

15 | 前端技术应用（二）：如何设计一个报表工具？

2019-09-16 宫文学

编译原理之美

[进入课程 >](#)



讲述：宫文学

时长 13:14 大小 12.14M



众所周知，很多软件都需要面向开发者甚至最终用户提供自定义功能，在[开篇词](#)里，我提到自己曾经做过 workflow 软件和电子表单软件，它们都需要提供自定义功能，报表软件也是其中的典型代表。

在每个应用系统中，我们对数据的处理大致会分成两类：一类是在线交易，叫做 OLTP，比如在网上下订单；一类是在线分析，叫做 OLAP，它是对应用中积累的数据进行进一步分析利用。而报表工具就是最简单，但也是最常用的数据分析和利用的工具。

本节课，我们就来分析一下，如果我们要做一个通用的报表工具，需要用到哪些编译技术，又该怎样去实现。


报表工具所需要的编译技术

如果要做一个报表软件，我们要想清楚软件面对的用户是谁。有一类报表工具面向的用户是程序员，那么这种软件可以暴露更多技术细节。比如，如果报表要从数据库获取数据，你可以写一个 SQL 语句作为数据源。

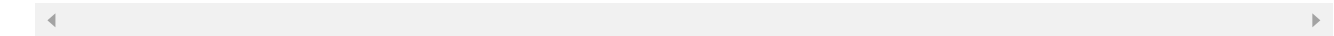
还有一类软件是给业务级的用户使用的，很多 BI 软件包都是这种类型。带有 IT 背景的顾问给用户做一些基础配置，然后用户就可以用这个软件包了。Excel 可以看做是这种报表工具，IT 人员建立 Excel 与数据库之间的连接，剩下的就是业务人员自己去操作了。

这些业务人员可以采用一个图形化的界面设计报表，对数据进行加工处理。我们来看看几个场景。

第一个场景是计算字段。计算字段的意思是，原始数据里没有这个数据，我们需要基于原始数据，通过一个自定义的公式来把它计算出来，比如在某个 CRM 系统中保存着销售数据。我们有每个部门的总销售额，也有每个部门的人数，要想在报表中展示每个部门的人均销售额，这个时候就可以用到计算公式功能，计算公式如下：

 复制代码


1 人均销售额 = 部门销售额 / 部门人数



得到的结果如下图所示：

部门	人数	销售额	人均销售额
电话销售部	10	2345	234.5
现场销售部	20	5860	293.0
电子商务部	15	3045	203.0
...
...
...
...
...

进一步，我们可以在计算字段中支持函数。比如我们可以把各个部门按照人均销售额排名次。这可以用一个函数来计算：

 复制代码

```
1 =rank(人均销售额)
```

rank 就是排名次的意思，其他统计函数还包括：

min(), 求最小值。

max(), 求最大值。

avg(), 求平均值。


sum(), 求和。

还有一些更有意思的函数，比如：

runningsum(), 累计汇总值。

runningavg(), 累计平均值。

这些有意思的函数是什么意思呢？因为很多明细性的报表，都是逐行显示的，累计汇总值和累计平均值，就是累计到当前行的计算结果。当然了，我们还可以支持更多的函数，比如当前日期、当前页数等等。更有意思的是，上述字段也好、函数也好，都可以用来组合成计算字段的公式，比如：


 复制代码

```
1 = 部门销售额 /sum(部门销售额)    // 本部门的销售额在全部销售额的占比
2 =max(部门销售额)- 部门销售额      // 本部门的销售额与最高部门的差距
3 =max(部门销售额 / 部门人数)- 部门销售额 / 部门人数    // 本部门人均销售额与最高的那个部门的差
4 =sum(部门销售额)/sum(人数)- 部门销售额 / 部门人数    // 本部门的人均销售额与全公司人均销售额的差
```

怎么样，是不是越来越有意思了呢？现在你已经知道了在报表中会用到普通字段和各种各样的计算公式，那么，我们如何用这样的字段和公式来定义一张报表呢？


如何设计报表

假设我们的报表是一行一行地展现数据，也就是最简单的那种。那我们将报表的定义做成一个 XML 文件，可能是下面这样的，它定义了表格中每一列的标题和所采用字段或公式：

