

ExcelUtility 类库操作说明（模板导出示例）

设计/开发：左文俊

```
/// <summary>
/// 测试方法：测试依据模板+DataTable 来生成 EXCEL
/// </summary>
[TestMethod]
public void TestExportToExcelWithTemplateByDataTable()
{
    DataTable dt = GetDataTable();//获取数据
    string templateFilePath = AppDomain.CurrentDomain.BaseDirectory + "/excel.xlsx"; //获得 EXCEL 模板路径
    SheetFormatterContainer formatterContainers = new SheetFormatterContainer(); //实例化一个模板数据格式化容器

    PartFormatterBuilder partFormatterBuilder = new PartFormatterBuilder();//实例化一个局部元素格式化器
    partFormatterBuilder.AddFormatter("Title", "跨越 IT 学员");//将模板表格中 Title 的值设置为跨越 IT 学员
    formatterContainers.AppendFormatterBuilder(partFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

    CellFormatterBuilder cellFormatterBuilder = new CellFormatterBuilder();//实例化一个单元格格式化器
    cellFormatterBuilder.AddFormatter("rptdate", DateTime.Today.ToString("yyyy-MM-dd HH:mm"));//将模板表格中 rptdate 的值设置为当前日期
    formatterContainers.AppendFormatterBuilder(cellFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

    //实例化一个表格格式化器，dt.Select()是将 DataTable 转换成 DataRow[], name 表示的模板表格中第一行第一个单元格要填充的数据参数名
    TableFormatterBuilder<DataRow> tableFormatterBuilder = new TableFormatterBuilder<DataRow>(dt.Select(), "name");
    tableFormatterBuilder.AddFormatters(new Dictionary<string, Func<DataRow, object>>{
        {"name", r=>r["Col1"]},//将模板表格中 name 对应 DataTable 中的列 Col1
        {"sex", r=>r["Col2"]},//将模板表格中 sex 对应 DataTable 中的列 Col2
        {"km", r=>r["Col3"]},//将模板表格中 km 对应 DataTable 中的列 Col3
        {"score", r=>r["Col4"]},//将模板表格中 score 对应 DataTable 中的列 Col4
        {"result", r=>r["Col5"]}//将模板表格中 result 对应 DataTable 中的列 Co5
    });
    formatterContainers.AppendFormatterBuilder(tableFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

    string excelPath = ExcelUtility.Export.ToExcelWithTemplate(templateFilePath, "table", formatterContainers);
    Assert.IsTrue(File.Exists(excelPath));
}
```

模板如下：


```
tableFormatterBuilder.AddFormatters(new Dictionary<string, Func<Student, object>>{
    {"name",r=>r.Name},//将模板表格中 name 对应 Student 对象中的属性 Name
    {"sex",r=>r.Sex},//将模板表格中 sex 对应 Student 对象中的属性 Sex
    {"km",r=>r.KM},//将模板表格中 km 对应 Student 对象中的属性 KM
    {"score",r=>r.Score},//将模板表格中 score 对应 Student 对象中的属性 Score
    {"result",r=>r.Result},//将模板表格中 result 对应 Student 对象中的属性 Result
});
formatterContainers.AppendFormatterBuilder(tableFormatterBuilder);

string excelPath = ExcelUtility.Export.ToExcelWithTemplate(templateFilePath, "table", formatterContainers);
Assert.IsTrue(File.Exists(excelPath));

}
```

