

XlsxToLua

(v7.2)

使用说明

张齐

2018-11-4

软件简介

本软件的目标是为游戏开发建立一套完善的表格配置和数据转换的方案。使得游戏策划人员通过 **Excel** 表格管理游戏配置数据，并通过本软件完成配表数据的检查以及客户端、服务器所使用的数据形式的生成。而今，为了实现游戏产品的热更新，越来越多的游戏采用 **lua** 作为主要开发语言，本软件的核心功能之一便是可将 **Excel** 表按配置的格式要求转换为 **lua** 文件，可供游戏项目直接读取使用。为了适应更多的应用场景，本软件也支持通用数据格式如 **csv**、**json** 的导出

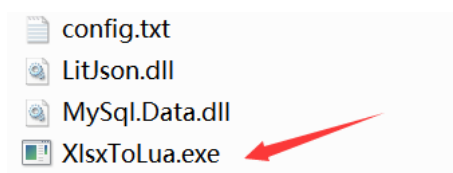
本软件大致提供以下功能：

- 1、 [表格检查功能](#)。游戏表格配置错误对于上线游戏所造成的后果往往十分严重，而通常项目采用的 **Excel** 表格查错方式为借助 **Excel** 公式、有效性验证或者由程序为每张表格单独编写检查函数，工作量较大。本软件针对游戏配置表格中常见的检查类型，抽象出一系列通用的检查规则，比如值范围检查、非空检查、唯一性检查、引用关系检查等等，可以直接针对某列进行检查，也可以根据具体逻辑需求，在本软件提供的框架下编写自定义的检查函数来满足对列或整表的特殊逻辑检查需求
- 2、 [导出表格为 lua 文件功能](#)。此功能可将 **Excel** 表导出为 **lua table** 的形式，从而能被 **lua** 语言直接解析使用，省去了如果用 **csv** 等通用格式作为数据载体，程序不得不为每张配置表写数据读取转换代码的麻烦

- 3、 [导出表格到 MySQL 数据库功能](#)。如果服务器采用 MySQL 数据库，并直接将配置表格作为数据库表进行存储，便可以使用本软件提供的此功能实现将 Excel 表同步到 MySQL 数据库中（包含表结构和数据）
- 4、 [导出表格为 csv、json 等通用数据格式的功能](#)。为适应更多的应用场景，本软件还提供将 Excel 表导出为常见的通用数据格式功能，供不使用 lua 语言的项目使用

安装说明

本软件基于微软.NET 2.0 平台使用 C#编程语言开发，故需在 Windows 系统下安装.NET 2.0 运行环境才可运行，软件本身无需安装，只需通过命令行方式传入运行参数打开执行，具体参数参看[软件运行参数](#)

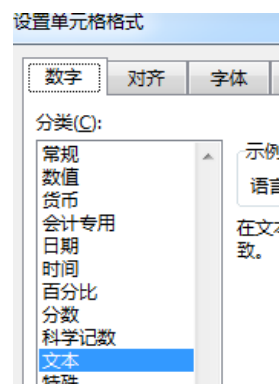


但在使用前还需要安装以下依赖组件以及进行如下设置：

- 1、 微软 Excel 2007 及以上版本
- 2、 微软官方 Office 2007 数据连接组件（<http://www.microsoft.com/en-US/download/details.aspx?id=23734>），安装之后本软件才可以通过效率较高的 OleDb 方式对 Excel 文件进行读取
- 3、 因为微软 OleDb 方式读取的策略，使得 Excel 文件中超过 256 字符的单元格内容只能读取到前 256 个字符，需要修改 Windows 系统注册表解决此问题。本软件提供辅助工具，自动修改注册表，只需以管理员身份运行 ChangeExcelRegedit.exe，出现如下图所示窗口后，说明修改成功

```
本程序用于自动修改Windows7及以上版本系统注册表指定项，解决OleDb方式读取Excel文件  
时超过256字符的单元格中的内容无法读取完整的问题  
  
修改"SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Excel\TypeGuessRows":  
值已设为0，无需修改  
修改"SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Lotus\TypeGuessRows":  
值已设为0，无需修改  
修改"SOFTWARE\Wow6432Node\Microsoft\Office\12.0\Access Connectivity Engine\Engin  
es\Excel\TypeGuessRows":  
值已由"8"修改为"0"  
修改"SOFTWARE\Wow6432Node\Microsoft\Office\12.0\Access Connectivity Engine\Engin  
es\Lotus\TypeGuessRows":  
值已由"8"修改为"0"  
修改"SOFTWARE\Wow6432Node\Microsoft\Office\14.0\Access Connectivity Engine\Engin  
es\Excel\TypeGuessRows":  
不存在，无需修改  
修改"SOFTWARE\Wow6432Node\Microsoft\Office\15.0\Access Connectivity Engine\Engin  
es\Excel\TypeGuessRows":  
不存在，无需修改  
修改"SOFTWARE\Wow6432Node\Microsoft\Office\16.0\Access Connectivity Engine\Engin  
es\Excel\TypeGuessRows":  
不存在，无需修改  
  
修改完毕，按任意键退出
```

经测试发现，还需将 Excel 文件的单元格格式改为“文本”格式，才能保证正确读取超过 256 字符的单元格



若所有 Excel 表格中都不会出现超过 256 字符的单元格，则无需进行此步骤的任何操作

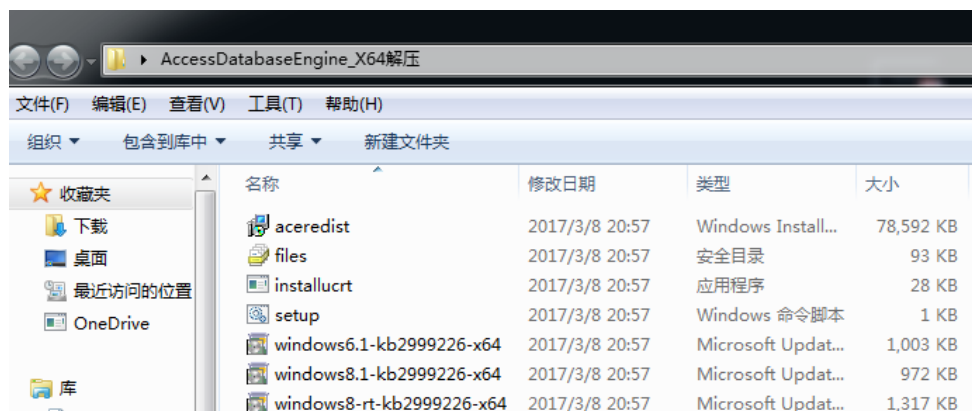
注意：微软官方 Office 2007 数据连接组件，经测试也适用于更高版本的 Office（包含 2010、2013、2016 版本）。

如果用户想安装更高版本的数据连接组件，请至微软官网下载。数据连接组件自 2010 版本以后，提供 32 位和 64 位的下载，比如最新 2016 版本 <https://www.microsoft.com/en-us/download/details.aspx?id=54920>，但安装数据连接组件选 32 位还是 64 位必须与 Office 版本的位数相匹配，否则安装程序会弹出错误提示并阻止安装

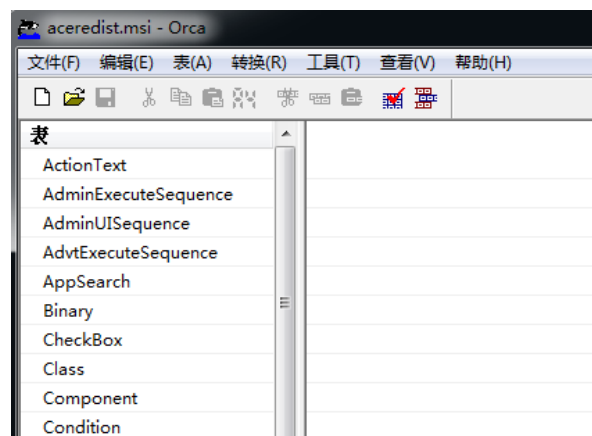
但极个别的电脑中会出现，明明安装的是 64 位的 Office，在安装微软 64 位数据连接组件时却提示 Office 是 32 位的，不得已重新安装 32 位 Office 后再安装 32 位数据连接组件时又提示 Office 是 32 位的。对于这种情况，只能通过修改数据连接组件的安装包，具体方法如下（以上面网址提供的 64 位 2016 版数据连接组件为例）：

1、下载微软出品的用于对 Windows Installer 数据库表进行编辑的 Orca 软件

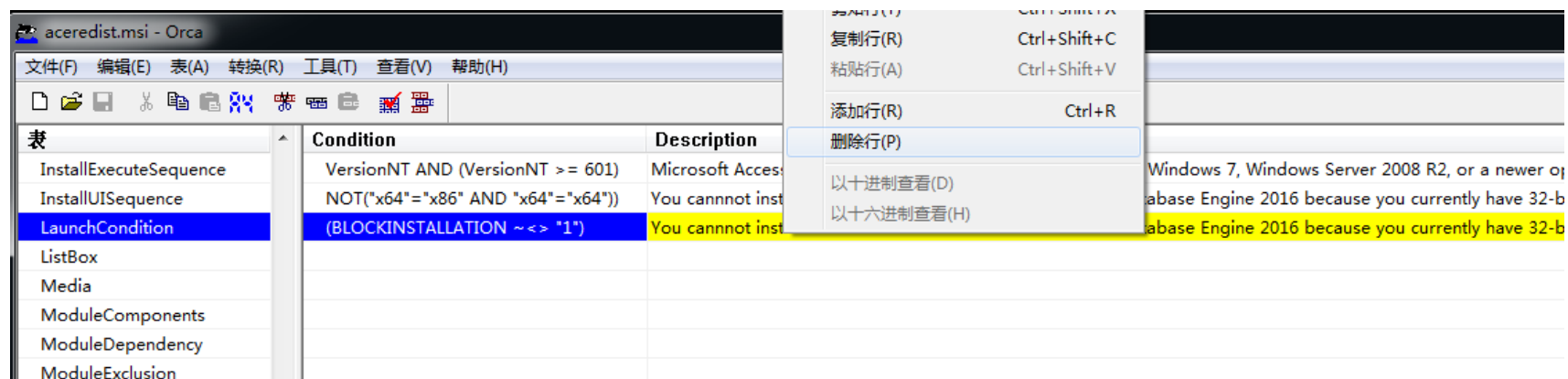
2、用 WinRAR 或 7-Zip 等压缩软件，打开下载的 64 位 2016 版数据连接组件安装包 AccessDatabaseEngine_X64.exe，然后解压到一个文件夹中，如下图所示：



3、用 Orca 软件打开上图中的 aceredist.msi，如下图所示：



4、在左侧找到 LaunchCondition 项并点击，然后删除右侧 BLOCKINSTALLATION 项后保存编辑后的 msi



5、直接在解压到的文件夹中，运行 aceredist.msi，就可跳过位数的强制要求进行数据连接组件的安装

本软件中的数据类型

本软件目前支持的数据类型如下表所示，注意下列数据类型声明中所使用的括号均为英文括号：

名称	说明
string	最普通的字符串类型，如果仅是声明为 string 类型，单元格中填写数据中的空格均会被保留。若希望进行各种导出时自动去掉首尾的空白字符，请声明为 string(trim)
int	整形数字，该类型的数据默认情况下不允许为空，除非设置了 -allowedNullNumber 运行参数
long	长整形数字，要求同上
float	可带小数的数字，在 C# 中用 double 存储，要求同上
bool	填写 true 或数字 1 表示真， false 或者数字 0 表示假，其他值非法，不允许为空值
date	<p>日期型，包含年月日时分秒的日期格式，格式声明形如 date(input=yyyy-MM-dd HH:mm:ss toLua=#dateTable toDatabase=#1970sec)，其中三个可选参数 input、toLua 和 toDatabase，它们的声明需要用 隔开。3 个参数的作用如下：</p> <p>①input 参数规定 Excel 单元格中填写的格式，有以下 2 种选项：(1)可以填写符合 C#类库要求的标准时间格式，如 yyyy-MM-dd HH:mm:ss。(2)设置为#1970sec 表示填写的是距离 1970 年 1 月 1 日的秒数，</p>

	<p>设置为#1970msec 则为毫秒数。若未声明 input 参数则使用 config 配置文件中的 defaultDateInputFormat 配置项</p> <p>②toLua 参数规定导出至 lua 文件中的形式，有以下 3 种选项：(1)可以填写符合 C#类库要求的标准时间格式，本软件将会按此格式将日期转为 string 字符串形式，但 lua 自身类库中不含将 string 转为时间的库函数，不利于对时间进行后续操作，除非自己编写相应的转换函数或者此时间仅作为界面展示之用，否则不推荐使用这种 toLua 的输出形式。(2)设置为#1970sec 或#1970msec，含义同上。(3)设置为#dateTable，本软件将导出为 os.date("!*t",xxx)这样的代码，其中 xxx 为时间转成的距离 1970 年 1 月 1 日的秒数，lua 语言在解析导出的 lua 文件时，上面的 lua 库函数可将日期转为 lua table 形式。若未声明 toLua 参数则使用 config 配置文件中的 defaultDateToLuaFormat 配置项</p> <p>③toDatabase 参数规定导出至数据库中的形式，有以下 2 种选项：(1)可以填写符合 C#类库要求的标准时间格式，MySQL 数据库中的 datetime 类型可识别并赋值，也可保存为 varchar 等文本类型。(2)设置为#1970sec 或#1970msec，适合导出到数值类型的字段。若未声明 toDatabase 参数则使用 config 配置文件中的 defaultDateToDatabaseFormat 配置项</p> <p>注意：时间戳以本地时区下，距离 1970-01-01 00:00:00 的时间计</p>
time	时 间 型 ， 用 时 分 秒 表 示 一 个 时 间 ， 格 式 声 明 形 如

