ExcelReport 1.5.2 使用手册

ExcelReport 是一款基于 NPOI 开发的报表引擎组件。它基于关注点分离的理念，将数据与样式、格式分离。让模板承载样式、格式等 NPOI 不怎么擅长且实现繁琐的信息，结合 NPOI 对数据的处理的优点将 Excel 报表的生成化繁为简。同时，对报表组成的基本元素进行了抽象，进一步简化了 Excel 报表的生成过程。

官方站点

NuGet：https://www.nuget.org/packages/ExcelReport

```
PM> Install-Package ExcelReport
```

GitHub：https://github.com/hanzhaoxin/ExcelReport
cnblogs：http://www.cnblogs.com/hanzhaoxin/tag/ExcelReport /

团队介绍

Jensen
来自于中国深圳，是这个项目的发起人和开发人员，2014 年 11 月开始了 ExcelReport 的开发，负责 ExcelReport 的开发、测试和 bug 修复。
个人博客地址：http:// hanzhaoxin.cnblogs.com

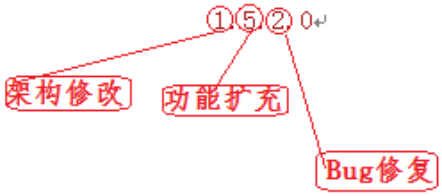
许可证说明

ExcelReport 采用的是 MIT 许可，这意味着它可以被用于任何商业或非商业项目，你不用担心因为使用它而必须开放你自己的源代码，且可借助 ExcelReport 的影响力推广你的产品。

当然作为一个开源许可证，肯定也是有一些义务。例如：在软件和软件的所有副本中都必须包含以上版权声明和本许可声明。

完整的 MIT 许可证请见：http://mit-license.org/

版本升级说明：



如果您项目中使用了 ExcelReport，Bug 修复和功能扩充的升级版本请放心升级，这类升级不能影响您现有的项目。如果架构修改版本信息发生了变化，请谨慎升级，这类升级有可能影响您现有的项目。

为什么要用 ExcelReport？

回答这个问题前，我们先看看 Tony Qu 是怎么回答为什么要用 NPOI 的？
1）你不需要在服务器上安装微软的 Office，可以避免版权问题。
2）NPOI 使用起来比 Office PIA 的 API 更加方便，更人性化。
3）你不用去花大力气维护 NPOI，NPOI Team 会不断更新、改善 NPOI，绝对省成本。
4）很多事情是 html 和 cvs 法做不到的，比如说公式计算[Cell C1]=A1+B1*A2 单元格高级样式（如文本旋转、对齐、宽度）等，其中公式计算可以适当减轻服务器端的计算压力

回到我们的问题，为什么要用 ExcelReport 呢？
1） 复杂的事情简单了，不可能的事情可行了。
2） 你不需要了解 NPOI 大量的 API，你不需要用.NET 语言写蹩脚的 Java 的语法。
3） 套用 Tony 兄的描述，你不用去花大力气维护 ExcelReport，ExcelReport Team 会不断更新、改善 ExcelReport，绝对省成本。

目录

1 报表元素与元素格式化器

2 演示：使用 ExcelReport 生成报表

3 格式化器示例
3.1 局部格式化器
3.2 单元格格式化器
3.3 表格格式化器
3.4 重复单元格式化器

4 多 Sheet 报表生成

1 报表元素与元素格式化器

在开始示例之前，我们用一章的篇幅介绍 ExcelReport 是如何将报表的内容抽象为元素的？又是如何为元素填充数据的？为了说明第一个问题，我们从一个现有的报表开始。

如下报表有两个 Sheet：Sheet1 名为“工资表”Sheet2 名为“工资条”，其截图如下所示。

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
006	梁忠基	11.31	174	28	8	0	1967.94	491.02	0.00	0	2458.96	60	30	90	2368.960	
007	韦明珍	6.68	174	28	8	0	1162.32	296.56	0.00	0	1458.88	60	30	90	1368.880	
008	韦明初	5.46	174	28	8	0	950.04	245.32	0.00	0	1195.36	60	30	90	1105.360	
010	钟燕华	5.25	174	28	8	0	913.5	236.50	0.00	0	1150.00	60	30	90	1060.000	
011	谭兴燕	7.19	174	28	8	0	1251.06	317.98	0.00	0	1569.04	60	30	90	1479.040	
012	梁海玲	6.76	174	28	8	0	1176.24	299.92	0.00	0	1476.16	60	30	90	1386.160	
013	黄金红	5.64	174	28	8	0	981.36	252.88	0.00	0	1234.24	60	30	90	1144.240	

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
006	梁忠基	11.31	174	28	8	0	1967.94	491.02	0.00	0	2458.96	60	30	90	2368.960	

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
007	韦明珍	6.68	174	28	8	0	1162.32	296.56	0.00	0	1458.88	60	30	90	1368.880	

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
008	韦明初	5.46	174	28	8	0	950.04	245.32	0.00	0	1195.36	60	30	90	1105.360	

某工厂生产A部工资表

月份：	2015年4月

我们先把数据部用蓝色框线标出来。

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
006	梁忠基	11.31	174	28	8	0	1967.94	491.02	0.00	0	2458.96	60	30	90	2368.960	
007	韦明珍	6.68	174	28	8	0	1162.32	296.56	0.00	0	1458.88	60	30	90	1368.880	
008	韦明初	5.46	174	28	8	0	950.04	245.32	0.00	0	1195.36	60	30	90	1105.360	
010	钟燕华	5.25	174	28	8	0	913.5	236.50	0.00	0	1150.00	60	30	90	1060.000	
011	谭兴燕	7.19	174	28	8	0	1251.06	317.98	0.00	0	1569.04	60	30	90	1479.040	
012	梁海玲	6.76	174	28	8	0	1176.24	299.92	0.00	0	1476.16	60	30	90	1386.160	
013	黄金红	5.64	174	28	8	0	981.36	252.88	0.00	0	1234.24	60	30	90	1144.240	

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
006	梁忠基	11.31	174	28	8	0	1967.94	491.02	0.00	0	2458.96	60	30	90	2368.960	

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
007	韦明珍	6.68	174	28	8	0	1162.32	296.56	0.00	0	1458.88	60	30	90	1368.880	

某工厂生产A部工资表

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
008	韦明初	5.46	174	28	8	0	950.04	245.32	0.00	0	1195.36	60	30	90	1105.360	

某工厂生产A部工资表

月份：	2015年4月

名正才能言顺，先说说 ExcelReport 中报表元素的概念：

元素：填充到报表模板中的数据源对象，我们称之为元素。

局部元素：填充到报表模板中的数据源对象是一个单元格内容的一部分，我们称这样的数据源对象为局部元素。

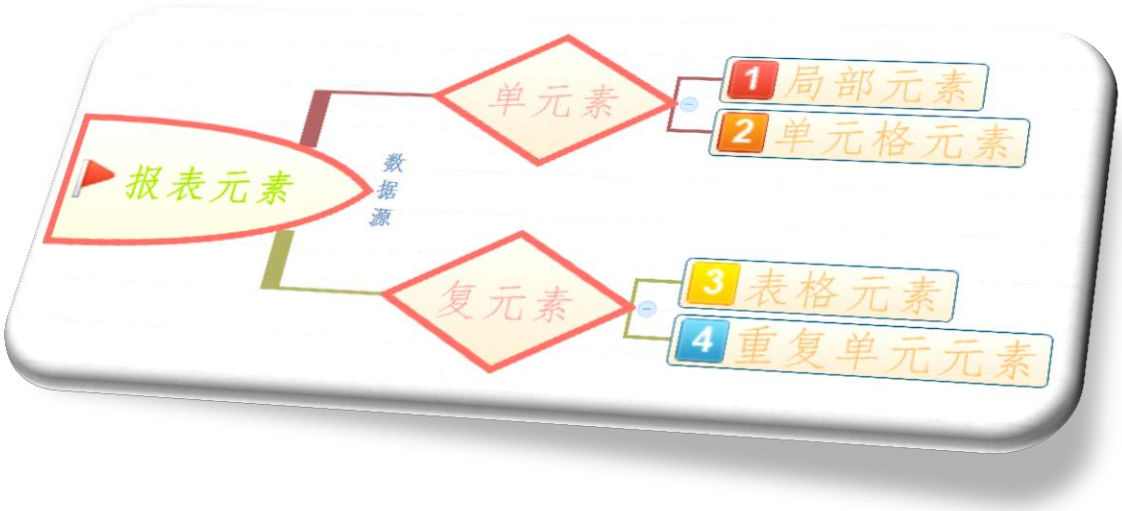
单元格元素：填充到报表模板中的数据源对象是一个单元格的内容，我们称这样的数据源对象为单元格元素。

