ExcelUtility 类库使用说明

编写：左文俊

日期：2016-1-8

- 1. ExcelUtility 功能：
  - 1.将数据导出到 EXCEL(支持 XLS,XLSX，支持多种类型模板，支持列宽自适应)
    - **类名：ExcelUtility. Export**
  - 2.将 EXCEL 数据导入到数据对象中（DataTable、Dataset，支持 XLS,XLSX）
    - **类名：ExcelUtility. Import**
- 2. ExcelUtility 依赖组件：
  - 1.NPOI →操作 EXCEL 核心类库
  - 2.NPOI.Extend →NPOI 扩展功能
  - 3. ExcelReport →基于 NPOI 的二次扩展，实现模板化导出功能
  - 4. System.Windows.Forms →导出或导入时，弹出文件选择对话框
- 3.具体使用方法介绍（示例代码）：
  - 1.导出功能：

```
/// <summary>
/// 测试方法：测试将 DataTable 导出到 EXCEL，无模板
/// </summary>
[TestMethod]
public void TestExportToExcelByDataTable()
{
   DataTable dt = GetDataTable();
   string excelPath = ExcelUtility.Export.ToExcel(dt, "导出结果");
   Assert.IsTrue(File.Exists(excelPath));
}
```

结果如下图示：

Col1	Col2	Col3	Col4	Col5	Col6
Name1	男	科目1	4	待定	0c3a1157f2324b2cb6e4cae318e58d96
Name2	女	科目2	8	待定	961b4a894fd345afacd3a8483a4ed645
Name3	男	科目3	12	待定	334a072ac66d4ec8933e83a34aca9045
Name4	女	科目4	16	待定	9778dc80695046ed9a56376e84728380
Name5	男	科目5	20	待定	3f2a5838de3a487abd45900050d4bc50
Name6	女	科目6	24	待定	b519af4b81c9447ba494cc402b5db85b
Name7	男	科目7	28	待定	fd4f00890946481cba2d421cbf943bb1
Name8	女	科目8	32	待定	e13b77c7c937441aa43038102085928b
Name9	男	科目9	36	待定	d852d7bd34024a399e64bb7553e31129
Name10	女	科目10	40	待定	4971063414ea4f759278c58d744128ae

```
/// <summary>
/// 测试方法：测试将 DataTable 导出到 EXCEL，无模板，且指定导出的列名
/// </summary>
[TestMethod]
public void TestExportToExcelByDataTable2()
{
   DataTable dt = GetDataTable();
```

```
    string[] expColNames = { "Col1", "Col2", "Col3", "Col4", "Col5" };
    string excelPath = ExcelUtility.Export.ToExcel(dt, "导出结果", null, expColNames);
    Assert.IsTrue(File.Exists(excelPath));
}
```

结果如下图示：

Col1	Col2	Col3	Col4	Col5
Name1	男	科目1	2	待定
Name2	女	科目2	4	待定
Name3	男	科目3	6	待定
Name4	女	科目4	8	待定
Name5	男	科目5	10	待定
Name6	女	科目6	12	待定
Name7	男	科目7	14	待定
Name8	女	科目8	16	待定
Name9	男	科目9	18	待定
Name10	女	科目10	20	待定

```
/// <summary>
/// 测试方法：测试将 DataTable 导出到 EXCEL，无模板，且指定导出的列名，以及导出列名的重命名
/// </summary>
[TestMethod]
public void TestExportToExcelByDataTable3()
{
    DataTable dt = GetDataTable();
    string[] expColNames = { "Col1", "Col2", "Col3", "Col4", "Col5" };
    Dictionary<string, string> expColAsNames = new Dictionary<string, string>() {
        {"Col1","列一"},
        {"Col2","列二"},
        {"Col3","列三"},
        {"Col4","列四"},
        {"Col5","列五"}
    };
    string excelPath = ExcelUtility.Export.ToExcel(dt, "导出结果", null, expColNames,expColAsNames);
    Assert.IsTrue(File.Exists(excelPath));
}
```

结果如下图示：

列一	列二	列三	列四	列五
Name1	男	科目1	3	待定
Name2	女	科目2	6	待定
Name3	男	科目3	9	待定
Name4	女	科目4	12	待定
Name5	男	科目5	15	待定
Name6	女	科目6	18	待定
Name7	男	科目7	21	待定
Name8	女	科目8	24	待定
Name9	男	科目9	27	待定
Name10	女	科目10	30	待定

```
/// <summary>
/// 测试方法：测试将 DataTable 导出到 EXCEL，无模板，且指定导出列名的重命名
/// </summary>
[TestMethod]
public void TestExportToExcelByDataTable4()
{
```

```
        DataTable dt = GetDataTable();
        Dictionary<string, string> expColAsNames = new Dictionary<string, string>() {
          {"Col1","列一"},
          {"Col5","列五"}
        };
        string excelPath = ExcelUtility.Export.ToExcel(dt, "导出结果", null, null, expColAsNames);
        Assert.IsTrue(File.Exists(excelPath));
    }
```

