**XMReport 1.1**

**使用说明**

**V1.0**

**目 录**

[1. 简介 1](#_Toc105929286)

[1.1. 关于XMReport 1](#_Toc105929287)

[1.2. 真正的“所见即所得” 1](#_Toc105929288)

[1.3. 真“强大”的表格 2](#_Toc105929289)

[1.4. 支持行分组 2](#_Toc105929290)

[1.5. 无限嵌套 2](#_Toc105929291)

[1.6. 丰富的文本样式，并支持富文本 2](#_Toc105929292)

[1.7. 强大的辅助设计 2](#_Toc105929293)

[1.8. 支持格式限制的页眉页脚 3](#_Toc105929294)

[2. 基本使用 3](#_Toc105929295)

[2.1. 启动服务 3](#_Toc105929296)

[2.2. 访问设计器 3](#_Toc105929297)

[2.3. 创建“列表式”报表 4](#_Toc105929298)

[2.3.1. 绘制报表 4](#_Toc105929299)

[2.3.1.1. 设置纸张 5](#_Toc105929300)

[2.3.1.2. 绘制表格 6](#_Toc105929301)

[2.3.2. 关于“变量” 10](#_Toc105929302)

[2.3.3. 通过JSON生成报表 10](#_Toc105929303)

[2.3.3.1. 传入JSON数据格式 10](#_Toc105929304)

[2.3.3.2. 设置数据/脚本 11](#_Toc105929305)

[2.3.3.3. 预览报表 12](#_Toc105929306)

[2.3.4. 通过查询数据库生成报表 14](#_Toc105929307)

[2.3.4.1. 设置数据/脚本 14](#_Toc105929308)

[2.3.4.2. 预览报表 16](#_Toc105929309)

[2.4. 创建“卡片式”报表 17](#_Toc105929310)

[3. 报表设计器进阶 18](#_Toc105929311)

[3.1. 加入图片 18](#_Toc105929312)

[3.2. 加入条码/二维码 20](#_Toc105929313)

[3.3. 表格计算 23](#_Toc105929314)

[3.4. 分组报表 24](#_Toc105929315)

[3.4.1. 列头合并分组报表 25](#_Toc105929316)

[3.4.2. 有分组头、分组尾的报表 27](#_Toc105929317)

[3.5. 图形报表 29](#_Toc105929318)

[3.6. 子报表（嵌套报表） 32](#_Toc105929319)

# 简介

## 关于XMReport

XMReport由国内开发者“mosmith”研发，是“国内首款基于HTML5的控件式报表，提供原生的在线模板编辑、维护功能”。

功能全部自主研发，设计，几乎没有第三方依赖，体积小，操作流畅。

支持多种数据源，包括JDBC查询，JavaBean，JSON等，基于可以通过脚本动态创建你所需要的数据源，原生支持多数据集，大大方便了业务数据源的开发，方便集成到现有的系统中。

支持javascript脚本，并支持在脚本中与java运行时交互，不需要学习复杂笨重的excel函数，只需要你会javascript就已足够。

报表产品中最强大的表格编辑功能（没有之一），并支持数据分组。快速开发各种复杂格式的单据。

强大的辅助设计功能，帮你精确设计文档的每个细节，彻底治愈具有强迫症的你。

众多中文样式支持，满足国内复杂文书，报告的需求。

原生自研图表控件，打印清晰，准确。

目前市面上的java报表产品，从国外控件式的报表，到国内同质化的所谓中国特色的Excel报表，多得让人眼花缭乱。国外控件式报表功能多，体积也是超级庞大，C/S架构居多，操作超级复杂，同时也缺少一些对中文的支持，国内类Excel报表操作相对简单，制作Excel类型的统计报表比较方便，但是所并非所见即所得，制作文书，报告类的模板真是生搬硬套，Excel的操作方式真是令人抓狂，缺少控件嵌套支持，还有很重要的一点是，数据源居然仍停留在上世纪的JDBC查询，与我们分布式，云化构架的数据结构格格不入。

## 真正的“所见即所得”

XMReport是真正的“所见即所得”，如下面的截图，你编辑时候可以看整个文档模板的页面，页眉，页脚，页边距，标尺，以及各个控件在页面的位置，另外按住ctrl+滚轮，页面会相应地对页面进行放大，以帮助你精确设计报表。这些是类Excel报表所不具备的。国外一些控件式的报表比如JasperReport，但JasperReport界面过于杂乱，乱七八糟的背景文字暗示着各个奇奇怪怪的功能，层次不清晰，ActiveReport也有类似的毛病。所以XMReport相对而言是最清晰明了的。

## 真“强大”的表格

XMReport采用的是类似Word中的表格，真正做到随心所欲地合并，分割。比如，在设计下面这个表格的时候，用市面上的类Excel编辑器来做话，我想大家大部分时候都在数一行有多少个格，然后在编辑下一行的时候重新调整上面的单元格宽度。笔者在做表格控件的时候也在思考这个问题，毫无疑问，这种类似Excel的假合并，假分割，不能嵌套的表格实现起来要简单太多，但是代价是开发者需要承受种种操作上的不方便，以及功能，样式受限。但通过全新的自主设计，XMReport完全突破了国内所谓类Excel的操作模式，可以毫不谦虚地说，我们的表格控件是报表产品里面最强大的！

## 支持行分组

XMReport支持对行进行分组，用户可以在分组行里面做一些事情比如分组小计，或者设置一些分组头的样式，用户只需要简单几个步骤即可完成分组样式的设置。

## 无限嵌套

相比于市面上常见采用子模板这种即拗口又超级难用的操作方式，XMReport原生支持控件无限嵌套，你可以将一个单元格设置成容器，然后双击它即可进入窗容器内编辑，真正做所“所见即所得”，如下图，几次点击即可完成控件嵌套样式，是不是很酷？

## 丰富的文本样式，并支持富文本

XMReport提供丰富的文本样式设置选项，常见的比如水平，垂直对齐，自动换行，缩进，行高，字/词间距等，另外，XMReport还支持财务单据上常见的填格样式。

另外，XMReport还支持Html格式的富文本，虽然目前还没有支持完备的HTML，但我想它已经能满足大部分的需求了，笔者日后将不断完善对HTML的支持。

## 强大的辅助设计

也许你经常会为了一个像素没有对齐而大发雷霆，很幸运，XMReport有强大的辅助设计功能，它可以帮助你轻松吸附到具有相同坐标，宽度，高度，甚至是间隔的控件。当然，如果你不喜欢的话，你可以关闭它。

## 支持格式限制的页眉页脚

在XMReport，页眉页脚与内容并没有太多的区别，你可以往页眉页脚里面放置任何控件比如图表，表格等。

# 基本使用

## 启动服务

使用XMReport报表服务有两种方式：

* 直接将作者提供的web-report-springboot.zip中的工程编译，形成可执行的jar包，并运行；
* 自己将web-report-springboot.zip工程中的代码集成到现有服务中，访问/report/designer.html页面。

