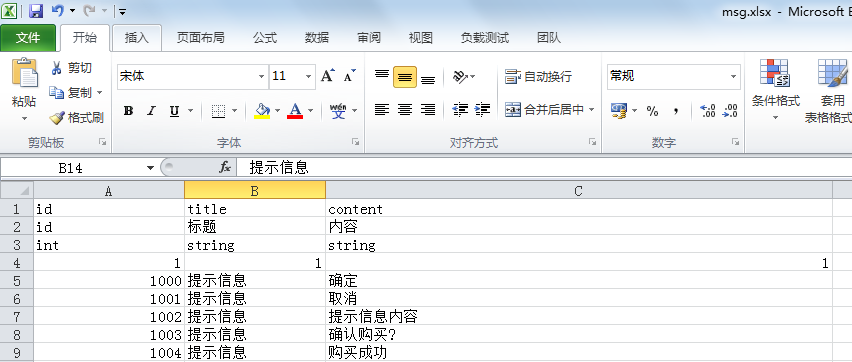
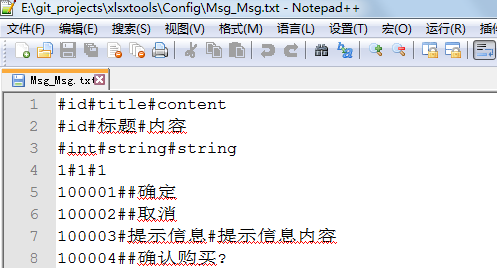
1. **功能**

1.xlsx文件转换成类CSV格式(以#号分隔)的txt文本文件

2.转换对应语言的代码解析txt文本文件（C#/Java/OC已经支持,C/CPP将支持）  
3.xlsx文件数据：

 4.txt文件数据：



1. **xlsx文件规范（msg.xlsx为例）**

1.支持数据类型：int(整型)，float(浮点型)，string(字符串)，数组类型

2.前三行为固定格式，描述xlsx表数据格式

如msg.xlsx的msg表

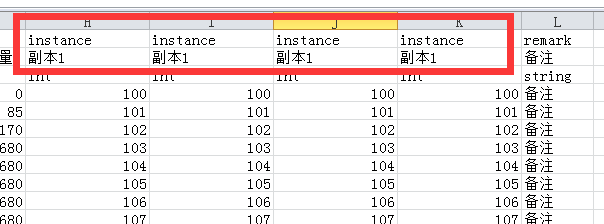
第一行：字段名字(id,title,content)