结果如下图示：

列一	Col2	Col3	Col4	列五	Col6
Name1	男	科目1	1	待定	5f0b6874e93e407a97c86475fba72511
Name2	女	科目2	2	待定	3f5961d74ad741a5af65eb4e8a1488c0
Name3	男	科目3	3	待定	47c31106830240 2e9878e72fe755517f
Name4	女	科目4	4	待定	bdd00961314e4abda81822e23573d5ee
Name5	男	科目5	5	待定	7991f6f769c54800b341e5b5872890fb
Name6	女	科目6	6	待定	86edf14457a1451bafa0455e28b53b31
Name7	男	科目7	7	待定	0ec6465f72e94fa78400ff452f3052c8
Name8	女	科目8	8	待定	df09c141eefc4f3089a84509e8880e17
Name9	男	科目9	9	待定	50d0db79788f40c1ab0f6aab3b218a44
Name10	女	科目10	10	待定	ce8103ec7b8648af9faffa9b346c0d35

```
    /// <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<DataRow> formatterContainers = new SheetFormatterContainer<DataRow>(); //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);//添加到工作薄格式容器中，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);//添加到工作薄格式容器中，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 Col
          {"result",r=>r["Col5"]}//将模板表格中 result 对应 DataTable 中的列 Co5
        });
        formatterContainers.AppendFormatterBuilder(tableFormatterBuilder);//添加到工作薄格式容器中，注意只有添加进去了才会生效
```

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

模板如下图示：

结果如下图示：

跨越IT学员成绩表

日期： 2016-01-08 00:00

姓名	性别	科目	得分	结果
Name1	男	科目1	2	待定
Name2	女	科目2	4	待定
Name3	男	科目3	6	待定
Name4	女	科目4	8	待定
Name5	男	科目5	10	待定
Name6	女	科目6	12	待定
Name7	男	科目7	14	待定
Name8	女	科目8	16	待定
Name9	男	科目9	18	待定
Name10	女	科目10	20	待定

```
    /// <summary>
    /// 测试方法：测试依据模板+List 来生成 EXCEL
    /// </summary>
    [TestMethod]
    public void TestExportToExcelWithTemplateByList()
    {
        List<Student> studentList = GetStudentList();//获取数据
        string templateFilePath = AppDomain.CurrentDomain.BaseDirectory + "/excel.xlsx"; //获得 EXCEL 模板路径
        SheetFormatterContainer<Student> formatterContainers = new SheetFormatterContainer<Student>(); //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);//添加到工作薄格式容器中，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);//添加到工作薄格式容器中，注意只有添加进去了才会生效

        //实例化一个表格格式化器，studentList 本身就是可枚举的无需转换，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<Student> tableFormatterBuilder = new TableFormatterBuilder<Student>(studentList, "name");
```

```
        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<Student>(templateFilePath, "table", formatterContainers);
        Assert.IsTrue(File.Exists(excelPath));

    }
```

结果如下图示：（模板与上面相同）

跨越IT学员成绩表

日期：2016-01-08 00:00

姓名	性别	科目	得分	结果
Name1	男	科目1	3	待定
Name2	女	科目2	6	待定
Name3	男	科目3	9	待定
Name4	女	科目4	12	待定
Name5	男	科目5	15	待定
Name6	女	科目6	18	待定
Name7	男	科目7	21	待定
Name8	女	科目8	24	待定
Name9	男	科目9	27	待定
Name10	女	科目10	30	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 NPOI 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<DataRow> formatterContainers = new SheetFormatterContainer<DataRow>(); //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 Col
          {"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);//添加到工作薄格式容器中，注意只有添加进去了才会生效