	<p>time(input=HH:mm:ss toLua=#sec toDatabase=HH:mm:ss)，其中有三个可选参数 input、toLua 和 toDatabase，它们的声明需要用 隔开。3 个参数的作用如下：</p> <p>①input 参数规定 Excel 单元格中填写的格式，有以下 2 种选项：(1)可以填写符合 C#类库要求的标准时间格式，如 HH:mm:ss。(2)设置为#sec 表示填写的是距离今日 0 点已过去的秒数。若未声明 input 参数则使用 config 配置文件中的 defaultTimeInputFormat 配置项</p> <p>②toLua 参数规定导出至 lua 文件中的形式，有以下 2 种选项：(1)可以填写符合 C#类库要求的标准时间格式，本软件将会按此格式将日期转为 string 字符串形式。(2)设置为#sec，含义同上。若未声明 toLua 参数则使用 config 配置文件中的 defaultTimeToLuaFormat 配置项</p> <p>③toDatabase 参数规定导出至数据库中的形式，有以下 2 种选项：(1)可以填写符合 C#类库要求的标准时间格式，MySQL 数据库中的 time 类型可识别并赋值，也可保存为 varchar 等文本类型。(2)设置为#sec，适合导出到数值类型的字段。若未声明 toDatabase 参数则使用 config 配置文件中的 defaTimeToDatabaseFormat 配置项</p>
dict	<p>字典类型，声明格式为 dict[子元素个数]，用于表示组成一个复杂数据结构的相关元素的集合，比如用一个 dict 可以表示一个奖励项，dict[3]其后三列为 type(int)、id(int)、count(int)分别表示奖励项的类型、id、奖励数量。dict 下的子元素变量名不允许重复。dict 类型的子元素可以为任意数据类型，dict 型可</p>

以与 dict 或 array 型形成嵌套，且不限制嵌套层数。注意：dict 的声明独占一行，如果某行数据希望该列有效则留空，如果填-1 表示该行数据不需要此字段信息，即如果某行不填写奖励项，则在“奖励物”列的单元格中填-1

奖励物	类型	id	数量
reward	type	id	count
dict[3]	int	int	int
	1	10001	20

array

数组类型，声明格式为 array[类型:个数]，用于表示多个相同数据结构的字段组成的数组，比如想通过 3 个数字表示某个游戏关卡的低中高三个难度的开放等级，则可定义为 array[int:3]。注意 array 类型中的元素会按 1、2、3……依次作为 table 的 key，不允许为 array 型下属子元素列像 dict 型子元素列那样指定 key 名。array 类型的子元素可以为任意数据类型，array 型可以与 dict 或 array 型形成嵌套，且不限制嵌套层数。注意：同 dict 型一样，array 的声明也独占一行，也是填-1 表示该行数据不需要此字段信息。注意 array 下属的 dict 或 array 类型的子字段若用-1 声明为无效，则后续的子元素也必须声明为无效的。比如配置的某一关推图的奖励列表为 array[dict[3]:5]，如果配置时本关卡只奖励两种道具，则必须配置在第 1、2 个子元素中，后 3 个子元素标为无效

	<table><tr><td>开放等级</td><td>低难度</td><td>中难度</td><td>高难度</td></tr><tr><td>openRank</td><td></td><td></td><td></td></tr><tr><td>array[int:3]</td><td>int</td><td>int</td><td>int</td></tr><tr><td></td><td>10</td><td>30</td><td>50</td></tr></table>	开放等级	低难度	中难度	高难度	openRank				array[int:3]	int	int	int		10	30	50
开放等级	低难度	中难度	高难度														
openRank																	
array[int:3]	int	int	int														
	10	30	50														
lang	<p>国际化文本类型，出多国语言版本的游戏或软件，通常为每种语言创建一个 lang 文件，配置每个 key 对应指定语种的译文。本软件提供的 lang 类型便针对上述场景，策划人员只需在 Excel 表格中填写 lang 文件中的 key，在导出时指定本次所使用的 lang 文件，本软件便可根据所填写的 key 在 lang 文件中查找到其对应的译文然后进行导出</p> <p>lang 文件必须是以 UTF-8 作为编码的文本文件，key 与 value 之间用英文冒号分隔，每行声明一个键值对。空行会被本软件忽略，若一行以#开头，本软件视为注释行，不进行读取。lang 文件路径通过运行参数设置</p> <p>如果某个字段在 lang 文件中的 key 名组成规则相同，都是由其他字段按相同规则值拼接而成，可以进行统一配置而不必在每个单元格中一一配置，比如道具表中的道具描述在 lang 文件中的 key 都是 propDesc 开头后面紧跟道具的 id 字段的值，则 lang 类型定义可填写为 lang(propDesc{id})，即所引用的字段名在花括号内声明，本软件自动拼接出 key 名然后查找对应的译文。注意所引用的字段必须在这个 lang 字段之前声明且不为 dict、array 集合类型子元素，否则无法找到</p>																

json	<p>json 字符串，需符合 json 格式要求。不建议使用此数据类型，并推荐使用上面所述的 dict 和 array 实现复杂嵌套结构的定义，因为 json 格式不直观，且需要手工重复声明键名，若由于手误写错根本无法检查出来</p>
tableString	<p>本软件特有的一种以字符串形式表达复杂嵌套 table 结构的数据类型。格式为 tableString[格式]，每组数据的 key 和 value 分为用 k、v 声明，中间用 分隔。特殊符号 #seq 表示自动按顺序编号，#true 表示自动填写 value 值为 true，#table 表示 value 值是一个 table，#后加数字 表示用填写的该组数据中的第几个表示，自定义的变量在值后面需要在()内标明数据类型，只支持 int、long、float、string、lang、bool 这几种最基础类型。填写的具体数据中，两个数据之间用英文分号隔开，一个数据中的不同变量用英文逗号隔开。注意整个输入的 tableString 字符串中不允许出现英文引号、斜杠，string 类型字符串中不允许出现英文逗号、分号，lang 类型数据所填的 key 必须在 lang 文件中能找到对应的字符串值，且字符串中不含有英文引号、斜杠、逗号、分号</p> <p>提供几个示例如下：</p> <p>(1)填写了某一场 PVE 战斗中的怪物组为 10001;10003;10021 填写格式为 tableString[k:#1(int) v:#true]</p> <p>最终生成的 lua table 为：</p> <pre>{</pre>

```
[10001] = true,  
[10003] = true,  
[10021] = true,  
}
```

如果格式写为 `tableString[k:#seq|v:#1(int)]`，最终则会生成：

```
{  
  [1] = 10001,  
  [2] = 10003,  
  [3] = 10021,  
}
```

推荐采用第 1 种生成方式，lua 语言可以通过直接索引获知填写的数据中有没有指定数据，而第 2 种生成的结果就只能 for 循环遍历后才能获知

(2) 填写了某场战斗后奖励物为 1,90001,500;2,90002,10 填写格式为

`tableString[k:#seq|v:#table(type=#1(int),id=#2(int),count=#3(int))]`

最终生成的 lua table 为：

	<pre>{ [1] = { type = 1, id = 90001, count = 500, }, [2] = }</pre>
mapString	<p>本软件特有的另一种以字符串形式表达复杂嵌套 table 结构的数据类型。它与 tableString 类型的区别是：tableString 类型对格式的要求比较严格，比如上面 tableString 实例中声明了一个奖励项用 type、id、count 三个参数表示，则在 Excel 表中填写数据必须严格按顺序声明且三个参数缺一不可。但项目开发过程中，经常会有在特定情况下，使用某一些参数，而在另外一些情况下，使用其他参数的需求。比如为新手引导步骤表格配置每一步骤的功能，当此步骤是显示对话时，需要填写对话内容 content、对话框所在位置 position 等，而当此步骤是要求玩家点击某个按钮时，则需要填写按钮的名称 buttonName 等。此情况下，用 mapString 类型可以灵活填写具体情况下需要配置的数据。以上面情况</p>

为例，声明数据类型为 `mapString[content=string,position=(x=float,y=float),buttonName=string]`，则当此步骤是显示对话时在 Excel 单元格中填写数据为 `content="xxx",position=(x=0,y=10)`，当此步骤是要求玩家点击某个按钮时，填写数据为 `buttonName="xxx"`。此情况下使用 `mapString` 类型而不是 `dict` 类型，可以避免在参数类型很多时，为每个参数都配置一列，导致表格过长，以及每一行数据需要填写的参数种类不多，产生大量空白单元格的问题

`mapString` 声明格式如上面实例所述，在英文小括号中声明所有可用的参数，**每个参数需要在=左右分别声明参数名和数据类型**（只支持 `int`、`long`、`float`、`string`、`lang`、`bool` 这几种最基础类型），**参数与参数之间用英文逗号分隔**，**`string`、`lang` 型数据必须用英文引号进行包裹**，`lang` 类型数据所填的 `key` 必须在 `lang` 文件中能找到对应的字符串值。**`mapString` 类型支持嵌套多层 `mapString` 型参数使用**，正如上面实例那样，参数 `position` 属于嵌套的 `mapString` 类型，后面直接在小括号内声明下属成员的参数名及类型即可

本软件在读取 Excel 表格数据时，会对 `mapString` 类型的数据进行检查，不允许出现未声明的参数名，并且填写参数对应的类型必须与声明类型相同

注意：在导出为 `lua` 或 `json` 文件时，`string` 类型以及 `lang` 类型 `key` 对应的 `value` 字符串中不对转义字符“\”进行处理，会原样输出。这样会使得 `lua` 或 `json` 在解析时可以对转义字符进行处理，比如在 Excel 单元格中按下 `Alt+Enter`

进行手工换行或者输入“\n”，最终转换并解析后就变成一个真实的换行。而如果想输出一个“\”，则必须在 Excel 单元格中输入“\\”。另外，本软件会将字符串中的“”强制变为“\”，以适应 lua 字符串中对英文引号的转义处理

本软件配置文件说明

在 XlsxToLua.exe 所在目录下的 **config.txt** 为本软件的配置文件，编码格式为 **UTF-8**，该文件配置部分功能所需的参数，如果您的项目不使用相应的功能，可以没有配置文件。配置文件以每行一个键值对的形式配置，键值对通过英文冒号分隔，空行和以#开头的注释行会被本软件忽略，不允许出现同名的 **key**

目前，以下参数为使用到某些功能的必配参数：

参数名	参数作用	在何种情况下必须配置
connectMySQLString	连接到 MySQL 数据库的连接字符串，例如： server=127.0.0.1;port=3306;uid=root;password=root; database=mydb;Charset=utf8;	若使用导出到 MySQL 数据库功能，则必须配置此参数
createDatabaseTableExtraParam	导出到 MySQL 数据库进行建表时额外添加的参数字符串，比如可以通过设置使用的编码格式，避免插入的中文变成乱码。例如：ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin	根据项目需要自行决定是否配置此参数，不强制要求配置
defaultDateInputFormat	未声明 date 型的输入格式时所采用的默认格式	当声明某个 date 或 time 型

defaultDateToLuaFormat	未声明 date 型导出至 lua 文件的格式时所采用的默认格式	字段，但未指定输入格式、导出至 lua 文件的格式或者导出至 MySQL 数据库的格式时，要保证对应默认配置必须存在。如果不存在未指定格式的情况，这些配置可以不存在
defaultDateToDatabaseFormat	未声明 date 型导出至 MySQL 数据库的格式时所采用的默认格式	
defaultTimeInputFormat	未声明 time 型的输入格式时所采用的默认格式	
defaultTimeToLuaFormat	未声明 time 型导出至 lua 文件的格式时所采用的默认格式	
defaultTimeToDatabaseFormat	未声明 time 型导出至 MySQL 数据库的格式时所采用的默认格式	

除此之外，**config.txt** 配置文件中还可以声明一系列自定义的键值对，用来定义自己项目中被多处使用的相同的表格检查规则或者一些常量配置，详细请参看[在配置文件中声明检查规则](#)