 复制代码

```
1 <playreport title="Report 1">
2   <section>
3     <column>
4       <title> 部门 </title>
5       <field>dept</field>
6     </column>
7     <column>
8       <title> 人数 </title>
9       <field>num_person</field>
10    </column>
11    <column>
12      <title> 销售额 </title>
13      <field>sales_amount</field>
14    </column>
15    <column>
16      <title> 人均销售额 </title>
17      <field>sales_amount/num_person</field>
18    </column>
19  </section>
20  <datasource>
21    <connection> 数据库连接信息...</connection>
22    <sql>select dept, num_person, sales_amount from sales</sql>
23  </datasource>
24 </playreport>
```

这个报表定义文件还是蛮简单的，它主要表达的是数据逻辑，忽略了表现层的信息。如果我们想要优先表达表现层的信息，例如字体大小、界面布局等，可以采用 HTML 模板的方式来定义报表，其实就是在一个 HTML 中嵌入了公式，比如：

 复制代码

```
1 <html>
2 <body>
3   <div class="report" datasource=" 这里放入数据源信息 ">
4     <div class="table_header">
5       <div class="column_header"> 部门 </div>
6       <div class="column_header"> 人数 </div>
7       <div class="column_header"> 销售额 </div>
8       <div class="column_header"> 人均销售额 </div>
```

```

9         </div>
10        <div class="table_body">
11            <div class="field">{=dept}</div>
12            <div class="field">{=num_person}</div>
13            <div class="field">{=sales_amount}</div>
14            <div class="field">{=sales_amount/num_person}</div>
15        </div>
16    </div>
17 </body>
18 </html>

```

这样的 HTML 模板看上去是不是很熟悉？其实在很多语言里，比如 PHP，都提供模板引擎功能，实现界面设计和应用代码的分离。这样一个模板，可以直接解释执行，或者先翻译成 PHP 或 Java 代码，然后再执行。只要运用我们学到的编译技术，这些都可以实现。

我想你应该会发现，这样的模板文件，其实就是一个特定领域语言，也就是我们常说的 DSL。DSL 可以屏蔽掉实现细节，让我们专注于领域问题，像上面这样的 DSL，哪怕没有技术背景的工作人员，也可以迅速地编写出来。


而这个简单的报表，在报表设计界面上可能是下图这样的形式：

部门	人数	销售额	人均销售额
dept	num_person	sales_amount	sales_amount/num_person

分析完如何设计报表之后，接下来，我们看看如何定义报表所需要的公式规则。

编写所需要的语法规则

我们设计了 PlayReport.g4 规则文件，这里面的很多规则，是把 PlayScript.g4 里的规则拿过来改一改用的：

 复制代码

```

1 bracedExpression
2     : '{' '=' expression '}'
3     ;
4
5 expression

```

```

6      : primary
7      | functionCall
8      | expression bop=('*' | '/' | '%') expression
9      | expression bop=('+' | '-') expression
10     | expression bop('<=' | '>=' | '>' | '<') expression
11     | expression bop('=' | '!=') expression
12     | expression bop='&&' expression
13     | expression bop='||' expression
14     ;
15
16 primary
17     : '(' expression ')'
18     | literal
19     | IDENTIFIER
20     ;
21
22 expressionList
23     : expression (',' expression)*
24     ;
25
26 functionCall
27     : IDENTIFIER '(' expressionList? ')'
28     ;
29
30 literal
31     : integerLiteral
32     | floatLiteral
33     | CHAR_LITERAL
34     | STRING_LITERAL
35     | BOOL_LITERAL
36     | NULL_LITERAL
37     ;
38
39 integerLiteral
40     : DECIMAL_LITERAL
41     | HEX_LITERAL
42     | OCT_LITERAL
43     | BINARY_LITERAL
44     ;
45
46 floatLiteral
47     : FLOAT_LITERAL
48     | HEX_FLOAT_LITERAL
49     ;

```

这里面，其实就是用了表达式的语法，包括支持加减乘除等各种运算，用来书写公式。我们还特意支持 functionCall 功能，也就是能够调用函数。因为我们内部实现了很多内置函数，比如求最大值、平均值等，可以在公式里调用这些函数。

现在呢，我们已经做好了一个最简单的报表定义，接下来，就一起实现一个简单的报表引擎，这样就能实际生成报表了！

实现一个简单的报表引擎

报表引擎的工作，是要根据报表的定义和数据源中的数据，生成最后报表的呈现格式。具体来说，可以分为以下几步：

解析报表的定义。我们首先要将报表定义形成 Java 对象。这里只是简单地生成了一个测试用的报表模板。

从数据源获取数据。我们设计了一个 `TabularData` 类，用来保存类似数据库表那样的数据。

实现一个 `FieldEvaluator` 类，能够在运行时对字段和公式进行计算。这个类是 `playscript` 中 `ASTEvaluator` 类的简化版。我们甚至连语义分析都简化了。数据类型信息作为 `S` 属性，在求值的同时自底向上地进行类型推导。当然，如果做的完善一点儿，我们还需要多做一点儿语义分析，比如公式里的字段是不是数据源中能够提供的？而这时需要用到报表数据的元数据。

