XML写入

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Xml;//XML命名空间

public class XMLWriter : MonoBehaviour {
    void Start () {
        XmlDocument xmlDoc = new XmlDocument();
        XmlElement xmlElem = xmlDoc.CreateElement("Root");
        xmlElem.SetAttribute("url", "http://baidu.com");
        xmlDoc.AppendChild(xmlElem);

        XmlNode root= xmlDoc.SelectSingleNode("Root");
        XmlElement xmlElem2= xmlDoc.CreateElement("Title");
        xmlElem2.InnerText = "百度首页";
        root.AppendChild(xmlElem2);

        for (int i = 0; i < 3; i++)
        {
           XmlElement xmlElem3= xmlDoc.CreateElement("Body");
           xmlElem3.SetAttribute("title", "百科");
           root.AppendChild(xmlElem3);
        }

        XmlElement xmlElem4 = xmlDoc.CreateElement("End");
        xmlElem4.InnerText = "关于百度";
        root.AppendChild(xmlElem4);

        xmlDoc.Save(Application.dataPath + "/BaiduConfig.xml");
    }
}

```

生成文件格式

```
<Root url="http://baidu.com">
```

```
  <Title>百度首页</Title>
  <Body title="百科" />
  <Body title="百科" />
  <Body title="百科" />
  <End>关于百度</End>
</Root>
```

XML读取

```
<?xml version="1.0" encoding="UTF-8"?>
<PlayerConfig>
    <Player name="张无忌" id="1">
        <Atk>1000</Atk>
        <Atk>800</Atk>
        <Atk>500</Atk>
    </Player>
    <Player name="赵敏" id="2">
        <Atk></Atk>
        <Atk></Atk>
        <Atk>200</Atk>
    </Player>
    <Player name="周芷若" id="3">
        <Atk>500</Atk>
        <Atk>100</Atk>
        <Atk>200</Atk>
    </Player>
</PlayerConfig>
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Xml;

public class Player
{
```

```
    public string name;
    public int id;
    public List<int> datas;
}

public class XMLLoad : MonoBehaviour {

    private List<Player> players = new List<Player>();

    void Start () {
        LoadXML();
    }

    void LoadXML()
    {
        TextAsset textAsset = Resources.Load("PlayerConfig") as
TextAsset;
        XmlDocument xml = new XmlDocument();

        //加载xml字符串
        xml.LoadXml(textAsset.text);

        //获取当前xml字符串中的根元素
        XmlElement node=xml.DocumentElement;

        foreach(XmlElement nodeChild in node.ChildNodes)
        {
            Player player = new Player();
            player.name = nodeChild.GetAttribute("name");
            player.id = int.Parse(nodeChild.GetAttribute("id"));
            player.datas=new List<int>();

            foreach(XmlElement child in nodeChild.ChildNodes)
            {
                player.datas.Add(int.Parse(child.InnerText));
            }

            players.Add(player);
        }
    }
}

```

```xml
<Root url="http://baidu.com">
  <Title>百度首页</Title>
  <Body title="百科" />
  <Body title="百科" />
  <Body title="百科" />
  <End>关于百度</End>
  <Body title="百科" />
</Root>
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Xml;


public class Root
{
    public string url;
    public string title;
    public string end;
    public List<string> bodys=new List<string>();
}

public class XMLLoad : MonoBehaviour {

    void Start () {
        LoadXML();
    }

    void LoadXML()
    {
        TextAsset textAsset=Resources.Load("BaiduConfig") as
TextAsset;

        XmlDocument xml = new XmlDocument();
```

```
26
27          xml.LoadXml(textAsset.text);
28
29          Root root = new Root();
30
31          XmlElement node=xml.DocumentElement;
32
33          root.url=node.GetAttribute("url");
34
35          foreach (XmlElement nodeChild in node.ChildNodes)
36          {
37              if (nodeChild.Name.Equals("Title"))
38              {
39                  root.title = nodeChild.InnerText;
40              }
41              if (nodeChild.Name.Equals("Body"))
42              {
43                  root.bodys.Add(nodeChild.GetAttribute("title"));
44              }
45              if (nodeChild.Name.Equals("End"))
46              {
47                  root.end = nodeChild.InnerText;
48              }
49          }
50      }
51
52  }
53
```

XML框架

Config类 所有配置文件的基类 可以不继承Mono 封装读取的虚函数

ConfigDataBase 单例 管理所有的配置文件 提供XML加载 查找获取的方法

PlayerConfig类 重写Config类的虚方法 存储读取得到的数据 提供一些查找和遍历的方法

LoadManager用来初始化读取所有配置文件的脚本

注意：

1 ConfigDataBase可以继承于Mono 但是单例的写法和C#不同

Config 可以不继承Mono 因为涉及不到Unity的生命周期

2 加载配置文件用WWW加载 得到text 读取时使用xml.LoadXml(text)

3 读取StreamingAssets文件夹 必须要考虑不同平台下的路径问题

4 读取原理是从根元素开始 查找子元素 Attribute和InnerText

5 获取配置文件 使用泛型 更方便效率更高

Config类

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//配置文件的基类
public class Config {

    public virtual bool ReadLoad(string text)
    {
        if (string.IsNullOrEmpty(text))
            return false;
        return true;
    }
}