表格元素：填充到报表模板中的数据源是某对象的集合，该对象是一行中多个单元格的内容集合，我们称这样的数据源集合为表格元素。

重复单元元素：填充到报表模板中的数据源是某对象的集合，该对象是多行中多个单元格的内容集合，我们称这样的数据源集合为重复单元元素。

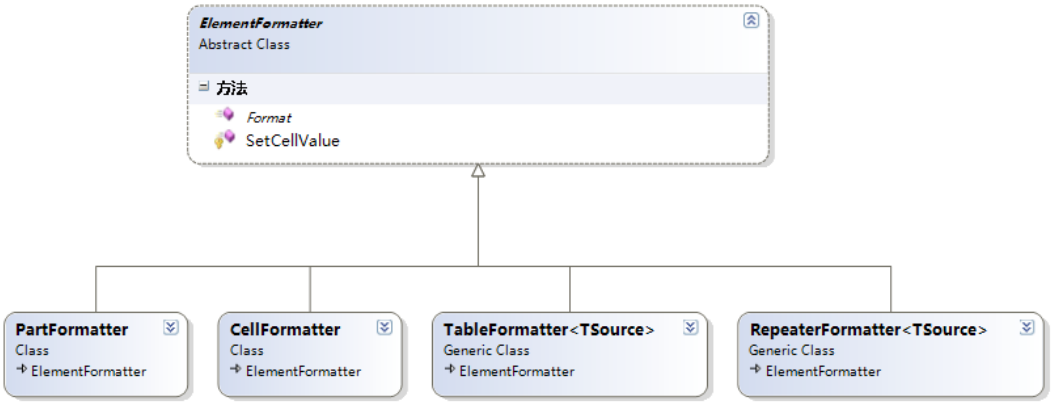
【表格元素与重复单元元素数据源以集合出现。为复元素。】

【对应的，局部元素和单元格元素称之为单元素。】



有了定义，我们让元素们各就各位。

那么，ExcelReport 又是如何为元素填充数据的？



话说元素格式化器为此而生。

报表元素	元素格式化器
局部元素	PartFormatter
单元格元素	CellFormatter
表格元素	TableFormatter
重复单元元素	RepeaterFormatter