表格格式要求

- 1、 每张 Excel 表格都必须含有一个名为“data”的 Sheet 表，在此 Sheet 中存放表格数据
- 2、 每张 Excel 表格都可以在一个名为“config”的 Sheet 表中定义此表配置（整表检查、导出规则等）
- 3、 其余 Sheet 表不会被本软件识别读取，可用作备注、历史版本备份等自定义用途

data 表的格式要求

- 1、 表格前五行必须依次声明如下图所示的定义内容，从第六行开始声明具体数据

	A	B	C	D	E
1	英雄ID	英雄名称	英雄名称	稀有度（11-13）	英雄职业（1：法师，2：战士，3：牧师，4：勇士）
2	heroId	name		rare	type
3	int	lang		int	int
4		notEmpty		\$heroRare	\$heroType
5	heroId(INT)	name (VARCHAR(20))		rare (INT)	type (INT)
6	1	heroName1	英雄法师	11	1
7	2	heroName2	英雄战士	11	2
8	3	heroName3	英雄牧师	11	3

各行的作用分别为：

(1)字段描述，仅用于方便填表者备注，可不填。**注意：整张表格第一行的字段描述不要都空着**，微软 OleDb 方式读取 Excel 文件时，表格前几行的空行会被忽略，从而导致本软件根据行号依次读取前五行配置行时出现错行

(2)字段名，会被作为导出的 lua table 中的 key 生成，如果某列不填写该字段名，且不填写第五行中导出数据库配置，则视该列为无效列（如上图所示的 C 列），该列数据不会被读取、生成。**注意：字段名必须以英文字母开头，且只允许使用英文字母、数字和下划线**

(3)字段数据类型，填写本软件支持的[数据类型](#)，凡是声明了字段名的有效字段，都必须声明其数据类型

(4)字段检查规则，填写本软件支持的[字段检查规则](#)，如果该列无需进行检查则留空

(5)导出到 MySQL 数据库中的字段名及对应 MySQL 的数据类型，声明格式形如 propName(VARCHAR(20))，即前面是字段名后面在英文括号中声明数据类型。**注意：1、数据库字段名不允许同名。2、导出到 MySQL 数据库的为该 Excel 表中所有有效字段。array 或 dict 类型的父字段上不需要进行导出至 MySQL 数据库的声明，只需逐个在子元素字段声明即可 3、对于 lang 型字段，导出到 MySQL 中的数据为 lang 文件中对应的 value 值。4、若不进行此配置，该列数据在导出到数据库时将被忽略**

- 2、**表格首列为主键列，只能为 int、long 或 string 型**，如果为 string 型主键，每行填写的值必须以英文字母开头，只能由英文字母、数字、下划线组成且不能为空或纯空格。另外，**每行所填的值不允许重复**

提示：若客户端和服务端共用同一张表格，且表格中含有部分仅一方所需要的数据列，可用以下方式实现按配置进行特定导出。如果某列仅**客户端**（会导出 lua、csv、csv 对应 C#类或 Java 类文件、json 形式）需要，则只需填写第二行配置的字段名而不填写第五行配置的导出数据库信息。反之如果某列仅**服务器端**（会导出到 MySQL 数据库）需要，则只需填写第五行导出数据库信息，而留空第二行字段名。如果某列这两个配置均留空，则视此列为无效列，不进行数据读取等任何操作。但**注意：**dict 下属的子元素中如果某列没有指定字段名但设置了数据库导出信息，也会被直接视为无效列不做数据库导出操作

config 表的格式要求

每列配置一个参数名及配置，列的首行声明参数名，之后的行用于填写配置值

	A	B
1	exportDatabaseTableName	addKeyToLuaTable
2	hero	true
3		
4		
	data	config
	Sheet3	

目前本软件支持的配置参数如下表所示：

参数名	作用	填写方法
addKeyToLuaTable	用于配置将表格导出为 lua table 时， 是否 将主键列的值也作为 table 中的元素	若要将主键列的值作为 table 中的元素， 填写值为 true，否则为 false。如果 不声 明此参数，默认为 false
exportDatabaseTableName	用于配置当需要将表格 导出到 MySQL 数 据库时的表名	在第二行中填写符合 MySQL 数据库要求的 表名即可。 注意：如果不填写就不将 此表导出到 MySQL 数据库

exportDatabaseTableComment	用于配置将表格导出到 MySQL 数据库时为表格填写的 Comment	在第二行中填写自定义的 Comment 即可。如果不声明此参数，默认为空
exportDatabaseWriteNull ForEmptyString	用于配置将表格导出到 MySQL 数据库时 string 型字段中的空白单元格导出为数据库中的 NULL 而不是空字符串	若要导出到数据库中为 NULL，填写值为 true，否则为 false。如果不声明此参数，默认为 false
tableCheckRule	用于配置某张 Excel 表的整表检查规则	参看 整表检查 功能说明，在下面的单元格中依次声明要对这张表格执行的自定义整表检查函数名，一个单元格中填写一个，总个数不限
tableExportConfig	用于配置某张 Excel 表按索引嵌套方式导出 lua 文件的规则	参看 按索引嵌套方式导出 lua 文件 的功能说明
exportCsvClassName	用于配置某张 Excel 表导出 csv 对应的 C# 类和 Java 类文件时的类名	请注意遵循 C# 类及 Java 类命名规范，采用以大写英文开头的驼峰式命名。如果不声明此参数，将自动将表格名转为类型，若发现表名中含有下划线，将认为

		<p>是以下划线分隔单词，否则认为通过单词首字母大写区分。若想让本软件自动生成类型，但想在类名中添加前缀或者后缀，请使用 -autoNameCsvClassParam 软件运行参数 中进行设置</p>
--	--	--

软件运行参数

本软件在 Windows 系统下通过命令行方式传入运行参数打开执行，软件运行参数分为必填参数和可选参数两种。参数与参数之间需按照 Windows 系统要求用一个空格进行分隔。注意：因为参数本身中可能含有空格，这会被 Windows 系统误认为是参数的分隔，从而造成一个参数被截断从而引发错误。故强烈建议将每个参数都用英文引号包裹后再用空格分隔参数，Windows 系统不会把引号中的空格当做参数分隔符

另外，手工配置运行参数繁琐且容易出错，且非程序开发人员有一定的学习成本，强烈建议使用本软件的配套 [GUI 辅助工具 XlsxToLuaGUI](#) 进行运行参数的可视化配置

必填参数

运行本软件，必须依次传入以下必填参数，注意必须按下面的顺序进行声明

顺序	参数作用	填写方法
1	Excel 表格的所在路径	Excel 表格所在文件夹的路径（支持绝对路径与相对路径，下同），文件夹必须存在
2	导出的 lua 文件的存放路径	文件夹必须存在

3	项目 Client 目录的路径	<p>此参数声明项目根目录路径，配合文件存在性检查规则使用，执行 file 检查规则时，会以此路径为根目录，到 file 检查规则声明的相对于根目录的具体路径下查找指定文件是否存在</p> <p>如果项目中不含 file 检查规则，则无需指定项目根目录，但此参数不允许为空，必须填写为-noClient</p>
4	lang 文件路径	<p>对于用到 lang 数据类型的项目，需要通过此参数指定 lang 文件路径，如果项目中不含 lang 类型，则无需指定 lang 文件路径，但此参数不允许为空，必须填写为-noLang</p>

可选参数

可选参数为在有以下需求时可声明的配置参数，可选参数可以不声明（部分参数本软件会按默认值处理），参数的声明顺序随意

参数名	作用或填写方法
-exportIncludeSub folder	<p>若声明此参数，则将要导出的 Excel 文件夹下的各级子文件夹中的 Excel 文件也进行导出，不声明则仅导出选定文件夹直接下属的 Excel 文件。注意：即便不在同一文件夹下，要导出</p>

	的 Excel 文件名也不允许重复
-exportKeepDirectoryStructure	若声明了-exportIncludeSubfolder 参数，同时希望进行各种导出时，将生成的文件按原 Excel 文件所在的目录结构进行存储，则声明此参数，不声明则会将生成的文件均存放在同级目录下
-exportMySQL	若声明此参数，表明使用导出至 MySQL 数据库的功能，不声明则不使用
-columnInfo	若声明此参数，在生成的 lua 文件上方会以注释形式显示列信息，不声明此参数则默认不会生成
-unchecked	若声明此参数，则本软件在各种导出操作前，不对表格按设置的检查规则进行查错，不推荐使用此参数，这将导致表格配置错误但无法被发现
-printEmptyStringWhenLangNotMatching	当 lang 型数据 key 在 lang 文件中找不到对应值时，本软件在导出的 lua 文件中默认输出值为 nil，即 xx = nil，如果使用此参数，则输出为空字符串，即 xx = ""
-part	如果本次只想对 Excel 所在目录下的部分表格执行导出操作，则可以声明此参数。声明格式为-part 后在英文小括号内声明本次要导出的 Excel 文件名，用 分隔，如“-part(Hero System)”。不声明此参数，则默认情况下指定的表格目录下所有 Excel 文件都将被导出

	<p>注意：只有会被导出的表格才执行表格检查操作，但即便指定只导出部分 Excel 文件，所有 Excel 文件仍旧会被读取，因为要导出的 Excel 文件中若存在跨表的检查规则可能会涉及到本次不导出的表格</p>
-except	<p>如果本次要忽略对 Excel 所在目录下的部分表格执行导出操作，则可声明此参数。声明格式同上，但不允许对某张表格既设置-part 又设置-except</p>
-allowedNullNumber	<p>本软件默认情况下，不允许 int、long、float 这几种数值型字段中单元格为空。如果声明此参数则允许为空，不推荐使用此参数</p>
<p>声明要将指定的 Excel 表额外导出为 csv 文件同时需要以下 2 个参数：</p> <p>-exportCsv</p> <p>-exportCsvParam</p>	<p>-exportCsv 参数声明要将哪些 Excel 表额外导出为 csv 文件，填写格式同上面所述的-part 参数格式，需在后面的小括号内填写用 分隔的 Excel 文件名。若要将本次要导出 lua 的表格全部进行 csv 格式导出，则在小括号内填写\$all</p> <p>-exportCsvParam 参数声明导出 csv 文件的保存路径 exportPath、扩展名 extension、字段分隔符 splitString、是否在首行列举字段名称 isExportColumnName、是否在其后列举字段数据类型 isExportColumnDataType 等导出参数</p> <p>参数名和配置值之间用 = 分隔，参数之间用 分隔。例如 -exportCsvParam(exportPath=C:\csv extension=csv splitString=, isExportColumnName=false is</p>

	<p>ExportColumnDataType=false), 其中 exportPath 参数必须声明, extension 参数不声明则默认为 csv, splitString 参数不声明则默认为英文逗号, 注意若想指定分隔符为 “ ” 则需要赋值为 “\ ”, isExportColumnName、isExportColumnDataType 参数不声明则默认为 true</p> <p>注意: 如果未指定本次要导出某个 Excel 表, 那么即便声明要导出 csv 文件也不会生效, 这种情况下, 本软件会进行警告提示</p>
<p>声明要将指定的 Excel 表额外导出 csv 对应的 C#类同时需要以下 2 个参数:</p> <p>-exportCsClass</p> <p>-exportCsClassParam</p>	<p>-exportCsvClass 参数声明要将哪些 Excel 表额外导出为 csv 对应的 C#文件, 填写格式同上, 需在后面的小括号内填写用 分隔的 Excel 文件名。若要将本次要导出 lua 的表格全部进行 C#类的导出, 则在小括号内填写\$all</p> <p>-exportCsClassParam 参数声明导出 csv 对应 C#类的保存路径 exportPath、命名空间 namespace、引用类库 using 等导出参数</p> <p>参数名和配置值之间用 = 分隔, 参数之间用 分隔。例如 -exportCsClassParam (exportPath=C:\csv\csClass namespace=Assets.Scripts.CsClass using=System,System.Collections.Generic), 其中 exportPath 参数必须声明, namespace 配置生成的类文件中的命名空间, 若不声明此参数则不设置命名空间, using 配置引用的类库, 不同类库之间用英文逗号分隔, 若不声明此参数则不设置类库引用, 注意 array、dict 集合类型会使用 C#的 List 和</p>

	<p>Dictionary 类型,对含有此类型的 C#类文件,本软件会强制添加对 System.Collections.Generic 的引用</p> <p>每个生成的 C#类文件的类名可在 Excel 表的 config 表中用 exportCsvClassName 参数进行手工指定,若不指定,本软件将按规则以 Excel 表名作为类名进行自动生成,但若想为所有类名添加前缀和后缀,则通过-autoNameCsvClassParam 运行参数设置</p> <p>注意: 本功能与导出 csv 文件功能是独立的,可以仅使用本功能,而不进行 csv 文件的导出</p>
<p>声明要将指定的 Excel 表额外导出 csv 对应的 Java 类同时需要以下 2 个参数:</p> <p>-exportJavaClass</p> <p>-exportJavaClassParam</p>	<p>需在后面的小括号内填写用 分隔的 Excel 文件名。若要将本次要导出 lua 的表格全部进行 Java 类的导出,则在小括号内填写\$all</p> <p>-exportJavaClassParam 参数声明导出 csv 对应 Java 类的保存路径 exportPath、包名 package、引用类库 import、时间型转为 Date 而不是 Calendar 型的 isUseDate、是否为 Java 类生成无参构造函数的 isGenerateConstructorWithoutFields、是否为 Java 类生成含全部参数构造函数的 isGenerateConstructorWithAllFields 等导出参数</p> <p>参数名和配置值之间用=分隔,参数之间用 分隔。例如 -exportJavaClassParam (exportPath=C:\csv\javaClass package=org.xlsxtolua.pojo import=java.util.ArrayList,java.util.HashMap isUseDate=true isGenerateConstructorWithoutFields=true isGenerateConstructorWith</p>