**注意**：本报表工具需要使用数据库存储报表模板、脚本等信息，所以必须有数据库支持。默认需要使用MySQL数据库，如果需要使用其他数据库，需要加入对应数据库的驱动jar包，并修改配置文件。

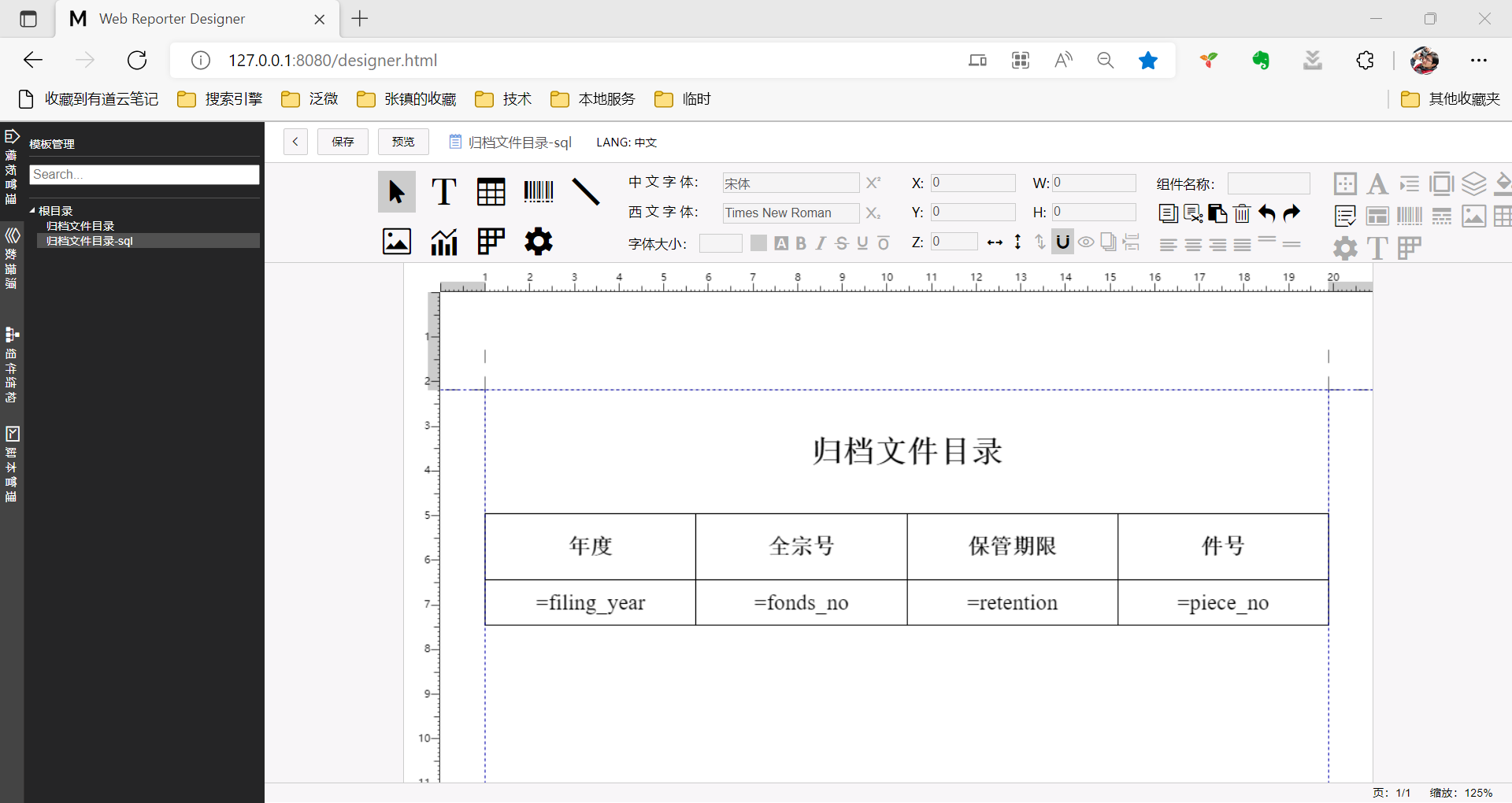
## 访问设计器

本例采用集成到当前“报表服务”中的版本进行说明。

注：报表服务已开源，Git地址为：https://github.com/zhangzhen1979/JIMU-ReportServer

启动本地的“报表服务”后，访问地址：http://127.0.0.1:8080/designer.html

显示报表设计器的界面。

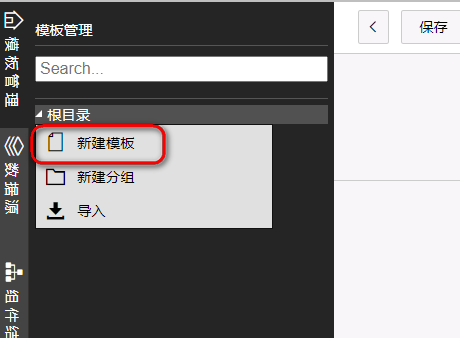


Let’s begin！！！接下来我们就可以创建一个空白报表开练！

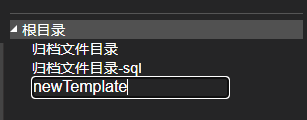
## 创建“列表式”报表

### 绘制报表

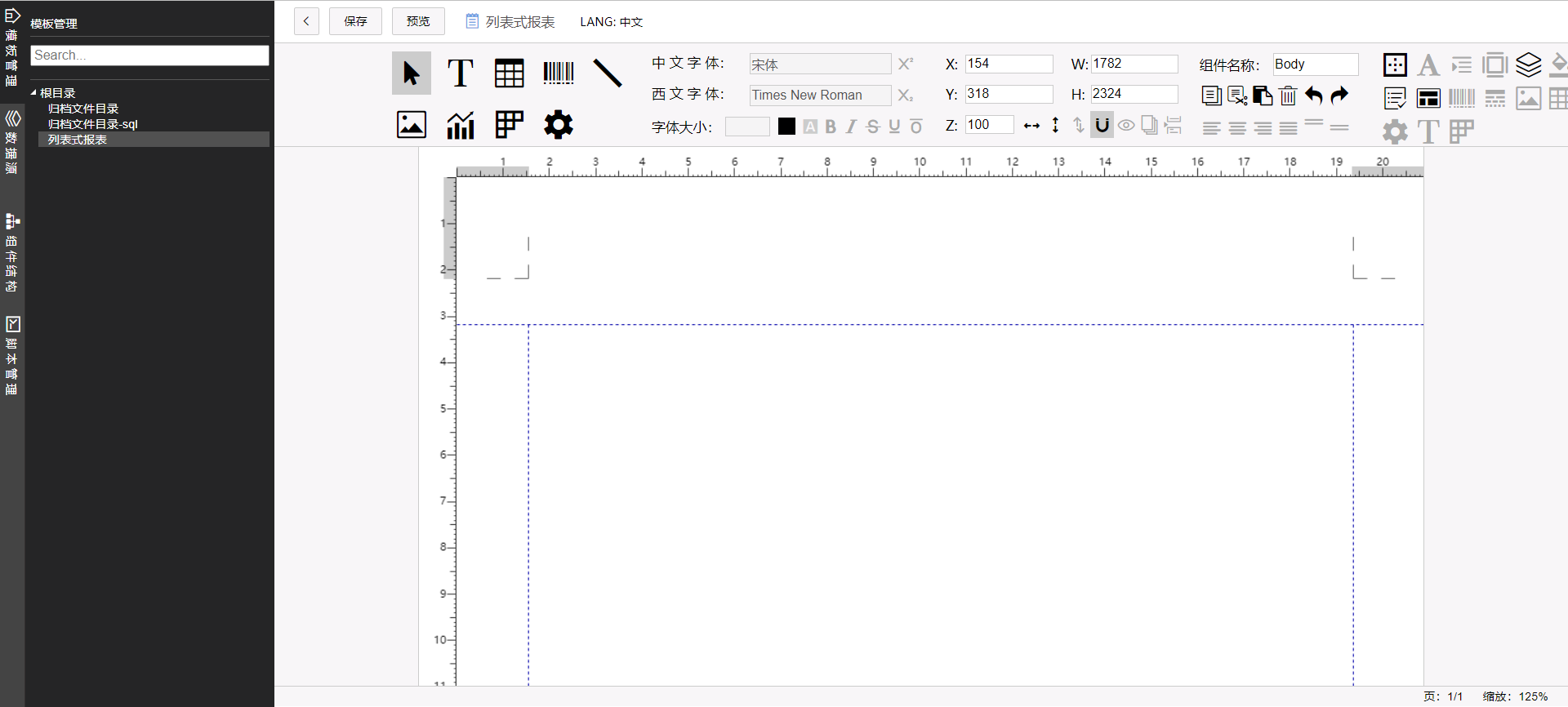
在左侧功能区中，选择【模板管理】，在“根目录”节点上点击鼠标右键，在弹出菜单中选择【新建模板】。



系统会在所选节点下创建一个新的空白报表模板。我们可以修改一下模板的名字，点击其他地方，即自动保存。



双击新建的报表模板名称，右侧工作区即显示一个空白的纸张模拟界面。

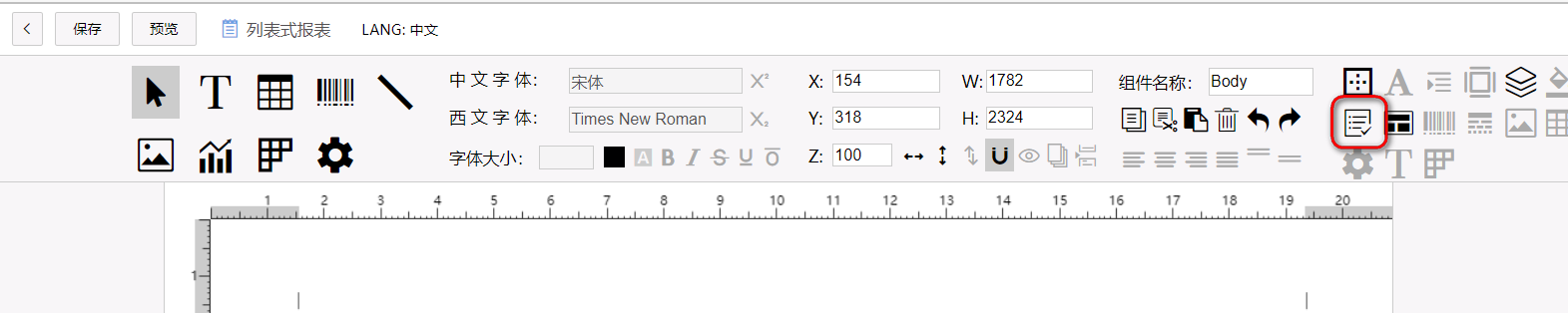


此界面中，顶部是工具栏；工具栏可以通过鼠标的滚轮左右拖动（右侧还有部分设置按钮）。

下方为报表绘制区，默认显示A4纸张的模拟界面。此处的标尺为“厘米”，便于对打印内容进行精确的定位。

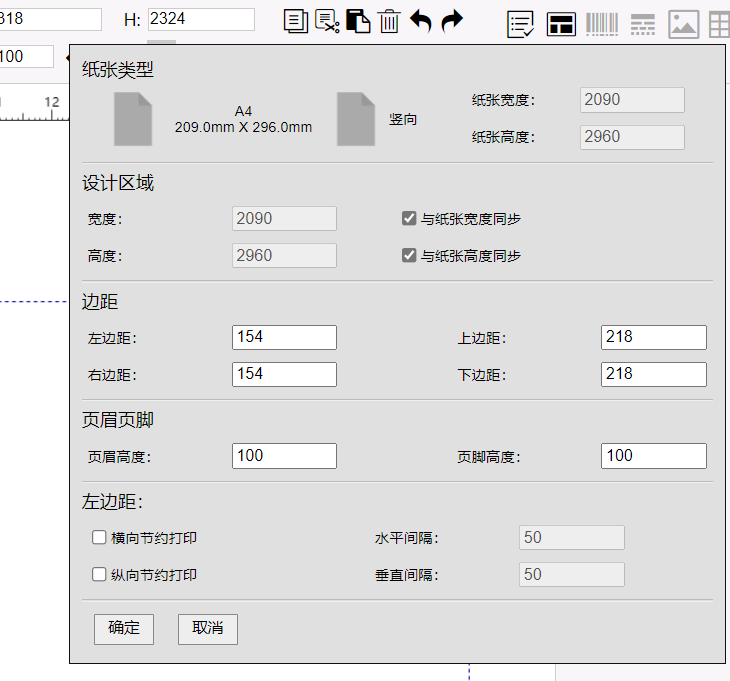
#### 设置纸张

制作报表时，首先要确定报表的纸张大小，这样才方便后续进行打印。



点击右上角的【页面设置】图标，系统显示如下界面。可以在此处选择：纸张类型、纸张方向、边距、页眉高度、页脚高度。

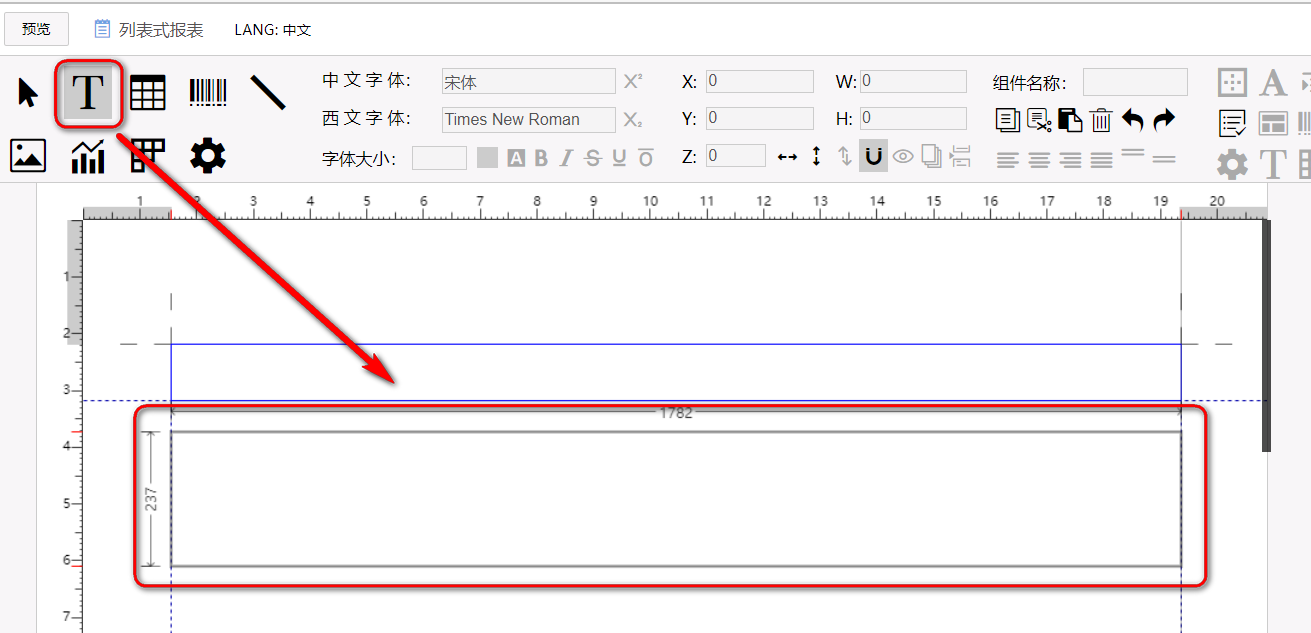
如果使用的是特殊大小的纸张，可以选择“Customized”，自定义纸张的宽度和高度。



#### 绘制表格

先添加报表的标题。

选择工具栏中的【T】（文字）图标



双击绘制出来的框，可以修改文字。

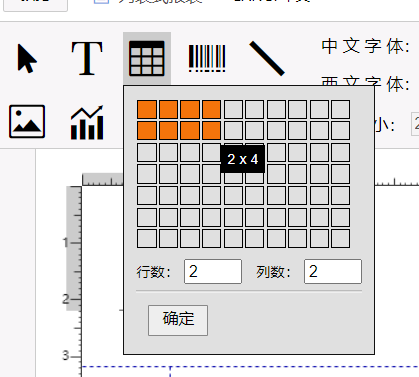
可以调整文字大小、加粗；可以设置组件【横向充满】（也可设置【纵向充满】，本例就不用了……）。