渲染报表。我们要把上面几个功能组合在一起，对每一行、每一列求值，获得最后的报表输出。

主控程序我放在了下面，用一个示例报表模板和报表数据来生成报表：

 复制代码

```
1 public static void main(String args[]) {
2     System.out.println("Play Report!");
3
4     PlayReport report = new PlayReport();
5
6     // 打印报表 1
7     String reportString = report.renderReport(ReportTemplate.sampleReport1(), TabularData);
8     System.out.println(reportString);
9 }
```

`renderReport` 方法用来渲染报表，它会调用解析器和报表数据的计算器：


 复制代码


```

1 public String renderReport(ReportTemplate template, TabularData data){
2     StringBuffer sb = new StringBuffer();
3
4     // 输出表格头
5     for (String columnHeader: template.columnHeaders){
6         sb.append(columnHeader).append('\t');
7     }
8     sb.append("\n");
9
10    // 编译报表的每个字段
11    List<BracedExpressionContext> fieldASTs = new LinkedList<BracedExpressionContext>()
12    for (String fieldExpr : template.fields){
13        // 这里会调用解析器
14        BracedExpressionContext tree = parse(fieldExpr);
15        fieldASTs.add(tree);
16    }
17
18    // 计算报表字段
19    FieldEvaluator evaluator = new FieldEvaluator(data);
20    List<String> fieldNames = new LinkedList<String>();
21    for (BracedExpressionContext fieldAST: fieldASTs){
22        String fieldName = fieldAST.expression().getText();
23        fieldNames.add(fieldName);
24        if (!data.hasField(fieldName)){
25            Object field = evaluator.visit(fieldAST);
26            data.setField(fieldName, field);
27        }
28    }
29
30    // 显示每一行数据
31    for (int row = 0; row< data.getNumRows(); row++){
32        for (String fieldName: fieldNames){
33            Object value = data.getFieldValue(fieldName, row);
34            sb.append(value).append("\t");
35        }
36        sb.append("\n");
37    }
38
39    return sb.toString();
40 }

```

程序的运行结果如下，它首先打印输出了每个公式的解析结果，然后输出报表：

 复制代码

```

1 Play Report!
2 (bracedExpression { = (expression (primary dept)) })
3 (bracedExpression { = (expression (primary num_person)) })

```



```
4 (bracedExpression { = (expression (primary sales_amount)) })
5 (bracedExpression { = (expression (expression (primary sales_amount)) / (expression (pr:
6 部门          人数      销售额   人均销售额
7 电话销售部      10        2345.0   234.5
8 现场销售部      20        5860.0   293.0
9 电子商务部      15        3045.0   203.0
10 渠道销售部      20        5500.0   275.0
11 微商销售部      12        3624.0   302.0
```