	<p>AllFields=true), 其中 exportPath、package 参数必须声明, import 配置引用的类库, 不同类库之间用英文逗号分隔, 若不声明此参数则不设置类库引用, 注意 array、dict 集合类型会使用 Java 的 java.util.ArrayList 和 java.util.HashMap 类型, 时间型会使用 java.util.Date 或 java.util.Calendar, 对含有此类型的 Java 类文件, 本软件会强制添加相应的引用。 isUseDate 配置生成的 Java 类中若有时间型, 是否转为 Date 型而不是 Calendar 型, 设为 true 则转为 Date 型, 否则为 Calendar 型, 若不进行设置, 默认为 true。</p> <p>isGenerateConstructorWithoutFields、isGenerateConstructorWithAllFields 参数配置是否为生成的 Java 类自动生成无参以及含全部参数的构造函数, 若不进行设置, 均默认为 false</p> <p>每个生成的 Java 类文件的类名设置同上面生成 C#类文件, 可手工指定或自动命名, 注意此规则同时对导出 C#类和 Java 类文件生效</p> <p>注意: 本功能与导出 csv 文件功能是独立的, 可以仅使用本功能, 而不进行 csv 文件的导出</p>
-autoNameCsvClassParam	<p>如果要将 Excel 表额外导出 csv 对应的 C#类或 Java 类, 并且希望在本软件根据 Excel 名自动为所有生成的类名统一添加类型前缀或后缀, 则可声明此参数。-autoNameCsvClassParam 参数可在括号中声明 classNamePrefix、classNamePostfix 两个子参数进行配置, 可以只设置前缀或只设置后缀, 参数之间用 分隔, 如 -</p>

	<p>autoNameCsvClassParam(classNamePrefix=TableData classNamePostfix=Pojo), 但注意如果某张 Excel 表已经在 config 表配置中单独指定了类名, 则不会对其增加前缀或后缀</p>
<p>声明要将指定的 Excel 表额外导出为 json 文件同时需要以下 2 个参数:</p> <p>-exportJson</p> <p>-exportJsonParam</p>	<p>-exportJson 参数声明要将哪些 Excel 表额外导出为 json 文件, 填写格式同上面所述的-part 参数格式, 需在后面的小括号内填写用 分隔的 Excel 文件名。若要将本次要导出 lua 的表格全部进行 json 格式导出, 则在小括号内填写\$all</p> <p>-exportJsonParam 参数声明导出 json 文件的保存路径 exportPath、扩展名 extension、是否带缩进格式 isFormat、是否生成 json array 形式 isExportJsonArrayFormat 等导出参数</p> <p>参数名和配置值之间用 = 分隔, 参数之间用 分隔。例如 -exportJsonParam(exportPath=C:\json extension=txt isFormat=true isExportJsonArrayFormat=false), 其中 exportPath 参数必须声明, extension 参数不声明则默认为 txt, isFormat 为 true 则将生成的 json 字符串整理为带缩进格式的形式, 否则 json 字符串中不含有空白字符, 不声明此参数则默认为 false。isExportJsonArrayFormat 参数配置选用生成 json 的两种不同形式, 如果设为 true 则生成为各行数据对应的 json object 包含在一个 json array 的形式, 否则生成为各行数据以主键列值为 key, 各字段信息为 value, 包含在一个 json object 的形式, 不声明此参数则默认值为 true。若选用后者的生成方式, 还可以通过</p>

	<p>isMapIncludeKeyColumnValue 参数设置每行字段信息对应的 json object 中是否还包含主键列对应的键值对，不声明此参数则默认值为 true</p> <p>注意：如果未指定本次要导出某个 Excel 表，那么即便声明要导出 json 文件也不会生效，这种情况下，本软件会进行警告提示</p>
--	---

表格检查功能

表格检查功能可根据您的设置，对某个字段或是整张表格设定检查规则，在进行导出转换前执行检查，以便及时发现填写错误。虽然可以声明[-unchecked 运行参数](#)来忽略对表格的检查，但不推荐采用这种会带来极高风险的做法

字段检查

字段检查仅针对 Excel 表中某个字段，本软件提供众多常用的检查方法满足实际项目中最普遍的检查需求，同时提供自定义检查规则的功能，以便您根据项目实际需要，编写个性化的检查函数

目前本软件提供以下这些通用的检查规则，注意声明格式中的括号、冒号均为英文字符：

作用	适用数据类型	填写方法及注意事项
值范围检查，确保填写的数字、时间在合法范围内，或者字符串的长度符合要求	int、long、float date、time string、lang	对于数值类型的 int、long、float 字段，比如填写形如[2,9)的规则声明，则要求输入的数字必须大于等于 2 且小于 9。其中上下限用英文逗号分隔，[]表示闭区间，()表示开区间，可以组合使用，比如(1,5]表示一个大于 1 小于或等于 5 的范围。如果不限制上下限范围，可以用*代替，比

		<p>如(*,5]或者[* ,5]表示一个小于等于 5 的数字，同理[5,*]或者[5,*)表示一个大于等于 5 的数字</p> <p>对于时间型的 date、time 字段，括号使用方法相同。但注意时间型字段值范围检查的上下限填写必须遵循 date 型格式为 yyyy-MM-dd HH:mm:ss，time 型为 HH:mm:ss。比如限定某个字段所填写的时间必须介于 2016 年 10 月到年末最后一天，则填写的规则为[2016-10-01 00:00:00,2016-12-31 23:59:59]</p> <p>对于 string 型字段，会检查字符串长度是否在设定的范围之内。对于 lang 型会检查对应 lang 文件中的 value 字符串的长度</p>
<p>有效性检查，确保填写的值必须为指定合法取值中的一个</p>	<p>int、long、float</p> <p>string</p> <p>date、time</p>	<p>除 string 型之外，声明格式均为{1,4,6}，即在英文花括号中用逗号分隔各个允许的有效值，其中时间型的 date、time 的填写格式，仍旧需要遵循 yyyy-MM-dd HH:mm:ss 和 HH:mm:ss</p> <p>string 型特殊，因为考虑到想设置的有效值字符串本身可能含有英文逗号，故提供声明 string 型有效性检查规则时自定义有效值分隔符的功能，填写格式形如{a,b;c;d};(;), 则代表允许的三个有效值分别为 “a,b”</p>

		“c” 和 “d”。其中需要在右花括号之后的小括号内声明各个有效值的分隔符，也可不进行分隔符声明，这会默认分隔符仍旧是英文逗号
非法值检查，确保填写的值不允许为指定非法取值集合中的一个	同上	非法值检查与有效性检查是正好对立的检查规则，规则声明格式与上面除了需要在左花括号前加英文叹号之外完全相同，即! <code>{1,4,6}</code>
值非空检查，确保填写了内容	int、long、float string、lang date、time json、tableString、 mapString	只需声明 <code>notEmpty</code> 。但注意如果是 <code>string</code> 类型，声明 <code>notEmpty</code> 仅要求字符串不能为空但允许为连续空格字符串，如果也不允许为连续空格字符串，需要声明为 <code>notEmpty[trim]</code> 。如果是 <code>lang</code> 类型，填写 <code>notEmpty[key]</code> 只检查是否填写了 <code>key</code> 值，填写 <code>notEmpty[value]</code> 只检查填写的 <code>key</code> 在相应的 <code>lang</code> 文件中能找到对应的 <code>value</code> ，填写 <code>notEmpty[key value]</code> 则既要求所有数据都填写 <code>lang</code> 文件中的 <code>key</code> ，也要求对应的 <code>value</code> 在 <code>lang</code> 文件中都能找到，如果填写 <code>notEmpty</code> 则与 <code>notEmpty[key value]</code> 相同。如果是数值型的 <code>int</code> 、 <code>long</code> 或 <code>float</code> 字段，并设置了此检查规则，即便声明了 -allowedNullNumber 运行参数 ，仍会进

		行检查
唯一性检查，确保该字段下的所有值不允许出现重复	int、long、float string、lang date、time	只需声明 unique 。但注意 string 型、 lang 型如果填写或者找到的 value 为空字符串，允许出现多次为空的情况。 lang 型默认只检查 key 不能重复，如果还想检查 value 需要声明为 unique[value]
值大小比较检查，确保一张表格中每一行的该字段值必须大于等于或大于另一字段的值	int、long、float date、time	<p>声明格式为>=或>后面跟同一表格中另一字段名，比如某张配置限时活动的表格中有活动开始时间 startTime 和结束时间 endTime 两个字段，要检查结束时间必须晚于开始时间，则在 endTime 字段上声明“>startTime”</p> <p>注意：1、对于要进行比较两字段，如果都属于数值类型（int、long 或 float），可以不管其具体的类型而进行比较，非数值类型的两列则必须数据类型完全相同才允许比较</p> <p>2、如果与之比较的字段为 array 或 dict 的下属子元素，需要按如下方式填写其字段名。若字段为某个 array 型（字段名为 openRankList）下属的第 2 个子元素，则需要填写为 openRankList[2]。若字段为某个 dict 型（字段名为 reward）下属的 count 字段，则需要填写为 reward.count</p>

		<p>3、虽然只提供了大于的检查，但所有小于的逻辑检查都可以转换为进行大于判断检查</p>
<p>值引用检查，确保该字段的所有取值存在于另外某个表格中的指定列中</p>	<p>int、long、float string</p>	<p>声明格式为 ref: 表格名-字段名，即在 ref 和英文冒号后声明所引用哪张表格下属的哪个字段。比如，在一张关卡表中要配置通关后的奖励道具，则需要确保填写的道具 id 必须在另一张道具表（表名为 Prop）的主键列（字段名为 id）中定义过，检查规则填写为 ref:Prop-id。另外，如果像这种引用另一张表格主键列的情况，可以不显式声明字段名，即只需填写 ref:Prop 即可</p> <p>注意：1、所引用的字段必须与要检查的字段数据类型相同。2、如果要检查的列为 string 类型，会忽略对空字符串的引用检查。3、如果要引用的字段为 array 或 dict 型的子元素，字段名的声明参看上面唯一性检查声明中的格式。4、此功能支持排除对某些特殊值的引用检查，比如填写关卡胜利后奖励道具 id，约定如果填-1 表示不奖励道具，则进行引用检查时，应忽略对-1 的检查。这种情况下，只需在 ref 检查规则后面追加(except{-1})，即 export 后用同有效值声明的方式去声明排除检查</p>

		<p>的值，注意 <code>string</code> 型字段仍旧可以声明分隔符，即 <code>ref:Prop-propName(except{经验药水;体力丹};)</code></p>
<p>文件存在性检查，确保填写的路径字符串对应的文件真实存在</p>	<p><code>string</code></p>	<p>项目中往往会在表格中配置资源文件的路径，比如在英雄配置表中，可能会配置每个英雄登场时播放的口头禅音频存放路径，配置英雄二维头像图片存放路径等，这个时候使用文件存在性检查，可确保文件路径填写正确，能够找到对应的资源文件</p> <p>声明格式为 <code>file(扩展名):路径</code>，形如 <code>file(png):\Assert\Resources\Texture\</code>。其中路径为相对于项目 <code>Client</code> 目录的路径（需配置项目 Client 目录运行参数）。如果 <code>file</code> 后在小括号中声明一个扩展名，则该字段下所有填写的路径字符串中无需带有扩展名，而使用这个统一的扩展名。如果 <code>file</code> 后未声明，则每个单元格中都需要显式带有扩展名。注意：空单元格会忽略进行存在性检查</p>
<p><code>mapString</code> 型选用参数检查，确保当逻辑上某列对应不同的类</p>	<p><code>mapString</code></p>	<p>比如新手引导配置表格中，通过 <code>int</code> 型的 <code>type</code> 列声明该某步骤的类型，类型 1 为显示对话框，类型 2 为要求玩家点击某个按钮。通过 <code>mapString</code> 型的 <code>params</code> 列声明具体的显示参数，格式定义为</p>