```

PlayerConfig

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Xml;

// PlayerConfig.xml对应的加载类
//Player类型 用来存储 配置文件中需要的结构
public class Player
```

```
{
    public string name;
    public int id;
    public List<int> datas;
}
public class PlayerConfig : Config {

    //配置文件中多个Player的集合
    public List<Player> players = new List<Player>();

    //子类重写父类Config类型提供的虚方法  为实现多态
    public override bool ReadLoad(string text)
    {
        if (!base.ReadLoad(text))
            return false;

        //记载配置文件的代码
        XmlDocument xml = new XmlDocument();
        xml.LoadXml(text);

        //获取当前xml字符串中的根元素
        XmlElement node = xml.DocumentElement;

        foreach (XmlElement nodeChild in node.ChildNodes)
        {
            Player player = new Player();
            player.name = nodeChild.GetAttribute("name");
            player.id = int.Parse(nodeChild.GetAttribute("id"));
            player.datas = new List<int>();

            foreach (XmlElement child in nodeChild.ChildNodes)
            {
                player.datas.Add(int.Parse(child.InnerText));
            }

            players.Add(player);
        }
        return true;
    }

    //根据PlayerID 得到Player
    public Player GetPlayerById(int id)
```

```
    {
        foreach (Player p in players)
        {
            if (p.id == id)
                return p;
        }
        return null;
    }
}
```

ConfigDataBase

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

//配置管理类
public class ConfigDataBase:MonoBehaviour{

    private static ConfigDataBase instance;
    public static ConfigDataBase Instance
    {
        get
        {
            return instance;
        }
    }

    //存储所有的配置文件类
    private Dictionary<Type, Config> configDic = new Dictionary<Type,
Config>();

    //StreamingAssets的文件夹路径
    private string path;
    void Awake()
    {
        //继承Mono单例的写法
        instance = this;
        DontDestroyOnLoad(gameObject);
```

```
    }

    public void AddConfig<T>() where T:Config,new()
    {
        var config = new T();
        StartCoroutine(LoadXml(config));
    }
    IEnumerator LoadXml(Config config)
    {
        //根据运行平台 构建不同的StreamingAssets路径
        path =
#if UNITY_ANDROID&&!UNITY_EDITOR
        "jar:file://" + Application.dataPath + "!/assets/" +
config.ToString() + ".xml";
#elif UNITY_IPHONE
        "file://" + Application.dataPath + "/Raw/" + config.ToString()
+ ".xml";
#else
        "file://"+Application.dataPath + "/StreamingAssets/" +
config.ToString() + ".xml";
#endif

        WWW www = new WWW(path);

        yield return www;

        //调用配置文件类的读取虚方法
        config.ReadLoad(www.text);
        //添加到管理类的字典中
        configDic.Add(config.GetType(), config);

    }

    //根据类型获取配置文件类
    public T GetConfig<T>() where T:Config
    {
        if (!configDic.ContainsKey(typeof(T)))
            return null;

        return configDic[typeof(T)] as T;
    }
}
```

```
67
```

LoadManager测试

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadManager : MonoBehaviour {

    void Start () {
        ConfigDataBase.Instance.AddConfig<PlayerConfig>();
        ConfigDataBase.Instance.AddConfig<ItemConfig>();
    }

    // Update is called once per frame
    void OnGUI () {
        if (GUILayout.Button("获取玩家信息"))
        {
            PlayerConfig config=
ConfigDataBase.Instance.GetConfig<PlayerConfig>();
            Player p = config.GetPlayerById(2);
            Debug.Log(p.name);
        }
    }
}
```