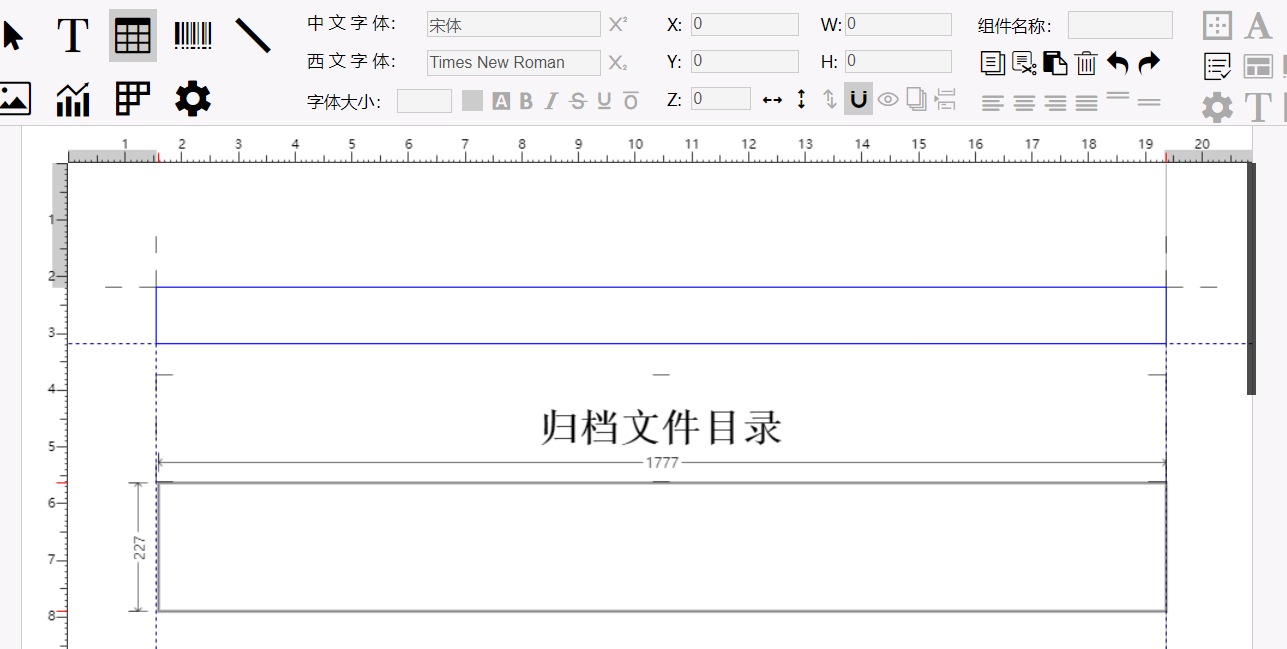
接下来，可以绘制报表的表格了。

本例中，暂定表格为4列（后续可以插入新的列）；需要加入两行，1行表头，1行数据。

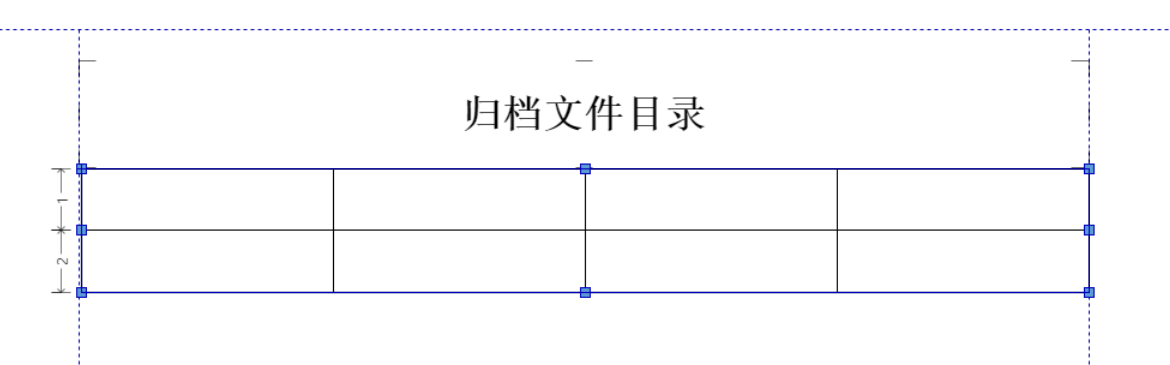
点击工具栏中的【表格】按钮，选择表格的大小（2×4）。



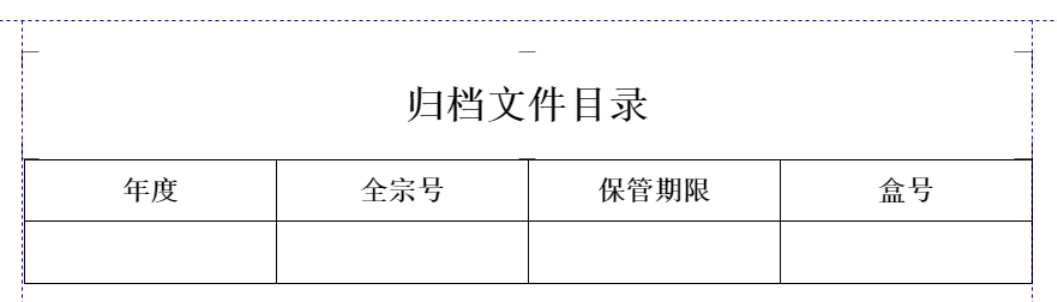
然后拖动鼠标，在工作区中绘制表格。



绘制完毕，效果如下。

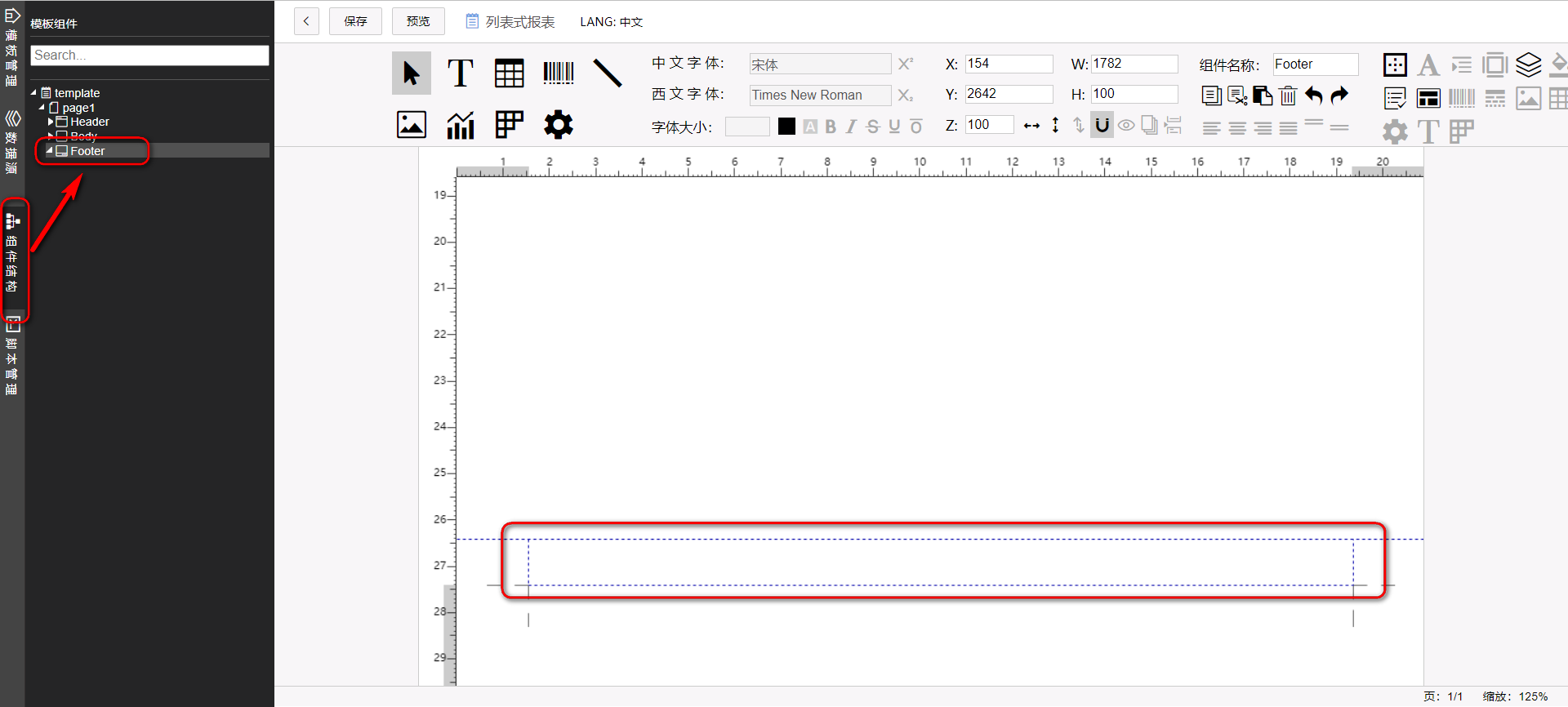


双击第一行的每一个表格，输入表头文字，设置字体大小。效果如下。

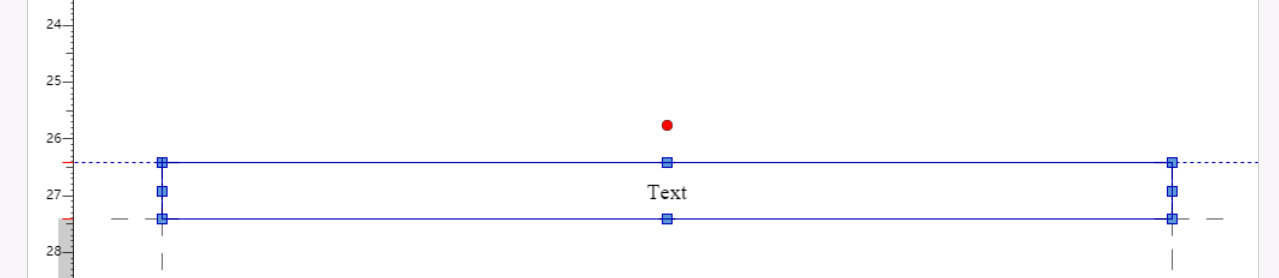


最后，添加一下“页码”。这个操作有些复杂（有Bug……）

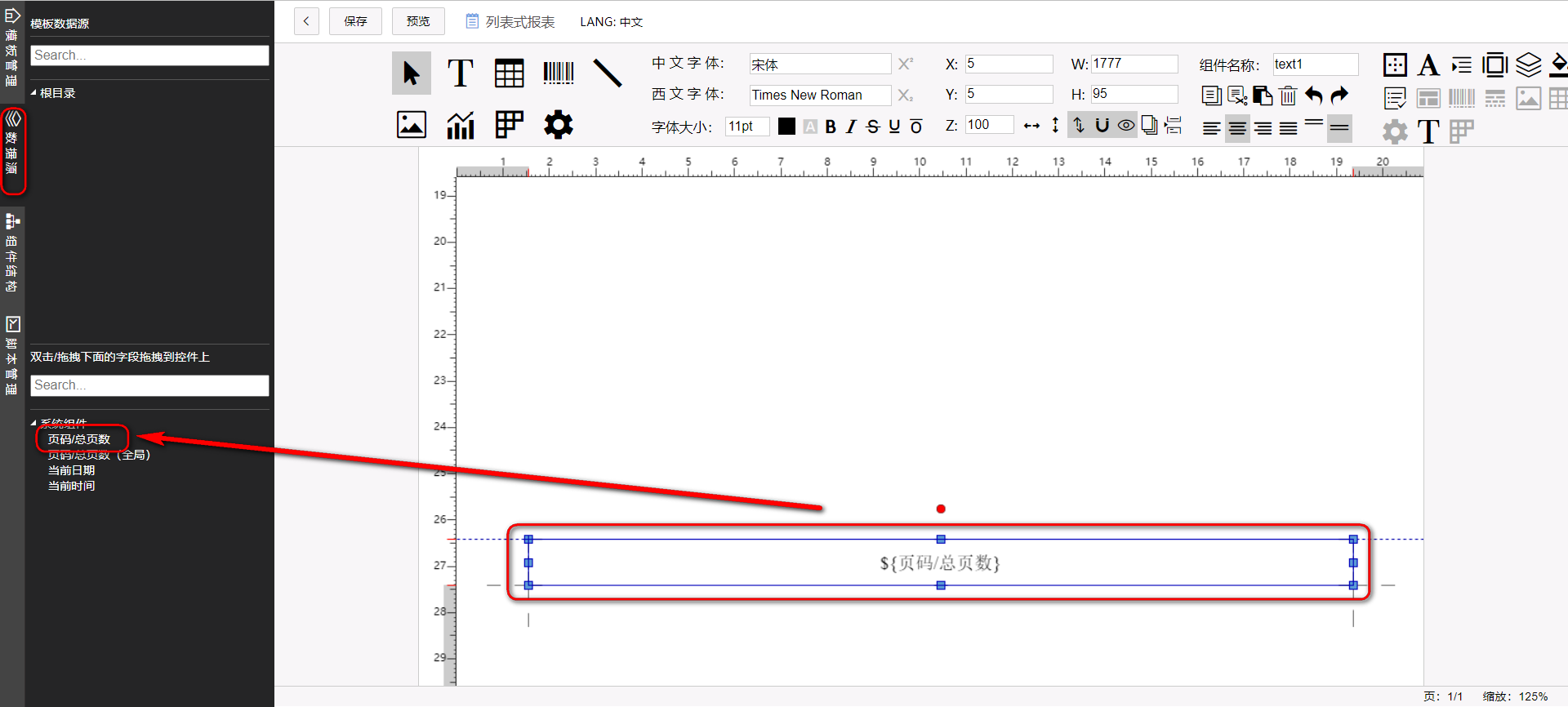
点击左侧功能区中的【组件结构】，在【模板组件】中打开“page1”-“Footer”，此时右侧工作区在页脚区显示虚线框。



在此区域中添加一个【文本】控件：点击【T】按钮，在页脚虚线框中拖拽、释放。位置不准没有关系，通过【横向填充】、【纵向填充】按钮，实现完美居中。不要试图手动调整大小，文本框会跑出页脚，并且回不来了……此为Bug……



保持此文本框为选中状态，点击左侧功能区中的【数据源】，在【系统组件】中用鼠标双击【页码/总页数】，系统即自动将页码赋予此组件。



设置完毕，点击【保存】按钮，保存模板。

### 关于“变量”

在生成报表的时候，我们需要为组件绑定数据；有的时候，还需要在字符串中加入变量，将传入的参数值加入其中（比如组装SQL）。这个时候就需要用到“变量”。

变量名可以是传入JSON的key，也可以是SQL查询后的字段名，还可以是JavaBean的属性名。

XMReport中变量有两种使用方式：

* =变量名：例如：=year。这种方式只能用于为组件绑定数据。例如文本框、单元格绑定传入的数据key。
* ${变量名}：例如：${table}。这种方式可用于组件数据绑定（传入数据的key、SQL查询后的字段名），可以用于字符串中拼装中使用。比如需要替换SQL中的表名，就可以这么写：select \* from ${table}。这里的“table”就是传入JSON中的key，系统自动将传入的数据表名加入到这个语句中，拼装为完整的查询语句。

### 通过JSON生成报表

#### 传入JSON数据格式

使用JSON数据生成报表时，需要先规划传入的JSON数据格式。

本例中，采用如下数据格式：

{

    "reportFile": "template-e7e2ad60-596b-41d4-80f8-65991da9049a",

    "data": [

        {

            "year": "2022",

            "fonds": "维森集团",

            "retention": "30年",

            "box\_no": "0001",

            "barcode": "1234567891"

        },

        {

            "year": "2022",

            "fonds": "维森集团",

            "retention": "10年",

            "box\_no": "0002",

            "barcode": "1234567892"

        },

        {

            "year": "2022",

            "fonds": "维森集团",

            "retention": "10年",

            "box\_no": "0003",

            "barcode": "1234567893"