模板同上一个模板

导出结果如下：

| | A | B | C | D | E |
|----|-----------|----|------------------|----|----|
| 1 | 跨越IT学员成绩表 | | | | |
| 2 | 日期： | | 2016-01-23 00:00 | | |
| 3 | 姓名 | 性别 | 科目 | 得分 | 结果 |
| 4 | Name1 | 男 | 科目1 | 4 | 待定 |
| 5 | Name2 | 女 | 科目2 | 8 | 待定 |
| 6 | Name3 | 男 | 科目3 | 12 | 待定 |
| 7 | Name4 | 女 | 科目4 | 16 | 待定 |
| 8 | Name5 | 男 | 科目5 | 20 | 待定 |
| 9 | Name6 | 女 | 科目6 | 24 | 待定 |
| 10 | Name7 | 男 | 科目7 | 28 | 待定 |
| 11 | Name8 | 女 | 科目8 | 32 | 待定 |

```
/// <summary>
/// 测试方法：测试依据模板+DataTable 来生成多表格 EXCEL（注意：由于 ExcelReport 框架限制，目前仅支持模板文件格式为：
xls）
/// </summary>
[TestMethod]
public void TestExportToRepeaterExcelWithTemplateByDataTable()
{
    DataTable dt = GetDataTable();//获取数据
    string templateFilePath = AppDomain.CurrentDomain.BaseDirectory + "/excel2.xls";//获得 EXCEL 模板路径
    SheetFormatterContainer formatterContainers = new SheetFormatterContainer();//实例化一个模板数据格式化容器

    //实例化一个可重复表格格式化器，dt.Select()是将 DataTable 转换成 DataRow[], rpt_begin 表示的模板表格开始位置参数名，
    rpt_end 表示的模板表格结束位置参数名
    RepeaterFormatterBuilder<DataRow> tableFormatterBuilder = new RepeaterFormatterBuilder<DataRow>(dt.Select(), "rpt_begin",
    "rpt_end");
```

```
tableFormatterBuilder.AddFormatters(new Dictionary<string, Func<DataRow, object>>{
    {"sex",r=>r["Col2"]},//将模板表格中 sex 对应 DataTable 中的列 Col2
    {"km",r=>r["Col3"]},//将模板表格中 km 对应 DataTable 中的列 Col3
    {"score",r=>r["Col4"]},//将模板表格中 score 对应 DataTable 中的列 Col4
    {"result",r=>r["Col5"]},//将模板表格中 result 对应 DataTable 中的列 Co5
});
```

PartFormatterBuilder<DataRow> partFormatterBuilder2 = new PartFormatterBuilder<DataRow>();//实例化一个可嵌套的局部元素格式化器

```
partFormatterBuilder2.AddFormatter("name", r => r["Col1"]);//将模板表格中 name 对应 DataTable 中的列 Col1
tableFormatterBuilder.AppendFormatterBuilder(partFormatterBuilder2);//添加到可重复表格格式化器中，作为其子格式化器
```

CellFormatterBuilder<DataRow> cellFormatterBuilder = new CellFormatterBuilder<DataRow>();//实例化一个可嵌套的单元格格式化器

cellFormatterBuilder.AddFormatter("rptdate", r => DateTime.Today.ToString("yyyy-MM-dd HH:mm"));//将模板表格中 rptdate 的值设置为当前日期

```
tableFormatterBuilder.AppendFormatterBuilder(cellFormatterBuilder);//添加到可重复表格格式化器中，作为其子格式化器
```

formatterContainers.AppendFormatterBuilder(tableFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

```
string excelPath = ExcelUtility.Export.ToExcelWithTemplate(templateFilePath, "multitable", formatterContainers);
Assert.IsTrue(File.Exists(excelPath));
}
```

模板如下：

| | A | B | C | D | |
|---|----------------|--------|-----------|-------------|--|
| 1 | [\$rpt_begin] | | | | |
| 2 | [\$name]的个人成绩单 | | | | |
| 3 | | | 日期： | [\$rptdate] | |
| 4 | 性别 | 科目 | 得分 | 结果 | |
| 5 | [\$sex] | [\$km] | [\$score] | [\$result] | |
| 6 | [\$rpt_end] | | | | |
| 7 | | | | | |
| 8 | | | | | |

导出结果如下：

formatterContainers.AppendFormatterBuilder(tableFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

```
RepeaterFormatterBuilder<ClassInfo> repeaterFormatterBuilder = new RepeaterFormatterBuilder<ClassInfo>(classList, "lv_begin",
"lv_end");//实例化一个可重复表格格式化器
repeaterFormatterBuilder.AddFormatters(new Dictionary<string, Func<ClassInfo, object>> {
    {"class",r=>r.ClassName}, //模板参数与数据源 ClassInfo 属性对应关系，下同
    {"stucount",r=>r.StudentCount},
    {"clsmaster",r=>r.Master},
    {"lvitem",r=>r.LevelName}
});
formatterContainers.AppendFormatterBuilder(repeaterFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

string excelPath = ExcelUtility.Export.ToExcelWithTemplate(templateFilePath, "school", formatterContainers);
Assert.IsTrue(File.Exists(excelPath));

}
```

模板如下：

| | A | B | C | D |
|----|-----------------|--------------|---------------|------------|
| 1 | \$[school]综合信息表 | | | |
| 2 | | | | |
| 3 | 年级 | 班级数 | 年级主任 | |
| 4 | [\$lv] | [\$clscount] | [\$lvmaster] | |
| 5 | | | | |
| 6 | 合计 | 0 | 1 | |
| 7 | | | | |
| 8 | [\$lv_begin] | | | |
| 9 | 班级 | 学生数 | 班主任 | 所属年级 |
| 10 | [\$class] | [\$stucount] | [\$clsmaster] | [\$lvitem] |
| 11 | | | | |
| 12 | [\$lv_end] | | | |
| 13 | | | | |

