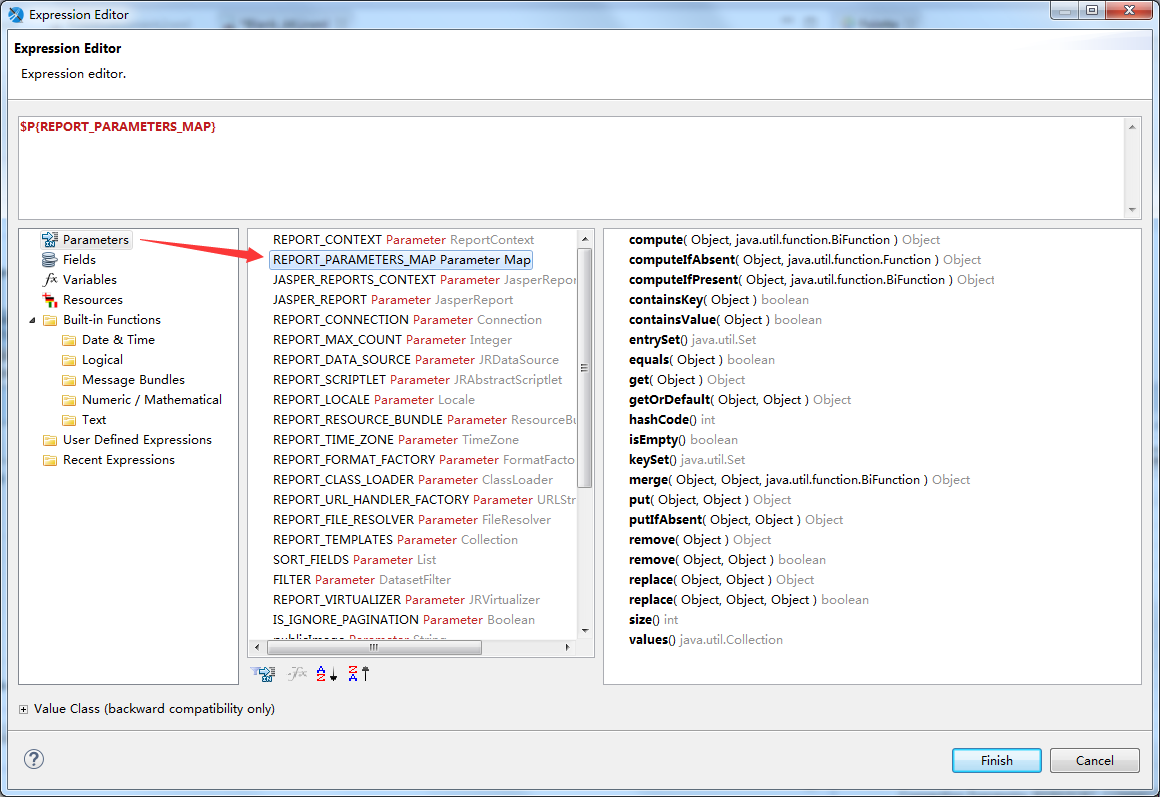


这样就将主表中的Parameter1传递给了table使用的dataset1的publicImage

点击Parameter Map Expression：

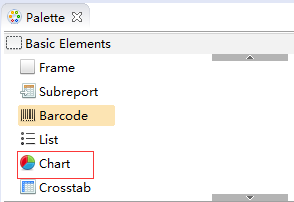


选择一下的值，表示使用主页中的参数集map

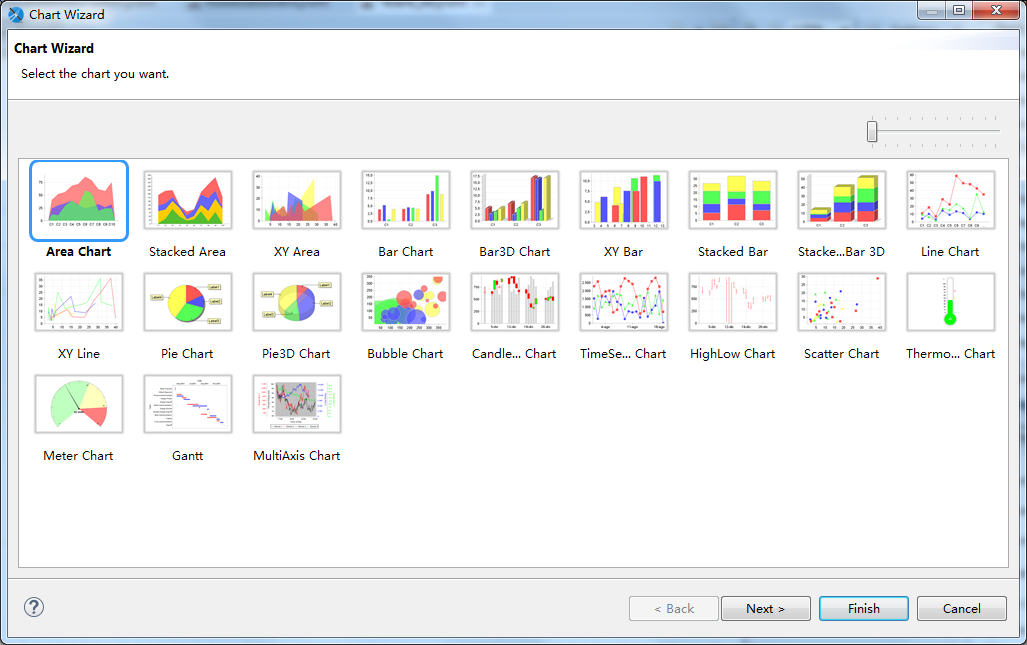


只要是使用dataset作为数据源的控件的数据传递方式都是一致的，可以举一反三，例如list、chart、subreport。

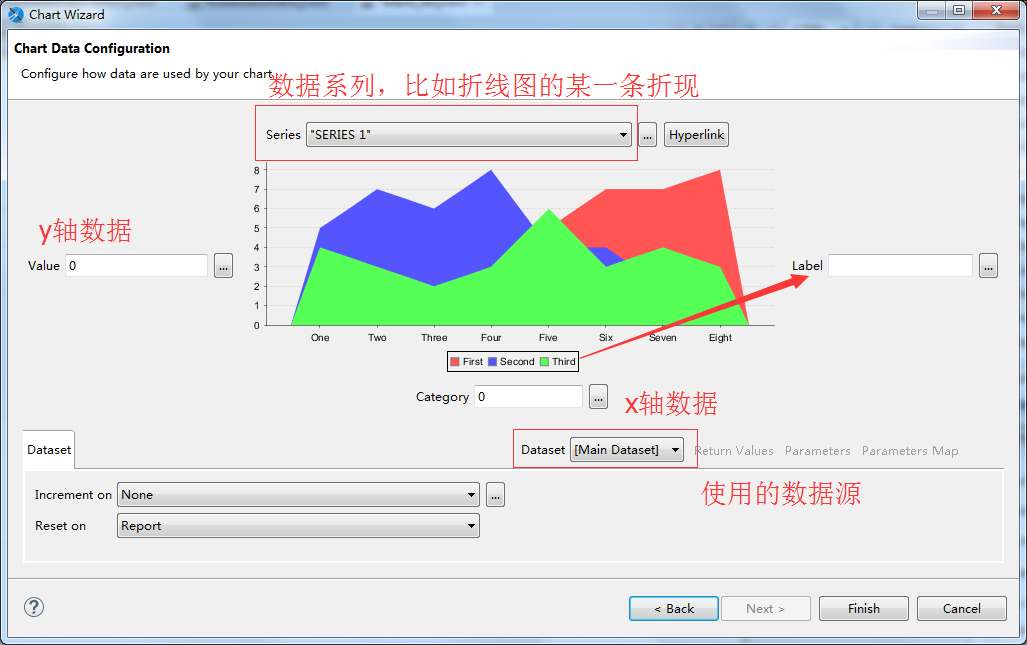
#### 10、图表Chart



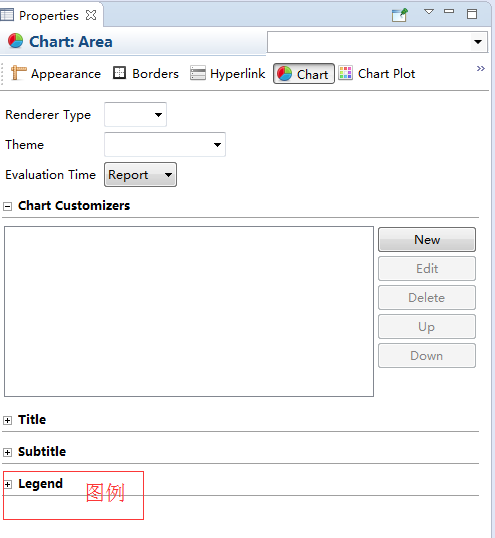