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

模板如下图示：（**注意：该模板仅支持 XLS 格式文件，XLSX 下存在问题**）

结果如下图示：

Name1的个人成绩单

日期： 2016-01-08 00:00

性别	科目	得分	结果
男	科目1	4	待定

Name2的个人成绩单

日期： 2016-01-08 00:00

性别	科目	得分	结果
女	科目2	8	待定

Name3的个人成绩单

日期： 2016-01-08 00:00

性别	科目	得分	结果
男	科目3	12	待定

Name4的个人成绩单

日期： 2016-01-08 00:00

性别	科目	得分	结果
女	科目4	16	待定

Name5的个人成绩单

日期： 2016-01-08 00:00

性别	科目	得分	结果
男	科目5	20	待定

Name6的个人成绩单

日期： 2016-01-08 00:00

性别	科目	得分	结果
女	科目6	24	待定

Name7的个人成绩单

以下是模拟数据来源所定义的方法（配合测试）：

```
private DataTable GetDataTable()
{
    DataTable dt = new DataTable();
    for (int i = 1; i <= 6; i++)
    {
        if (i == 4)
        {
            dt.Columns.Add("Col" + i.ToString(), typeof(double));
        }
        else
        {
            dt.Columns.Add("Col" + i.ToString(), typeof(string));
        }
    }

    for (int i = 1; i <= 10; i++)
    {
        dt.Rows.Add("Name" + i.ToString(), (i % 2) > 0 ? "男" : "女", "科目" + i.ToString(), i * new Random().Next(1, 5), "待定", Guid.NewGuid().ToString("N"));
```

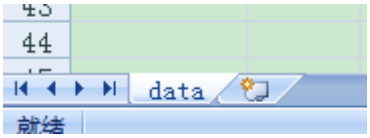
```
    }

    return dt;
}

private List<Student> GetStudentList()
{
    List<Student> studentList = new List<Student>();
    for (int i = 1; i <= 10; i++)
    {
        studentList.Add(new Student
        {
            Name = "Name" + i.ToString(),
            Sex = (i % 2) > 0 ? "男" : "女",
            KM = "科目" + i.ToString(),
            Score = i * new Random().Next(1, 5),
            Result = "待定"
        });
    }
    return studentList;
}

class Student
{
    public string Name { get; set; }

    public string Sex { get; set; }

    public string KM { get; set; }

    public double Score { get; set; }

    public string Result { get; set; }
}
```

- 2.导入功能：

```
        /// <summary>
        /// 测试方法：测试将指定的 EXCEL 数据导入到 DataTable
        /// </summary>
        [TestMethod]
        public void TestImportToDataTableFromExcel()
        {
            //null 表示由用户选择 EXCEL 文件路径，data 表示要导入的 sheet 名,0 表示数据标题行
            DataTable dt=  ExcelUtility.Import.ToDataTable(null, "data", 0);
            Assert.AreNotEqual(0, dt.Rows.Count);
        }
```

数据源文件内容如下图示：

	A	B	C	D	E	F
1	Col1	Col2	Col3	Col4	Col5	Col6
2	Name1	男	科目1	4	待定	0c3a1157f2324b2cb6e4cae318e58d96
3	Name2	女	科目2	8	待定	961b4a894fd345afacd3a8483a4ed645
4	Name3	男	科目3	12	待定	334a072ac66d4ec8933e83a34aca9045
5	Name4	女	科目4	16	待定	9778dc80695046ed9a56376e84728380
6	Name5	男	科目5	20	待定	3f2a5838de3a487abd45900050d4bc50
7	Name6	女	科目6	24	待定	b519af4b81c9447ba494cc402b5db85b
8	Name7	男	科目7	28	待定	fd4f00890946481cba2d421cbf943bb1
9	Name8	女	科目8	32	待定	e13b77c7c937441aa43038102085928b
10	Name9	男	科目9	36	待定	d852d7bd34024a399e64bb7553e31129
11	Name10	女	科目10	40	待定	4971063414ea4f759278c58d744128ae
12						
13						



- 4.其它说明：
  - 1.无模板导出方法，如果未指定参数 **filePath 或设为 null,**则将会弹出导出对话框让用户选择导出路径；
  - 2.无模板导出方法，支持 XLS 及 XLSX 格式，不论客户端是否安装 OFFICE，均不受影响
  - 3.有模板导出方法，模板若以 XLSX 格式制作，则导出可选择生成 XLS 及 XLSX 格式，但若以 XLS 格式制作，则只能生成 XLS，即：高版本兼容低版本。**特殊情况：多表格模板导出方法，模板只支持 XLS 格式，若使用 XLSX，则导出的数据会存在问题，经分析是 NPOI 或 ExcelReport 组件的 BUG**
  - 4. ExcelUtility 类库导出与导入方法，我均作了深度封装，大家在使用的时候无需学习 NPOI 的相关知识就能轻松实现导出与导入方法，降低了使用难度，若在使用过程中发现问题或不解的地方，可及时向我反馈，我会尽力解决，谢谢！

    联系方式：QQ: 3345272365　　E-MAIL:kyezuo@126.com