导出结果如下：

| | A | B | C | D |
|----|-----------|-----|------|------|
| 1 | 跨越小学综合信息表 | | | |
| 2 | | | | |
| 3 | 年级 | 班级数 | 年级主任 | |
| 4 | 1年级 | 35 | 牛1 | |
| 5 | 2年级 | 40 | 牛2 | |
| 6 | 3年级 | 45 | 牛3 | |
| 7 | 4年级 | 50 | 牛4 | |
| 8 | 5年级 | 55 | 牛5 | |
| 9 | 6年级 | 60 | 牛6 | |
| 10 | 7年级 | 65 | 牛7 | |
| 11 | 8年级 | 70 | 牛8 | |
| 12 | 9年级 | 75 | 牛9 | |
| 13 | | | | |
| 14 | 合计 | 495 | 9 | |
| 15 | | | | |
| 16 | 班级 | 学生数 | 班主任 | 所属年级 |
| 17 | 1-1班 | 10 | 谢某1 | 1年级 |
| 18 | | | | |
| 19 | 班级 | 学生数 | 班主任 | 所属年级 |
| 20 | 1-2班 | 15 | 张某1 | 1年级 |
| 21 | | | | |
| 22 | 班级 | 学生数 | 班主任 | 所属年级 |
| 23 | 1-3班 | 20 | 赵某1 | 1年级 |
| 24 | | | | |
| 25 | 班级 | 学生数 | 班主任 | 所属年级 |
| 26 | 1-4班 | 25 | 张某1 | 1年级 |

```
/// <summary>
```

/// 测试方法：测试依据复杂模板（含固定表格，可重复表格中嵌套表格）+DataTable 来生成 EXCEL（注意：由于 ExcelReport 框架限制，目前仅支持模板文件格式为：xls）

```
/// </summary>
```

```
[TestMethod]
```

```
public void TestExportToExcelWithTemplateByList3()
```

```
{
```

```
    var schoolLevelList = SchoolLevel.GetList();
```

```
    var classList = ClassInfo.GetListWithLevels();
```

```
string templateFilePath = AppDomain.CurrentDomain.BaseDirectory + "/mb1.xls"; //获得 EXCEL 模板路径
SheetFormatterContainer formatterContainers = new SheetFormatterContainer(); //实例化一个模板数据格式化容器
```

```
PartFormatterBuilder partFormatterBuilder = new PartFormatterBuilder();
partFormatterBuilder.AddFormatter("school", "跨越小学");
formatterContainers.AppendFormatterBuilder(partFormatterBuilder);
```

TableFormatterBuilder<SchoolLevel> tableFormatterBuilder = new TableFormatterBuilder<SchoolLevel>(schoolLevelList, "lv");//实例化一个表格格式化器

```
tableFormatterBuilder.AddFormatters(new Dictionary<string, Func<SchoolLevel, object>>
{
    {"lv", r=>r.LevelName}, //模板参数与数据源 SchoolLevel 属性对应关系，下同
    {"clscount", r=>r.ClassCount},
    {"lvmaster", r=>r.Master}
});
```

formatterContainers.AppendFormatterBuilder(tableFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

```
RepeaterFormatterBuilder<KeyValuePair<string, List<ClassInfo>>> repeaterFormatterBuilder = new
RepeaterFormatterBuilder<KeyValuePair<string, List<ClassInfo>>>(classList, "lv_begin", "lv_end");
repeaterFormatterBuilder.AddFormatter("lvitem", r=>r.Key);
```

```
TableFormatterBuilder<KeyValuePair<string, List<ClassInfo>>, ClassInfo> tableFormatterBuilder2=new
TableFormatterBuilder<KeyValuePair<string, List<ClassInfo>>, ClassInfo>(r=>r.Value, "class");
tableFormatterBuilder2.AddFormatter("class", r=>r.ClassName);
tableFormatterBuilder2.AddFormatter("stucount", r=>r.StudentCount);
tableFormatterBuilder2.AddFormatter("clsmaster", r=>r.Master);
```

```
repeaterFormatterBuilder.AppendFormatterBuilder(tableFormatterBuilder2);
```

formatterContainers.AppendFormatterBuilder(repeaterFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生效

```
string excelPath = ExcelUtility.Export.ToExcelWithTemplate(templateFilePath, "school", formatterContainers);
Assert.IsTrue(File.Exists(excelPath));
```

```
}
```