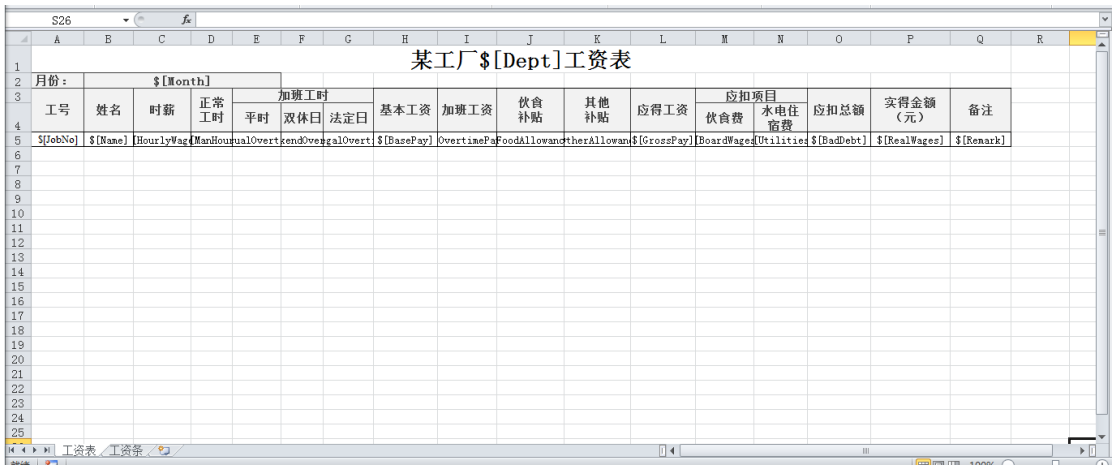
【元素格式化器的作用是格式化元素将其填充到报表模板。】

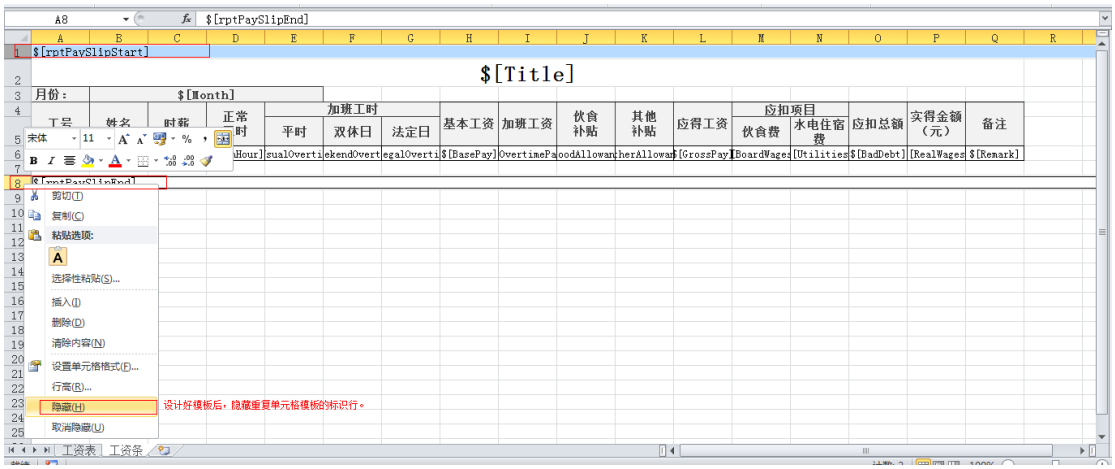
2 演示：使用 ExcelReport 生成报表

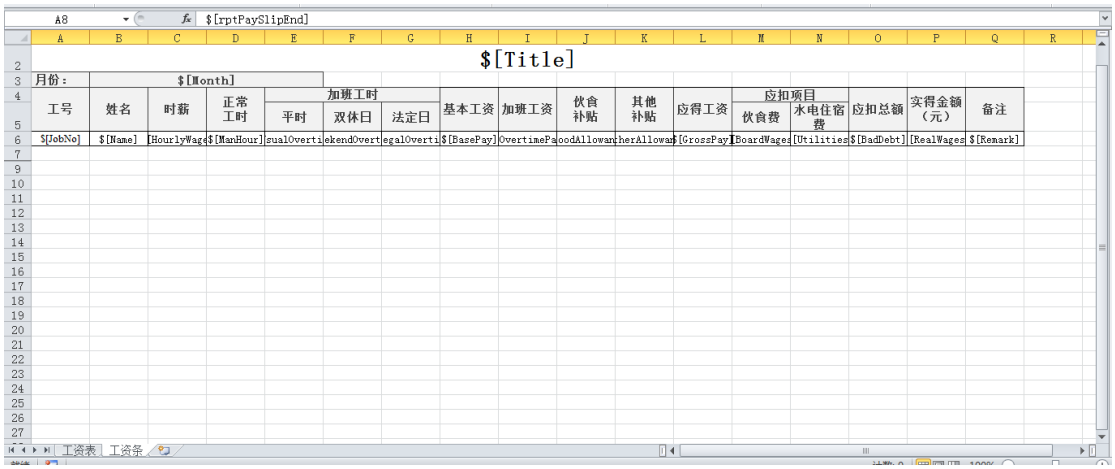
目标报表，生成上章分析的”工资表-工资条”报表。

第一步：设计模板

【模板中参数格式：$[ParameterName]】

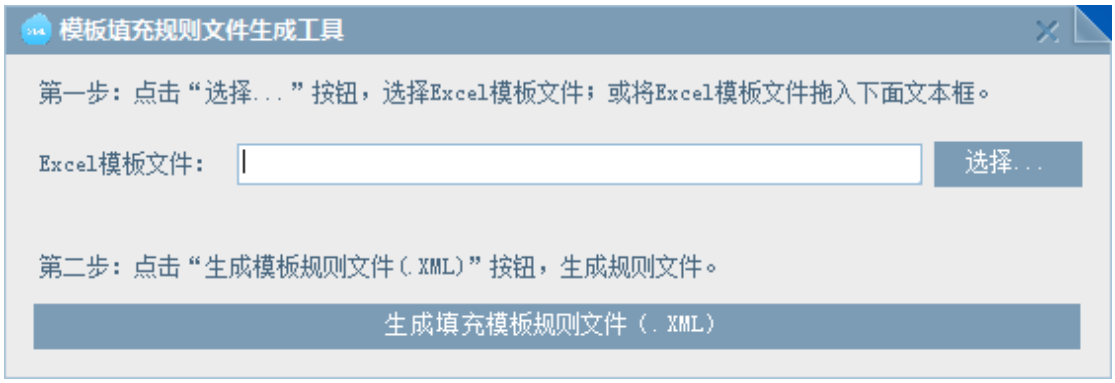






第二步：由模板生成模板填充规则文件

1） 打开模板填充规则文件生成工具



2） 将设计好的模板拖入 Excel 模板文件选择框

3） 点击“生成模板规则文件（.XML）”按钮，生成规则文件。

第三步：填充模板，生成报表【项目中要添加对 ExcelReport 及 NPOI（2.0 以上）的引用】

```
ParameterCollection collection = new ParameterCollection();
collection.Load(@"Template\演示：工资条模板.xml");
```

1、加载模板规则文件

```
List<ElementFormatter> payTableFormatters = new List<ElementFormatter>();
payTableFormatters.Add(new PartFormatter(collection["工资表", "Dept"], "Dept", "生产A部"));
payTableFormatters.Add(new CellFormatter(collection["工资表", "Month"], "2015年4月"));
payTableFormatters.Add(new TableFormatter<SalaryInfo>(collection["工资表", "JobNo"].X, SalaryInfoLogic.GetList(),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "JobNo"].Y, t => t.JobNo),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "Name"].Y, t => t.Name),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "HourlyWage"].Y, t => t.HourlyWage),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "ManHour"].Y, t => t.ManHour),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "UsualOvertime"].Y, t => t.UsualOvertime),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "WeekendOvertime"].Y, t => t.WeekendOvertime),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "LegalOvertime"].Y, t => t.LegalOvertime),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "BasePay"].Y, t => t.BasePay),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "OvertimePay"].Y, t => t.OvertimePay),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "FoodAllowance"].Y, t => t.FoodAllowance),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "OtherAllowance"].Y, t => t.OtherAllowance),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "GrossPay"].Y, t => t.GrossPay),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "BoardWages"].Y, t => t.BoardWages),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "Utilities"].Y, t => t.Utilities),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "BadDebt"].Y, t => t.BadDebt),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "RealWages"].Y, t => t.RealWages),
    new TableColumnInfo<SalaryInfo>(collection["工资表", "Remark"].Y, t => t.Remark)
    ));
```

2、Sheet“工资表”的格式化规则

```
List<ElementFormatter> payStripFormatters = new List<ElementFormatter>();
payStripFormatters.Add(new RepeaterFormatter<SalaryInfo>(collection["工资条", "rptPaySlipStart"], collection["工资条", "rptPaySlipEnd"], SalaryInfoLogic.GetList(),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "Title"], t => "某工厂生产A部工资条"),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "Month"], t => "2015年4月"),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "JobNo"], t => t.JobNo),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "Name"], t => t.Name),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "HourlyWage"], t => t.HourlyWage),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "ManHour"], t => t.ManHour),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "UsualOvertime"], t => t.UsualOvertime),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "WeekendOvertime"], t => t.WeekendOvertime),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "LegalOvertime"], t => t.LegalOvertime),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "BasePay"], t => t.BasePay),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "OvertimePay"], t => t.OvertimePay),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "FoodAllowance"], t => t.FoodAllowance),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "OtherAllowance"], t => t.OtherAllowance),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "GrossPay"], t => t.GrossPay),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "BoardWages"], t => t.BoardWages),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "Utilities"], t => t.Utilities),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "BadDebt"], t => t.BadDebt),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "RealWages"], t => t.RealWages),
    new RepeaterCellInfo<SalaryInfo>(collection["工资条", "Remark"], t => t.Remark)
    ));
```

3、Sheet“工资条”的格式化规则

```
//导出文件到本地
ExportHelper.ExportToLocal(@"Template\演示：工资条模板.xls", saveFileDlg.FileName,
    new SheetFormatterContainer("工资表", payTableFormatters),
    new SheetFormatterContainer("工资条", payStripFormatters)
    );
```

4、生成并导出Excel报表到本地。

【注：导出的Web：ExportHelper.ExportToWeb 】

【注：导出到Buffer：Export.ExportToBuffer】

生成导出报表：

A1 某工厂生产A部工资表

某工厂生产A部工资表

月份：2015年4月

工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
006	梁忠基	11.31	174	28	8	0	1967.94	491.02	0.00	0.00	2458.96	60	30	90	2368.960	
007	韦明珍	6.68	174	28	8	0	1162.32	296.56	0.00	0.00	1458.88	60	30	90	1368.880	
008	韦明初	5.46	174	28	8	0	950.04	245.32	0.00	0.00	1195.36	60	30	90	1105.360	
010	钟燕华	5.25	174	28	8	0	913.50	236.50	0.00	0.00	1150.00	60	30	90	1060.000	
011	谭兴燕	7.19	174	28	8	0	1251.06	317.98	0.00	0.00	1569.04	60	30	90	1479.040	
012	梁海玲	6.76	174	28	8	0	1176.24	299.92	0.00	0.00	1476.16	60	30	90	1386.160	
013	黄金红	5.64	174	28	8	0	981.36	252.88	0.00	0.00	1234.24	60	30	90	1144.240	

工资表 工资条

某工厂生产A部工资条

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
006	梁忠基	11.31	174	28	8	0	1967.94	491.02	0.00	0.00	2458.96	60	30	90	2368.960	

某工厂生产A部工资条

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
007	韦明珍	6.68	174	28	8	0	1162.32	296.56	0.00	0.00	1458.88	60	30	90	1368.880	