        },

        {

            "year": "2022",

            "fonds": "维森集团",

            "retention": "30年",

            "box\_no": "0004",

            "barcode": "1234567894"

        }

    ],

    "options": {

        "docType": "PDF",

        "dividePage": **true**

    }

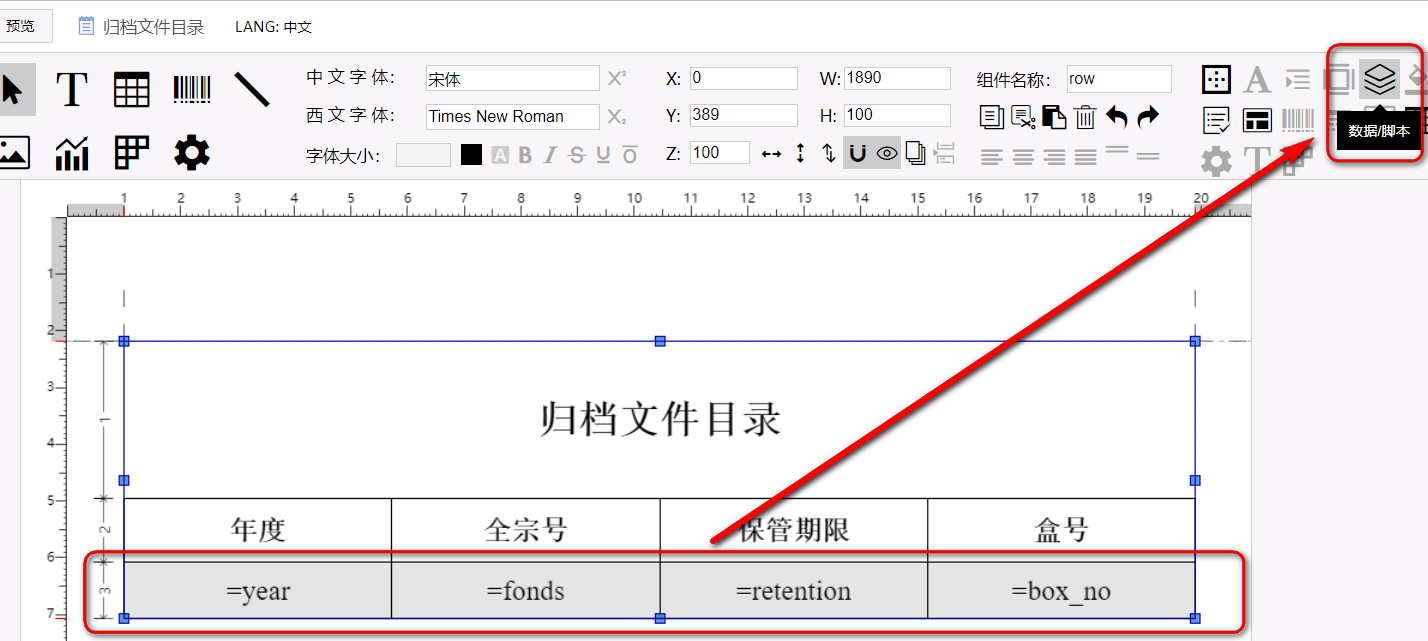
}

说明：

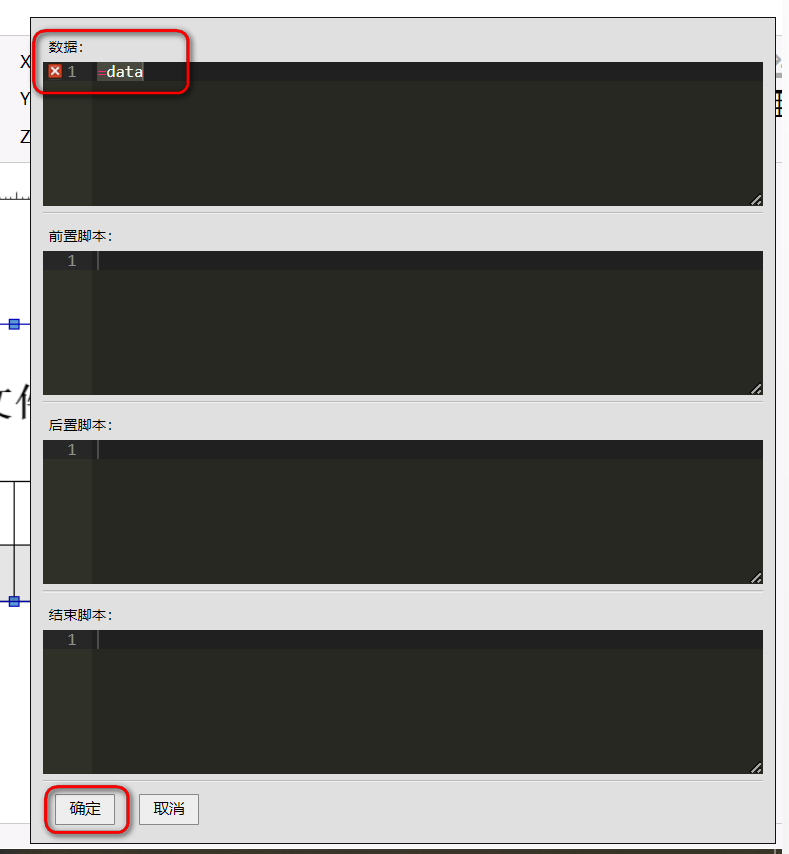
* reportFile：报表模板ID，接口调用时专用。设计器中此项无用，可不写。
* data：JSON数据内容。JSON模式中必填（key与后续设置的“数据”中输入的key对应，可修改），结构为JSON数组，具体内容与报表中单元格设置的key对应。

#### 设置数据/脚本

我们需要将“data”中的数据在表格的第二行循环输出。那么就要先将表格第二行绑定“数据”。



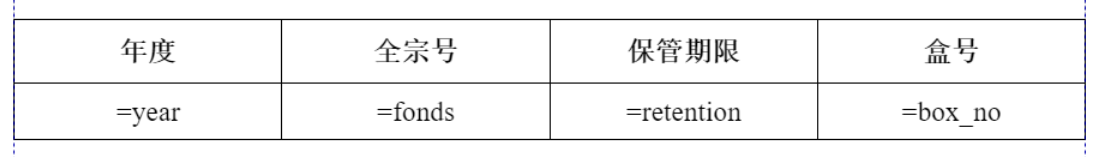
横向选中第二行整行，之后点击工具栏上的【数据/脚本】按钮。系统显示如下界面。



在【数据】框中填写“=data”，即将当前选中的行绑定输入JSON对象中的“data”的数组内容。填写完毕，一定要点击【确定】，保存到模板中。

然后就可以给第二行（需要循环输出的内容）绑定JSON数组中的key了。

双击第二行的单元格，填写每个格中需要获取的key的名称，形如：“=year”。效果如下。

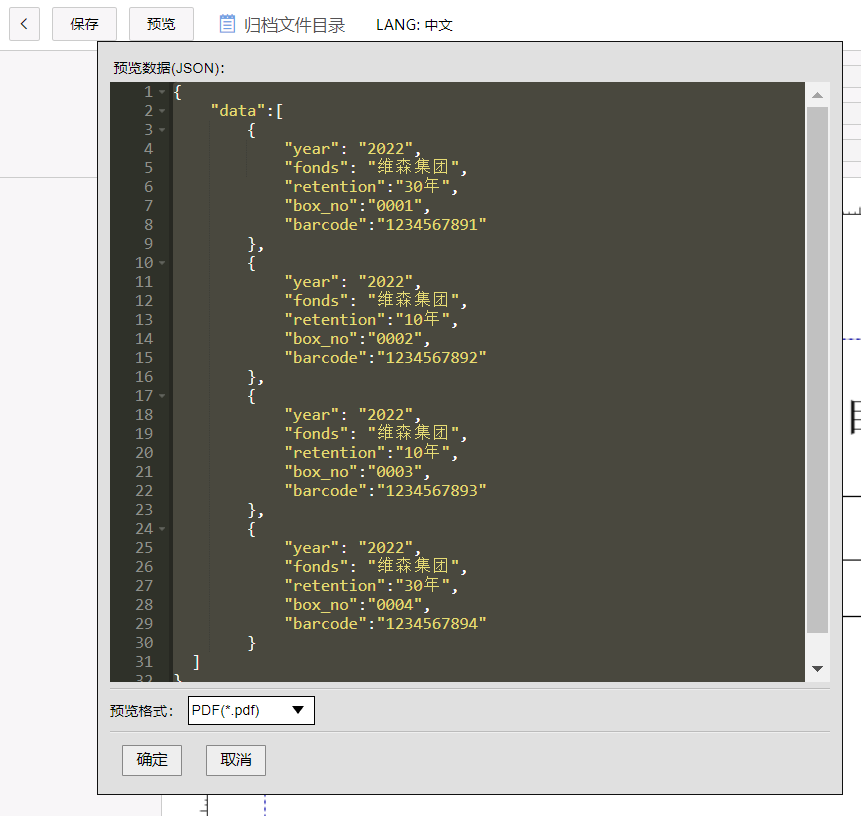


设置完毕，即可点击顶部的【保存】按钮（或Ctrl+S），保存模板。

#### 预览报表

万事俱备，可以试试了。

点击顶部的【预览】按钮，系统弹出如下界面。



在【预览数据（JSON）】框中，可以输入预览所需使用JSON数据。此处将前面定义的JSON内容复制到此处。

可以选择【预览格式】，支持：PDF、Word（docx）、Excel（xlsx）、HTML、Image（png）。

点击【确定】按钮，系统弹出【预览】页面，显示生成的报表内容。

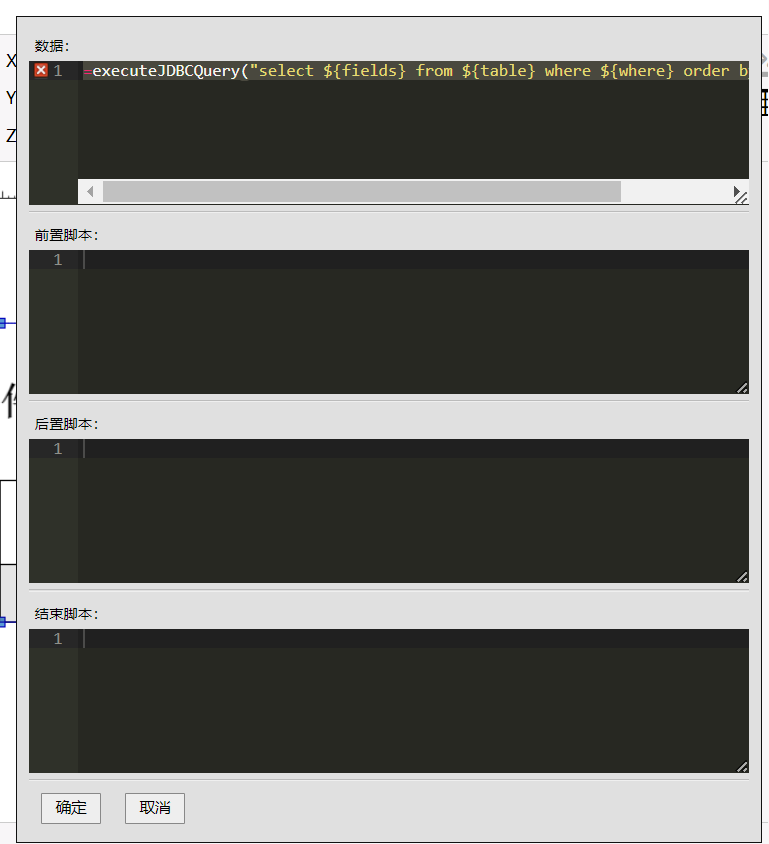


### 通过查询数据库生成报表

报表绘制与前述内容相同，不再赘述。

#### 设置数据/脚本

横向选中第二行整行，之后点击工具栏上的【数据/脚本】按钮。系统显示如下界面。



由于需要执行SQL进行查询，所以需要在【数据】中调用后台的SQL执行方法“executeJDBCQuery”，其中的SQL语句可以带入变量，变量内容为输入JSON中的可以。

本例中，【数据】中输入内容如下：

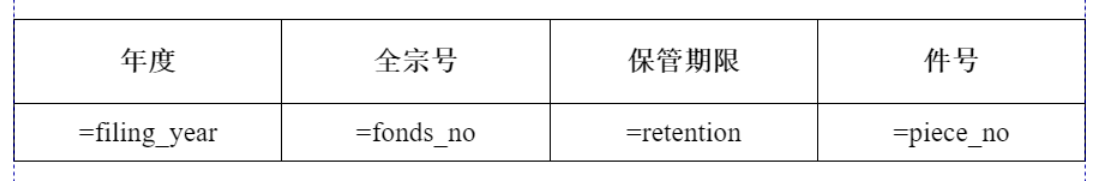
=executeJDBCQuery("select ${fields} from ${table} where ${where} order by ${orderBy}")