模板如下：

| | A | B | C |
|----|-----------------|--------------|---------------|
| 1 | \$[school]综合信息表 | | |
| 2 | | | |
| 3 | 年级 | 班级数 | 年级主任 |
| 4 | \$[lv] | \$[clscount] | \$[lvmaster] |
| 5 | | | |
| 6 | 合计 | 0 | 0 |
| 7 | | | |
| 8 | \$[lv_begin] | | |
| 9 | 年级: | \$[lvitem] | |
| 10 | 班级 | 学生数 | 班主任 |
| 11 | \$[class] | \$[stucount] | \$[clsmaster] |
| 12 | | | |
| 13 | \$[lv_end] | | |

导出结果如下:

| | A | B | C |
|----|-----------|-----|------|
| 1 | 跨越小学综合信息表 | | |
| 2 | | | |
| 3 | 年级 | 班级数 | 年级主任 |
| 4 | 1年级 | 30 | 牛1 |
| 5 | 2年级 | 35 | 牛2 |
| 6 | 3年级 | 40 | 牛3 |
| 7 | 4年级 | 45 | 牛4 |
| 8 | 5年级 | 50 | 牛5 |
| 9 | 6年级 | 55 | 牛6 |
| 10 | 7年级 | 60 | 牛7 |
| 11 | 8年级 | 65 | 牛8 |
| 12 | 9年级 | 70 | 牛9 |
| 13 | | | |
| 14 | 合计 | 450 | 0 |
| 15 | | | |
| 16 | 年级: | 1年级 | |
| 17 | 班级 | 学生数 | 班主任 |
| 18 | 1-1班 | 10 | 张某1 |
| 19 | 1-2班 | 15 | 李某1 |
| 20 | 1-3班 | 20 | 李某1 |
| 21 | 1-4班 | 25 | 张某1 |
| 22 | 1-5班 | 30 | 李某1 |
| 23 | 1-6班 | 35 | 张某1 |
| 24 | 1-7班 | 40 | 赵某1 |
| 25 | 1-8班 | 45 | 张某1 |
| 26 | 1-9班 | 50 | 赵某1 |
| 27 | 1-10班 | 55 | 李某1 |
| 28 | | | |
| 29 | 年级: | 2年级 | |
| 30 | 班级 | 学生数 | 班主任 |
| 31 | 2-1班 | 15 | 赵某2 |
| 32 | 2-2班 | 20 | 李某2 |
| 33 | 2-3班 | 25 | 赵某2 |
| 34 | 2-4班 | 30 | 张某2 |
| 35 | 2-5班 | 35 | 张某2 |
| 36 | 2-6班 | 40 | 张某2 |
| 37 | 2-7班 | 45 | 赵某2 |
| 38 | 2-8班 | 50 | 张某2 |
| 39 | 2-9班 | 55 | 赵某2 |
| 40 | 2-10班 | 60 | 李某2 |
| 41 | | | |
| 42 | 年级: | 3年级 | |
| 43 | 班级 | 学生数 | 班主任 |
| 44 | 3-1班 | 20 | 李某3 |
| 45 | 3-2班 | 25 | 李某3 |
| 46 | 3-3班 | 30 | 张某3 |