<p>型取值时，mapString 型数据按要求含有或不允许有某些元素</p>		<p>mapString[content=string,position=(x=float,y=float),buttonName=string]。</p> <p>此时，需要确保当 type 列填写值为 1 时，对应 params 列的 mapString 数据中必须含有 content 和 position 元素，而不能含有 buttonName，反之若是类型 2 则相反</p> <p>声明格式为 mapString:if(type=1)[content=1,position=1,buttonName=0] if(type=2)[content=0,position=0,buttonName=1]</p> <p>要求先声明“mapString:”，然后在 if 后的括号内声明其他列的数据满足什么情况下，检验此情况下的 mapString 型数据是否符合要求。需要用 = 分隔字段与值，仅支持对 int、long、float、bool、string 型进行值判断，其中对于数值型的 int、long、float 还支持 >、>=、<、<= 的判断。如果有多个条件要求，用英文逗号进行分隔，如 if(id>100,type=1)。如要求在任何情况下 mapString 型都要满足参数规则，则 if 后的括号中填写 all，比如必须存在 content 字段，则声明为 if(all)[content=1]。注意如果条件字段中某行对应的值为空，视为不满足此条件，但进行警告。被检查的 mapString 型字段中某行对应的值为空，会跳过检查，但进行警告</p>
---	--	---

		<p>在 if 条件声明后的英文方括号内声明 mapString 数据必须含有或不允许有哪些字段，用=分隔子参数名和配置，=1 是要求必须有，=0 则不允许有，未配置的子参数则可有可无。上面例子中的 position 声明为=1，则要求也必须声明 position 子参数下属的子参数 x、y，如果只要求必须填写 x 值，则声明为 position=(x=1)，如果还要求不允许填写 y，则声明为 position=(x=1,y=0)</p> <p>不同条件下对 mapString 数据的不同要求用 进行分隔</p>
--	--	---

注意：对于表格的主键列，无论是否填写检查规则，都会强制执行 **unique** 检查

此外，本软件支持自定义编写检查函数功能，请参看[自定义字段检查函数的编写](#)。完成相应函数的编写后，在 Excel 表的对应字段下声明检查规则格式为在 **func:**后加自己编写的检查函数的名字

注意：1、可以为同一字段指定任意多个由以上通用规则或自定义的检查规则所组合而成的检查规则。多个检查规则之间用**&&**进行分隔，比如要检查英雄表中每一行都必须填写英雄二维头像图片路径并且能在项目指定路径下找到资源文件，就需要组合使用值非空检查规则 and 文件存在性检查规则，声明为 **notEmpty && file(png):\Assert\Resources\Texture**。2、请自行保证同一字段上设置的多个检查规则不会互相冲突（比如[1,5] && {8,11}

这样本身就不可能同时满足的组合规则)，本软件不进行组合规则是否合理的判定

实际项目中，相同的检查规则可能被多次使用，比如凡是填写奖励道具时，都需要检查所填的道具 id 是否在道具表中定义过，再比如很多功能对玩家等级有限制，需要检查填写的功能开放等级数值是否在合法取值范围内（如果游戏中玩家等级为 1-60 级，那么填写开放等级为 500 级显然是笔误）。类似这些检查规则不仅可能被多次使用，而且还会进行修改（比如新版本中增加了玩家的养成内容，玩家等级从 60 级扩展到 80 级），如果这样的检查规则分散在各相关表格中配置，不仅容易配错，如果需要修改，还需手工把所有相关表格一一更新，很容易遗漏表格。因此本软件支持将可能会多次使用、可能日后会进行修改的检查规则声明到 **config.txt 配置文件** 中，即定义一条这样的**键值对，键名以\$开头，后面跟自己为该检查规则起的名字，键值则为检查规则**，比如用于检查所填作为奖励用的道具的 id 是否正确的规则可定义为\$checkPropId:ref:Prop-id(except{-1}) && notEmpty。之后，所有需要此检查的字段填写上\$checkPropId 即可

整表检查

整表检查针对一张表格中逻辑相关字段、行，提供编写自定义检查函数的功能，详细请参看[自定义整表检查规则的编写](#)。完成相应函数的编写后，在需要使用此整表检查函数的 Excel 表格的 config 表中配置 [tableCheckRule 参数](#) 即可

导出 lua 文件功能

将 Excel 表导出为 lua table 的形式供主要使用 lua 作为开发语言的项目直接读取使用配置数据，是本软件的核心功能之一。生成的 lua 文件中，都 return 一个由表格配置内容转换成的 lua table，您可根据项目需要，自行决定读取使用该文件的方式，比如可以读取文件为字符串，然后用 lua 提供的 dostring 方法解析 lua 文件。导出 lua 文件的统一存放路径通过[运行参数](#)进行设置

常规方式导出

常规导出方式，将每一行的数据作为最外层 table 中的一个键值对，其中主键列的值作为键名，其余字段的信息作为键值 table 中的各个子元素。如下图所示，为系统模块配置表按常规方式导出的 lua table，其中会通过缩进形式，更加清晰地展示字段嵌套层次

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	系统ID	系统名称	系统帮助信息	开启条件（同时满足三个条件）	所需玩家等级（不限填-1）	所需Vip等级（不限填-1）	所需通关关卡（不限填-1）	开启系统后赠送物	奖励物1	类型	id	数量	奖励物2	类型	id	数量
1																
2	systemId	systemName	help	openCondition	rankLimit	vipLimit	levelLimit	openRewards		type	id	count		type	id	count
3	int	lang	lang	dict[3]	int	int	int	array[dict[3]:2]	dict[3]	int	int	int	dict[3]	int	int	int
4		notEmpty	notEmpty					func:CheckRewardListField								
5	systemId (INT)	systemName (VARCHAR(20))	help (VARCHAR(100))		openConditionRankLimit (INT)	openConditionVipLimit (INT)	openConditionLevelLimit (INT)			rewardType1 (INT)	rewardId1 (INT)	rewardCount1 (INT)		rewardType2 (INT)	rewardId2 (INT)	rewardCount2 (INT)
6	1	sysNormalCombat	sysNormalCombatHelp	-1				-1								
7	2	sysEliteCombat	sysEliteCombatHelp	-1				-1								
8	3	sysArena	sysArenaHelp		25	-1	-1			1	100001	10	-1			
9	4	sysShop	sysShopHelp		15	-1	-1			1	100001	5	-1			

```
return {
  [1] = {
    systemName = "普通关卡",
    help = "通过普通关卡可以获得一定几率掉落的道具",
    openCondition = nil,
    openRewards = nil,
  },
  [2] = {
    systemName = "精英关卡",
    help = "通过精英关卡可以获得一定几率掉落的稀有道具",
    openCondition = nil,
    openRewards = nil,
  },
  [3] = {
    systemName = "竞技场",
    help = "这里是各位英雄彼此切磋较量的场所, 有丰富的奖励",
    openCondition = {
      rankLimit = 25,
      vipLimit = -1,
      levellimit = -1,
    },
    openRewards = {
      [1] = {
        type = 1,
        id = 100001,
        count = 10,
      },
      [2] = nil,
    },
  },
  [4] = {
    systemName = "商店"
```

按索引嵌套方式导出

英雄装备表以及按常规方式导出的 lua table 如下图所示

	A	B	C	D	E	F
	ID	英雄ID	英雄品阶	装备槽位序号	装备ID	穿戴等级限制
1						
2	id	heroId	heroQuality	seq	propId	equipRank
3	int	int	int	int	int	int
4		ref:Hero	\$heroQuality	\$heroEquipmentSeq	ref:Prop && (200000, 300000)	\$heroRank
5	id(INT)	heroId(INT)	heroQuality(INT)	seq(INT)	propId(INT)	equipRank(INT)
6	1	1	10	1	200001	1
7	2	1	10	2	200002	1
8	3	1	10	3	200003	1
9	4	1	10	4	200004	1
10	5	1	20	1	200005	5
11	6	1	20	2	200006	5
12	7	1	20	3	200007	5
13	8	1	20	4	200008	5
14	9	1	21	1	200001	10
15	10	1	21	2	200002	10

```
return {  
  [1] = {  
    heroId = 1,  
    heroQuality = 10,  
    seq = 1,  
    propId = 200001,  
    equipRank = 1,  
  },  
  [2] = {  
    heroId = 1,  
    heroQuality = 10,  
    seq = 2,  
    propId = 200002,  
    equipRank = 1,  
  },  
  [3] = {  
    heroId = 1,  
    heroQuality = 10,  
    seq = 3,  
    propId = 200003,  
    equipRank = 1,  
  },  
}
```

但通常情况下，游戏中的实际需求会是通过英雄 id 和品阶查找其所能穿戴的四件装备的信息，如果直接从这样的 lua table 中获取信息，每次都要遍历所有数据直到找到符合要求的一项，效率低下。虽然程序开发人员可以在游戏运行时将上面的 lua table 整理为依次按英雄 id、品阶、装备顺序编号层层索引的嵌套结构，以便快速找到指定的

信息，但会给开发人员带来额外工作量

索引嵌套导出方式是针对类似上面这种情况提供的特殊导出方式，导出的 lua table 如下图所示：

```
return {  
  [1] = {  
    [10] = {  
      [1] = {  
        propId = 200001,  
        equipRank = 1,  
      },  
      [2] = {  
        propId = 200002,  
        equipRank = 1,  
      },  
      [3] = {  
        propId = 200003,  
        equipRank = 1,  
      },  
    },  
  },  
}
```

声明按索引嵌套方式导出 lua 文件，需在相应 Excel 表的 config 表中声明 [tableExportConfig 参数](#)，上表中为 HeroEquipmentDict:herold-heroQuality-seq{propId,equipRank}。其中英文冒号前声明按这种规则导出的 lua 文件名，冒号之后声明索引层次，将一层的索引的字段用英文连字符连接，注意作为索引的字段只能为 int、long、float、string、lang 型的独立字段，且不允许出现空值，最后在花括号中声明索引出来的 table value 中包含哪些字段的信息，用英文逗号分隔各个字段名，如果没有在花括号内声明，则默认将索引字段之外的所有字段进行填充

注意：1、不允许出现在各个索引字段值都相同的两行数据，本软件发现这种情况会进行报错提示。2、默认情况下，即便声明了按索引嵌套方式进行导出，Excel 表也会按常规方式被导出为 lua 文件，如果只想对某张表格进行索引嵌套方式进行导出而不进行常规方式导出，可在 tableExportConfig 参数下赋值-notExportOriginalTable。3、可以

对一张 Excel 表设置若干种按索引嵌套方式导出的规则。但请自行确保各种导出规则声明的导出 lua 文件的文件名不同，否则同名的会被覆盖只留下按最后一个导出规则生成的 lua 文件。如果未声明不按常规方式导出，并且存在索引嵌套方式导出规则中声明的导出 lua 文件的文件名与表格名相同，只会保留按常规方式导出的文件

以上为导出的设置，但像上面所述的英雄装备表那样，往往需要进行逻辑上的检查，比如上表需要确保凡是在本表中配置过的英雄，都必须具有在所有品阶下的四件装备的信息，不允许缺少在任何品阶下任何一键装备的信息，也不允许出现重复配置或者多配置的情况。虽然用[整表检查](#)规则自行编写检查函数可以进行检查，但这样会给程序开发人员带来额外工作量。本软件可以在索引嵌套导出规则中对用作索引的字段设置并进行完整性检查。即在用作索引的字段的字段名后添加完整性声明，比如写为 herold-heroQuality({10,20,21,30,31,32,40,41,42,43})-seq([1-4]){propId,equipRank}则表示对于某一个 herold，必须具备在 10,20……43 这些品阶下的穿装信息，并且在所有品阶下都必须具备四件装备信息。注意括号中可使用有效性检查或值范围检查规则去声明对该字段所填数据的要求，采用有效性声明方式的字段必须为 int、long、float 或 string 型，采用值范围声明方式的字段只能为 int、long 型。括号中支持用\$引用在 config.txt 配置文件中声明的有效性检查或值范围检查规则，但不支持用&&声明多组规则

显示字段信息选项

本软件可以将表格各字段的信息以 lua 语言中的注释形式生成到导出的 lua 文件最上方，信息包含字段名、数据类型、字段描述，对于 array、dict 类型的子元素会采用缩进形式体现从属关系。如果需要此功能，需声明[_columnInfo](#)运行参数

上面常规方式导出功能中提及的系统模块配置表，如果声明要显示字段信息，则会在生成的 lua 文件最上方打印如下内容：

```
1 -- systemId      int      系统ID
2 -- systemName    lang     系统名称
3 -- help          lang     系统帮助信息
4 -- openCondition dict[3]   开启条件（同时满足三个条件）
5 --   rankLimit   int      所需玩家等级（不限填-1）
6 --   vipLimit    int      所需Vip等级（不限填-1）
7 --   levelLimit  int      所需通关关卡（不限填-1）
8 -- openRewards   array[dict[3]:2] 开启系统后赠送物
9 --   [1]         dict[3]   奖励物1
10 --     type      int      类型
11 --     id        int      id
12 --     count     int      数量
13 --   [2]         dict[3]   奖励物2
14 --     type      int      类型
15 --     id        int      id
16 --     count     int      数量
17
```