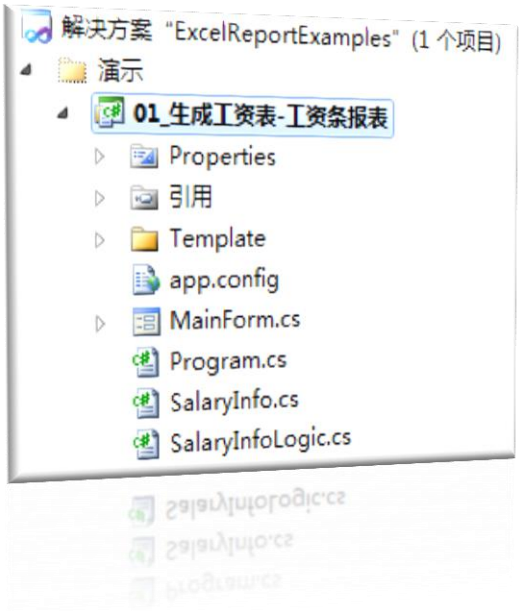
某工厂生产A部工资条

月份：	2015年4月															
工号	姓名	时薪	正常工时	加班工时			基本工资	加班工资	伙食补贴	其他补贴	应得工资	应扣项目		应扣总额	实得金额（元）	备注
				平时	双休日	法定日						伙食费	水电住宿费			
008	韦明初	5.46	174	28	8	0	950.04	245.32	0.00	0.00	1195.36	60	30	90	1105.360	

某工厂生产A部工资条

月份：	2015年4月

【注：演示示例“生成工资表-工资条报表”源码见解决方案：ExcelReportExamples】



3 格式化器示例

通过上一章的示例，我们了解了 ExcelReport 生成报表的步骤，也用到了各种格式化器。在本章，我们将对格式化器逐一展开讲解，以更详细的了解它们。

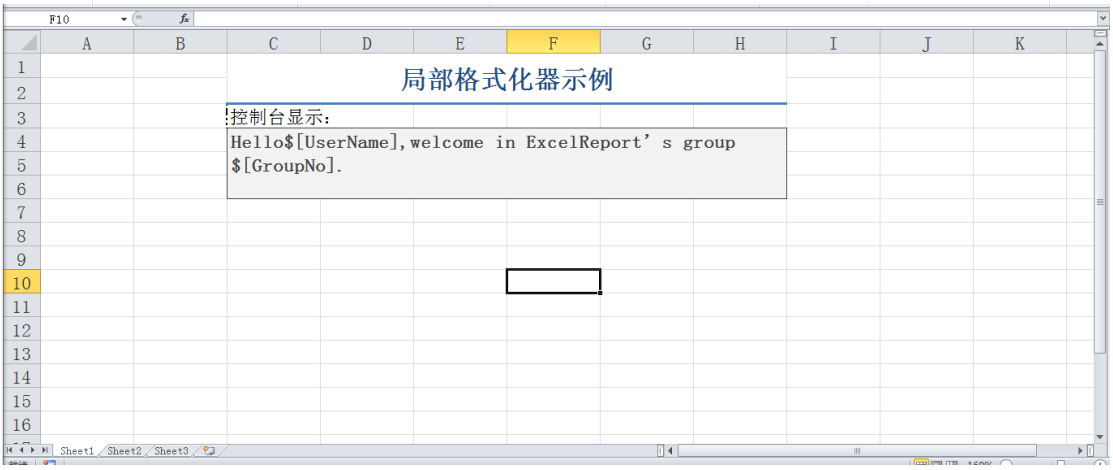
3.1 局部格式化器

局部格式化器用于格式化填充一个单元格内容的一部分。填充数据类型为 string。
构成函数：

```
/// 构造函数 ...
public PartFormatter(Point cellPoint, string parameterName ,string value)
{
    this._cellPoint = cellPoint;
    this._parameterName = parameterName;
    this._value = value;
```

参数	描述
cellPoint	参数所在单元格位置
parameterName	参数名
value	要格式化填充的值

示例：（局部格式化器示例：Hello$[UserName],welcome in ExcelReport’s group $[GroupNo].）
第一步：设计模板

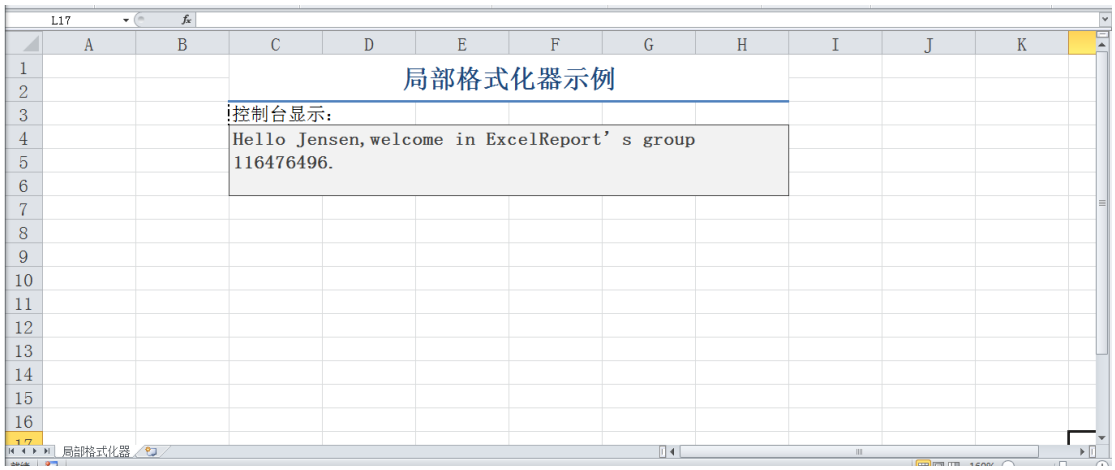


第二步：由模板生成模板填充规则文件

第三步：填充模板，生成报表

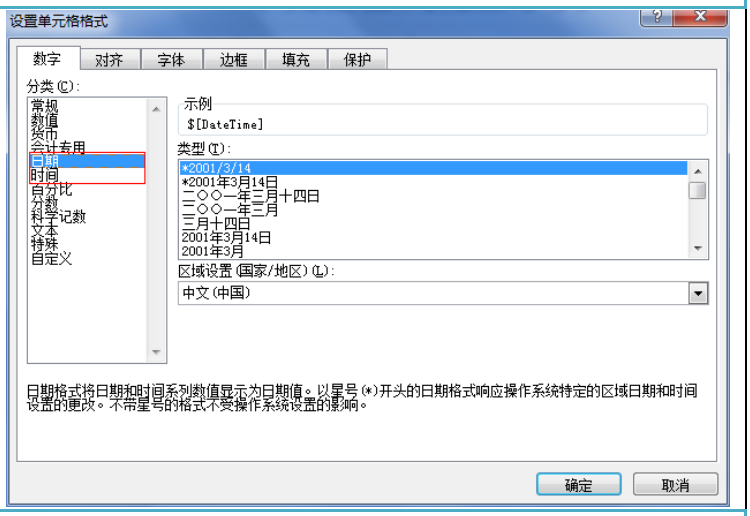
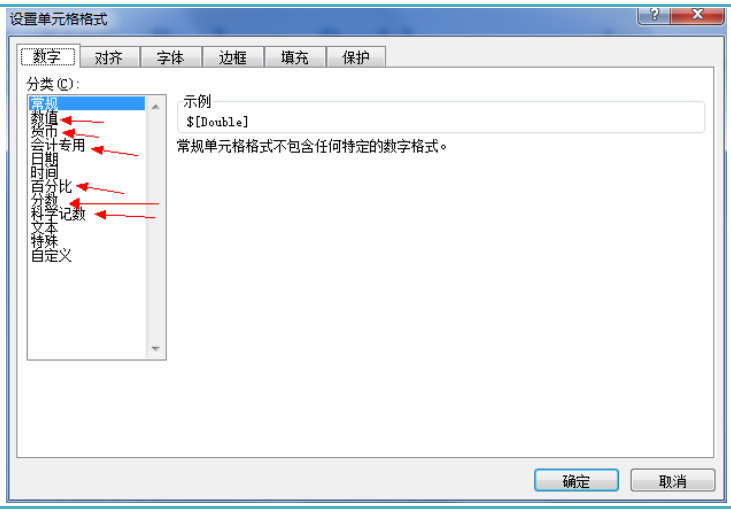
```
ParameterCollection collection = new ParameterCollection();
collection.Load(@"Template\Template.xml");

List<ElementFormatter> formatters = new List<ElementFormatter>();
formatters.Add(new PartFormatter(collection["局部格式化器", "UserName"], "UserName", "Jensen"));
formatters.Add(new PartFormatter(collection["局部格式化器", "GroupNo"], "GroupNo", "116476496"));

//导出文件到本地
ExportHelper.ExportToLocal(@"Template\Template.xls", saveFileDlg.FileName,
    new SheetFormatterContainer("局部格式化器", formatters)
    );
```