/// <summary>

```

/// 测试方法：测试依据复杂模板（多工作簿，且含固定表格，可重复表格）+DataSet 来生成 EXCEL
/// </summary>
[TestMethod]
public void TestExportToExcelWithTemplateByDataSet()
{
    var ds = GetDataSet();

    string templateFilePath = AppDomain.CurrentDomain.BaseDirectory + "/mb2.xls"; //获得 EXCEL 模板路径
    Dictionary<string, SheetFormatterContainer> formatterContainerDic = new Dictionary<string, SheetFormatterContainer>(); //实例
    化一个模板数据格式化容器数组，包含两个 SheetFormatterContainer 用于格式化两个工作簿

    #region 创建第一个工作簿格式化容器，并设置相关参数对应关系

    SheetFormatterContainer formatterContainer1 = new SheetFormatterContainer();

    PartFormatterBuilder partFormatterBuilder = new PartFormatterBuilder();
    partFormatterBuilder.AddFormatter("school", "跨越小学");
    formatterContainer1.AppendFormatterBuilder(partFormatterBuilder);

    TableFormatterBuilder<DataRow> tableFormatterBuilder = new TableFormatterBuilder<DataRow>(ds.Tables[0].Select(), "lv");//实例
    化一个表格格式化器
    tableFormatterBuilder.AddFormatters(new Dictionary<string, Func<DataRow, object>>
    {
        {"lv",r=>r["Col1"]}, //模板参数与数据源 DataTable 属性对应关系，下同
        {"clscount",r=>r["Col2"]},
        {"lvmaster",r=>r["Col3"]}
    });
    formatterContainer1.AppendFormatterBuilder(tableFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才会生
    效

    RepeaterFormatterBuilder<DataRow> repeaterFormatterBuilder = new RepeaterFormatterBuilder<DataRow>(ds.Tables[1].Select(),
    "lv_begin", "lv_end");//实例化一个可重复表格格式化器
    repeaterFormatterBuilder.AddFormatters(new Dictionary<string, Func<DataRow, object>> {
        {"class",r=>r["Col1"]}, //模板参数与数据源 ClassInfo 属性对应关系，下同
        {"stucount",r=>r["Col2"]},
        {"clsmaster",r=>r["Col3"]},
        {"lvitem",r=>r["Col4"]}
    });
    formatterContainer1.AppendFormatterBuilder(repeaterFormatterBuilder);//添加到工作簿格式容器中，注意只有添加进去了才
    会生效

    formatterContainerDic.Add("table1", formatterContainer1);//添加到工作簿格式容器数组中，注意此处的 Key 值为模板上工作
    簿的名称，此处即为：table1

    #endregion

    #region 创建第二个工作簿格式化容器，并设置相关参数对应关系

```

```
SheetFormatterContainer formatterContainer2 = new SheetFormatterContainer(); //实例化一个模板数据格式化容器
```

```
PartFormatterBuilder partFormatterBuilder2 = new PartFormatterBuilder(); //实例化一个局部元素格式化器
```

```
partFormatterBuilder2.AddFormatter("Title", "跨越 IT 学员");//将模板表格中 Title 的值设置为跨越 IT 学员
```

```
formatterContainer2.AppendFormatterBuilder(partFormatterBuilder2); //添加到工作簿格式容器中，注意只有添加进去了才会生效
```

```
CellFormatterBuilder cellFormatterBuilder2 = new CellFormatterBuilder(); //实例化一个单元格格式化器
```

```
cellFormatterBuilder2.AddFormatter("rptdate", DateTime.Today.ToString("yyyy-MM-dd HH:mm"));//将模板表格中 rptdate 的值设置为当前日期
```

```
formatterContainer2.AppendFormatterBuilder(cellFormatterBuilder2); //添加到工作簿格式容器中，注意只有添加进去了才会生效
```

```
//实例化一个表格格式化器，dt.Select()是将 DataTable 转换成 DataRow[]，name 表示的模板表格中第一行第一个单元格要填充的数据参数名
```

```
TableFormatterBuilder<DataRow> tableFormatterBuilder2 = new TableFormatterBuilder<DataRow>(ds.Tables[2].Select(), "name");
```

```
tableFormatterBuilder2.AddFormatters(new Dictionary<string, Func<DataRow, object>>{
```

```
    {"name", r => r["Col1"]}, //将模板表格中 name 对应 DataTable 中的列 Col1
```

```
    {"sex", r => r["Col2"]}, //将模板表格中 sex 对应 DataTable 中的列 Col2
```

```
    {"km", r => r["Col3"]}, //将模板表格中 km 对应 DataTable 中的列 Col3
```

```
    {"score", r => r["Col4"]}, //将模板表格中 score 对应 DataTable 中的列 Col4
```

```
    {"result", r => r["Col5"]}, //将模板表格中 result 对应 DataTable 中的列 Col5
```

```
});
```

```
formatterContainer2.AppendFormatterBuilder(tableFormatterBuilder2); //添加到工作簿格式容器中，注意只有添加进去了才会生效
```

```
formatterContainerDic.Add("table2", formatterContainer2); //添加到工作簿格式容器数组中，注意此处的 Key 值为模板上工作簿的名称，此处即为：table2
```

```
#endregion
```

```
string excelPath = ExcelUtility.Export.ToExcelWithTemplate(templateFilePath, formatterContainerDic);
```

```
Assert.IsTrue(File.Exists(excelPath));
```

```
}
```

