unity存储

步骤是:1，创建/打开一个文件流，2，BinaryFormatter序列化一 个Save对象， 3，写入硬盘中 4，最后关闭文件流

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;//必须引用这个命名空间才能使用BinaryFormatter

public class GameSaveManager : MonoBehaviour
{
public MyBag myBagManager;


public void SaveGame()
{
   Debug.Log(Application.persistentDataPath);
   if (!Directory.Exists(Application.persistentDataPath + "/game_SaveDate"))//创建文件夹
   {
      Directory.CreateDirectory(Application.persistentDataPath + "/game_SaveDate");
   }

   BinaryFormatter formatter = new BinaryFormatter();//进行二进制转换。
   1，FileStream file = File.Create(Application.persistentDataPath + "/game_SaveDate/myBag.txt");

   2，var json = JsonUtility.ToJson(myBagManager);//将数据转换为string类型，

   3，formatter.Serialize(file, json);//json为要转换的对象
  4，  file.Close();
}
步骤是:创建/打开一个文件流，BinaryFormatter序列化一 个Save对象， 写入硬盘中
最后关闭文件流
public void LoadGame()
{
   BinaryFormatter bf = new BinaryFormatter();
   if(File.Exists(Application.persistentDataPath + "/game_SaveDate/myBag.txt"))
   //加载文件夹
   {
      FileStream file = File.Open(Application.persistentDataPath + "/game_SaveDate/myBag.txt", FileMode.Open);
      JsonUtility.FromJsonOverwrite((string)bf.Deserialize(file), myBagManager);
```

```

        file.Close();
    }
}
```

}

`udp.json` 文件保存在：

`Application.PersistentDataPath + '/Unity' + /Application.CloudProjectID + /udp/udp.json`

其中：

- `Application.PersistentDataPath` 是游戏的持久数据路径
- `Application.CloudProjectID` 是游戏的 Unity 项目 ID

以下是一个 `udp.json` 文件路径示例：

`path/storage/emulated/0/Android/data/com.mystudio.mygame/files/Unity/c83d2de2-de74-4b75-83fc-ade948bda064/udp/udp.json`

其中：

- `Application.PersistentDataPath` = `path/storage/emulated/0/Android/data/com.mystudio.mygame`
- `Application.CloudProjectID` = `c83d2de2-de74-4b75-83fc-ade948bda064`

([(**PlayerPrefs)** 是player preferences的简写)
[playerPrefs) 是Unity内置的一个静态类，可以用于[存储读取] -些简单的数据类型，是一个特殊的[Caching System缓存系统]用来储存读取游戏中的[简单设置和数据]

其中**PlayerPrefs.SetInt(Key，value)**方法用做[保存数据]
即将value的值，以键key的方式储存起来，可以理解为一把钥匙(键key)储存一个数据(值value)

[PlayerPrefs.GetInt(Key)方法用做 [读取数据]通过键名来读取数据(如果这个键存在的话)

序列化是将[对象的状态信息]转换为[Unity可以存储的形式] 的自动化处理过程
(即:序列化是一个过程，以某种存储形式使自定义对象得以保存)

简单来说，把[对象]转化为[可传输的字节序列]过程称为序列化

可以将对象转换为字节序列后，存储在数据库，内存或者文件中

[**File类]**，[**FileStream类**]和(**BinaryFormatter类**]　使用File类的前提条件是使用[System.Io]命名空间

(即文件类，文件流类，二进制格式化类)

当你想新建，剪切，复制一个文件的时候，声明的是File类的对象(实例)，当你想将文件读取或者写入时候，就要使用FileStream类

BinaryFormatte序列化(保存的过程) :将对象转化成二进制
BinaryFormatter.Serialize(Stream Obiect)方法: 将对象序列化到给定的流

BinaryFormatter.Serialize(Stream, Object)方法: 将对象序列化到给定的流这是(保存过程 的核心方法

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

public class MenuManager : MonoBehaviour
{
public GameObject gameMenuImage;
public Player play;
public Toggle BGMtoggle;
private AudioSource audioSource;
// Start is called before the first frame update
void Start()
{
gameMenuImage.gameObject.SetActive(false);
audioSource = GetComponent<AudioSource>();
}


// Update is called once per frame
void Update()
{
    if (Input.GetKeyDown(KeyCode.M))
    {
        if (GameManager.instance.isPause)
        {
            Refuse();//恢复游戏
        }
        else
        {
            Pasue();//暂停游戏
        }
    }
   BGMManager();
}
public void  Pasue()
```

```
{
    gameMenuImage.gameObject.SetActive(true);//
    Time.timeScale = 0f;
    GameManager.instance.isPause = true;
}
public void Refuse()
{
    gameMenuImage.gameObject.SetActive(false);
    Time.timeScale = 1.0f;
    GameManager.instance.isPause = false;
}
public void BGMtoggleButten()//背景音乐
{
    if (BGMtoggle.isOn)
    {
        PlayerPrefs.SetInt("BGM", 1);
        Debug.Log(PlayerPrefs.GetInt("BGM"));
    }
    else
    {
        PlayerPrefs.SetInt("BGM", 0);
        Debug.Log(PlayerPrefs.GetInt("Bgm"));
    }
}
private void BGMManager()
{
    if (PlayerPrefs.GetInt("BGM") == 1)
    {
        BGMtoggle.isOn = true;
        audioSource.enabled = true;

    }
    else if (PlayerPrefs.GetInt("BGM") == 0)
    {
        BGMtoggle.isOn = false;
        audioSource.enabled = false;
    }
}

public void SaveButten()
{
    SaveByPlayPre();
}
public void LoadButten()
{
    LoadBYPlayerPfe();//加载数据
    Refuse();//恢复游戏
}
```

```
private void SaveByPlayPre()
{
    //存储数据
    PlayerPrefs.SetInt("score", play.score);
    PlayerPrefs.SetInt("HP", play.currentHp);
}
private void LoadBYPlayerPfe()
{
    if(PlayerPrefs.HasKey("score"))
    play.score =PlayerPrefs.GetInt("score");//加载数据
    if (PlayerPrefs.HasKey("HP"))
        play.currentHp = PlayerPrefs.GetInt("HP");
}
```