生成导出报表：



3.2 单元格格式化器

单元格格式化器用于格式化填充一个单元格。填充数据类型为 object，详见下表：

填充数据类型	Excel 中对应的数据类型	可视化设置项
String	String	
DateTime	DateTime	
Boolean	Boolean	
Int16	Double	
Int32		
Int64		
Byte		
Single		
Double		
UInt16		
UInt32		
UInt64		
Byte[]	Image	

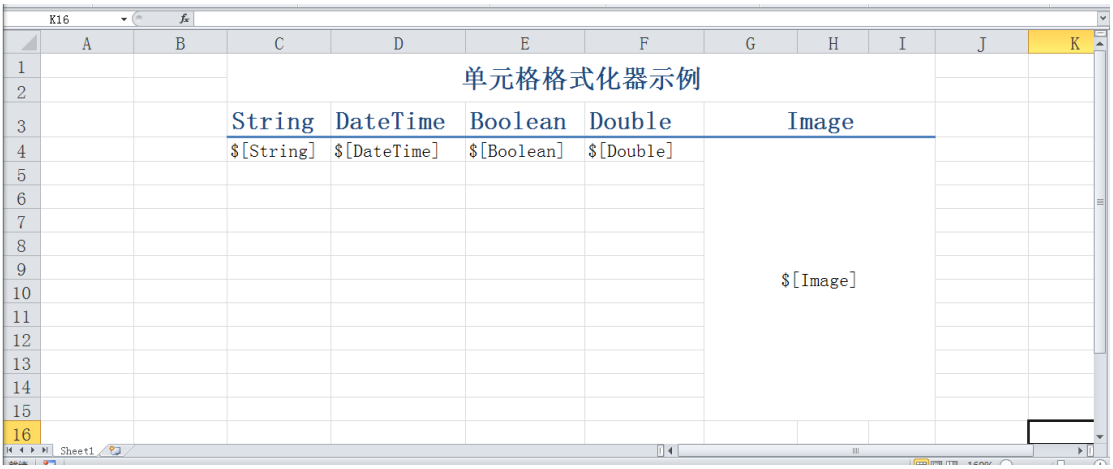
构成函数：

```
/// 构造函数 ...
public CellFormatter(Point cellPoint, object value)
{
    _cellPoint = cellPoint;
    _value = value;
}
```

参数	描述
cellPoint	参数所在单元格位置
value	要格式化填充的值

示例：（单元格格式化器示例）

第一步：设计模板

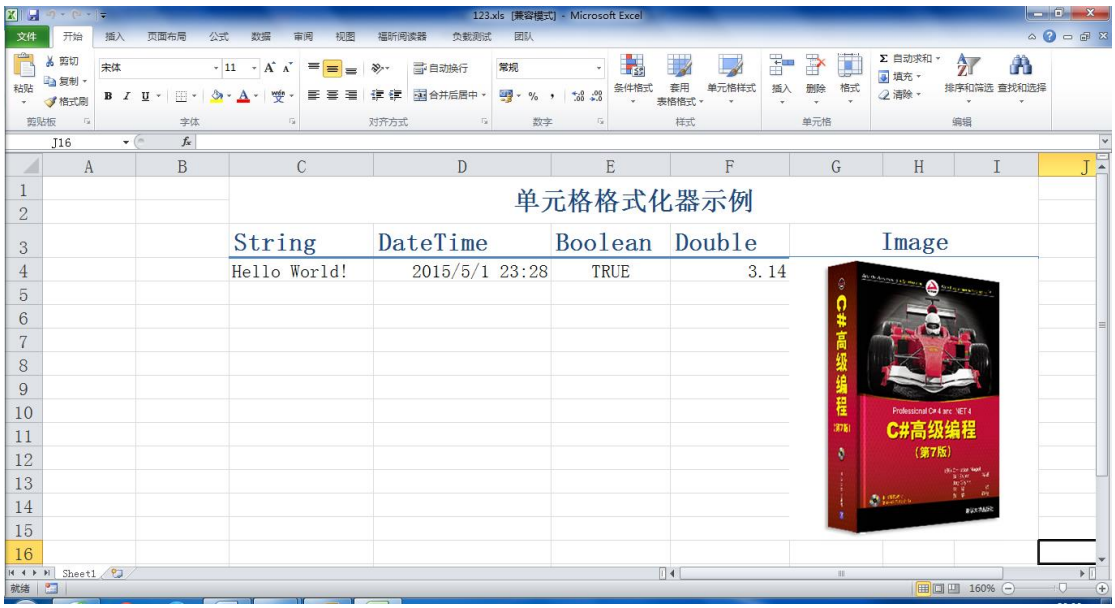


第二步：由模板生成模板填充规则文件（略，详参见第 2 章“演示”）

第三步：填充模板，生成报表

```
ParameterCollection collection = new ParameterCollection();
collection.Load(@"Template\Template.xml");

List<ElementFormatter> formatters = new List<ElementFormatter>();
formatters.Add(new CellFormatter(collection["Sheet1","String"],"Hello World!"));
formatters.Add(new CellFormatter(collection["Sheet1", "DateTime"], DateTime.Now));
formatters.Add(new CellFormatter(collection["Sheet1", "Boolean"], true));
formatters.Add(new CellFormatter(collection["Sheet1", "Double"], 3.14));
formatters.Add(new CellFormatter(collection["Sheet1", "Image"], Image.FromFile("Image/C#高级编程.jpg").ToBuffer()));

//导出文件到本地
ExportHelper.ExportToLocal(@"Template\Template.xls", saveFileDlg.FileName,
    new SheetFormatterContainer("Sheet1", formatters)
    );
```

生成导出报表：



3.3 表格格式化器

表格格式化器用于格式化填充一个表格。填充数据类型为IEnumerable<TSource>。
构造函数：

```
/// 构造函数 ...
public TableFormatter(int templateRowIndex, IEnumerable<TSource> dataSource, params TableColumnInfo<TSource>[] columnInfos)
{
    _templateRowIndex = templateRowIndex;
    _dataSource = dataSource;
    _columnInfoList = new List<TableColumnInfo<TSource>>();
    if (null != columnInfos && columnInfos.Length > 0)
    {
        _columnInfoList.AddRange(columnInfos);
    }
```

