UWA
MAKE IT SIMPLE

Android平台代码热更新
(Unity Based)

UWA 优化日 - 上海站

侑虎科技（上海）有限公司

日期：2016.7.31

Outline

- 热更新原理
- 如何热更新
- 后续维护

代码热更新

- Lua Based: uLua等
- Java Based (Android Only)
- C# Based

Scripts

- C#
- Unity Script

Resources

- .unity
- *.prefab*
- .asset
- AssetBundle

Script & Resources

- 绑定
- 序列化与反序列化

资源和脚本绑定示例

UwaTest.cs: 8e11ef1010cb643fa9473476ebb280ed

```
public class UwaTest : MonoBehaviour {

    public int temperature;
    public bool isHot;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
    }
}
```

资源和脚本绑定示例

```
--- !u!114 &11438854
MonoBehaviour:
  m_ObjectHideFlags: 1
  m_PrefabParentObject: {fileID: 0}
  m_PrefabInternal: {fileID: 100100000}
  m_GameObject: {fileID: 149300}
  m_Enabled: 1
  m_EditorHideFlags: 0
  m_Script: {fileID: 11500000, guid: 8e11ef1010cb643fa9473476ebb280ed, type: 3}
  m_Name:
  m_EditorClassIdentifier:
  temperature: 40
  isHot: 1
```

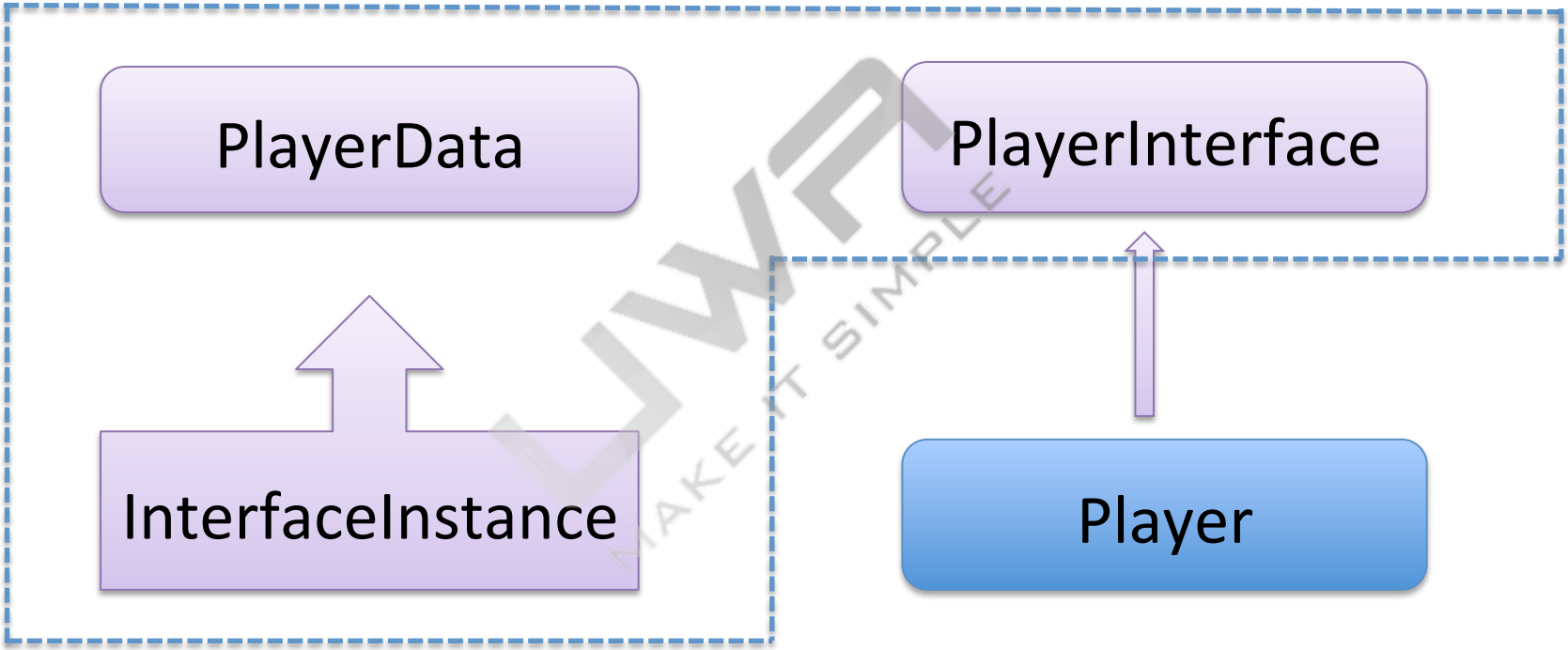
C# based

- **分离脚本与资源之间的绑定**
- 脚本的数据部分和资源绑定
- 逻辑部分动态加载

如何分离

- 接口与实现分离
- 数据和实现分离

接口与实现分离



接口与实现分离

- 具体实现

接口与实现分离

```
public interface PlayerInterface {
    void Update();
}
```

接口与实现分离

```
public class PlayerData : MonoBehaviour {
    public string playerName  = string.Empty;
    public int    playerLevel = 0;

    void Awake() {
        player = (PlayerInterface)LogicAssembly.instance.CreateInstance("Player", this);
        if (player != null) {
            player.Awake();
        }
    }

    void Update() {
        if (player != null) {
            player.Update();
        }
    }
    public PlayerInterface player {
        get;
        private set;
    }
}
```