但按索引嵌套方式进行导出时生成的字段信息则与之不同，会以缩进形式依次显示作为索引字段的信息，然后在花括号中显示作为 table value 的字段信息

下面两图分别为常规方式导出和按索引嵌套方式导出上面提及的英雄装备表时，生成的字段信息对比：

```
1  -- id                int                ID
2  -- heroId            int                英雄ID
3  -- heroQuality       int                英雄品阶
4  -- seq               int                装备槽位序号
5  -- propId            int                装备ID
6  -- equipRank         int                穿戴等级限制
7
```

```
1  -- heroId            int                英雄ID
2  --      heroQuality  int                英雄品阶
3  --      seq          int                装备槽位序号
4  --      {
5  --          propId    int                装备ID
6  --          equipRank int                穿戴等级限制
7  --      }
```

导出到 MySQL 数据库功能

本软件提供将 Excel 表导出到 MySQL 数据库的功能，会按照您的配置在 MySQL 数据库中创建指定名称的表并插入数据。**注意：导出之前本软件会先删除 MySQL 数据库中的同名旧表，然后重新创建新表并插入数据**

使用导出至 MySQL 数据库功能，所需要进行的配置如下：

1、在软件运行参数中声明[-exportMySQL 参数](#)

2、必须在 config.txt 配置文件中声明 [connectMySQLString 参数](#)所配置的数据库连接字符串，如果有需要还可以配置 [createDatabaseTableExtraParam 参数](#)设置建表时额外添加的参数字符串

3、在需要导出至 MySQL 数据库的 Excel 表的 config 表中通过 [exportDatabaseTableName 参数](#)配置导出到 MySQL 数据库中的表名。还可以额外通过配置 [exportDatabaseTableComment 参数](#)指定 MySQL 表的 Comment。string 型字段中的空单元格在导出至 MySQL 数据库时默认为空字符串，若要导出为 NULL，需要额外配置 [exportDatabaseWriteNullForEmptyString 参数](#)

4、在 Excel 表中，为每个字段[配置导出到 MySQL 数据库中的字段名及对应 MySQL 的数据类型](#)。但**注意本软件不会对配置进行检查**，比如一个 string 型的字段如果设为导出为 MySQL 中 int 型，只有在执行 SQL 语句进行插入时，会打印出 MySQL 输出的报错信息

注意：Windows 版本的 MySQL 默认不分区表名大小写，而 Linux 下区分（虽然 Windows 下可通过修改 MySQL 配置文件 my.ini 中 lower_case_table_names 参数为 2 开启大小写敏感，但 MySQL 也会报兼容性警告，并且表名虽然可以出现大写，但不支持仅大小写不同的表名），故在使用本软件时需自行确认 MySQL 数据库的设置，否则即便定义表名中包含大写也会被 MySQL 强制转为小写形式

其他导出功能

导出 csv 文件功能

该功能可将指定的 Excel 表格额外导出为 csv 文件，即每行中通过指定分隔符将各个字段内容进行分隔。使用此功能，需配置对应的[软件运行参数](#)

注意：1、date 和 time 型导出为 csv 文件时，按 toLua 的格式进行导出，若 date 型声明 toLua 的格式为 dateTable，则按 input 格式进行导出。tableString 和 mapString 型会被原样输出 2、array、dict 型字段所占用的用于标识每行中该字段是否生效的列也会被导出为 csv 文件中的一列，值为-1 表示该行中此字段不生效（可能是此字段或者其父字段被标为无效），值为空则表示生效。3、array、dict 型下属子元素列的字段名按如下规则生成：array 下属的子元素，字段名生成格式为“array 字段名[从 1 开始的下标序号]”。dict 下属的子元素，生成格式为“dict 字段名.下属字段名”

导出 csv 对应的 C#类文件功能

该功能可将指定的 Excel 表格额外导出为 csv 对应的 C#类文件，使用此功能需配置对应的[软件运行参数](#)

比如样例中的 System 表转为 C#类如下图所示：

```
1 using System;
2 using System.Collections.Generic;
3
4 namespace Assets.Scripts.CsvClass
5 {
6     public class SystemData
7     {
8         public int systemId { get; set; }
9         public string systemName { get; set; }
10        public string help { get; set; }
11        public Dictionary<string, object> openCondition { get; set; }
12        public List<Dictionary<string, object>> openRewards { get; set; }
13    }
14 }
```

注意：1、自定义类和文件名、命名空间、引用类库的参数设置参看软件运行参数。2、array、dict 型会被转为 C#中对应的 List 和 Dictionary 型。3、时间型除了指定导出为时间戳形式(被转为数值型)以外，会被统一转为 DateTime 型。4、json、mapString 型会被转为 LitJson 类型

导出 csv 对应的 Java 类文件功能

该功能可将指定的 Excel 表格额外导出为 csv 对应的 Java 类文件，使用此功能需配置对应的[软件运行参数](#)

比如样例中的 System 表转为 Java 类如下图所示：

```
1 package org.xlsxtolua.pojo;
2
3 import java.util.ArrayList;
4 import java.util.HashMap;
5
6 public class SystemData {
7     private int systemId;
8     private String systemName;
9     private String help;
10    private HashMap<String, Object> openCondition;
11    private ArrayList<HashMap<String, Object>> openRewards;
12
13    public int getSystemId() {
14        return systemId;
15    }
16
17    public void setSystemId(int systemId) {
18        this.systemId = systemId;
19    }
20
21    public String getSystemName() {
22        return systemName;
23    }
24
25    public void setSystemName(String systemName) {
```

注意：1、自定义类的文件名、包名、引用类库的参数设置以及下述的各种选项设置参看软件运行参数。2、array、dict 型会被转为 Java 中对应的 ArrayList 和 HashMap 型。3、时间型除了指定导出为时间戳形式（被转为数值型）以外，按用户选择转为 Date 或 Calendar。4、json 型会被 Gson（Google 提供的用来在 Java 对象和 Json 数据之间进行映射的 Java 类库，被广泛使用）转为 Object 类型，但可在使用时将这个 Object 型强制转为对应 Json 的具体类型。

mapString 型的处理与 json 型相同 5、转出的 Java 文件会依照 Java 编码规范，将各个字段声明为 private 的变量，然后提供 public 的 get 与 set 方法。6、用户可选是否为生成的 Java 类提供无参构造函数和含全部参数的构造函数

导出 json 文件功能

该功能可将指定的 Excel 表格额外导出为 json 文件，即用 json 字符串表示的数据。使用此功能，需配置对应的

[软件运行参数](#)

每张表格生成 json 的形式有两种选择：第一种是将 Excel 表每行数据对应的信息作为一个 json object，整张表作为一个 json array，将各行信息对应的 json object 包含进来。另一种则是将整张表作为一个 json object，将每行数据以表格主键列为 key，各字段信息组成的 json object 为 value。两种方式生成 json 的不同形式见下面两图：

```
1▼ [
2▼   {
3     "id":1,
4     "heroId":1,
5     "heroQuality":10,
6     "seq":1,
7     "propId":200001,
8     "equipRank":1
9   },
10▼  {
11    "id":2,
12    "heroId":1,
13    "heroQuality":10,
14    "seq":2,
15    "propId":200002,
16    "equipRank":1
17  },
18▼  {
19    "id":3,
20    "heroId":1,
21    "heroQuality":10,
22    "seq":3,
23    "propId":200003,
24    "equipRank":1
25  }
26 ]
27
28
```

```
1▼ {
2▼   "1":{
3     "id":1,
4     "heroId":1,
5     "heroQuality":10,
6     "seq":1,
7     "propId":200001,
8     "equipRank":1
9   },
10▼  "2":{
11    "id":2,
12    "heroId":1,
13    "heroQuality":10,
14    "seq":2,
15    "propId":200002,
16    "equipRank":1
17  },
18▼  "3":{
19    "id":3,
20    "heroId":1,
21    "heroQuality":10,
22    "seq":3,
23    "propId":200003,
24    "equipRank":1
25  }
26 }
```

支持带缩进格式与不带两种不同的输出形式，输出结果如下图所示：

```
[
  {
    "heroId":1,
    "name":"英雄法师",
    "rare":11,
    "type":1,
    "defaultStar":1,
    "isOpen":true,
    "attributes":{
      "attack":{
        "physical":20,
        "magic":100,
        "canCrit":true,
        "hitRate":0.9,
        "ult":[
          {
            "name":"CoupDeGrace",
            "params":[
              "circle",
              1,
              0,
              true
            ],
            "cd":5
          }
        ]
      },
      "defence":{
        "physical":10,
        "magic":60
      }
    }
  },
  {"heroId":2,"name":"英雄战士","rare":11,"type":2,"defaultStar":1,"isOpen":true,"attributes":null}, {"heroId":3,"name":"英雄牧师","rare":11,"type":3,"defaultStar":1,"isOpen":false,"attributes":null}, {"heroId":4,"name":"英雄勇士","rare":11,"type":4,"defaultStar":1,"isOpen":false,"attributes":{"attack":{"physical":140,"magic":0,"canCrit":true,"hitRate":0.9,"ult":[{"name":"CoupDeGrace","params":["circle",1,0,true],"cd":5}]},"defence":{"physical":10,"magic":60}},"hp":200,"modelSize":[4,5,10.5],"petPhrase":"I will kill you!"}}]
```

注意：date 和 time 型导出为 json 文件时，按 toLua 的格式进行导出，若 date 型声明 toLua 的格式为 dateTable，则按 input 格式进行导出

辅助工具介绍

MySQLToExcel

可能您想开始使用 XlsxToLua 软件，但项目已经进行了很久，会担心新建符合本软件要求格式的 Excel 可能带来很多工作量。如果您的项目目前用 MySQL 数据库存储配置表格，可以利用 MySQLToExcel 这个辅助工具直接将指定的 MySQL 中的表格连同数据生成为符合 XlsxToLua 软件要求的 Excel 配置表，从而省去您手工建立 Excel 表所需的工作量。如果您的项目目前使用 csv 文件作为配置表载体，也可使用 MySQL 官方提供的工具将 csv 文件导入 MySQL 数据库，然后利用 MySQLToExcel 辅助工具实现 Excel 表的生成

首先需要根据您的项目的具体情况，[对 MySQLToExcel 进行二次开发](#)，指定项目中用到的 MySQL 中各种数据类型转换为哪种 XlsxToLua 软件支持的数据类型

MySQLToExcel 辅助工具[所在目录下的 config.txt 是配置文件](#)，设置软件运行参数，其格式要求同[XlsxToLua 配置文件格式要求](#)，[该配置文件必须存在](#)，否则程序无法正常运行

本辅助工具会将 MySQL 数据库表的字段注释生成到 Excel 表格中 data 表的第一行作为字段描述，将数据库表的列名生成到第二行作为字段名，将数据库表的数据类型转换为 XlsxToLua 软件支持的数据类型后生成到第三行作为字段数据类型，将数据库表中部分 notEmpty、unique 规则生成到第四行作为字段检查规则，将数据库表的数据类型

声明生成到第五行作为导出至 MySQL 数据库的数据类型配置，同时数据库表中填写的数据也会被一一生成到 Excel 表中。此外，还会自动生成 config 表，并在其中生成 [exportDatabaseTableName 参数](#)，自动填写上该数据库表的表名，生成 [exportDatabaseTableComment 参数](#)，自动填写上该数据库表的 Comment

本辅助工具还会对生成的 Excel 表格进行格式上的美化，设置单元格自动列宽（使得列宽根据内容自动调整，每个单元格在一行中可显示完整内容），设置单元格的`最大宽度`（根据您在配置文件中的设置），对于因内容过多而通过自动列宽后超过最大列宽的单元格，强制缩小列宽到所允许的最大宽度。最后设置单元格内容自动换行，使得单元格自动扩大高度以显示所有内容。为了使列与列的分隔更加明显，您还可以在配置文件中设置每列的背景色。除此之外，您还可以在配置文件中为 data、config 两个 Sheet 表指定 Sheet 表标签按钮的背景色

目前本辅助工具可进行的配置如下表所示：

参数名	作用	填写方法
connectMySQLString	连接到 MySQL 数据库的连接字符串	例如：server=127.0.0.1;port=3306;uid=root;password=root;database=mydb;Charset=utf8;
exportExcelPath	导出 Excel 文件的存储目录	填写已存在的文件夹
exportDataTableNames	需要导出的数据库表名	用 分隔各个表名
columnMaxWidth	对生成的 Excel 表进行美化时，设置列的	填写合法的列宽数值

	最大宽度	
columnBackgroundColor	每列的背景色	依次为各列填写微软支持的颜色编号（详见 https://msdn.microsoft.com/en-us/library/cc296089.aspx 其中 0 代表无背景色），列与列的背景色颜色编号之间用英文逗号分隔，本辅助工具将为各列按配置循环设置颜色，即如果配置了 3 列的颜色，则会为第 4 列使用与第 1 列相同的背景色。此参数可以不配置，则不对各列设置背景色
dataSheetTabColor	data 表的标签按钮颜色	填写微软支持的颜色编号，如不想设置则可以不声明此参数
configSheetTabColor	config 表的标签按钮颜色	同上