参数	描述
templateRowIndex	模板行行标
dataSource	要格式化填充的数据
columnInfos	列信息集合

TableColumnInfo 构造函数：

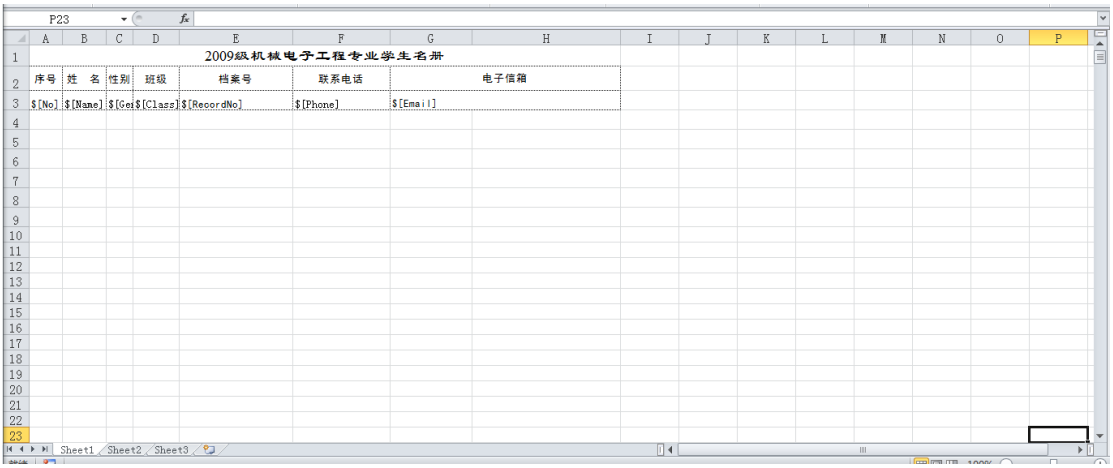
```
/// 构造函数 ...
public TableColumnInfo(int columnIndex, Func<TSource, object> dgSetValue)
{
    _columnIndex = columnIndex;
    _dgSetValue = dgSetValue;
}
```

参数	描述
columnIndex	列标
dgSetValue	Func<TSource, object>类型的委托对象（返回值Object，可参照

	第二节填充数据类型与Excel中数据类型对应表。）

示例：（表格格式化器示例）

第一步：设计模板



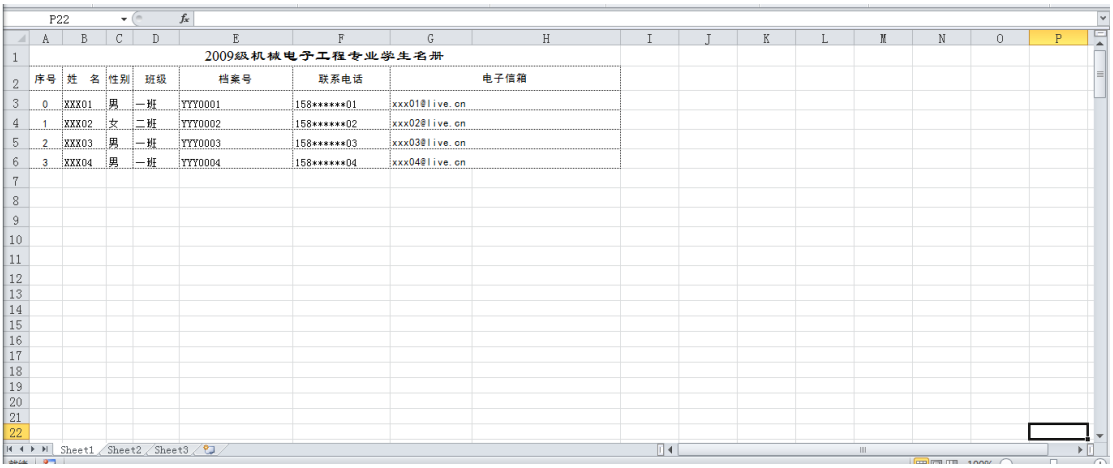
2009级机械电子工程专业学生名册						
序号	姓　名	性别	班级	档案号	联系电话	电子信箱
$[No]	$[Name]	$[Gen	$[Class]	$[RecordNo]	$[Phone]	$[Email]

第二步：由模板生成模板填充规则文件（略，详参见第 2 章“演示”）

第三步：填充模板，生成报表

```
ParameterCollection collection = new ParameterCollection();
collection.Load(@"Template\Template.xml");
int num = 0;
List<ElementFormatter> formatters = new List<ElementFormatter>();
formatters.Add(new TableFormatter<StudentInfo>(collection["Sheet1", "No"].X, StudentLogic.GetList(),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "No"].Y, t => num++),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "Name"].Y, t => t.Name),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "Gender"].Y, t => t.Gender ? "男" : "女"),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "Class"].Y, t => t.Class),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "RecordNo"].Y, t => t.RecordNo),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "Phone"].Y, t => t.Phone),
    new TableColumnInfo<StudentInfo>(collection["Sheet1", "Email"].Y, t => t.Email)
    ));

//导出文件到本地
ExportHelper.ExportToLocal(@"Template\Template.xls", saveFileDlg.FileName,
    new SheetFormatterContainer("Sheet1", formatters)
    );
```

生成导出报表：



2009级机械电子工程专业学生名册						
序号	姓　名	性别	班级	档案号	联系电话	电子信箱
0	XXX01	男	一班	YYY0001	158******01	xxx01@live.cn
1	XXX02	女	二班	YYY0002	158******02	xxx02@live.cn
2	XXX03	男	一班	YYY0003	158******03	xxx03@live.cn
3	XXX04	男	一班	YYY0004	158******04	xxx04@live.cn

提问回复：【关于 TableFormatter 有以下知识点，多次被提问，这里做出统一回答。】

问题一：TableFormatter 的数据源可以是 DataTable 吗？

答：你可以这样写：

```
formatters.Add(new TableFormatter<DataRow>(collection["Sheet1", "No"].X, dt.Select(),
    new TableColumnInfo<DataRow>(collection["Sheet1", "No"].Y, t => num++),
    new TableColumnInfo<DataRow>(collection["Sheet1", "Name"].Y, t => t["Name"])
    ));
```

问题二：我的实体类中没有实现外链，在实体对象中有一个 ID 属性，我想导出的内容是 ID 对应的 Name，这怎么办？

答：和示例中的“性别”没什么区别吧。好吧，贴行代码：

```
formatters.Add(new TableFormatter<StudentInfo>(collection["Sheet1", "No"].X, StudentLogic.GetList(),
   new TableColumnInfo<StudentInfo>(collection["Sheet1", "ClassId"].Y, t => GetClassName(t.ClassId))
   ));
```

3.4 重复单元格式化器

重复单元格式化器用于格式化填充一个重复单元元素。填充数据类型为IEnumerable<TSource>。构造函数：

```
/// 构造函数 ...
public RepeaterFormatter(Point startTagCell, Point endTagCell, IEnumerable<TSource> dataSource, params RepeaterCellInfo<TSource>[] cellInfos)
{
    _startTagCell = startTagCell;
    _endTagCell = endTagCell;
    _dataSource = dataSource;
    _cellInfoList = new List<RepeaterCellInfo<TSource>>();
    if (null != cellInfos && cellInfos.Length > 0)
    {
        _cellInfoList.AddRange(cellInfos);
    }
}
```

参数	描述
startTagCell	重复单元（开始）标识单元格
endTagCell	重复单元（结束）标识单元格
dataSource	要格式化填充的数据
cellInfos	重复单元包含的单元格信息集合

RepeaterCellInfo 构造函数：

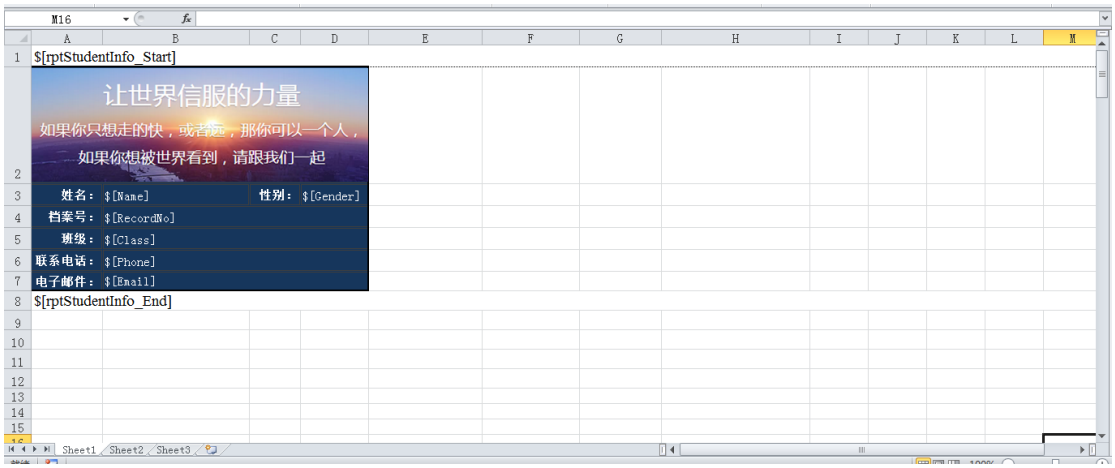
```
/// 构造函数 ...
public RepeaterCellInfo(Point cellPoint, Func<TSource, object> dgSetValue)
{
    _cellPoint = cellPoint;
    _dgSetValue = dgSetValue;
}
```