新建一个折线图：



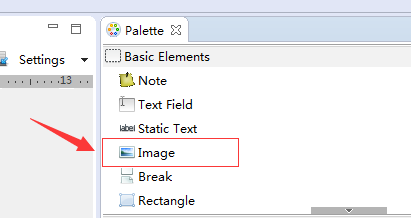
选择显示的数据：

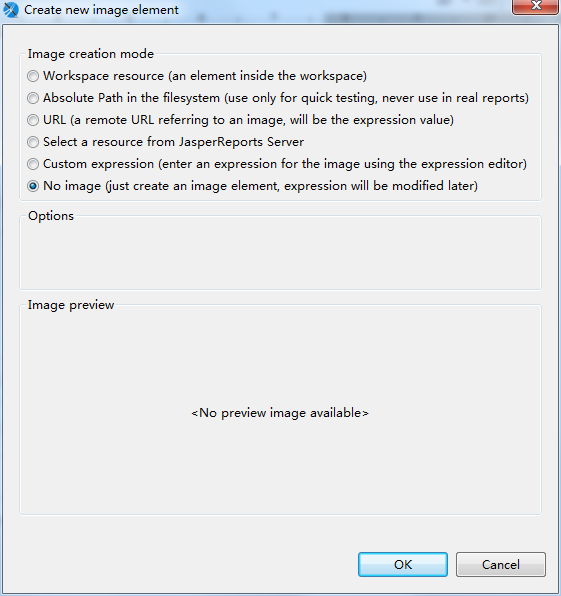


图表的属性设置边框、字体、颜色之类的属性：



#### 11、模板公共图片



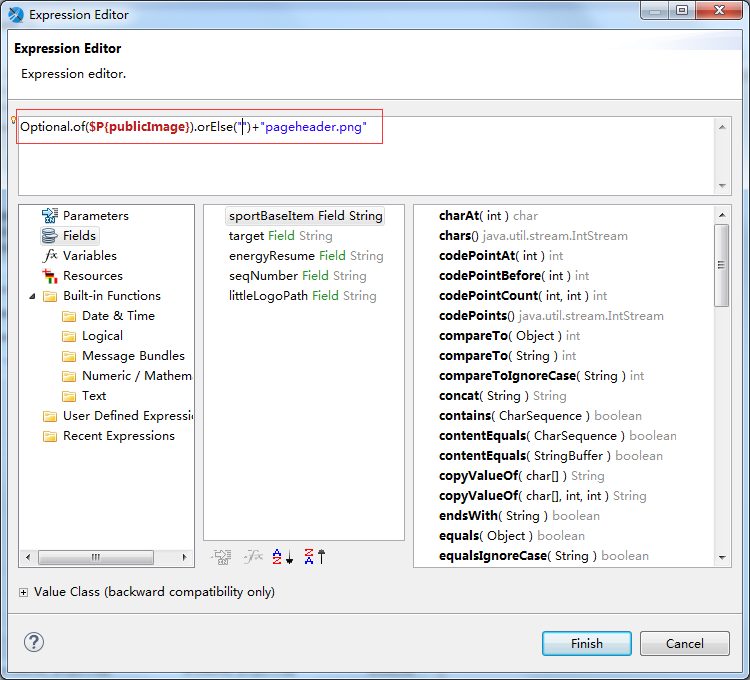


点击放到页面上，然后如上图设置图片路径。

如果该图片不是项目要去设置的，是与模板固定的，约定以以下方式：



路径以：publicImage + “文件名”



publicImage是一个参数，通过生成后台传入。

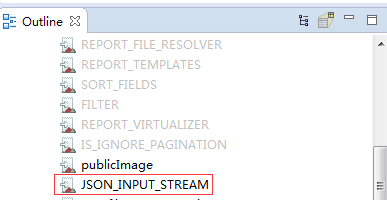
推荐使用 JDK8 的Optional类判断null，设置默认值。

#### 12、进阶：后台注入数据源

现在画好了模板，需要怎么将json数据传入页面呢？

很简单：后台添加参数传入





主页面（相对于table之类的子页面）的JSON\_INPUT\_STREAM不是自己手动建立的，是自动会给你产生的，只要在代码中传入。JSON\_INPUT\_STREAM是一个输入流。

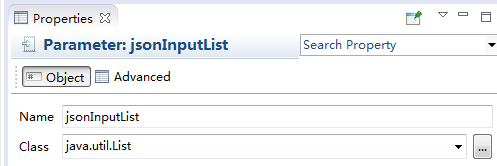
#### 13、进阶：给控件的dataset传入数据源

主页的数据源传入是不用去操作，那么像table使用的dataset是不是也可以直接使用呢？

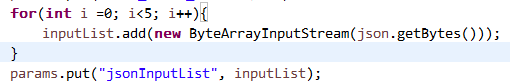
这是不可以的，直接使用通常在生成时会抛出json解析出错，所以要通过模板中配置数据源。经过我几天的测试，发现了一种简单的传递方法，通过主表将参数传递给子数据源即可。