第二行：描述，代码注释

第三行：类型，支持三种（int,float,string）

3.数组定义

连续字段认为为数组,如



4.子结构数组

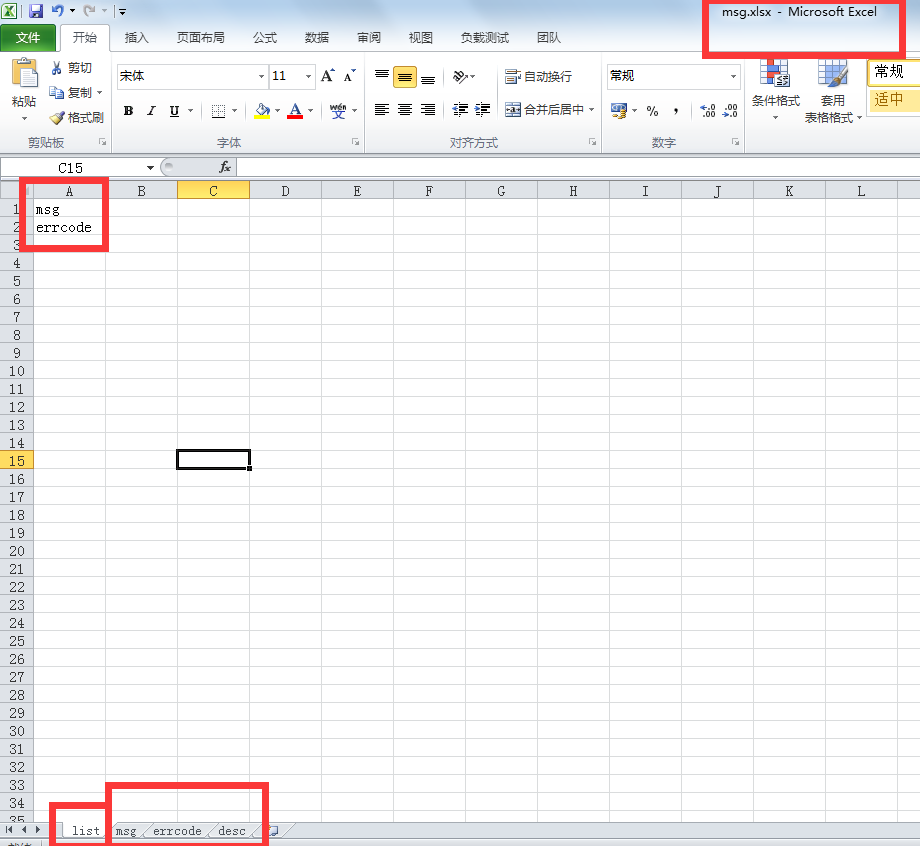
如在字段定义中出现了”-”(中划线)，则说明是数组，如



会转出相应的的Tip子结构数组

4. list表标识需要转换的表

如.msg.xlsx文件下有list,msg,errcode,desc四个表,list表标识转换msg，errcode表，不转换desc 表



1. **转换txt文件与代码文件命名规范（msg.xlsx为例）**
2. **txt命名规范**

文件名：文件名(首字母大写) + ”\_” + 表名（首字母大写）

如：Msg\_Msg.txt,Msg\_Errcode.txt

1. **代码文件规范**

**a.文件名：文件名(首字母大写) + 表名（首字母大写）+ “ConfigTable”**

**b.结构名：文件名+表名+”Config”**

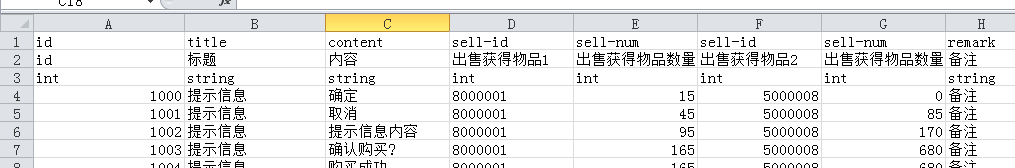
**c.子结构名：文件名+表名+子结构字段名+”Config”**

如：

C#：MsgMsgConfigTable.cs, MsgDescConfigTable.cs

Java：MsgMsgConfigTable.java, MsgErrcodeConfigTable.java

如MsgMsgConfigTable.cs对应msg.xlsx文件的Msg表



第一行：字段名字(id,title,content)

第二行：描述，代码注释

第三行：类型，支持三种（int,float,string）

**数据结构定义**  
public class MsgMsgConfig

{

public int id; //id

public string title; //标题

public string content; //内容

public MsgMsgSellConfig[] sells;

public int[] instance;

public string remark; //备注

}

public class MsgMsgSellConfig

{

public int id; //出售获得物品1

public int num; //出售获得物品数量1

}

**解析txt文件代码**

public class MsgMsgConfigTable

{

Dictionary<int,MsgMsgConfig> m\_configs = new Dictionary<int,MsgMsgConfig>();

public Dictionary<int, MsgMsgConfig> configs

{

get {

return m\_configs;

}

}

public object ConfigProcess(string[] row)

{

if (row.Length < 3)

{

return null;

}

RowHelper rh = new RowHelper(row);

MsgMsgConfig rec = new MsgMsgConfig();

rec.id = CSVUtility.ToInt(rh.Read()); //id

rec.title = rh.Read(); //标题

rec.content = rh.Read(); //内容

rec.sells = new MsgMsgSellConfig[2];

for (int i = 0;i < 2;i ++) {

rec.sells[i] = new MsgMsgSellConfig();

