

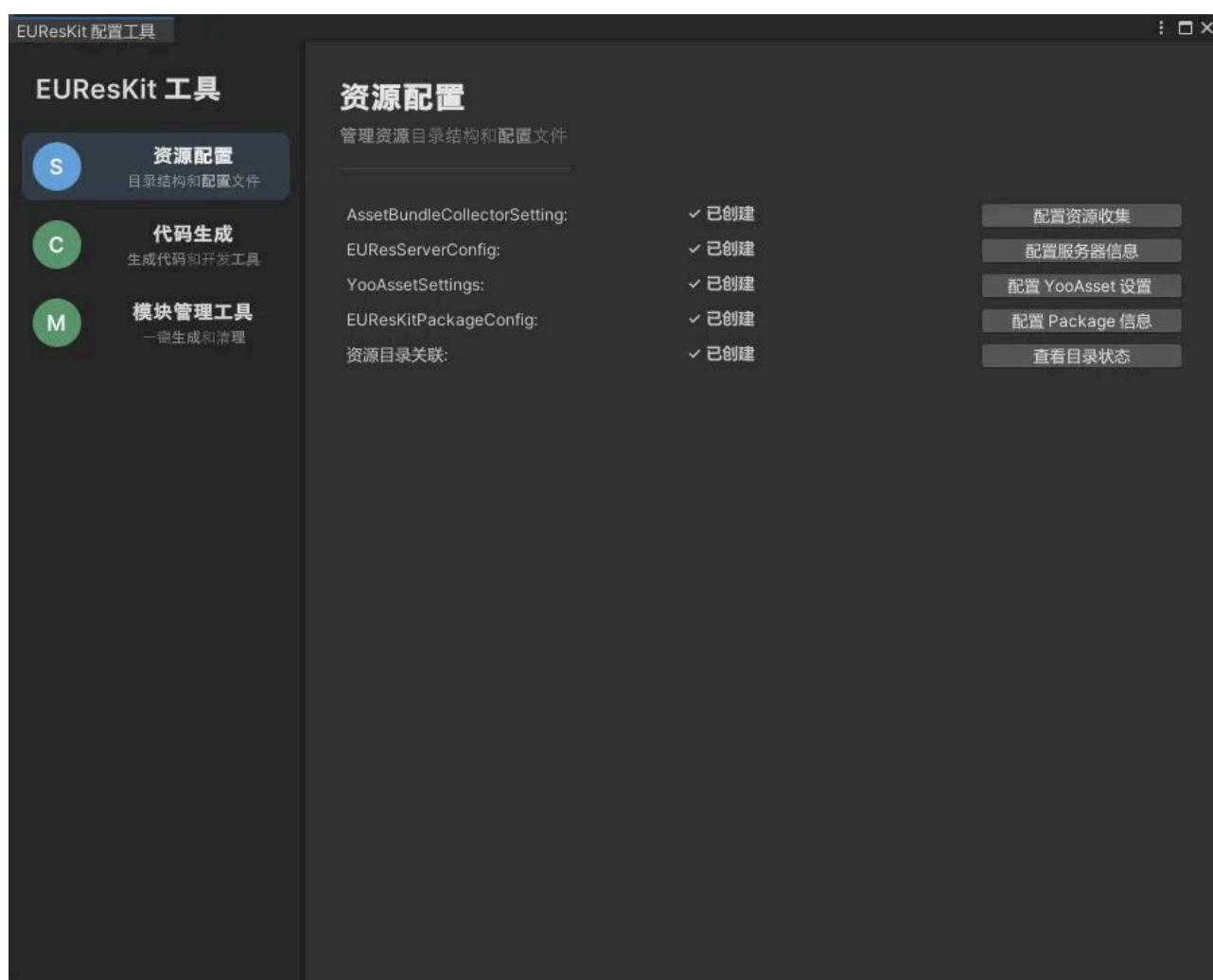
# EUResKit

## 框架的入口面板

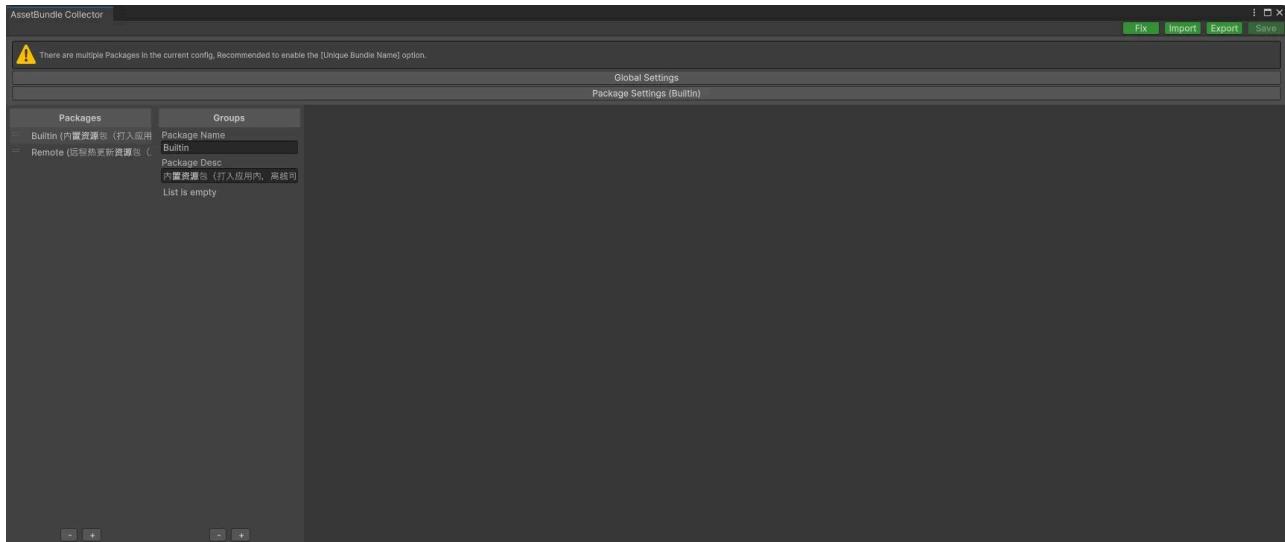
- 资源配置
- 代码生成
- 模块管理工具

### 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

A screenshot of the Resource Configuration interface. The title is "资源配置" (Resource Configuration). Below it is a subtitle: "管理资源目录结构和配置文件" (Manage resource directory structure and configuration files). The main content area has several sections: 1. "AssetBundleCollectorSetting": Status is "✓ 已创建" (Created), with a button "配置资源收集" (Configure Resource Collection). 2. "EUResServerConfig": Status is "✓ 已创建" (Created), with a button "收起配置" (Collapse Configuration). 3. "YooAssetSettings": Status is "✓ 已创建" (Created), with a button "配置 YooAsset 设置" (Configure YooAsset Settings). 4. "EUResKitPackageConfig": Status is "✓ 已创建" (Created), with a button "配置 Package 信息" (Configure Package Information). 5. "资源目录关联": Status is "✓ 已创建" (Created), with a button "查看目录状态" (View Directory Status). Below these sections is a "资源服务器配置" (Resource Server Configuration) section with the following fields: 1. 协议类型 (Protocol Type): HTTP (selected). 2. 服务器地址 (Server Address): 127.0.0.1. 3. 端口号 (Port Number): A slider set to 80. 4. 应用版本 (App Version): 1.0.0. At the bottom left is a note with an exclamation mark: "完整服务器地址: http://127.0.0.1:80" (Full server address: http://127.0.0.1:80).

YooAssetSettings 为 YooAsset配置打包情况的处理



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



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



## 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        EUResKit.SetDownloadProgressCallback((packageName, totalCount,
11        currentCount, totalBytes, currentBytes) =>
12        {
13            float progress = (float)currentBytes / totalBytes;
14            Debug.Log($"[{packageName}] 下载进度: {progress * 100:F1}%");
15        });
16
17        // 初始化所有资源包
18        // 如果检测到更新，会自动弹出内置 UI 提示用户下载
19        bool success = await EUResKit.InitializeAllPackagesAsync();
20
21        if (success)
22        {
23            Debug.Log("资源初始化成功");
24            StartGame();
25        }
26        else
27        {
28            Debug.LogError("资源初始化失败");
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 = EUResKit.GetPackage().LoadAssetAsync<GameObject>
5     ("Assets/Prefabs/Player.prefab");
6 await handle.ToUniTask();
7
8 if (handle.Status == EOperationStatus.Succeed)
9 {
10     GameObject player = handle.AssetObject as GameObject;
11     Instantiate(player);
12 }

```

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