其中${fields}、${table}、${where}、${orderBy}均为需要替换的变量，对应输入JSON中的Key。

特别需要注意：fields必须是结果字段名的列表，不允许用\*号（会获取不到字段）。

然后就可以给第二行（需要循环输出的内容）绑定查询结果中的字段名了。

双击第二行的单元格，填写每个格中需要获取的字段名，形如：“=filing\_year”。效果如下。

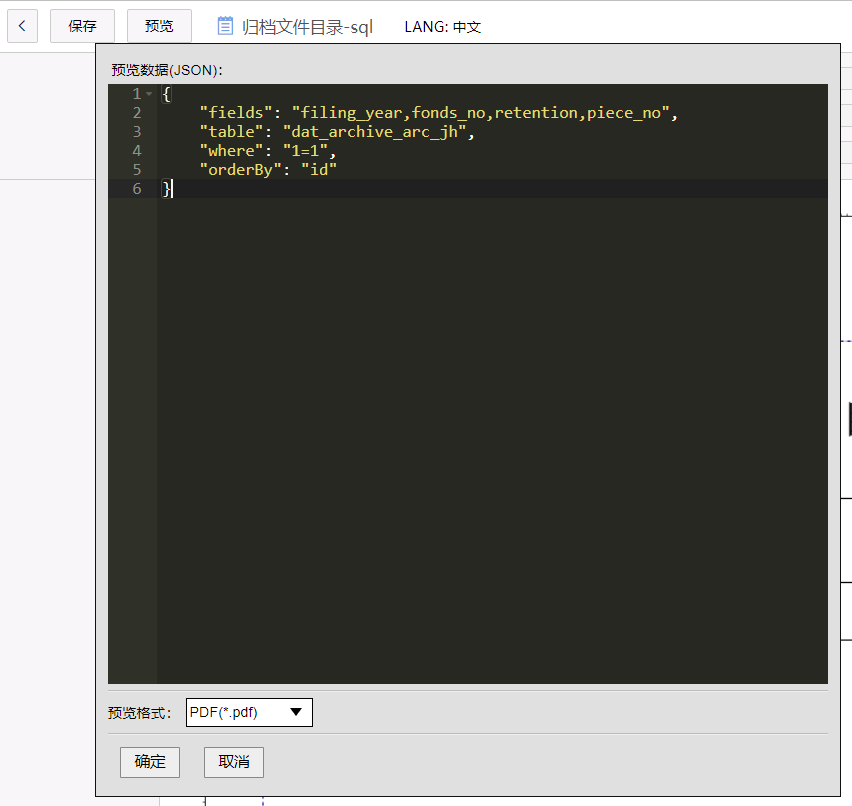


设置完毕，即可点击顶部的【保存】按钮（或Ctrl+S），保存模板。

#### 预览报表

又万事俱备，可以试试了。

点击顶部的【预览】按钮，系统弹出如下界面。



在【预览数据（JSON）】框中，可以输入预览所需使用JSON数据。

本例中使用的JSON如下：

{

"fields": "filing\_year,fonds\_no,retention,piece\_no",

"table": "dat\_archive\_arc\_jh",

"where": "1=1",

"orderBy": "id"

}

点击【确定】按钮，系统弹出【预览】页面，显示生成的报表内容。



## 创建“卡片式”报表

卡片式报表，即为每页只输出一条记录。

处理方式非常简单。

以本例为例，为页面中的组件【数据】绑定到JSON数组/数据库记录结果对象即可。

例如，本例JSON示例中，我们可以选中【表格】组件，点击【数据/脚本】按钮，在【数据】中填写“=data”，然后【确定】-【保存】即可。

系统即会按照数据的条数，自动进行循环，每页生成一条数据。

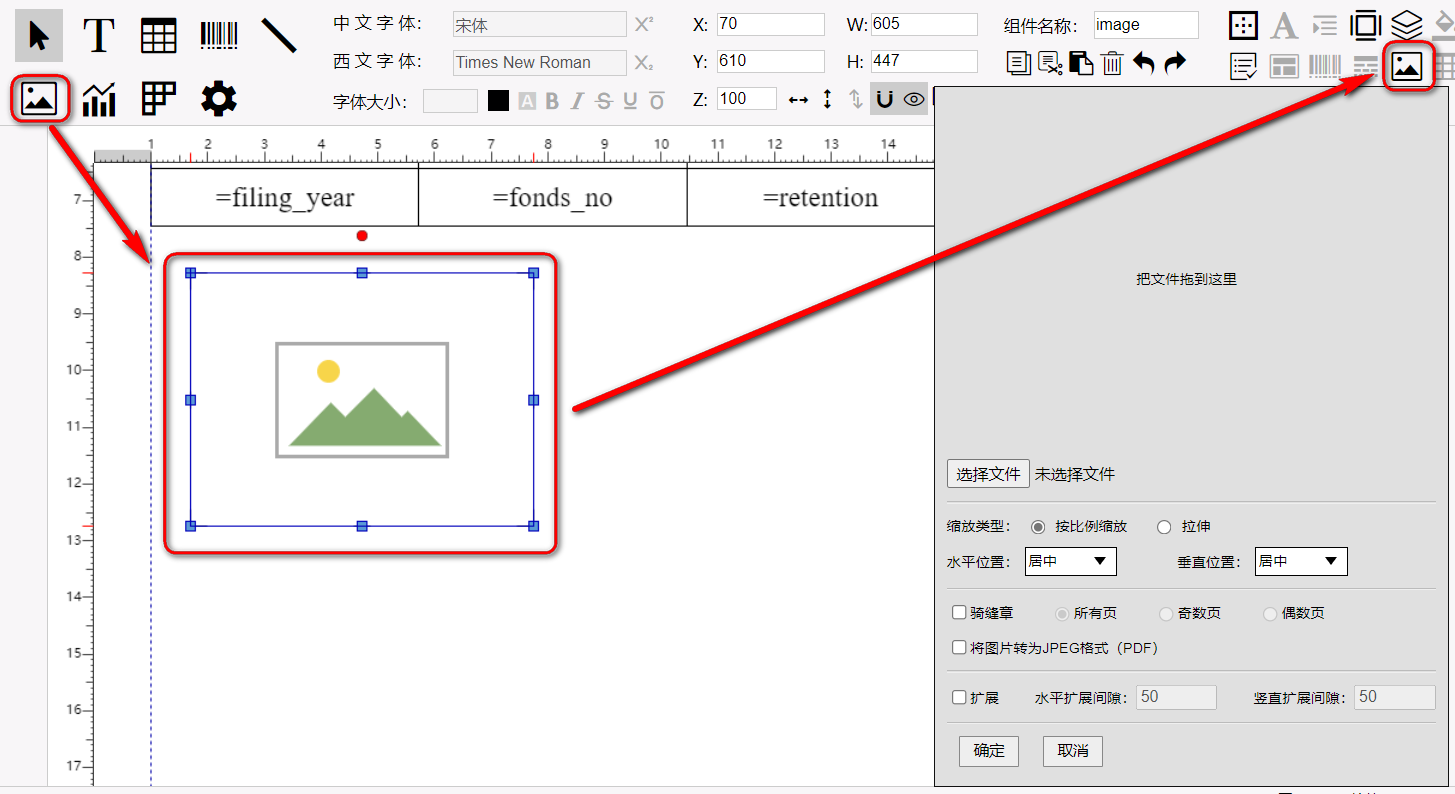
此种方式，建议直接在报表中加入一个【表格】，作为所有工作的“底板”，便于绑定数据、精确定位。

# 报表设计器进阶

## 加入图片

当报表中需要加入Logo等图片时，可以使用【图像】组件。

点击工具栏中的【图像】按钮，在工作区中拖拽鼠标，确定组件大小。

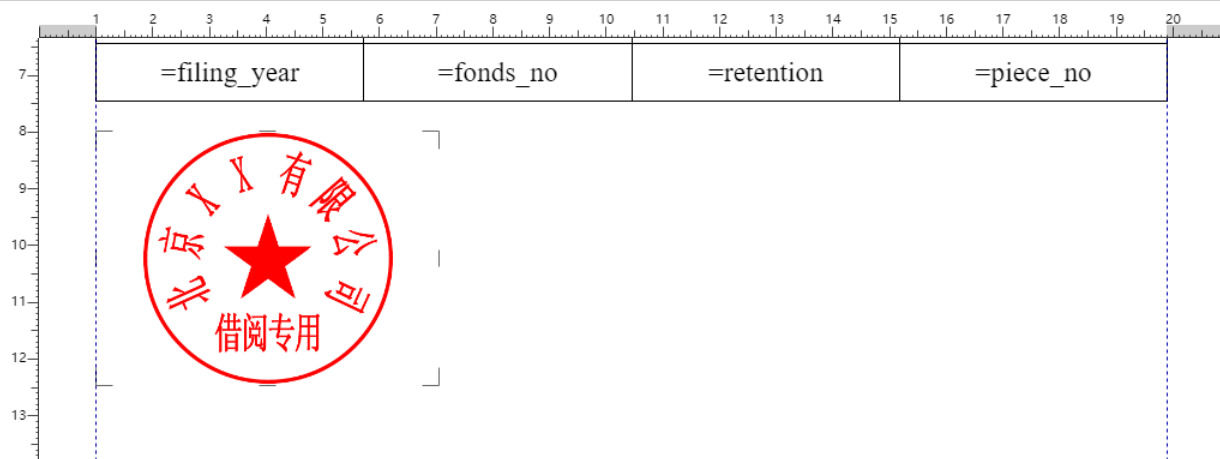


选中新加入的图像组件，点击工具栏右侧的【图片属性】按钮，系统显示图片设置页面。

可以选择本地的图片文件，或者直接将图片拖拽到上传区中。



可以对图片进行精细化设置，点击【确定】，即可看到效果。



预览效果如下。

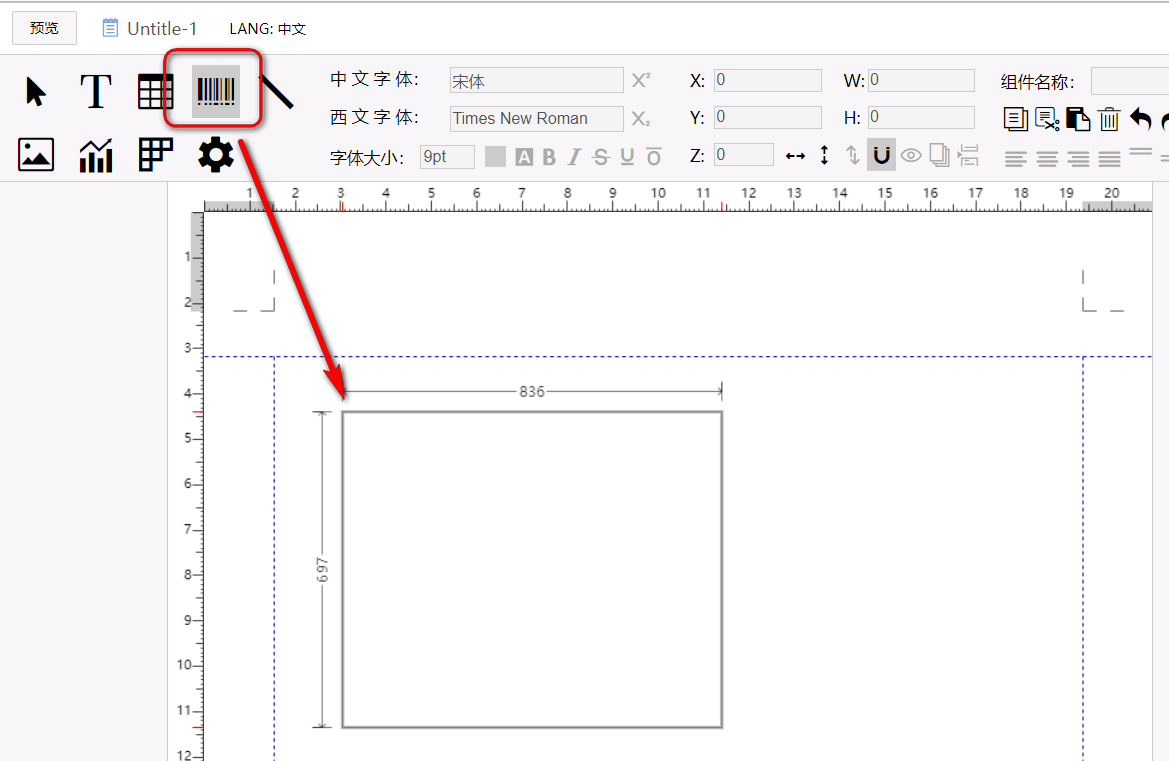


## 加入条码/二维码

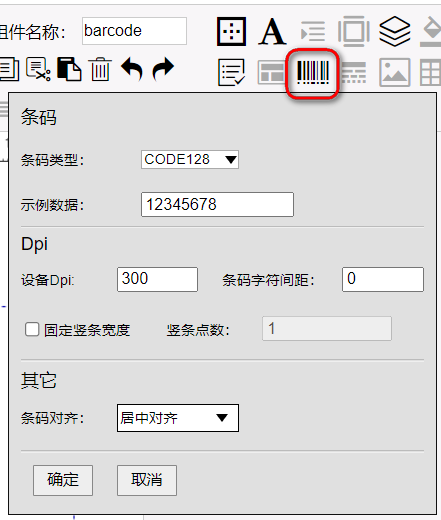
在打印报表的时候，经常会需要在报表中加入条码或二维码，方便打印为纸质报表后，通过扫码枪、App等直接进行扫码查询。