}

rec.sells[0].id = CSVUtility.ToInt(rh.Read()); //出售获得物品1

rec.sells[0].num = CSVUtility.ToInt(rh.Read()); //出售获得物品数量1

rec.sells[1].id = CSVUtility.ToInt(rh.Read()); //出售获得物品2

rec.sells[1].num = CSVUtility.ToInt(rh.Read()); //出售获得物品数量2

rec.instance = new int[7];

rec.instance[0] = CSVUtility.ToInt(rh.Read()); //副本1

rec.instance[1] = CSVUtility.ToInt(rh.Read()); //副本2

rec.instance[2] = CSVUtility.ToInt(rh.Read()); //副本3

rec.instance[3] = CSVUtility.ToInt(rh.Read()); //副本4

rec.instance[4] = CSVUtility.ToInt(rh.Read()); //副本5

rec.instance[5] = CSVUtility.ToInt(rh.Read()); //副本6

rec.instance[6] = CSVUtility.ToInt(rh.Read()); //副本7

rec. remark = rh.Read(); //备注

return rec;

}

public void Load()

{

CSVReader reader = new CSVReader();

reader.LoadText("Config/Msg\_Msg.txt", 3);

int rows = reader.GetRowCount();

for (int r = 0; r < rows; ++r)

{

string[] row = reader.GetRow(r);

MsgMsgConfig ac = ConfigProcess(row) as MsgMsgConfig;

configs.Add(ac.id, ac);

}

}

}

**3.生成对应的配置表管理类ConfigMgr.cs，加载所有配置表**

public class ConfigMgr

{

public static ConfigMgr GetInstance()

{

if(null == mInstance)

{

mInstance = new ConfigMgr();

mInstance.Init();

}

return mInstance;

}

private static ConfigMgr mInstance = null;

public ItemItemConfigTable ItemItem;

public MsgDescConfigTable MsgDesc;

public MsgErrcodeConfigTable MsgErrcode;

public MsgFormatConfigTable MsgFormat;

public MsgMsgConfigTable MsgMsg;

public void Init ()

{

ItemItem = new ItemItemConfigTable();

ItemItem.Load(); MsgDesc = new MsgDescConfigTable();

MsgDesc.Load(); MsgErrcode = new MsgErrcodeConfigTable();

MsgErrcode.Load(); MsgFormat = new MsgFormatConfigTable();

MsgFormat.Load(); MsgMsg = new MsgMsgConfigTable();

MsgMsg.Load();

}

}

**4.txt文件解析类CSVReader.cs**

using UnityEngine;

using System.Collections;

using System.IO;

using System.Collections.Generic;

public class RowHelper

{

string[] rows;

int idx;

public RowHelper(string[] rs)

{

rows = rs;

idx = 0;

}

public string Read()

{

return rows[idx++];

}

}

public class CSVReader

{

const int MaxColumn = 128;

public struct FieldData

{

public string[] FieldText;

}

private List<FieldData> items = new List<FieldData>();

private int columnCount = 0;

public bool LoadText(string textfileName, int skinLine)

{

string strText = ResLoad.msInst.LoadText(textfileName);

if(strText == null)

{

Utility.LogError("Load text file " + textfileName + " failed !");

return false;

}

string[] lines = strText.Split(new char[]{'\n'});

int iStartLine = Mathf.Max(0, skinLine);

for(int i = iStartLine; i < lines.Length; i++)

{

ParseLine(lines[i]);

}

return true;

}

public string GetString(int iRow, int iColumn)

{

if(iRow < 0 || iRow >items.Count)

{

return "";

}

if(iColumn < 0 || iColumn > GetColumnCount())

{

return "";

}

return items[iRow].FieldText[iColumn];

}

public int FindRowIndex(string key, int iFieldIndex)

{

for(int i = 0; i < items.Count; i++)

{

if(items[i].FieldText[iFieldIndex] == key)

{

return i;

}

}

return -1;

}

public int GetColumnCount() { return columnCount; }

public int GetRowCount() { return items.Count; }

public string[] GetRow(int idx)

{

return items[idx].FieldText;

}

private void ParseLine(string lineText)

{

string[] vals = lineText.Split(new char[]{'#'});

FieldData item = new FieldData();

item.FieldText = new string[MaxColumn];

if(columnCount == 0)

{

columnCount = vals.Length;

//empty table

if(columnCount <= 1)

{

return;

}

}

if(columnCount == vals.Length)

{

for(int i = 0; i < vals.Length; i++)

{

item.FieldText[i] = vals[i].Trim();

}

items.Add(item);

}

}

}