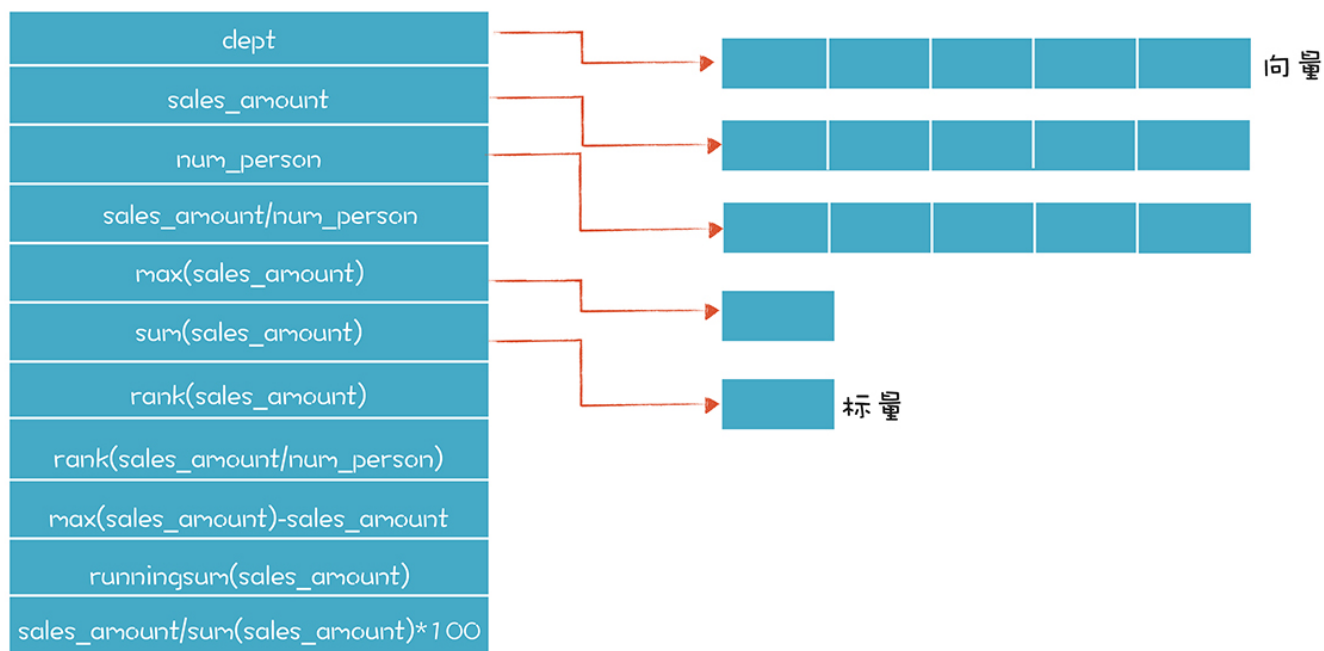


你可以看到，报表工具准确地得出了计算字段的数据。接下来，我再讲一讲报表数据计算的细节。

如果你看一看 `FieldEvaluator.java` 这个类，就会发现我实际上实现了一个简单的向量数据的计算器。在计算机科学里，向量是数据的有序列表，可以看做一个数组。相对应的，标量只是一个单独的数据。运用向量计算，我们在计算人均销售额的时候，会把“销售额”和“人数”作为两个向量，每个向量都有 5 个数据。把这两个向量相除，会得到第三个向量，就是“人均销售额”。这样就不需要为每行数据运行一次计算器，会提高性能，也会简化程序。

其实，这个向量计算器还能够把向量和标量做混合运算。因为我们的报表里有时候确实会用到标量，比如对销售额求最大值 $\{=\max(\text{sales_amount})\}$ ，就是一个标量。而如果计算销售额与最大销售额的差距 $\{=\max(\text{sales_amount})-\text{sales_amount}\}$ ，就是标量和向量的混合运算，返回结果是一个向量。

`TabularData.java` 这个类是用来做报表数据的存储的。我简单地用了一个 `Map`，把字段的名称对应到一个向量或标量上，其中字段的名称可以是公式：



在报表数据计算过程中，我们还做了一个优化。公式计算的中间结果会被存起来，如果下一个公式刚好用到这个数据，可以复用。比如，在计算 `rank(sales_amount/num_person)` 这个公式的时候，它会查一下括号中的 `sales_amount/num_person` 这个子公式的值是不是以前已经计算过，如果计算过，就复用，否则，就计算一下，并且把这个中间结果也存起来。

我们把这个报表再复杂化一点，形成下面一个报表模板。这个报表模版用到了好几个函数，包括排序、汇总值、累计汇总值和最大值，并通过公式定义出一些相对复杂的计算字段，包括最高销售额、销售额的差距、销售额排序、人均销售额排序、销售额累计汇总、部门销售额在总销售额中的占比，等等。

复制代码

```

1 public static ReportTemplate sampleReport2(){
2     ReportTemplate template = new ReportTemplate();
3
4     template.columnHeaders.add(" 部门 ");
5     template.columnHeaders.add(" 人数 ");
6     template.columnHeaders.add(" 销售额 ");
7     template.columnHeaders.add(" 最高额 ");
8     template.columnHeaders.add(" 差距 ");
9     template.columnHeaders.add(" 排序 ");
10    template.columnHeaders.add(" 人均 ");
11    template.columnHeaders.add(" 人均排序 ");
12    template.columnHeaders.add(" 累计汇总 ");
13    template.columnHeaders.add(" 占比 %");
14
15    template.fields.add("{=dept}");