XMReport提供在报表画布上添加【条码】组件的功能，通过此组件，可以选择常用的条码编码或二维码。

在编制报表的时候，可以在组件中找到【条码】组件，然后在画布中拖拽组件大小即可。



放置之后，可以点击右侧属性栏中的【条码属性】按钮，显示如下界面。



【条码类型】中可以更改条码的编码，也可以选择“QrCode”，将组件改为二维码。



【示例数据】只是在编制报表时显示用的，没有实际意义。实际生成条码/二维码，还是需要点击【数据/脚本】按钮，在【数据】区中设置绑定字段才会生效。

设定好后，点击【确定】按钮，设置才会生效。

常用的使用场景有两种：

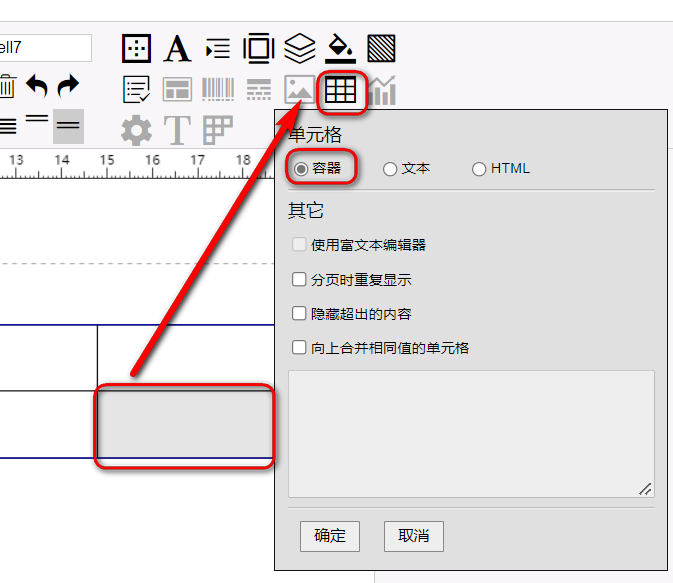
1. 在表单头部打印条码/二维码。此时只要在报表顶部放置【条码】组件；传入的数据中，第一层数据（例如JSON的第一层）可以直接获取到需要生成条码的值即可。
2. 循环输出多个条码。

很不幸，XMReport不支持横向遍历，只支持纵向循环。

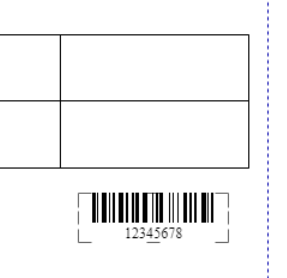
此时，需要使用【单元格】作为“容器”获取值。即，以“表格”获取结果集，以“行”进行循环获取每条数据，以“单元格”作为“容器”放置【条码】组件，获取条码的值。

操作如下：

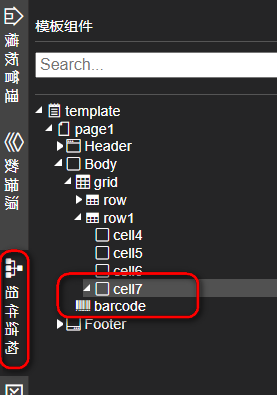
1. 设置表格的行绑定数据为结果集。本例为：=data（操作见前述章节）
2. 先调整好数据行的高度和“条码”单元格的宽度。
3. 选中此单元格，点击【表格属性】按钮，设置为“容器”。点击【确定】。



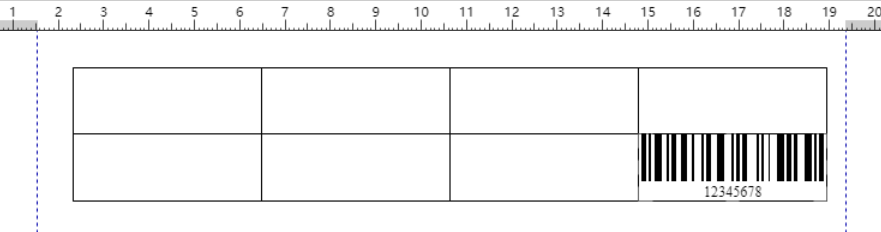
1. 在表格空白处添加一个【条码】组件，调整好宽度、高度，使之可以放在单元格中。点击【数据/脚本】按钮，【数据】区设置为：=barcode（预览数据见前述章节）



1. 点击左侧【组件结构】页，打开【Body】-【grid】-【row1】，可以看到当前的单元格（cell7）前面有一个三角号，代表它已经是个容器了，里面可以放其他的组件了。不过，不能通过直接拖拽的方式放进去。



1. 能看到【barcode】与【grid】同级。需要在【barcode】上点击右键，选择【剪切】，然后在【cell7】上点击右键，选择【粘贴】，即完成组装



点击【预览】，使用前述章节的预览数据，可以看到实际输出的效果如下。



## 表格计算

和 Excel 类似，你可以做一些表格计算操作，比如求行（sum），平均（average），计 数（count），但是，相比于 Excel，它有以下几点不同：

1. 进行表格计算的组件需要放在目标单元格之后，因为被未执行的行在上下文中是不可见的。如果你需要把计算结果放到目标单元格之前，那么你需要在目标单元格之后的单元格的脚本先执行计算操作，然后设置到前面的单元格里面去，如：targetCell.setValue(sum("cellToSum"));

2. 计算目标单元格范围是用单元格的 ID 来表示的，如： sum("cell1', "cell8");

3. 有 4 种类型的统计计算，分别为全局统计、分组统计、当页统计和当前行统计。全局统计会对所有输出的目标单元格进行统计；而分组统计只会对分组内（包括子分组）的单元格进行统计；而当前页统计只会对当前页内的输出的单元格进行统计；当前行统计只会对当前行内输出的单元格进行统计。

1) sum/average/count（全局统计）

2) gsum/gaverage/gcount（分组统计）

3) psum/paverage/pcount（当前页统计）

4) rsum/raverage/rcount（当前行统计）

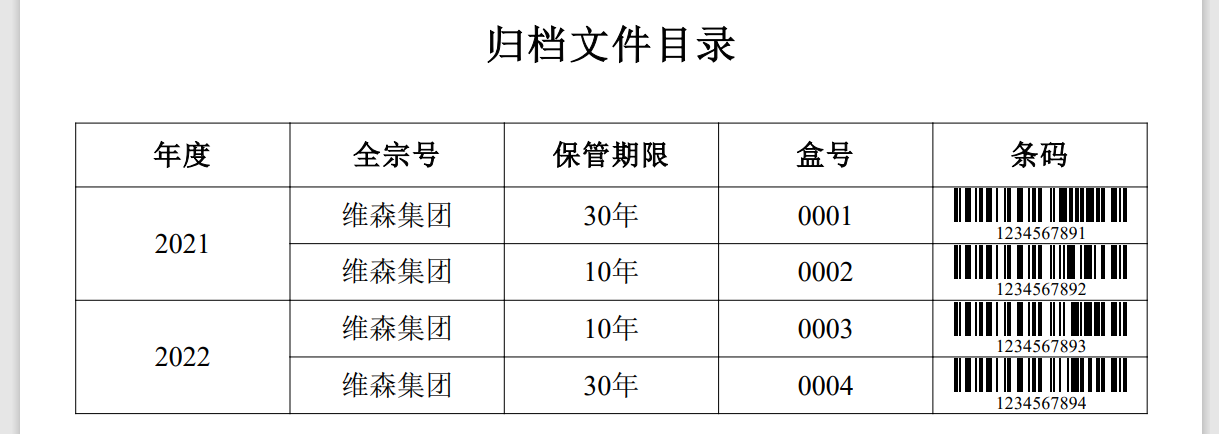
获取页码和总页数可以分别通过调用 getPageNo 和 getTotalPages 两个方法。需要注意的是，当 getPageNo 和 getTotalPages 在分组行或者分组行单元格中时，返回的是当前分组的页码和总页数；在普通单元格中，它返回的是表格的页码和总页数；其它情况它返回的是全局页码和总页数。其实 getPageNo 和 getTotalPage 只是根据他们所在的控件进行一个重定向，如果你想直接获取指定的页码或者总页数，可以直接调用对应的方法。