}

BinaryFormatte序列化(保存的过程) :将对象转化成二进制

BinaryFormatter.Serialize(Stream Obiect)方法: 将对象序列化到给定的流

```
}

private void SaveByPlayerPrefs()
{
    //THE KEY NAME IS CUSTOMIZED AND CHOOSE ONE EASY TO READ FOR OTHERS
    PlayerPrefs.SetInt("Coins", GameManager.instance.coins);//THE KEY is the "Coins"
    PlayerPrefs.SetInt("Diamonds", GameManager.instance.diamonds);//THE VALUE IS ONE VARIABLE

    //SAVE the player gameobject position
    PlayerPrefs.SetFloat("PlayerPosX", player.transform.position.x);
    PlayerPrefs.SetFloat("PlayerPosY", player.transform.position.y);
    Debug.Log("SAVE THE DATA");
    StartCoroutine(DisplayHintCo("SAVED"));
}

private void LoadByPlayerPrefs()
{
    if (PlayerPrefs.HasKey("Coins") && PlayerPrefs.HasKey("Diamonds") && PlayerPrefs.HasKey("PlayerPosX") && PlayerPrefs.HasKey("PlayerPos
    {
        GameManager.instance.coins = PlayerPrefs.GetInt("Coins");//Assign the saving data to our variable
        GameManager.instance.diamonds = PlayerPrefs.GetInt("Diamonds");
        player.playerPosX = PlayerPrefs.GetFloat("PlayerPosX");
        player.playerPosY = PlayerPrefs.GetFloat("PlayerPosY");
        StartCoroutine(DisplayHintCo("LOADED"));
        Debug.Log("LOAD THE DATA");
    }
    else
    {
        StartCoroutine(DisplayHintCo("NOT FOUND"));
    }
```

private void SaveBySerialization()

{

Save save = createSaveGame0bject():

BinaryFormatter bf = new BinaryFormatter();

FiLestream filestream = FiLe.CreatelAppllcation persistentDataPath + "/Data. text");//新建文件流

bf.Serialize(fileStream, save);//序列化Save对象

fileStream.Close();//关闭文件流

}这就是序列化(保存)全过程

(创建一个二进制格式化对象-新建文件流->序列化Save对象->关闭文件流)

```


 private void SaveBySerialization()
{
Save save = createSaveGame0bject():
BinaryFormatter bf = new BinaryFormatter();
FiLestream filestream = FiLe.CreatelApplcation persistentDataPath + "/Data. text");//新建文件流
bf.Serialize(fileStream, save);//序列化Save对象
fileStream.Close();//关闭文件流
}这就是序列化(保存)全过程
(创建一个二进制格式化对象-新建文件流->序列化Save对象->关闭文件流)


 private void LoadByDeSerialization()
{
if (File.Exists(Application.persistentDataPath + "/Data. text"))
    //LOAD THE GAME
     BinaryFormatter bf = new BinaryFormatter();
    FiLeStream fiLeStream = FiLe.Open(AppLicat ion.persistentDataPath + "/Data. text",
FileMode.open);
    Save save = bf.Deserialize(fileStream) as Save;//You I
    fileStream.Close();
    else
    {
        //REPORT THE ERROR
        Debug.Log("NOT FOUND THIS FILE");
    }
}
```