参数	描述
cellPoint	单元格坐标
dgSetValue	Func<TSource, object>类型的委托对象（返回值Object，可参照

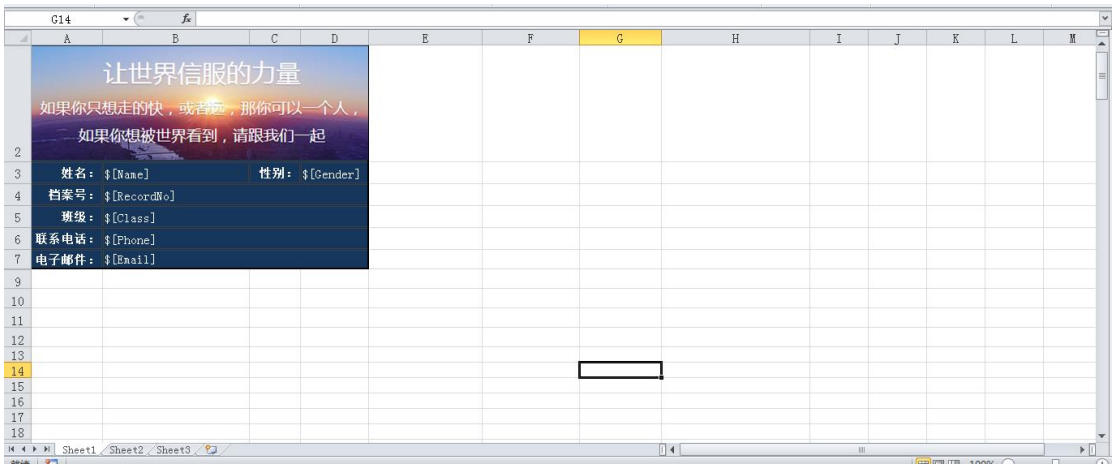
	第二节填充数据类型与Excel中数据类型对应表。）

示例：（重复单元格式化器示例）

第一步：设计模板



根据需求，隐藏重复单元标识行。

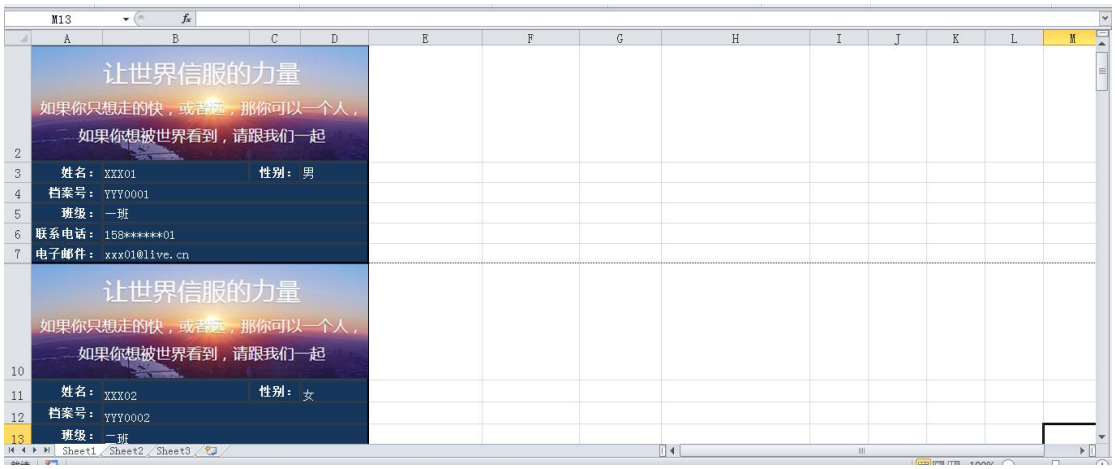


第二步：由模板生成模板填充规则文件（略，详参见第 2 章“演示”）

第三步：填充模板，生成报表

```
ParameterCollection collection = new ParameterCollection();
collection.Load(@"Template\Template.xml");
int num = 0;
List<ElementFormatter> formatters = new List<ElementFormatter>();
formatters.Add(new RepeaterFormatter<StudentInfo>(collection["Sheet1", "rptStudentInfo_Start"],collection["Sheet1", "rptStudentInfo_End"], StudentLogic.GetList(),
    new RepeaterCellInfo<StudentInfo>(collection["Sheet1", "Name"], t => t.Name),
    new RepeaterCellInfo<StudentInfo>(collection["Sheet1", "Gender"], t => t.Gender ? "男" : "女"),
    new RepeaterCellInfo<StudentInfo>(collection["Sheet1", "Class"], t => t.Class),
    new RepeaterCellInfo<StudentInfo>(collection["Sheet1", "RecordNo"], t => t.RecordNo),
    new RepeaterCellInfo<StudentInfo>(collection["Sheet1", "Phone"], t => t.Phone),
    new RepeaterCellInfo<StudentInfo>(collection["Sheet1", "Email"], t => t.Email)
    ));

//导出文件到本地
ExportHelper.ExportToLocal(@"Template\Template.xls", saveFileDlg.FileName,
    new SheetFormatterContainer("Sheet1", formatters)
    );
```

生成导出报表：


M13
让世界信服的力量
如果你只想走的快，或者远，那你可以一个人，
如果你想被世界看到，请跟我们一起
姓名： XXX01
性别： 男
档案号： YYY0001
班级： 一班
联系电话： 158******01
电子邮件： xxx01@live.cn
让世界信服的力量
如果你只想走的快，或者远，那你可以一个人，
如果你想被世界看到，请跟我们一起
姓名： XXX02
性别： 女
档案号： YYY0002
班级： 一班
Sheet1
Sheet2
Sheet3

4 多 Sheet 报表生成

到目前为止，我们所有导出都直接使用的是：ExportHelper类中的静态方法。其实在ExcelReport组件的设计中Export类便是终点了，ExportHelper类正如它的名字只是为了方便操作提供的助手类而已。

【哦，它还是部分类，你可以根据需求扩展。】

```
public static partial class ExportHelper
{

    /// 导出到本地 ...
    public static void ExportToLocal(string templateFile, string targetFile, params SheetFormatterContainer[] containers)...

    /// 导出到Web ...
    public static void ExportToWeb(string templateFile, string targetFile, params SheetFormatterContainer[] containers)...
}
```