```

```
16     template.fields.add("{=num_person}");
17     template.fields.add("{=sales_amount}");
18     template.fields.add("{=max(sales_amount)}");
19     template.fields.add("{=max(sales_amount)-sales_amount}");
20     template.fields.add("{=rank(sales_amount)}");
21     template.fields.add("{=sales_amount/num_person}");
22     template.fields.add("{=rank(sales_amount/num_person)}");
23     template.fields.add("{=runningsum(sales_amount)}");
24     template.fields.add("{=sales_amount/sum(sales_amount)*100}");
25
26     return template;
27 }
```

最后输出的报表截屏如下，怎么样，现在看起来功能还是挺强的吧！

部门	人数	销售额	最高额	差距	排序	人均	人均排序	累计汇总	占比%
电话销售部	10	2345.0	5860.0	-3515.0	5	234.5	4	2345.0	11.509767350544813
现场销售部	20	5860.0	5860.0	0.0	1	293.0	2	8205.0	28.762147835476586
电子商务部	15	3045.0	5860.0	-2815.0	4	203.0	5	11250.0	14.945518798468637
渠道销售部	20	5500.0	5860.0	-360.0	2	275.0	3	16750.0	26.995189947972907
微商销售部	12	3624.0	5860.0	-2236.0	3	302.0	1	20374.0	17.787376067537057

当然了，这个程序只是拿很短的时间写的一个 Demo，如果要变成一个成熟的产品，还要在很多地方做工作。比如：

可以把字段名称用中文显示，这样更便于非技术人员使用；

除了支持行列报表，还要支持交叉表，用于统计分析；

支持多维数据计算。

.....

在报表工具中，编译技术除了用来做字段的计算，还可以用于其他功能，比如条件格式。我们可以在人均销售额低于某个数值时，给这行显示成红色，其中的判断条件，也是一个公式。

甚至你还可以为报表工具添加自定义公式功能。我们给用户提供脚本功能，用户可以自己做一个函数，实现某个领域的一个专业功能。我十分建议你在这个示例程序的基础上进一步加工，看看能做否做出一些让自己惊喜的功能。

课程小结

本节课我们做了一个示例性的报表工具。你能在这个过程中看到，像报表工具这样的软件，如果有编译技术的支持，真的可以做得很灵活、很强大。你完全可以借鉴本节课的思路，去尝试做一下其他需要自定义功能的软件工具或产品。

与此同时，我们能看到编译技术可以跟某个应用领域结合在一起，内置在产品中，同时形成领域的 DSL，比如报表的模板文件。这样，我们就相当于赋予了普通用户在某个领域内的编程能力，比如用户只需要编写一个报表模板，就可以生成报表了。了解这些内容之后，我来带你回顾一下，这个应用是怎么运用编译器前端技术的。

词法分析和语法分析都很简单，我们就是简单地用了表达式和函数调用的功能。而语义分析除了需要检查类型以外，还要检查所用到的字段和函数是否合法，这是另一种意义上的引用消解。而且这个例子中的运算的含义是向量运算，同样是加减乘除，每个操作都会处理一组数据，这也是一种语义上的区别。

我在学习了这两节课之后，你能对如何在某个应用领域应用编译技术有更直观的了解，甚至有了很多的启发。

一课一思

你在自己的工作领域中，是否发现有哪些需要用户自定义功能的需求？你又是如何实现这些需求的？编译技术会不会在这些地方帮助到你？欢迎在留言区分享你的发现。

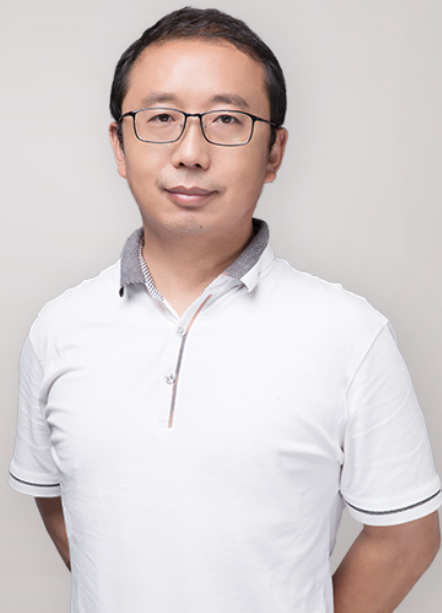
最后，感谢你的阅读，如果这篇文章让你有所收获，欢迎你将它分享给更多的朋友。

编译原理之美

手把手教你实现一个编译器

宫文学

北京物演科技CEO



新版升级：点击「👤 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 14 | 前端技术应用（一）：如何透明地支持数据库分库分表？

精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。