具体操作如下：

1、在主页中增加parameter: **jsonInputList，类型是List**

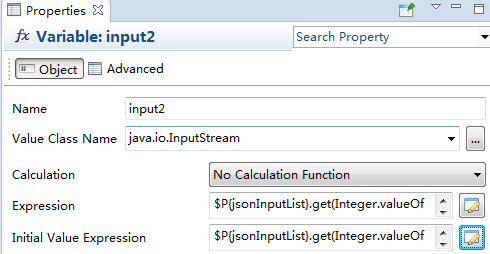


后台代码：



提供了5个可用输入流，这个输入流与主页中的JSON\_INPUT\_STREAM数据是一样的。

2、增加一个variable变量

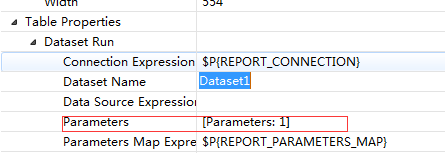


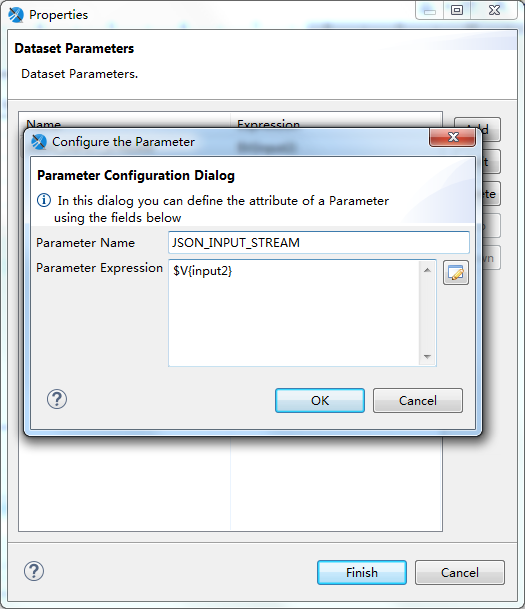
类型是InputStream，表达式与初始表达式都从jsonInputList获取。

例如：**$P{jsonInputList}**.get(Integer.valueOf("1"))

