

# EUResKit

## 框架的入口面板

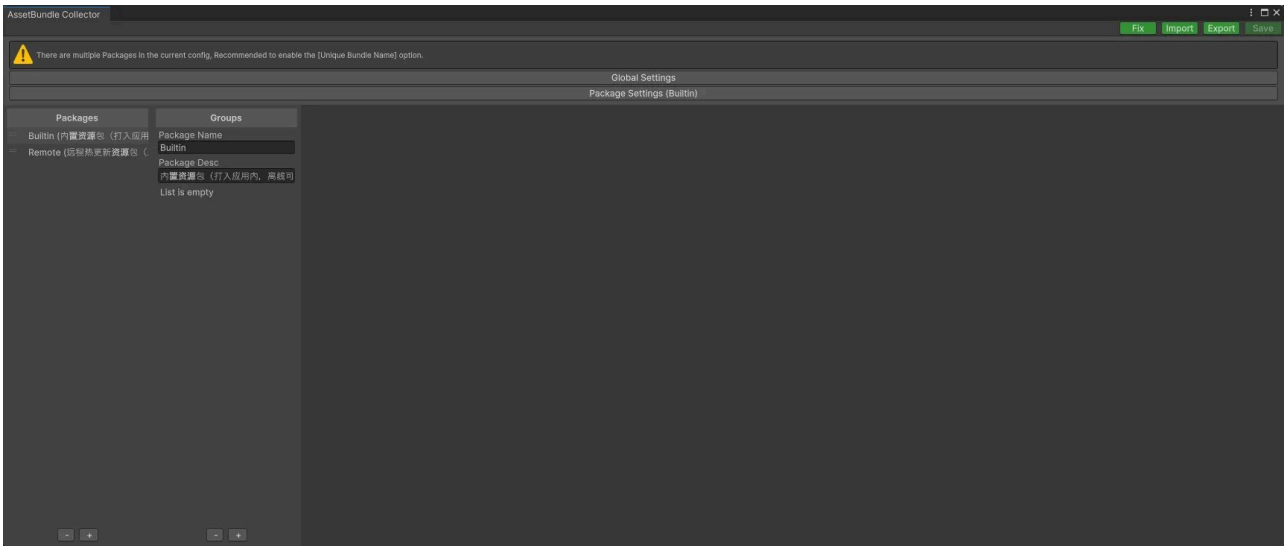
- 资源配置
- 代码生成
- 模块管理工具

## 资源配置

### EURes的SO管理



AssetBundleCollectorSetting 为YooAssets自身的资源收集器



EUResServerConfig 为 代码读取的服务器配置 如果资源为远程加载的情况下进行配置



YooAssetSettings 为 YooAsset配置打包情况的处理

YooAsset 设置

YooAsset 文件夹名称

yoo

资源清单前缀

!

YooAsset 文件夹名称用于缓存和资源目录，清单前缀用于多包配置

EUResKitPackageConfig 每个包初始化时的加载方式的配置 回字节从 AssetBundleCollectorSetting中获取包数据

只需要从AssetBundleCollector同步即可

Package 运行配置（仅配置模式，不可添加/删除）

!

配置说明：

- 本界面仅用于配置 Package 的运行参数
- Package 列表完全由 AssetBundleCollector 管理
- 不支持手动添加、删除或重命名 Package
- 可配置项：运行模式（PlayMode）、默认包设置

数据管理

从 AssetBundleCollector 同步

✓ 验证数据一致性

清理重复数据

Package 1

Package 名称

Builtin

运行模式

Editor Simulate Mode

是否为默认包

☒

包描述

内置资源包（打入应用内，离线可用）

Package 2

Package 名称

Remote

运行模式

Editor Simulate Mode

是否为默认包

☐

包描述

远程热更新资源包（从服务器下载，支持热更新）

验证配置

资源目录关联 创建默认目录结构

## 资源目录状态

### 目录结构

- EUResources/Builtin/ ✓
- EUResources/Excluded/ ✓ (不打包)
- EUResources/Remote/ ✓

### Collector Packages

- Builtin ✓
- Remote ✓
- Excluded (不需要配置)