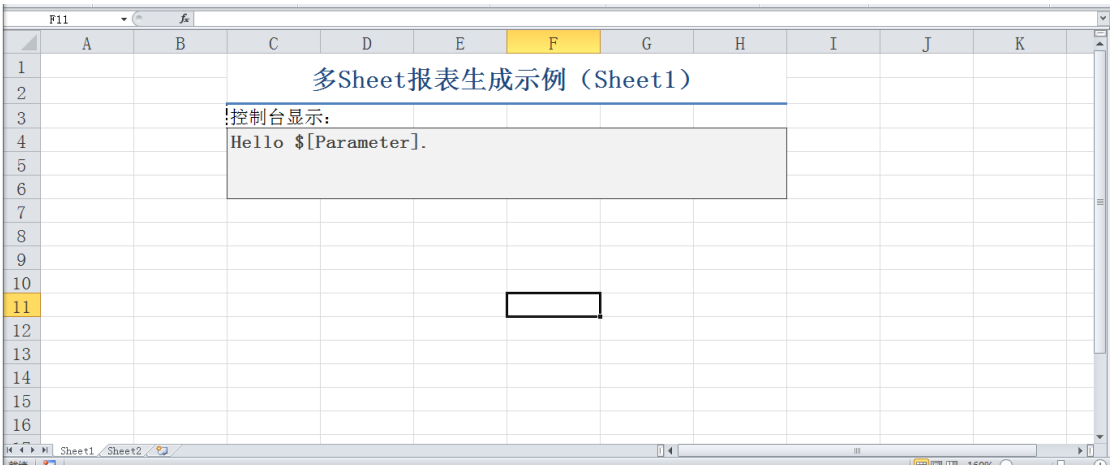
回到本章的主题，我们要谈的是多Sheet报表的生成。什么？这和前边啰嗦的那些有什么关系？好吧，其实关系也不大，我只是想说关注报表的生成，你看Export类就对了。

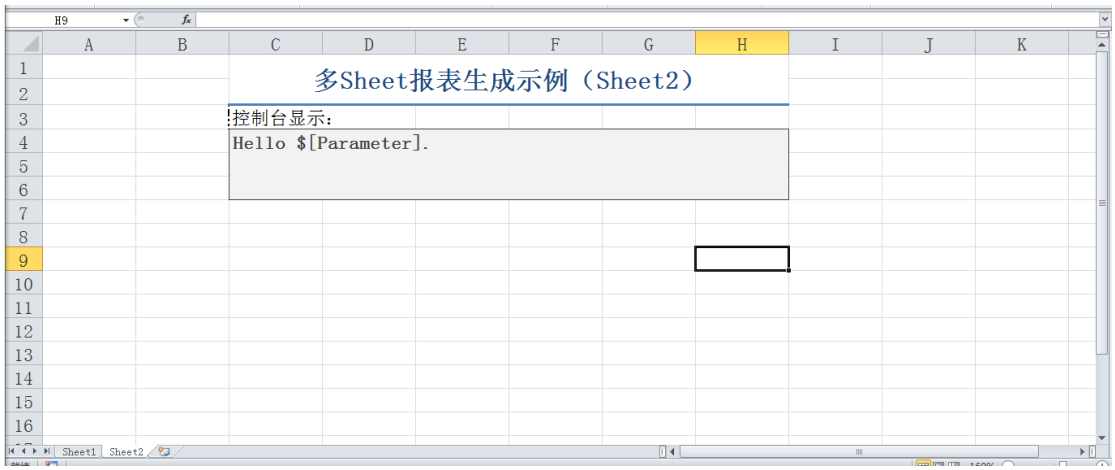
```
public static class Export
{
    /// 导出格式化处理后的文件到二进制文件流 ...
    public static byte[] ExportToBuffer(string templateFile, params SheetFormatterContainer[] containers)...
}
```

参数	描述
templateFile	模板文件路径
containers	Sheet 格式化容器的数组【数组中的每个元素对应一个 Sheet】

示例：（多 Sheet 报表生成示例）

第一步：设计模板



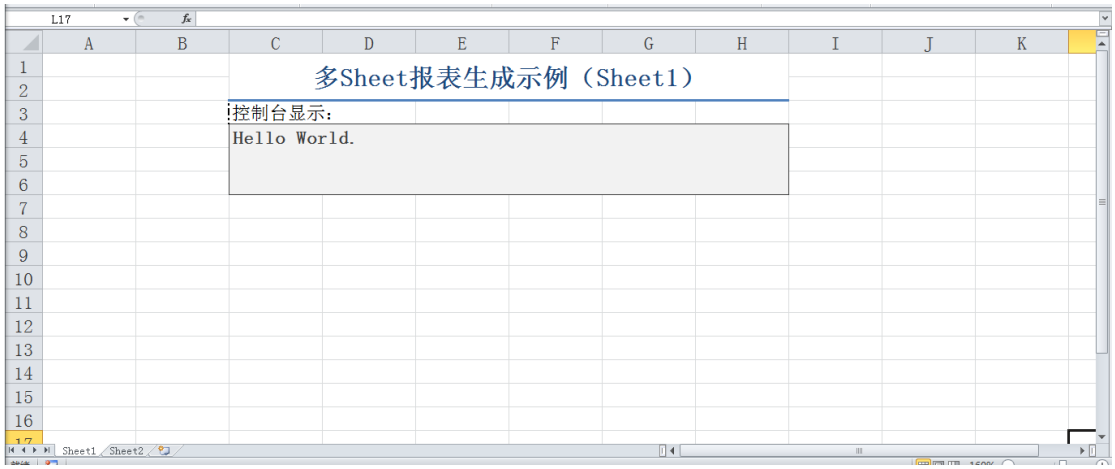


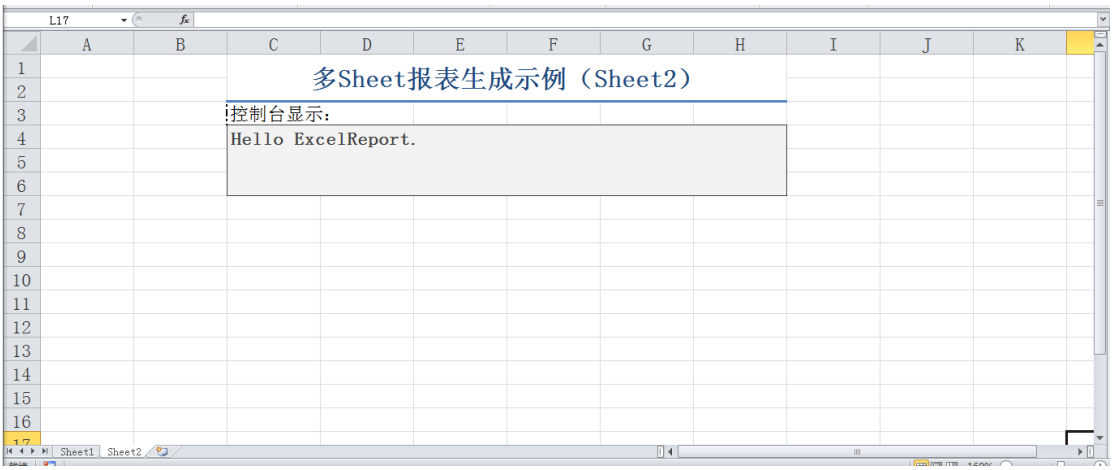
第二步：由模板生成模板填充规则文件（略，详参见第 2 章“演示”）

第三步：填充模板，生成报表

```
ParameterCollection collection = new ParameterCollection();
collection.Load(@"Template\Template.xml");

List<ElementFormatter> sheet1Formatters = new List<ElementFormatter>();
sheet1Formatters.Add(new PartFormatter(collection["Sheet1", "Parameter"], "Parameter","World"));

List<ElementFormatter> sheet2Formatters = new List<ElementFormatter>();
sheet2Formatters.Add(new PartFormatter(collection["Sheet2", "Parameter"], "Parameter", "ExcelReport"));

//导出文件到本地
ExportHelper.ExportToLocal(@"Template\Template.xls", saveFileDlg.FileName,
    new SheetFormatterContainer("Sheet1", sheet1Formatters),
    new SheetFormatterContainer("Sheet2", sheet2Formatters)
    );
```

生成导出报表：




L17
A
B
C
D
E
F
G
H
I
J
K
多Sheet报表生成示例（Sheet2）
控制台显示：
Hello ExcelReport.
Sheet1
Sheet2