XlsxToLuaGUI

XlsxToLua 软件采用命令行方式传参后启动运行，这对于程序开发人员来说没有使用难度，但对于完全不懂程序的策划等人员来说，上手难度不低。XlsxToLuaGUI 辅助工具为解决这个问题而编写，它采用图形化界面方式呈现 XlsxToLua 各种运行参数及设置，任何人员通过简单的勾选、配置就可以完成 XlsxToLua 运行参数的配置，从而让 XlsxToLua 按您的要求工作。同时 XlsxToLuaGUI 辅助工具可以保存您设置的运行参数，避免每次都要重复设置

软件界面如下图所示，在界面中将 XlsxToLua 软件的运行参数以图形化方式呈现给您，您可以通过在文本框中输入参数或勾选用于设置选项的多选框实现参数配置。其中需要注意：导出部分 Excel 表功能、导出 csv 或 json 文件功能中选择 Excel 表的按钮，在被点击之前，必须设置过界面上方第二项的“Excel 文件所在目录”，然后在弹出的文件选择对话框中，会直接定位到此目录，然后您可以直接用鼠标选择指定要操作的 Excel 表，可以通过按住 Shift、Ctrl 键进行多选

XlsxToLua GUI by 张齐

XlsxToLua所在路径: C:\Project\Tools\XlsxToLua.exe 选择

Excel文件所在目录: C:\Project\Excel 选择

lua文件导出目录: C:\Project\Client\Assert\lua\data 选择

项目Client所在目录: C:\Project\Client\ 选择

lang文件所在路径: C:\Project\Client\Assert\Resources\Text 选择

可选参数:

☒ -columnInfo (在生成lua文件的最上方用注释形式显示列信息, 默认不开启)

☐ -unchecked (不对表格进行查错, 不推荐使用)

☐ -printEmptyStringWhenLangNotMatching (当lang数据类型key在lang文件中找不到对应值时, 在lua文件输出字段值为空字符串即xx = "", 默认为输出nil)

☒ -exportMySQL (将表格数据导出到MySQL数据库中, 默认不导出)

☐ -allowedNullNumber (允许int、float型字段下填写空值, 不推荐使用)

☐ -part (后面在英文小括号内声明本次要导出的Excel文件名, 用|分隔, 未声明的文件将被本工具忽略)

选择

☐ -except (后面在英文小括号内声明本次忽略导出的Excel文件名, 用|分隔, 注意不能与-part参数冲突)

选择

☐ 额外导出部分表格为json文件

需导出的表格名(用|分隔, 全部则用\$all): 选择

json文件导出路径: 选择

json文件扩展名: txt ☐ 将生成的json字符串整理为带缩进格式的形式

☒ 生成各行数据对应的json object包含在一个json array的形式

☒ 若生成包含在一个json object的形式, 使每行字段信息对应的json object中包含主键列对应的键值对

☐ 额外导出部分表格为csv文件

需导出的表格名(用|分隔, 全部则用\$all): 选择

csv文件导出路径: 选择

csv文件扩展名: csv 字段分隔符: ,

☒ 在首行列举字段名称 ☒ 在其后列举字段数据类型

自动生成C#或Java类名时的类名前缀: 后缀:

☐ 额外导出部分表格为csv对应C#类文件

需导出的表格名(用|分隔, 全部则用\$all): 选择

C#类文件导出路径: 选择

C#类所属命名空间:

C#类引用类库(用英文逗号分隔):

☐ 额外导出部分表格为csv对应Java类文件

需导出的表格名(用|分隔, 全部则用\$all): 选择

Java类文件导出路径: 选择

Java类对应包名:

Java类引用类库(用英文逗号分隔):

☒ 时间型使用Date而不是Calendar ☐ 生成无参构造函数 ☐ 生成含所有参数的构造函数

☐ 使用相对路径进行配置(相对于本工具路径)

保存配置 载入配置 执行 生成bat批处理脚本

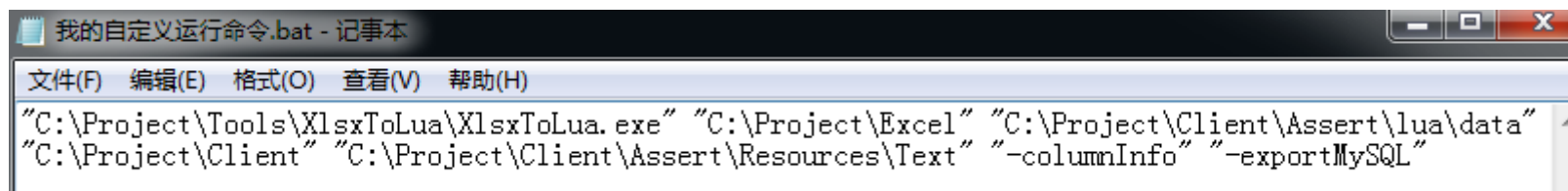
设置完成后, 点击“执行”按钮, 该辅助工具就会调用 XlsxToLua 软件按您设置的运行参数执行

点击“生成 bat 批处理脚本”按钮, 并选择存储路径后, 就会在选择的存储路径下生成按本次设置的运行参数

直接调用 XlsxToLua 软件运行的 bat 批处理脚本，如下图所示：



通过文本编辑软件可以打开此文件，可见其内容如下图所示：



XlsxToLua 软件运行后有返回值，0 为成功，-1 为发现错误，在使用 bat 调用本软件后可通过 %errorlevel% 判断导出是否成功从而进一步处理

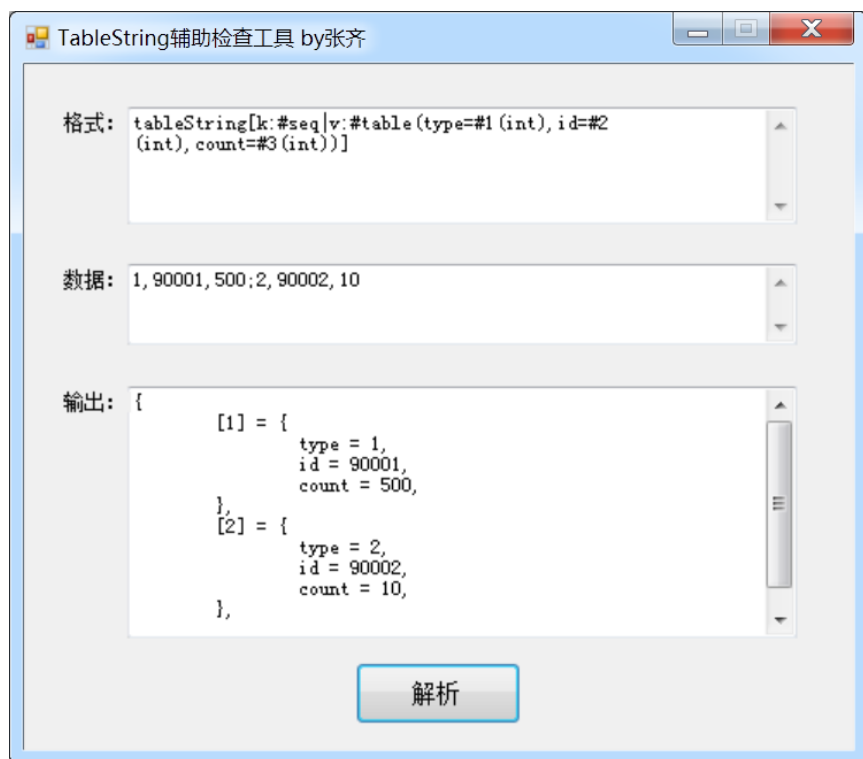
之后，在 Windows 系统下直接运行此脚本，便可按照本次设置的运行参数运行 XlsxToLua 软件，避免每次都要设置运行参数的麻烦

另外，本辅助提供“保存配置”和“载入配置”功能，可将当前设置的情况进行存储与读取，方便下次使用时直接载入之前的配置进行微调

TableStringChecker

XlsxToLua 软件中自定义了 `tableString` 这种为方便声明较复杂的嵌套结构而设计的数据结构，TableStringChecker 辅助工具以图形化界面方式提供给您检验 `tableString` 数据结构声明及测试的功能

使用方法为在如下图所示的界面中，输入格式定义及测试数据，点击“解析”按钮，本辅助工具便会在输出文本框中将转换为的 `lua table` 显示出来，若格式定义或数据有误，也会进行错误提示

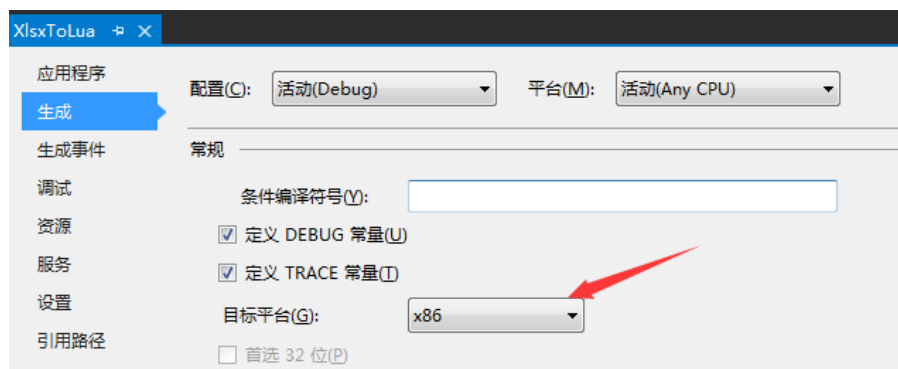


二次开发、编译说明

XlsxToLua

编译说明

- 1、 本软件使用 Microsoft Visual Studio 2013 基于 .NET Framework 2.0 进行开发编译，编程语言为 C#
- 2、 本软件在 Microsoft Visual Studio 2013 下设置为编译为 32 位的应用程序，如下图所示：

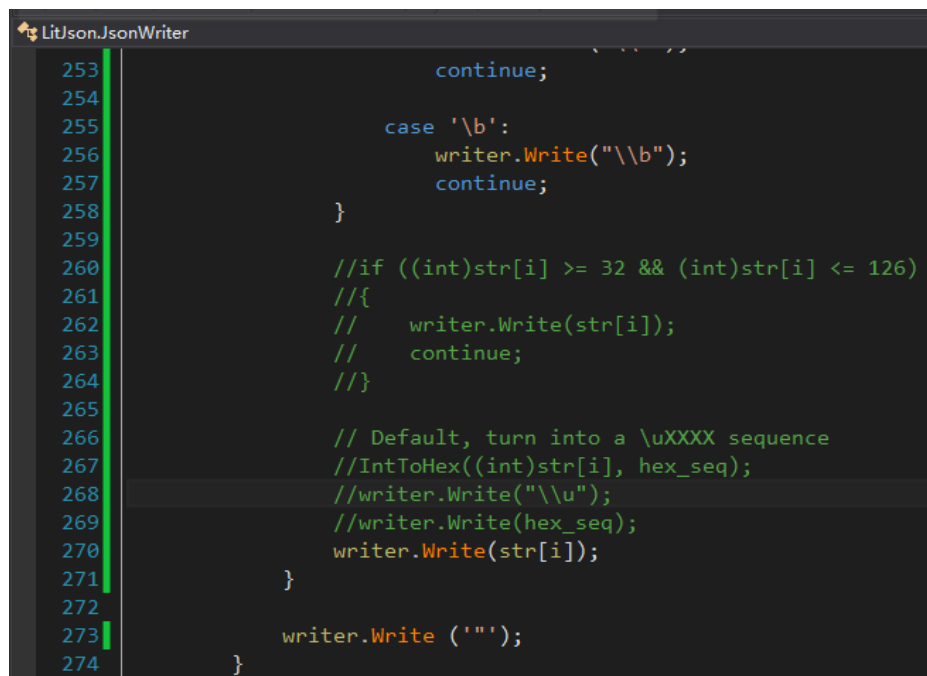


原因是为了更好的兼容性，因为本软件需要微软官方 Office 2007 数据连接组件，如果使用本软件的电脑为 64 位 Windows 操作系统但安装的是 32 位的 Office，编译出的 64 位的本软件运行时仍旧无法读取 Excel 文件。若你的应用环境中不存在此情况，可以自行编译 64 位的本软件使用

- 3、 本软件解析 json 字符串功能使用开源的 LitJson (<http://lbv.github.io/litjson/>)，操作 MySQL 数据库功能使

用官方提供的 C#类库（<http://dev.mysql.com/downloads/connector/net/1.0.html>），在进行编译时注意以上类库是否正确引用

- 4、 本软件对 LitJson 类库进行了修改，LitJson 类库打印的 json 字符串默认会将中文转为 Unicode 编码的形式，如“测试”显示为“\u6d4b\u8bd5”。为了让显示更加直观，本软件修改了 LitJson 类库中的 JsonWriter.cs 文件，将其中的 PutString 函数改为无论字符是什么，都直接 `writer.Write(str[i]);`而不是写入 `IntToHex` 后的形式。本软件中的 LitJson 类库已替换为进行如上修改后重新编译的版本，如果想使用 LitJson 默认处理中文的方式，可以自行到官网下载原版类库进行替换



```
253         continue;
254
255         case '\b':
256             writer.Write("\\b");
257             continue;
258     }
259
260     //if ((int)str[i] >= 32 && (int)str[i] <= 126)
261     //{
262         //    writer.Write(str[i]);
263         //    continue;
264     //}
265
266     // Default, turn into a \uXXXX sequence
267     //IntToHex((int)str[i], hex_seq);
268     //writer.Write("\\u");
269     //writer.Write(hex_seq);
270     writer.Write(str[i]);
271 }
272
273 writer.Write ('');
274 }
```