说明:



- 创建标准目录结构: Builtin / Excluded / Remote
- 在 YooAsset Collector 中创建 Builtin 和 Remote 两个 Package
- Excluded 仅作为本地目录, 不参与打包
- Package 创建后, 请在 YooAsset Collector 中手动添加 Group 和 Collector



✓ 资源目录结构已完整创建

打开 EUResources 目录

重新同步配置

## 代码生成

## 代码生成

生成资源管理代码和开发工具

### UI Prefab 和脚本

EUResKitUserOpePopUp.cs: ✓ 已生成

EUResKitUserOpePopUp.prefab: ✓ 已生成



▲ 业务脚本：用户可自定义 UI 交互逻辑，请勿覆盖！  
Prefab：位于 Resources/EUResKitUI/ 目录

定位到脚本

定位到 Prefab

### EUResKit 分部类 (Partial Class)

EUResKit.Generated.cs: ✓ 已生成

EUResKit.cs: ✓ 已生成



分部类说明：

- EUResKit.Generated.cs - 自动生成的基础工具类（可重新生成）
- EUResKit.cs - 用户编辑的业务逻辑类（请勿覆盖）
- 两个文件作为 partial class 相互引用，必须同时存在

定位到 Generated

定位到用户脚本

重新生成 Generated 部分



提示：一键生成、删除文件、刷新命名空间等功能已移至【模块管理工具】面板

## 1. 用户操作界面的交互代码

生成EUResKitUserOpePopUp.cs.sbn 文件的代码 以及对应的prefab(修改为游戏内对应的游戏样式) 用于管理资源更新时，用户的操作和通知

