# Demo5-加载配置文件

2018年3月12日 18:07

加载配置文件也算是这个框架的基础，很多事情都依赖于配置文件（比如加载实体也是先读取配置文件，然后再创建对象的），本Demo将和大家一起了解GameFramework是如何加载配置文件的。

关于加载配置文件，官网也是有文档的，大家可以选择看文档：
http://gameframework.cn/archives/category/module/buildin/datatable

本文只做简单的补充说明，和要注意的地方。

## 1.配置文件格式

官方默认的配置文件格式是“制表符分隔”文本，即用tab作为分隔符的文件。
比如我们新建一个Hero.txt配置文件，如下格式（使用Excel编辑）：

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | # | 配置文件测试 | | | |
| 2 | # | int | string | int | |
| 3 | # | ID | 名称 | 攻击 | |
| 4 | | 1 | musicvs | 1 | |
| 5 | | 2 | mutou | 999 | |
| 6 | | | | | |
| 7 | | | | | |

带有#号的行为注释行，不带#号的为正式数据。

## 2.导出配置文件

在框架中是不能直接加载Excel配置文件的，需要保存为制表符分隔的文本文件，使用Excel自带的另存为功能即可：

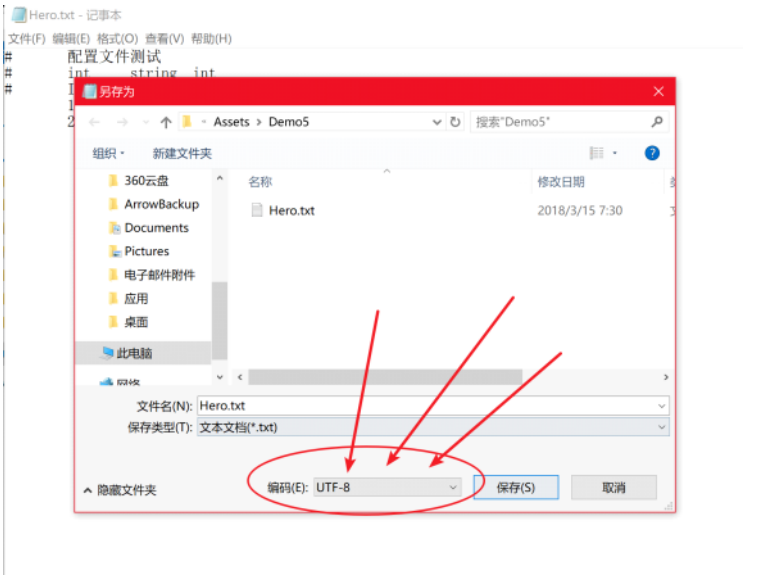

## 3.关于文件格式的坑

官方的文档特意说明了，导出的文本文件编码必须是ANSI，为此我遇到了一点问题——文件总是无法加载出来，内容一直是空的。

我以为是我的文件格式有问题，所以折腾了很久（这也是为什么这篇Demo隔了这么久才发布）。

原因是我的文件里带有中文，官方明确指出，除了注释行，其他地方不要出现中文。
然而，我只是注释行里出现了中文，依然无法加载。

最终，我把Hero.txt文件再另存为UTF-8格式，问题就解决了，大家也多留意一下。




## 4.与配置文件对应的类

因为文件需要加载到内存，当然需要对象来存储，我们首先要建一个和文件字段一一对应的类。
新建一个脚本，命名我DRHero.cs，内容如下：

```
using System.Collections;
using System.Collections.Generic;
using GameFramework.DataTable;
using UnityEngine;
public class DRHero : IDataRow
{
public int Id { get; protected set; }
public string Name { get; private set; }
public int Atk { get; private set; }
public void ParseDataRow(string dataRowText)
{
string[] text = dataRowText.Split('\t');
        int index = 0;
index++; // 跳过#注释列
        Id = int.Parse(text[index++]);
        Name = text[index++];
        Atk = int.Parse(text[index++]);
}
}
```