自定义字段检查函数的编写

编写自定义字段检查函数，用于比较复杂、个性的逻辑检查，以弥补通用检查规则无法涵盖所有检查需求的情况。比如，如下图所示的 `array[dict[3]:2]` 型字段用于配置游戏中某个系统模块开放时给予玩家的奖励物。需要通过检查确保满足如下要求：要求字段的数据结构必须为 `array[dict[3]:n]`，定义一种奖励类型的三个 `int` 型字段分别叫 `type`、`id`、`count`，每个奖励项的类型必须存在，除道具类型之外不允许奖励同一种类型，奖励数量必须大于 0，如果是道具类型则道具 `id` 在道具表中要存在

I	J	K	L	M	N	O	P	Q
开启系统后赠送物	奖励物1	类型	id	数量	奖励物2	类型	id	数量
openRewards		type	id	count		type	id	count
<code>array[dict[3]:2]</code>	<code>dict[3]</code>	int	int	int	<code>dict[3]</code>	int	int	int
func:CheckRewardListField								
		rewardType1 (INT)	rewardId1 (INT)	rewardCount1 (INT)		rewardType2 (INT)	rewardId2 (INT)	rewardCount2 (INT)
-1								
-1								
		1	100001	10	-1			
		1	100001	5	-1			

函数的开发需在本软件项目工程代码路径下的 `MyCheckFunction.cs` 中进行，函数必须形如 `public static bool 函数名(FieldInfo fieldInfo, out string errorString)`，建议函数名为 `CheckXxxField`。其中函数返回值为 `true` 表示通过检查，`false` 则为存在错误。本软件传入的 `fieldInfo` 是要检查字段的信息，包含字段属性、每行填写的值等，具体请看 `TableInfo.cs` 中对 `FieldInfo` 类的定义。而 `errorString` 是要向外传出的错误信息，如果进行检查后确定填写内容无误，

函数返回 `true` 的同时，需要将 `errorString` 赋值为 `null`，反之如果发现了错误，需要对 `errorString` 赋值为供软件使用者可见的错误原因和错误数据所在位置等信息，以便使用者根据该信息了解错误原因以及定位错误位置，从而进行快速更正。在 `MyCheckFunction.cs` 中，名为 `CheckRewardListField` 的函数针对上面所述奖励项字段进行检查，是提供给开发者进行编写的样例

```
public static class MyCheckFunction
{
    /// <summary>
    /// 检查奖励列表字段是否配置正确，要求字段的数据结构必须为array[dict[3]:n]，定义一种奖励类型的三个int型字段分别叫type、id、count
    /// </summary>
    public static bool CheckRewardListField(FieldInfo fieldInfo, out string errorString)
    {
        // 道具类型对应的type
        int PROP_TYPE;
        const string PROP_TYPE_CONFIG_KEY = "propType";
        if (AppValues.ConfigData.ContainsKey(PROP_TYPE_CONFIG_KEY))
        {
            string configValue = AppValues.ConfigData[PROP_TYPE_CONFIG_KEY];
            if (int.TryParse(configValue, out PROP_TYPE) == false)
            {
                errorString = string.Format("config配置中用于表示道具类型对应数字（名为\"{0}\"）的配置项所填值不是合法的数字（值为{0}）");
                return false;
            }
        }
    }
}
```

自定义整表检查函数的编写

编写自定义整表检查函数，用于对整表中的行与字段进行个性化的逻辑检查。比如英雄表中不同职业的英雄会有不同的能力属性偏向，但要求初始各属性的总和都相同。再比如对于英雄装备表，需要在逻辑上保证凡是英雄表

配置的英雄，在装备表中都为其配置该英雄在每个品阶下的四件装备的属性，如下图所示：

	A	B	C	D	E	F
	ID	英雄ID	英雄品阶	装备槽位序号	装备ID	穿戴等级限制
1						
2	id	heroId	heroQuality	seq	propId	equipRank
3	int	int	int	int	int	int
4		ref:Hero	\$heroQuality	\$heroEquipmentSeq	ref:Prop && (200000, 300000)	\$heroRank
5	id(INT)	heroId(INT)	heroQuality(INT)	seq(INT)	propId(INT)	equipRank(INT)
6	1	1	10	1	200001	1
7	2	1	10	2	200002	1
8	3	1	10	3	200003	1
9	4	1	10	4	200004	1
10	5	1	20	1	200005	5
11	6	1	20	2	200006	5
12	7	1	20	3	200007	5
13	8	1	20	4	200008	5
14	9	1	21	1	200001	10
15	10	1	21	2	200002	10

函数的开发同自定义字段检查函数那样，需在 `MyCheckFunction.cs` 中进行，函数必须形如 `public static bool 函数名(TableInfo tableInfo, out string errorString)`，建议函数名为 `CheckXxxTable`。其中返回值和 `errorString` 变量的含义同自定义字段检查函数。本软件传入的 `tableInfo` 是要检查表格的信息，包含表格属性、所有字段信息等，具体请看 `TableInfo.cs` 中对 `TableInfo` 类的定义。在 `MyCheckFunction.cs` 中，名为 `CheckHeroEquipmentTable` 的函数针对上面所述英雄装备表进行检查，是提供给开发者进行编写的样例

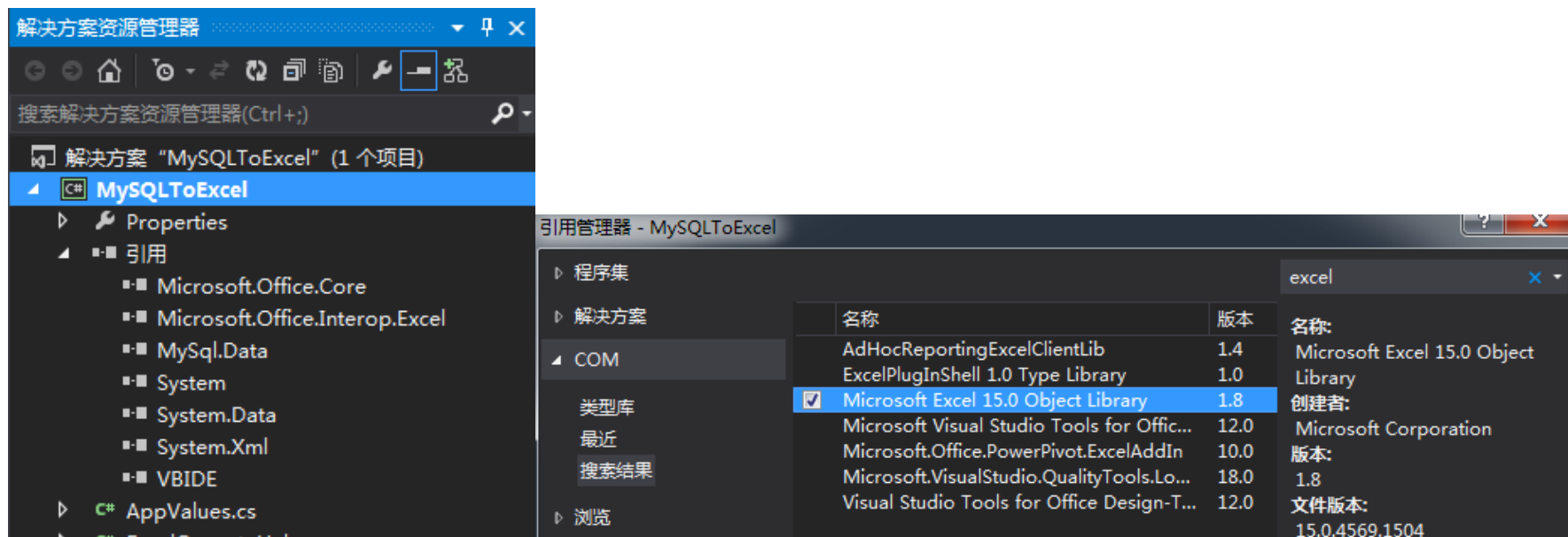
```
/// <summary>
/// 检查HeroEquipment表, 凡是填写的英雄都必须填写在所有品阶下四个槽位可穿戴装备信息
/// </summary>
0 个引用
public static bool CheckHeroEquipmentTable(TableInfo tableInfo, out string errorString)
{
    // 从配置文件中读取英雄的所有品阶 (key: 品阶, value: 恒为true)
    Dictionary<int, bool> HERO_QUALITY = new Dictionary<int, bool>();
    // 每品阶英雄所需穿戴的装备数量
    const int HERO_EQUIPMENT_COUNT = 4;

    const string HERO_QUALITY_CONFIG_KEY = "$heroQuality";
    if (AppValues.ConfigData.ContainsKey(HERO_QUALITY_CONFIG_KEY))
    {
        string configString = AppValues.ConfigData[HERO_QUALITY_CONFIG_KEY];
        // 去除首尾花括号后, 通过英文逗号分隔每个有效值即可
        if (!(configString.StartsWith("{") && configString.EndsWith("}")))
        {
            errorString = string.Format("表示英雄所有品阶的配置{0}错误, 必须在首尾用一对花括号包裹整个字
```

MySQLToExcel

编译说明

- 1、 同 [XlsxToLua 软件编译说明](#) 的前 2 条，并且也使用 MySQL 官方提供的 C#操作类库
- 2、 本软件写入 Excel 文件的方式采用微软官方的 Interop.Excel 类库，其需要本机中安装微软 Excel 软件的支持。本软件编译时的电脑安装的是 Excel 2013 版本，经测试也支持安装 Excel 2016 版本的电脑运行本软件。但在低于 2013 版本的电脑中无法运行，比如需要支持 2010 版本的 Excel，就需要在安装有 2010 版本的电脑上，修改 Microsoft Visual Studio 2013 中的引用，替换为本机中 Excel 版本的类库，然后进行编译



MySQL 数据类型转换为 XlsxToLua 规定的数据类型

不同项目使用 MySQL 数据库中数据类型的习惯千差万别，MySQLToExcel 软件需要使用者自定义每种用到的 MySQL 数据库中的数据类型转为 XlsxToLua 软件中哪种对应的数据类型。需要使用者修改 MySQLToExcel 源码中的 `ExcelOperateHelper.cs` 文件，在如下图所示的 `_ConvertDataType` 函数中适配项目中使用到的每种 MySQL 数据库的数据类型到 XlsxToLua 软件中的转换关系：

```
167  /// <summary>
168  /// 自定义MySQL数据库中的数据类型对应转换到XlsxToLua工具要求的数据类型
169  /// </summary>
170  1 个引用
170  private static string _ConvertDataType(string databaseDataType, string length)
171  {
172      switch (databaseDataType)
173      {
174          case "int":
175              return "int";
176          case "bigint":
177              return "long";
178          case "tinyint":
179              {
180                  if ("1".Equals(length))
181                      return "bool";
182                  else
183                      return "int";
184              }
185          case "float":
186          case "double":
187              return "float";
188          case "char":
```

其中传入的第一个参数 `databaseDataType` 为读取的 MySQL 数据库中某个字段的数据类型，第二个参数为该数据类型具体声明的长度。如 MySQL 数据库中某个字段为 `int(11)` 型，则传入的第一个参数值为 “int”，第二个参数为 “11”。您只需要根据传入的两个参数值，自己编写代码指定其转为 XlsxToLua 软件中的哪种数据类型即可

注意：必须为项目中所有使用过的 **MySQL** 数据类型指定转为 **XlsxToLua** 软件的哪种数据类型，否则软件会进行
报错提示

我的开源项目

GitHub 主页

<https://github.com/zhangqi-ulua>

XlsxToLua

<https://github.com/zhangqi-ulua/XlsxToLua>

Excel 表格数据导出为 Lua table、csv、json 形式的工具，兼带数据检查功能以及导出、导入 MySQL 数据库功能

国际化文本工具

<https://github.com/zhangqi-ulua/LangTextTools>

国际化文本工具，旨在帮助国际化人员从繁重的整理国际化 Excel 母表、在版本更新后收集需要新增翻译或重新翻译的内容然后外包给专业翻译公司、待翻译公司翻译完成发回后将翻译内容填充到母表中等工作中解放出来

ServerDatabaseVersionUpdateHelper

<https://github.com/zhangqi-ulua/ServerDatabaseVersionUpdateHelper>

本工具可以对比新旧两个版本的服务器软件所用数据库，并根据配置（可设置对某表格忽略对比、仅比较表结构、比较结构和数据），展示两库差异并生成将旧版本数据库的表结构和数据升级为新版本的 **SQL**

交流反馈方式

我的 QQ

2246549866

交流 QQ 群

132108644



群名称：XlsxToLua交流群

群 号：132108644