Easy Save3 入门中文文档

作者：文若

阅读文档：**EasySave3官网文档**

学习参考：Unity EasySave3中文图文教程详解-万能数据保存插件多平台支持 -- Chinar

翻译参考：Getting Started with Easy Save 3 -- 絮酱翻译

Getting started 开始

原文链接：Getting started

从Asset Store导入Easy Save以后，你可以立即从脚本或者PlayMaker中使用该插件。

关于使用Auto Save来保存和加载无代码的部分，详见自动保存指南 Auto Save guide

有关PlayerMaker操作的信息，详见PlayerMaker操作概述 PlayMaker Actions Overview

---

1. 基本类型保存加载

Easy Save将数据存储为键和值，很像字典

- 保存值，使用 ES3.Save
- 加载值，使用 ES3.Load

例如，把一个整数保存到名为“myInt”的键中，并再次加载

```
ES3.Save("myInt", 123);
myInt = ES3.Load<int>("myInt", defaultValue);
```

如果没有要加载的数据，它将返回`defaultValue`。

如果没有指定默认值`defaultValue`，则必须确保有加载的数据。例如，使用 ES3.KeyExists

```
if(ES3.KeyExists("myInt"))
    myInt = ES3.Load<int>("myInt");
```

把值加载的到现有的引用中，使用 ES3.LoadInto

```
// 直接将数据加载到Transform中，而不是创建一个新的Transform。
ES3.LoadInto("myTransform", this.transform);
```

注意，ES3.LoadInto只适用于引用类型，而不适用于基本类型（比如，int，string）或者值类型（structs）

2. Classes, Components 和 ScriptableObjects

保存和加载类，组件，数据容器的方式与保存加载任何其他数据的方式相同。然而，有一些重要的点需要注意：

- 类字段保存的条件
  - 是 *public* ，或者具有 *[SerializeField]* 属性
  - 不被 *const* 或 *readonly* 修饰'
  - 没有 *[Obsolete]* 或者 *[NonSerialized]* 属性
  - 是可支持类型
- UnityEngine.Object 按照引用和值储存
  - 这便意味着，如果要加载的实例化对象已经存在于场景中，它将把数据加载到这个实例中，而不是创建一个新的实例
- 如果类的字段是 UnityEngine.Object，（比如Component, ScriptableObject, Texture2D等等）将通过引用保存
  - 如果场景中不存在实例，则加载时不会创建新的实例，并且该字段设置为空。
  - 如果希望按照值保存它们，则需要使用单独的ES3保存它们。保存后调用必须在加载引用它们的任何内容之前加载它们

3. GameObjects和Prefabs

关于无代码自动保存和加载GameObjects的信息，详情参阅 Auto Save guide

3.1 GameObjects保存与加载

如果需要保存和加载GameObjects，场景中必须要有 **Easy Save 3 Manager** 。如果要将它添加到场景中，需要在Assets中右键选择 *Easy Save 3 > Add Manager to Scene*。

完成上一步操作后，可以像其他任何对象一样加载和保存GameObjects

```
// 保存GameObject.
ES3.Save("myGameObject", go);

// 加载游戏对象，将自动加载到现有的游戏对象
// 如果对象不存在，则会创建一个新的游戏对象
ES3.Load("myGameObject", defaultGo);
```

将保存和加载GameObject的以下内容：

- *layer*，*tag*，*name* 和 *hideFlags*
- 本机支持的类型列表中的组件，或使用ES#Type手动支持的组件
- 以上所有内容适用于GameObject的每一个子物体

3.2 Prefabs的保存与加载

要保存预制件实例，需要进行以下操作：

- 右键单击Prefab，选择 *Easy Save 3 > Enable Easy Save for Prefab*
- 使用 ES3.Save 保存实例，如果有多个实例，则需要使用实例数组
- 加载使用 ES3.Load 或者 ES3.LoadInto
- 如果场景中不存在实例，则会创建Prefab的实例

4. Collections

还可以保存和加载 Arrays, Lists, Dictionaries, Queues, Stacks 和 HashSets 和保存加载其他任何数据类型的方式相同

```
// Arrays
ES3.Save("myArray", myArray);
myArray = ES3.Load("myArray", defaultValue);

// List
ES3.Save("myList", myList);
myList = ES3.Load("myList", defaultValue);

// Dictionary
ES3.Save("myDictionary", myDictionary);
myDictionary = ES3.Load("myDictionary", defaultValue);

// 2DArray
ES3.Save("my2DArray", my2DArray);
my2DArray = ES3.Load("my2DArray", defaultValue);

// Queue
ES3.Save("myQueue", myQueue);
myQueue = ES3.Load("myQueue", defaultValue);

// HashSet
ES3.Save("myHashSet", myHashSet);
myHashSet = ES3.Load("myHashSet", defaultValue);

// Stack
ES3.Save("myStack", myStack);
```

```
myStack = ES3.Load("myStack", defaultValue);
```

5. 何时保存与加载

对于大多数项目，在Start()中加载，保存在移动设备的 OnApplicationQuit(), 或者 OnApplicationPause(true)

可以创建Save()和Load()方法，并把它们赋给保存和加载按钮

6. 更改文件名或路径

filePath 参数允许你指定数据的保存位置，如果文件或文件夹不存在，则自动创建该文件或文件夹

该参数可以是文件名、相对路径或绝对路径，有关详细信息，请参阅路径与位置指南 Paths and Locations

```
// 将值保存到默认位置的文件中
ES3.Save("myKey", myValue, "myFile.es3");

// 使用相对路径将文件保存到默认保存位置的文件夹中
ES3.Save("myKey", myValue, "myFolder/myFile.es3");

// 将值保存到绝对路径的文件中
ES3.Save("myKey", myValue, "C:/Users/User/Documents/myFile.es3);
```

7. Deleting 删除操作

ES3.DeleteKey 删除一个键

```
ES3.DeleteKey("myKey", "myFolder/myFile.es3");
```

ES3.DeleteFile 删除文件

```
ES3.DeleteFile("myFolder/myFile.es3");
```

ES3.DeleteDirectory 删除一个目录及其包含的所有文件

```
ES3.DeleteDirectory("myFolder/");
```

8. Changing settings 更改设置

更改默认设置需要通过顶部菜单 *Window > Easy Save 3 > Settings*

通过提供ES3Settings对象作为参数，在运行时启用设置，有关详情，请参阅 ES3Settings

Enable encryption 启用加密

```
var settings = new ES3Settings(ES3.EncryptionType.AES, "myPassword");

ES3.Save("key", data, settings);
ES3.Load("key", data, settings);
```

Enable compression 启用压缩

```
var settings = new ES3Settings(ES3.CompressionType.Gzip);

ES3.Save("key", data, settings);
ES3.Load("key", data, settings);
```

9. 可以保存和加载的内容

查看支持的类型，详情参阅 Supported Types

10. Error handling 错误处理

与所有代码一样，建议处理错误。使用 try/catch 块以捕获异常，并在必要时向用户发送适当得到消息。比如：

```
try
{
    ES3.Save("key", 123);
    ES3.Save("key2", 456);
}
catch(System.IO.IOException)
{
    DisplayMessageToUser("The file is open elsewhere or there was not enough storage space");
}
catch(System.Security.SecurityException)
{
    DisplayMessageToUser("You do not have the required permissions");
}
```

有关需要处理错误类型的更多信息，请参阅 Error Handling

11. Further Reading 进一步阅读

- Guides 向导
- Feature Overview 功能概述
- Scripting API 脚本API
- Saving and Loading GameObjects and Prefabs 保存加载GameObject和预制件