## 分组报表

“分组报表”即是对数据查询后，按照指定字段进行排序，之后“分组”进行展示的报表。

一般有两种展示模式：

1、列头合并分组报表。



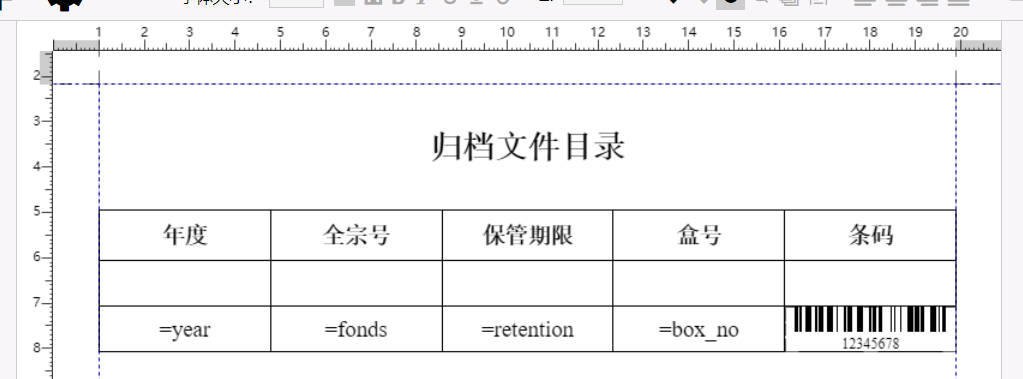
2、有分组头、分组尾的报表。



XMReport实现分组报表时，需要将表格行设置为“分录行”，并添加“分组起始行”、“分组结束行”（可不填）。

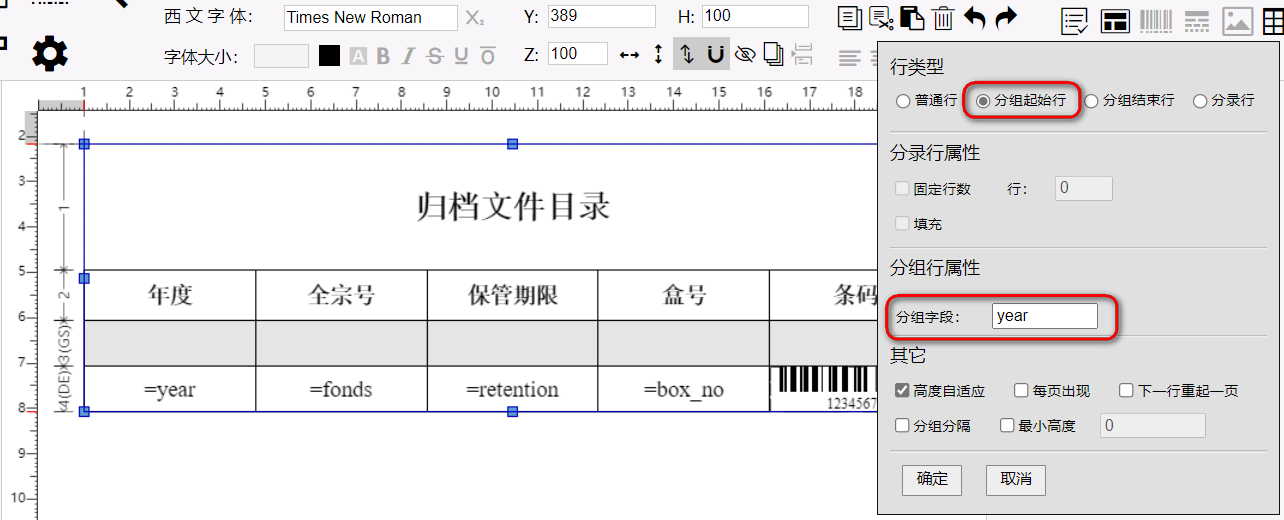
### 列头合并分组报表

绘制一个基本的表格（3行）。



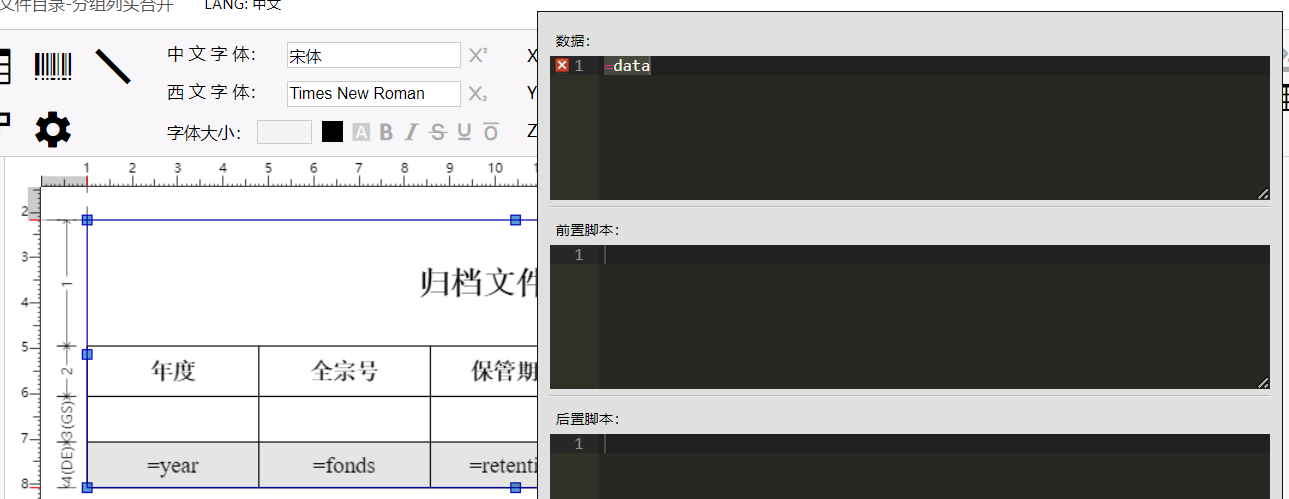
第一行：表头，固定文字。

第二行：选中整行，设置为【分组起始行】，并设定分组字段。本例为“year”。

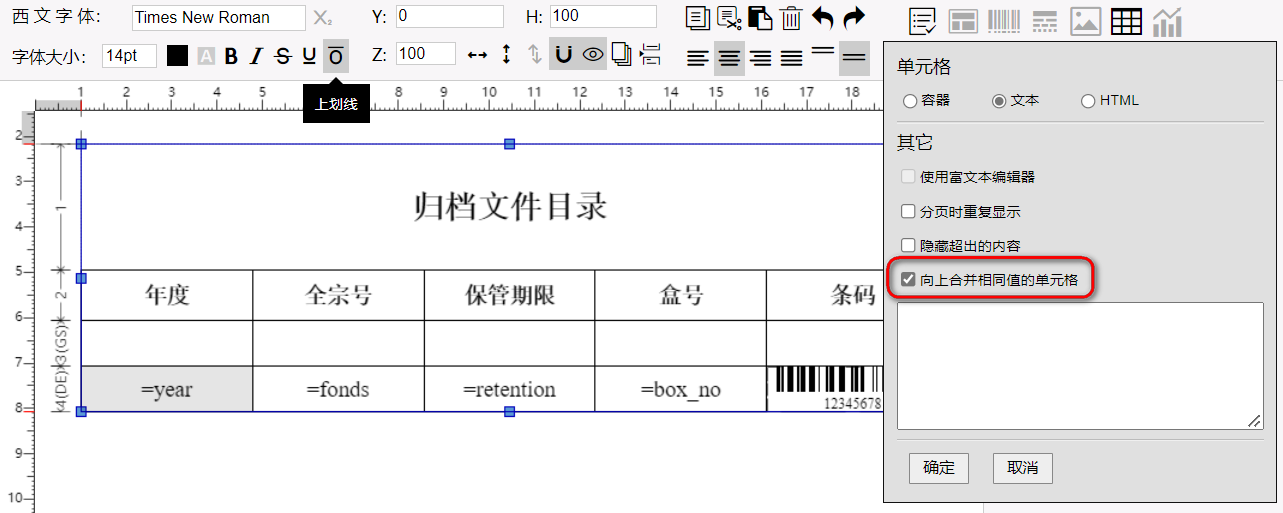


第三行：数据行，需要设置为“分录行”（否则无效哦）。同时，在【数据/脚本】中设置此行绑定数据，本例为“=data”。最后在每个单元格中设置绑定的字段key。





第三行需合并的字段：本例中，选择“年度”值的单元格（“=year”），点击【单元格属性】按钮，选中【向上合并相同值的单元格】。

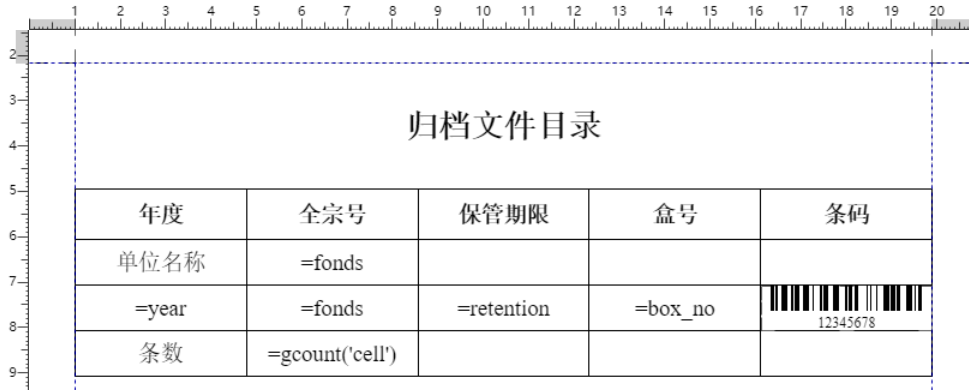


最后，我们可以使用前述章节的JSON数据进行预览。



### 有分组头、分组尾的报表

绘制一个基本的表格（4行）。

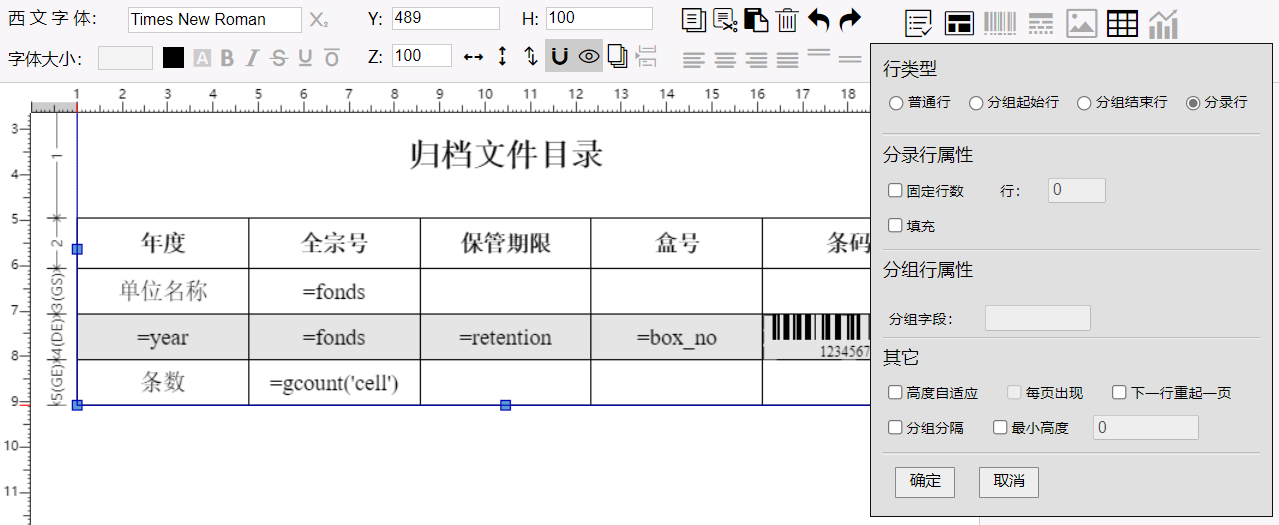


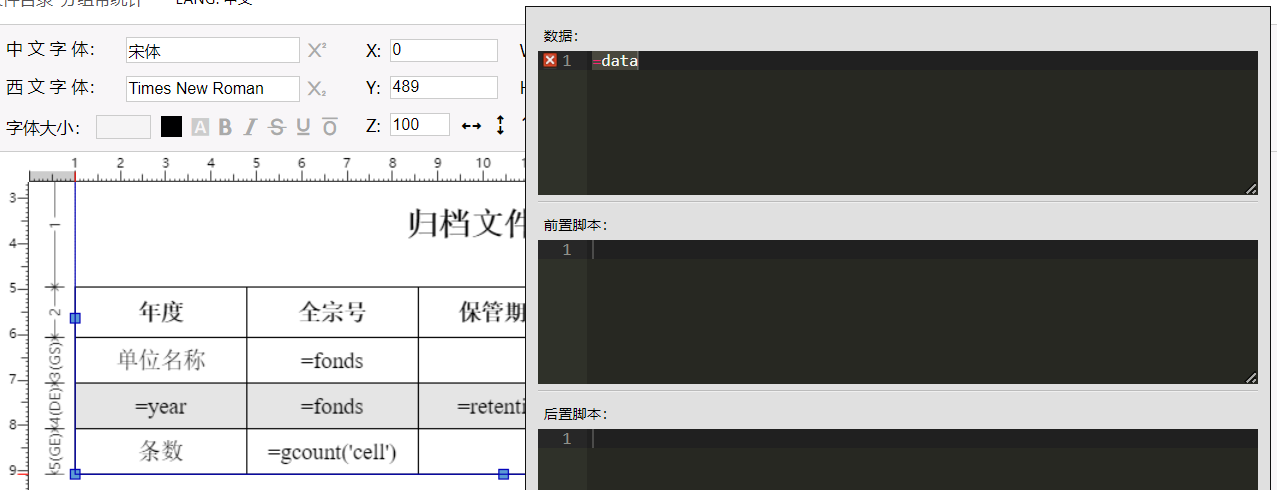
第一行：表头，固定文字。

第二行：选中整行，设置为【分组起始行】，并设定分组字段。本例为“year”。可以在此行中将需要单独显示的内容设定在此处，即固定文字、绑定数据（参见前述章节说明）。

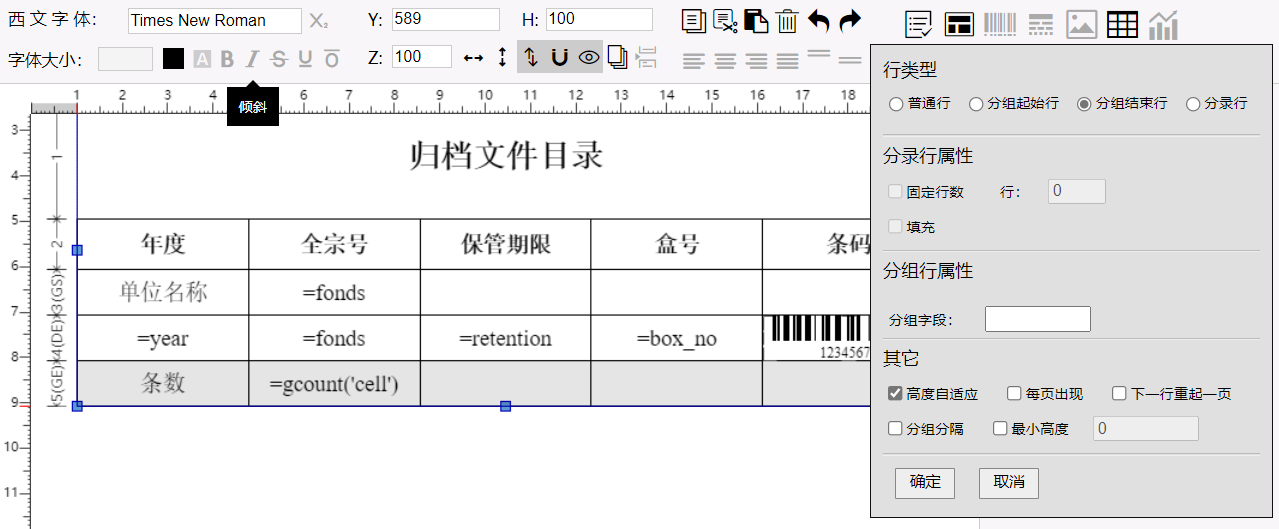


第三行：数据行，需要设置为“分录行”（否则无效哦）。同时，在【数据/脚本】中设置此行绑定数据，本例为“=data”。最后在每个单元格中设置绑定的字段key。





第四行：分组结束行。可以在此行设定表格计算函数，本例使用“gcount”计算数据的总条数。



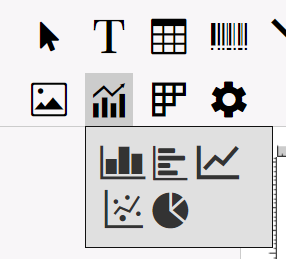
最后，我们可以使用前述章节的JSON数据进行预览。



## 图形报表

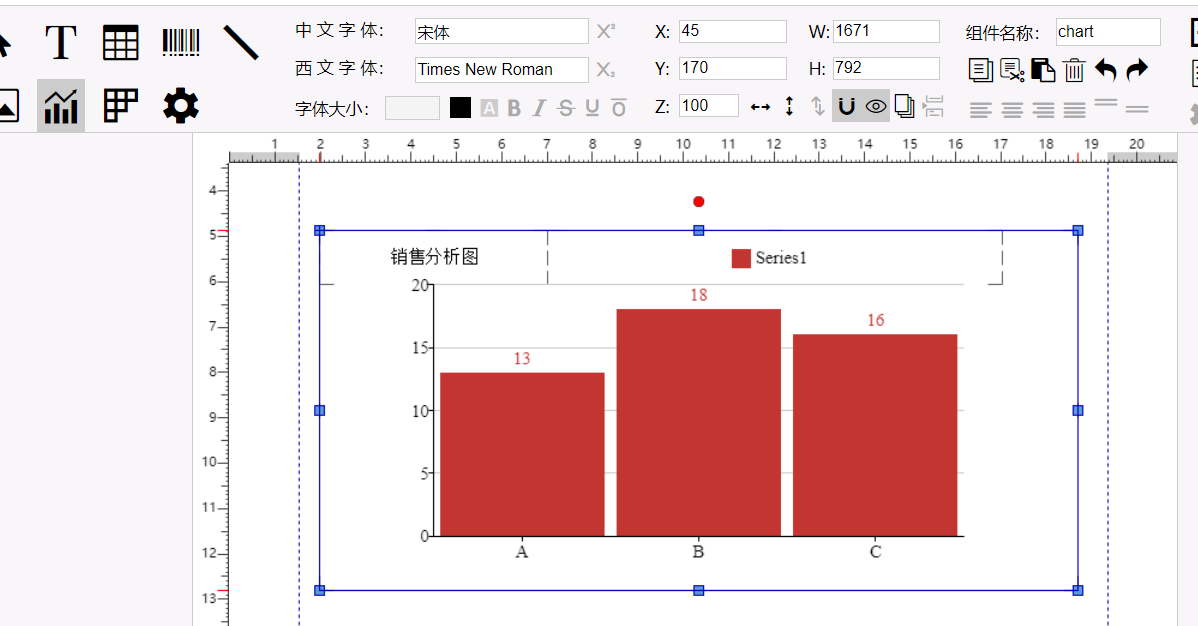
XMReport中的统计图形都是为了打印服务的，目标是输出到PDF、Word等文件中，所以都不提供“交互”功能（不能进行数据穿透）。