我们的Hero.txt文件里包含了Id、Name、Atk三个字段内容，所以DRHero类里也有对应的字段。
并且DRHero需要继承IDataRow，最重要的是ParseDataRow函数，这是解析配置文件的关键地方。

当框架读取配置文件时，会把文件的每一行内容作为参数调用ParseDataRow函数，这个函数负责解析字段内容。
由于Hero.txt是使用tab分隔的，所以直接把一行的内容按照tab拆分，然后一个个读取即可（注意，第1列是#号注释列，需要跳过）。

## 5.加载配置文件

新建一个Demo5场景，并且新建一个Demo5_ProcedureLaunch.cs脚本，作为初始流程。
我们在初始流程的OnEnter函数里加载配置文件：

```
    protected override void OnEnter(ProcedureOwner procedureOwner)
    {
        base.OnEnter(procedureOwner);
        // 获取框架事件组件
        EventComponent Event
            = UnityGameFramework.Runtime.GameEntry.GetComponent<EventComponent>
();
        // 获取框架数据表组件
        DataTableComponent DataTable
            =
UnityGameFramework.Runtime.GameEntry.GetComponent<DataTableComponent>();
        // 订阅加载成功事件
        Event.Subscribe(UnityGameFramework.Runtime.LoadDataTableSuccessEventArgs
.EventId, OnLoadDataTableSuccess);
        // 加载配置表
        DataTable.LoadDataTable<DRHero>("Hero", "Assets/Demo5/Hero.txt");
    }
```

由于GameFramework加载资源都是异步的，所以我们需要通过订阅LoadDataTableSuccessEvent事件来监听文件的加载。

加载配置文件需要用到框架的DataTableComponent组件的LoadDataTable函数。
函数需要指定配置表对应的类类型（这里是DRHero），第一参数是数据表名称，暂时没发有什么作用，第二个参数是配置文件的路径。

## 6.读取加载完成的文件数据

刚刚只是加载了配置文件，我们需要在很多地方用到文件的内容，所以接下里就要读取配置文件的内容。

之前已经订阅了配置表加载成功的事件，所以我们在这个事件里加载文件内容试试（实际上我们可以在其他地方加载，配置文件已经在内存里了，可以随时加载）

```
private void OnLoadDataTableSuccess(object sender, GameEventArgs e)
    {
```

```
        // 获取框架数据表组件
        DataTableComponent DataTable
            =
UnityGameFramework.Runtime.GameEntry.GetComponent<DataTableComponent>();
        // 获得数据表
        IDataTable<DRHero> dtScene = DataTable.GetDataTable<DRHero>();

        // 获得所有行
        DRHero[] drHeros = dtScene.GetAllDataRows();
        Log.Debug("drHeros:" + drHeros.Length);
        // 根据行号获得某一行
        DRHero drScene = dtScene.GetDataRow(1); // 或直接使用 dtScene[1]
        if (drScene != null)
        {
            // 此行存在，可以获取内容了
            string name = drScene.Name;
            int atk = drScene.Atk;
    Log.Debug("name:" + name + ", atk:" + atk);
        }
        else
        {
            // 此行不存在
        }
        // 获得满足条件的所有行
        DRHero[] drScenesWithCondition = dtScene.GetAllDataRows(x => x.Id > 0);

        // 获得满足条件的第一行
        DRHero drSceneWithCondition = dtScene.GetDataRow(x => x.Name ==
"mutou");

    }
```

加载配置表内容还是需要用到DataTableComponent组件，通过组件的GetDataTable函数即可获取配置表内容，获得IDataTable<DRHero>集合对象。
通过调用集合的GetAllDataRows可以获取配置表所有行的数据，通过GetDataRow可以获取某一个的数据。

并且，GetAllDataRows和GetDataRow支持传入查询表达式进行条件筛选，如果大家有用过LINQ的话，就知道这是有多方便的事情了。

## 7.唠叨一下

其实加载配置文件的核心步骤只有4个：

a.创建配置文件
b.创建文件类
c.加载配置文件到内存
d.在需要的地方读取配置文件内容

明白了这四个步骤，就没问题了。

OK，更多的内容大家还是参考官方文档吧，我在这里仅作补充说明。