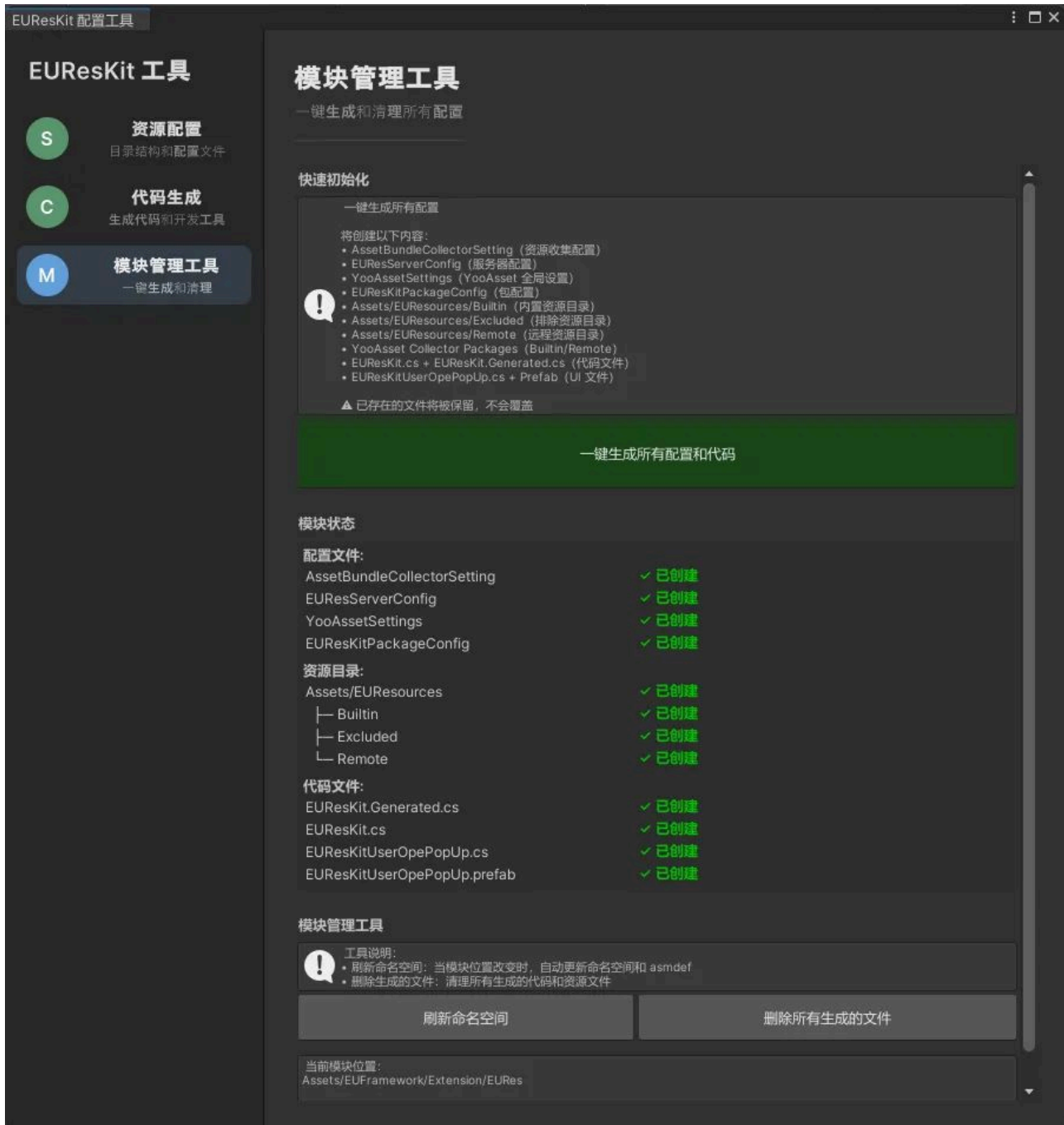
## 2. EUResKit代码。分布类

固定生成的部分DefaultResKit.Generated，只提供InitPackageResAsync GetPackage SetDefaultPackage IsInitialized--初始化，获取包，设置默认包，判断包是否初始化

DefaultResKit.cs，包含默认的UI交互部分的逻辑和一个初始化所有包的入口。用户可以重写，处理新的资源部分如LoadSprite之类的接口也可以写在这里。EUResKit的一个处理思路就是只负责包的初始化和获取包，至于Handle让各自的Kit或者在胶水层处理，如EUUIKit的内部处理

## 模块管理工具

一次性生成/删除 所有的配置和代码部分，刷新程序集引用更改后的命名空间设置。



## 快速入门

引入框架之后，在模块管理中点击一键生成所有配置和代码

处理好YooAssetCollector，设置好包的加载方式

代码中，游戏初始化处，如下调用资源包的初始化

C#

```
1 using EUFramework.Extension.EURes; // 根据实际命名空间修改
2 using Cysharp.Threading.Tasks;
3 using UnityEngine;
```

```

4
5 public class GameLauncher : MonoBehaviour
6 {
7     private async void Start()
8     {
9         // 可选：设置下载进度回调（用于自定义 Loading 条）
10        EResKit.SetDownloadProgressCallback((packageName, totalCount,
currentCount, totalBytes, currentBytes) =>
11        {
12            float progress = (float)currentBytes / totalBytes;
13            Debug.Log($"[{packageName}] 下载进度: {progress * 100:F1}%");
14        });
15
16        // 初始化所有资源包
17        // 如果检测到更新，会自动弹出内置 UI 提示用户下载
18        bool success = await EResKit.InitializeAllPackagesAsync();
19
20        if (success)
21        {
22            Debug.Log("资源初始化成功");
23            StartGame();
24        }
25        else
26        {
27            Debug.LogError("资源初始化失败");
28        }
29    }
30 }

```

如下调用资源的加载和句柄的释放

C#

```

1 using YooAsset;
2
3 // 异步加载预制体
4 var handle = EResKit.GetPackage().LoadAssetAsync<GameObject>
("Assets/Prefabs/Player.prefab");
5 await handle.ToUniTask();
6
7 if (handle.Status == EOperationStatus.Succeed)
8 {
9     GameObject player = handle.AssetObject as GameObject;
10    Instantiate(player);
11 }
12

```

```
13 // 使用完毕释放资源  
14 handle.Release();
```