点击工具栏中的【图表】按钮，可以看到提供的5中图表样式。

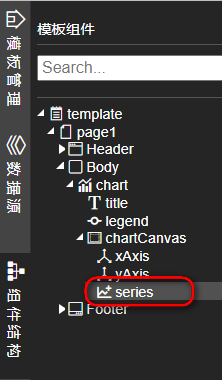


本例使用第一个图表，创建一个常见的柱状统计图。

选择第一个图表后，在画布中绘制，如下。图表是一个【容器】，其中的标题文字就是标准的文本框，可以根据实际情况修改文字内容。



打开左侧【组件结构】页，打开这个图表组件（chart）的对象结构，选中底层的“series”。



此时，点击【数据/脚本】按钮，在【数据】区中输入“=data”，即将图表与输入数据中的“data”绑定。

点击【预览】按钮，输入如下JSON字符串：

{

"data":[

{

"name": "成交",

"data": {"周一":10, "周二":12, "周三":21, "周四":54, "周五":260, "周六":830, "周日":710}

},

{

"name": "预购",

"data": {"周一":30, "周二":182, "周三":434, "周四":791, "周五":390, "周六":30, "周日":10}

},

{

"name": "意向",

"data": {"周一":1320, "周二":1132, "周三":601, "周四":234, "周五":120, "周六":90, "周日":20}

}

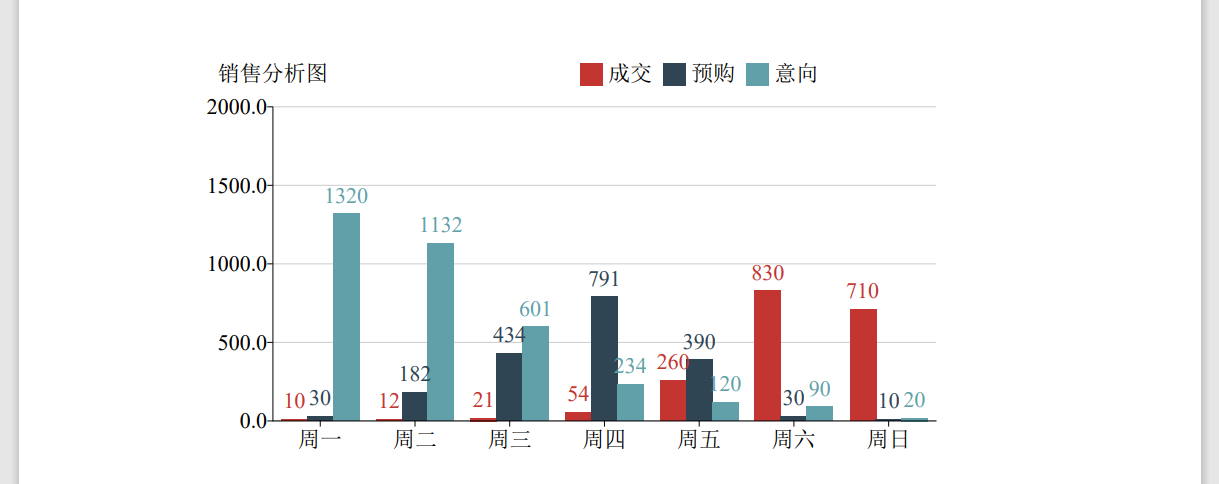
]

}

其中：

* 最外层的“data”这个key是与图表组件数据绑定的值相同，用于组件取值。其中内容为JSON数组。
* JSON数组中的每一个对象都对应着图表中的一组数据（一个柱）。
* **JSON对象中的key是固定的，必须为“name”和“data”。**
  + name：图表中每类数据的名称，对应一个柱。
  + data：显示的数据。每个key对应横轴（x轴）上的一个值，每个value对应纵轴（y轴）的高度。

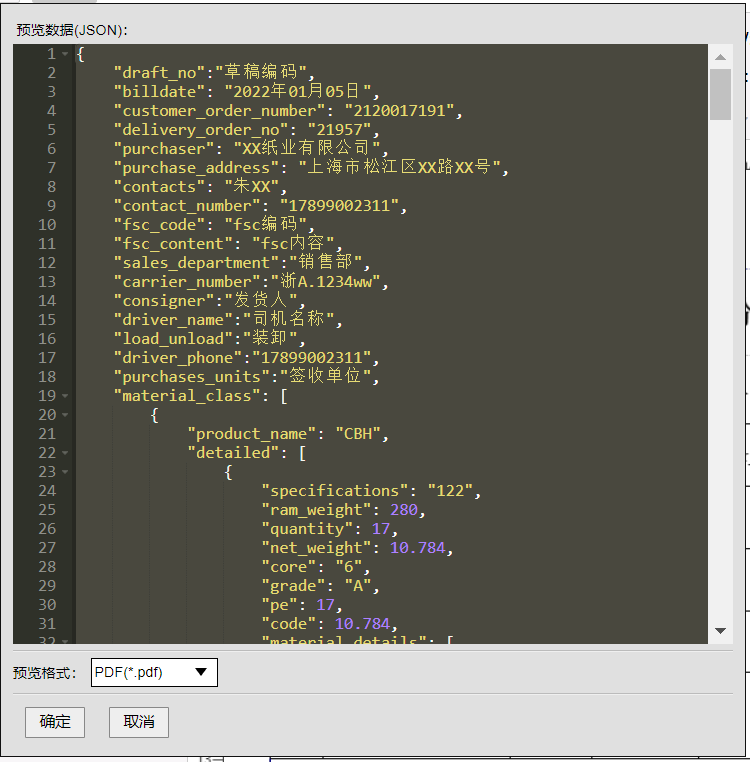
预览后，报表效果如下。



## 子报表（嵌套报表）

子报表，即多层报表嵌套。一般见于传入的数据结构层级多，需要逐级输出的情况。

例如，输入的数据格式如下：



输出的报表是多级的。



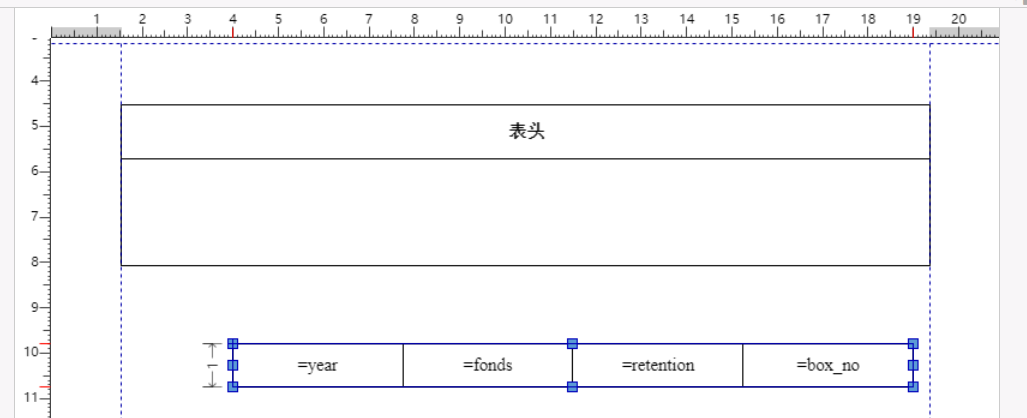
其实，所有复杂的报表都可以分解为简单的单层表格。

只需要对最终需要输出的表格进行逐级分解，对应输入数据进行逐层嵌套即可。

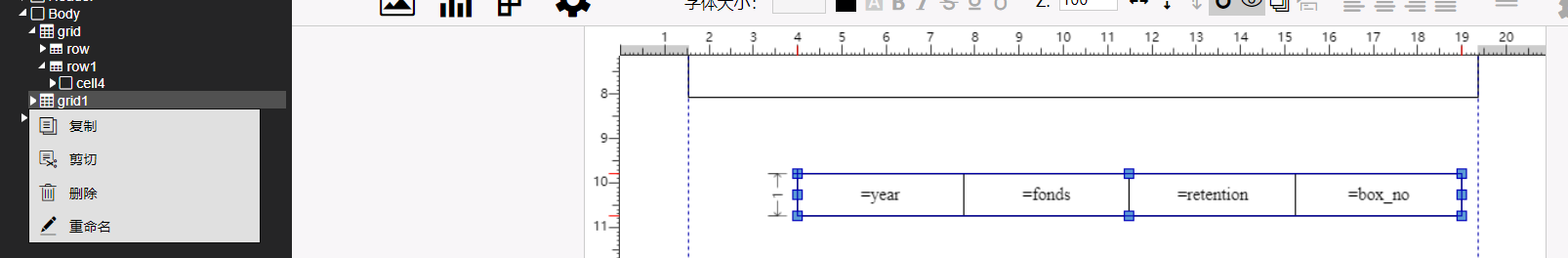
XMReport支持将【表格】中的单元格设置为“容器”，可以存放其他组件，同样也可以存放表格。单元格支持无限嵌套。

我们在实现这种复杂表格的时候，可以将数据逐层分解到【行】上，然后外层表格每行只保留一个单元格，作为下一层表格的【容器】，再利用【剪切】、【粘贴】的功能，将下一级表格挪到这个【容器】中就可以了。例如：

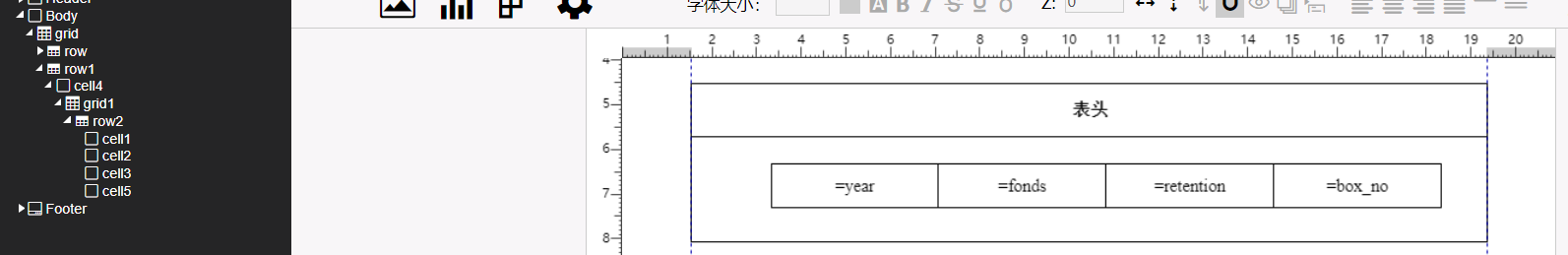
1. 加入一个2行1列的表格；第二行的单元格设定为“容器”；第二行【数据】绑定为传入JSON的第一层的一个key（本例为“data”）
2. 再加入一个1行4列的表格，将作为子表输出数据。



1. 设定每个单元格需要输出的数据key。
2. 在左侧【组件结构】页，打开第一个表格的对象，可以看到第二行的单元格前面有一个三角号，代表它已经是个容器了，里面可以放其他的组件了。
3. 在【组件结构】中，选中第二个表格对象，点击【剪切】。



1. 在第一个表格的第二行的单元格节点上，点击鼠标右键，选择【粘贴】，完成子报表的添加。



1. 预览后效果如下。



当然，为了更加美观，我们可以选中子报表对象后，通过点击【横向充满】、【纵向充满】按钮，调整表格的宽度、高度。