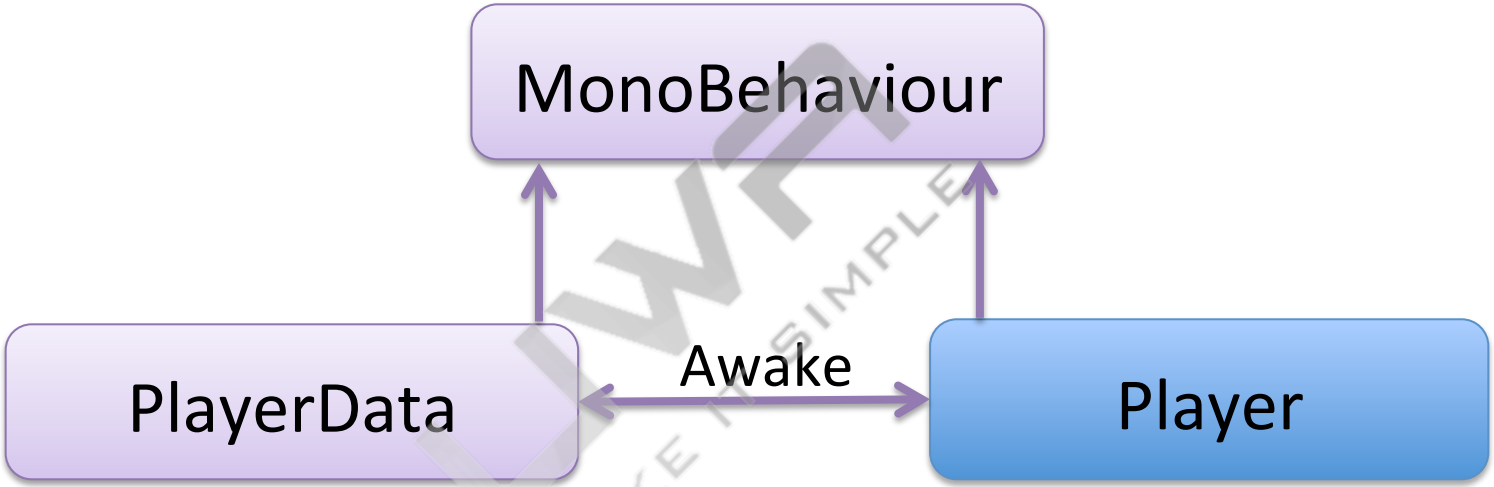
接口与实现分离

```
public class Player : PlayerInterface {
    private PlayerData m_data = null;

    public Player(PlayerData data) {
        m_data = data;
    }

    // Update is called once per frame
    public void Update() {
        // Update
    }
}
```

数据与实现分离



代码中的数据

- Mono: Public, SerializeField
- C#的序列化: [System.Serializable]
- Scriptable

分离Mono

- 删除不能系列化的成员变量
- 删除所有的方法
- 创建数据类对应的逻辑类
- 在数据类的Awake方法中Add逻辑类
- 在逻辑类的Awake方法中初始化

数据与实现分离

- 具体实现

示例(分离前)

```
public class UwaTest : MonoBehaviour {

    public int temperature;
    public bool isHot;
    private int pm25

    void LogicFunction() {
        // logic code
    }
}
```

示例(分离后)

```
public class UwaTestData : MonoBehaviour
{
    public int temperature;
    public bool isHot;

    void Awake () {
        // AddComponent<UwaTest>();
    }
}
```

```
public class UwaTest : MonoBehaviour
{
    public int temperature;
    public bool isHot;
    private int pm25;

    void LogicFunction() {
        // logic code
    }

    void Awake() {
        // copy field from UwaTestData
    }
}
```

分离Serializable & Scriptable

- 有逻辑方法：Wrapper
- 没有逻辑方法

示例(分离前)

```
[System.Serializable]
public class UwaSerializable
{
    public int temperature;
    public bool isHot;

    public void LogicFunctions() {
        // logic code
    }
}
```

示例(分离后)

```
[System.Serializable]
public class UwaSerializable
{
    public int temperature;
    public bool isHot;
}
```

示例(分离后)

```
public class UwaSerializableWrapper
{
    public UwaSerializable DataHolder { private set; get; }

    public UwaSerializableWrapper(UwaSerializable dataHolder) {
        DataHolder = dataHolder;
    }

    public void LogicFunctions() {
        // logic code
    }
}
```

如何解决依赖冲突

- 逻辑上去绕开
- 将逻辑代码移到数据层
- 反射：C# 或者MonoBehaviour

反射

- Type.GetField().GetValue()
- Type.GetMethod().Invoke()
- Type.GetProperty().GetValue()
- Type.GetMember().GetValue()

跨DLL加载 V.S. 序列化

- ~~Mono~~
- ~~Serializable~~
- ~~Scriptable~~
- 所以...

如何维护更新

- 自动化编译
- 管理好数据目录
- Android可增量更新，同时要兼顾iOS

性能问题

- 反射: Assembly.Load
- AddComponent: 内存&CPU

局限性

- IL2CPP
- iOS
- 新增脚本

如何进行代码保护

- 代码混淆
- 修改mono.so

基于此更新的方案的保护

- 将逻辑代码打包成Dll后，对其加密
- 用Android的plugin进行揭秘
- 直接通过内存加载Assembly.Load(Byte[])

使用场景

- 项目初期
- 项目后期

www.uwa4d.com