**注意数字的用法**

3、参考**1.3.9的参数传入**，传入一个参数





传入的参数是jsonInputList，来自主页的变量input2，这样在运行时就将数据源传入到了dataset。

## 2、后台生成

### 2.1生成流程



### 2.2添加字体支持

下载Jasperreport的示例工程

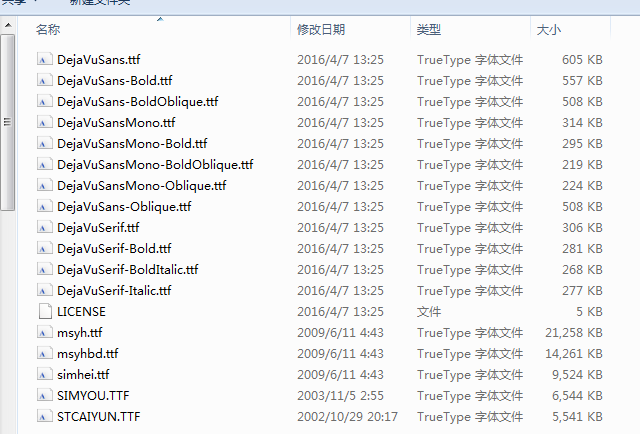


1、添加字体文件

找到以下路径

\jasperreports-6.3.0\demo\fonts\net\sf\jasperreports\fonts\dejavu\

该目录下都是字体文件，需要将要添加的字体文件添加到该目录下，一些常用的字体文件可以在操作系统目录的fonts文件夹下找到。



我已经加入了

黑体文件 simhei.ttf

微软雅黑 msyh.ttf

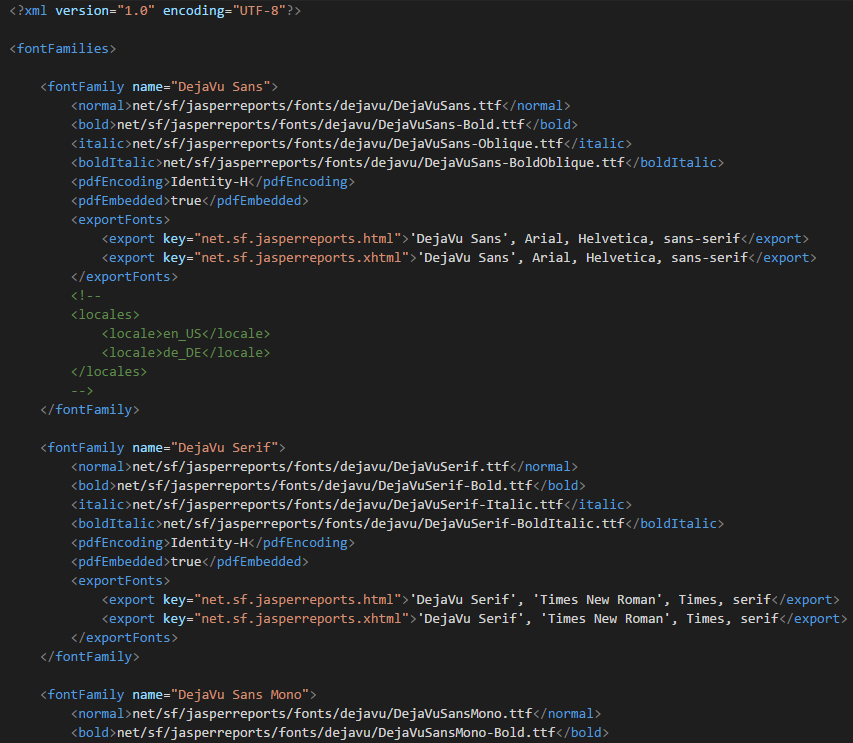
幼圆 SIMYOU.TTF

华文彩云 STCAIYUN.TTF

2、添加字体支持

返回上一个目录：\jasperreports-6.3.0\demo\fonts\net\sf\jasperreports\fonts

打开 fonts.xml文件



这里就描述了对应的字体，格式如下：

<fontFamily name="DejaVu Sans Mono">

        <normal>net/sf/jasperreports/fonts/dejavu/DejaVuSansMono.ttf</normal>

        <bold>net/sf/jasperreports/fonts/dejavu/DejaVuSansMono-Bold.ttf</bold>

        <italic>net/sf/jasperreports/fonts/dejavu/DejaVuSansMono-Oblique.ttf</italic>

        <boldItalic>net/sf/jasperreports/fonts/dejavu/DejaVuSansMono-BoldOblique.ttf</boldItalic>

        <pdfEncoding>Identity-H</pdfEncoding>

        <pdfEmbedded>true</pdfEmbedded>

        <exportFonts>

            <export key="net.sf.jasperreports.html">'DejaVu Sans Mono', 'Courier New', Courier, monospace</export>

            <export key="net.sf.jasperreports.xhtml">'DejaVu Sans Mono', 'Courier New', Courier, monospace</export>

        </exportFonts>

    </fontFamily>

<fontFamily name="黑体">字体的名字

<normal>net/sf/jasperreports/fonts/dejavu/simhei.ttf</normal>

<bold>net/sf/jasperreports/fonts/dejavu/simhei.ttf</bold>

<italic>net/sf/jasperreports/fonts/dejavu/simhei.ttf</italic>

**对应的字体文件路径**<boldItalic>net/sf/jasperreports/fonts/dejavu/simhei.ttf</boldItalic>

所以添加黑体如下：

<fontFamily name="黑体">

        <normal>net/sf/jasperreports/fonts/dejavu/simhei.ttf</normal>

<bold>net/sf/jasperreports/fonts/dejavu/simhei.ttf</bold>

<italic>net/sf/jasperreports/fonts/dejavu/simhei.ttf</italic>

<boldItalic>net/sf/jasperreports/fonts/dejavu/simhei.ttf</boldItalic>

<pdfEncoding>Identity-H</pdfEncoding>

<pdfEmbedded>true</pdfEmbedded>

        <exportFonts>

            <export key="net.sf.jasperreports.html">'黑体', Arial, Helvetica, sans-serif</export>

            <export key="net.sf.jasperreports.xhtml">'黑体', Arial, Helvetica, sans-serif</export>

        </exportFonts>

    </fontFamily>

找到\jasperreports-6.3.0\demo\fonts\target\ jasperreports-fonts-6.3.0.jar

然后用修改后的文件替换里面的文件

### 2.3报表生成代码

只描述生成部分：

Temppath是模板路径（也可以直接使用输入流传入），params传入页面的参数，source数据源。

JasperPrint print = JasperFillManager.fillReport(tempPath, params,source);

        JRPrintPage page1 = (JRPrintPage) JasperFillManager.fillReport(tempPath, params,source);

        JasperExportManager.exportReportToPdfFile(print, destFileName);

**详细的内容可以参考这篇http://blog.csdn.net/pigdreams/article/details/56680437**