模板如下：

| | | | | | |
|----|-----------------|--------------|---------------|------------|---|
| | A | B | C | D | E |
| 1 | \${school}综合信息表 | | | | |
| 2 | | | | | |
| 3 | 年级 | 班级数 | 年级主任 | | |
| 4 | \${lv} | \${clscount} | \${lvmaster} | | |
| 5 | | | | | |
| 6 | 合计 | 0 | 1 | | |
| 7 | | | | | |
| 8 | \${lv_begin} | | | | |
| 9 | 班级 | 学生数 | 班主任 | 所属年级 | |
| 10 | \${class} | \${stucount} | \${clsmaster} | \${lvitem} | |
| 11 | | | | | |
| 12 | \${lv_end} | | | | |
| 13 | | | | | |

| | | | | | |
|----|--------------|---------|--------|-------------|------------|
| | A | B | C | D | E |
| 1 | \${Title}成绩表 | | | | |
| 2 | | | 日期: | \${rptdate} | |
| 3 | 姓名 | 性别 | 科目 | 得分 | 结果 |
| 4 | \${name} | \${sex} | \${km} | \${score} | \${result} |
| 5 | | | | | |
| 6 | 总计 | | | 0 | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |

导出结果如下：

| | | | | | |
|-----|-----------|--------|------|------|---|
| P17 | | | | | |
| | A | B | C | D | E |
| 1 | 跨越小学综合信息表 | | | | |
| 2 | | | | | |
| 3 | 年级 | 班级数 | 年级主任 | | |
| 4 | 1年级 | 25 | 牛1 | | |
| 5 | 2年级 | 20 | 牛2 | | |
| 6 | 3年级 | 40 | 牛3 | | |
| 7 | 4年级 | 35 | 牛4 | | |
| 8 | 5年级 | 55 | 牛5 | | |
| 9 | 6年级 | 50 | 牛6 | | |
| 10 | 7年级 | 50 | 牛7 | | |
| 11 | 8年级 | 60 | 牛8 | | |
| 12 | 9年级 | 60 | 牛9 | | |
| 13 | | | | | |
| 14 | 合计 | 395 | 9 | | |
| 15 | | | | | |
| 16 | 班级 | 学生数 | 班主任 | 所属年级 | |
| 17 | 1年级 | 1-1班 | 10 | 李某1 | |
| 18 | | | | | |
| 19 | 班级 | 学生数 | 班主任 | 所属年级 | |
| 20 | 1年级 | 1-2班 | 15 | 张某1 | |
| 21 | | | | | |
| 22 | 班级 | 学生数 | 班主任 | 所属年级 | |
| | table1 | table2 | | | |
| 就绪 | | | | | |

| | | | | | |
|----|----------------------|--------|-----|----|--------------------|
| HY | | | | | |
| | A | B | C | D | E |
| 1 | 跨越IT学员成绩表 | | | | |
| 2 | 日期: 2016-01-23 00:00 | | | | |
| 3 | 姓名 | 性别 | 科目 | 得分 | 结果 |
| 4 | Name1 | 男 | 科目1 | 2 | 待定则则则则则则则则则则则则则则则则 |
| 5 | Name2 | 女 | 科目2 | 4 | 待定则则则则则则则则则则则则则则则则 |
| 6 | Name3 | 男 | 科目3 | 6 | 待定则则则则则则则则则则则则则则则则 |
| 7 | Name4 | 女 | 科目4 | 8 | 待定则则则则则则则则则则则则则则则则 |
| 8 | Name5 | 男 | 科目5 | 10 | 待定则则则则则则则则则则则则则则则则 |
| 9 | Name6 | 女 | 科目6 | 12 | 待定则则则则则则则则则则则则则则则则 |
| 10 | Name7 | 男 | 科目7 | 14 | 待定则则则则则则则则则则则则则则则则 |
| 11 | Name8 | 女 | 科目8 | 16 | 待定则则则则则则则则则则则则则则则则 |
| 12 | Name9 | 男 | 科目9 | 18 | 待定则则则则则则则则则则则则则则则则 |
| | table1 | table2 | | | |
| 就绪 | | | | | |

注意事项说明:

1. 模板文件格式建议以 XLS 为主, 因为可重复表格导出方法若采用 XLSX, 则导出的数据存在问题;
2. 模板导出方法支持多种情况, 可多种组合, 灵活多变, 基本可以满足所有的模板导出方法 (图片动态生成除外), 但上手相对也就复杂一些, 故大家若在使用的时候有不明白的地方, 可以联系我, 谢谢!
3. 模板导出方法支持图片固定位置导出, 但暂不支持透明图片浮动于文字上方, 后面我会继续研究图片浮动解决方案。