Server 模块说明

模块内包含 Socket 和 Http 两套通信框架。

目录：

一、Socket 通信说明

类：NetMngr
性质：单例

包含方法：

1、注册接口侦听
方法：public void AddListener(string name, Transmit.Callback call)；
参数：
name：接口名
call：回调函数，定义如下：
public delegate void Callback(Hashtable message);
示范：

```
private void Awake()
{
    NetMngr.GetSingleton().AddListener("TestInterface",TestCallback);
}

private void TestCallback(Hashtable data)
{
    Debug.Log(data["message"].ToString());
}
```

说明：注册接口侦听，当收到接口消息会执行 call 方法并传入接口回调数据

2、移除接口侦听
方法：public void RemoveListener(string name);
参数：
name：接口名
示范：

```
NetMngr.GetSingleton().RemoveListener("TestInterface");
```

说明：移除接口的侦听

3、发送接口消息
方法：public bool Send(string strMethod, object[] args,byte[] data=null);
参数：
strMethod：接口名
args：接口数据。当传输多个数据，按顺序排列。
data：接口数据。为 byte[]类型，当需要传输文件时转换为 byte[]传输，如传输音频文件。
示范：

```
NetMngr.GetSingleton().Send("TestInterface", new[] {"参数 1", "参数 2"});
```

（基础类型传输）
NetMngr.GetSingleton().Send("TestInterface", new object[] {}, bytes);(byte[]传输)
说明：发送消息，可以传输基础类型和 byte[]类型

4、获取网络延迟
方法：public float GetLatency();
参数：无
示范：
NetMngr.GetSingleton().GetLatency();
说明：获取当前网络延迟，单位为毫秒，在第一个心跳返回之前，返回为 0

5、清除消息
方法：public void ClearMessage();
参数：无
示范：
NetMngr.GetSingleton().ClearMessage();
说明：清除消息队列中的消息

5、重新连接 Socket
方法：public void ReInit();
参数：无
示范：
NetMngr.GetSingleton().ReInit();
说明：重新连接 Socket，可能会不太稳定，建议不要轻易使用

6、网络是否连接
方法：public bool IsConnect();
参数：无
示范：NetMngr.GetSingleton().IsConnect();
说明：获取当前网络是否连接，也可以直接访问 Constants.socketConnected

7、获取网络类别
方法：public int GetNetType();
参数：无
示范：NetMngr.GetSingleton().GetNetType();
说明：获取当前网络类型，0 为无网络连接，1 为 WIFI 连接，2 为流量连接，-1 为获取失败

包含事件：

1、网络连接成功
定义：public Action OnConnect;
返回参数：无
示范：

```
private void Awake()
{
    NetMngr.GetSingleton().OnConnect += OnNetConnect;
}

private void OnNetConnect()
{
    Debug.Log("服务器连接成功");
}
```

说明：当网络连接成功时触发

2、网络断开
定义：public Action OnNetDown;
返回参数：无
示范：

```
private void Awake()
{
    NetMngr.GetSingleton().OnNetDown += OnNetDown;
}

private void OnNetDown()
{
    Debug.Log("网络连接断开");
}
```

说明：网络连接断开时触发，**注意：当网络重新连接上之前会一直循环触发**

3、发送接口消息
定义：public Action OnSend;
返回参数：无
示范：

```
private void Awake()
{
    NetMngr.GetSingleton().OnSend += OnSendMessage;
}

private void OnSendMessage()
```

```
{
    //禁用交互
    eventSystem.SetActive(false);
}
```

说明：发送接口消息时触发，可以用于处理禁用交互，显示加载中等操作。

4、收到接口消息

定义：public Action OnReceive;

返回参数：无

示范：

```
private void Awake()
{
    NetMngr.GetSingleton().OnReceive += OnReceiveMessage;
}

private void OnReceiveMessage()
{
    //启用交互
    eventSystem.SetActive(true);
}
```

说明：收到接口消息时触发，可以用于启用交互，隐藏加载中等操作

5、收到心跳消息

定义：public Action<float> OnReceiveTicker;

返回参数：网络延迟，和 GetLatency()方法取到的数值相同

示范：

```
private void Awake()
{
    NetMngr.GetSingleton().OnReceiveTicker += OnReceiveTicker;
}

private void OnReceiveTicker(float latency)
{
    Debug.Log("当前网络延迟为：" + latency);
}
```

说明：当收到心跳消息时触发，心跳频率大概为 5 秒一次，可用于更新网络 ping 值显示

二、Http 通信说明

类：HttpMngr
性质：单例

包含方法：

1、发送 Get 请求
定义：public void Get(string interfaceName, Hashtable getData, Action<Hashtable> callBack = null);
参数：
interfaceName：接口名
getData：接口参数
callBack：接口回调，返回一个 Hashtable 类型的回调数据
示范：

```
private void Awake()
{
    Hashtable data = new Hashtable();
    data["key"] = "value";
    HttpMngr.GetSingleton().Get("testInterface", data, InterfaceCallback);
}

private void InterfaceCallback(Hashtable args)
{
    Debug.Log(args["message"].ToString());
}
```

说明：发送 Get 请求，只需要输入接口名，发送的 url 为 HttpManager 设置的 Url+接口名

2、发送 Get 请求（自带 Url）
定义：public void GetWithUrl(string url, Hashtable getData, Action<Hashtable> callBack = null);
参数：
url：接口地址
getData：接口参数
callBack：接口回调，返回一个 Hashtable 类型的回调数据
示范：

```
private void Awake()
{
    Hashtable data = new Hashtable();
    data["key"] = "value";
    HttpMngr.GetSingleton().GetWithUrl(url, data, InterfaceCallback);
}
```

```
private void InterfaceCallback(Hashtable args)
{
    Debug.Log(args["message"].ToString());
}
```

说明：和 Get 方法相同，区别：不使用 HttpManager 设置的 Url，使用自定义的接口地址

3、发送 Post 请求

定义：public void Post(string interfaceName, Hashtable postData, Action<Hashtable> callBack = null);

参数：

interfaceName：接口名

postData：接口参数

callBack：接口回调，返回一个 Hashtable 类型的回调数据

示范：

```
private void Awake()
{
    Hashtable data = new Hashtable();
    data["key"] = "value";
    HttpMngr.GetSingleton().Post("interfaceName", data, InterfaceCallback);
}

private void InterfaceCallback(Hashtable args)
{
    Debug.Log(args["message"].ToString());
}
```

说明：发送 Post 请求，只需要输入接口名，发送的 url 为 HttpManager 设置的 Url+接口名

4、发送 Post 请求（自带 Url）

定义：public void PostWithUrl(string url, Hashtable postData, Action<Hashtable> callBack = null);

参数：

url：接口地址

postData：接口参数

callBack：接口回调，返回一个 Hashtable 类型的回调数据

示范：

```
private void Awake()
{
    Hashtable data = new Hashtable();
    data["key"] = "value";
```

```
        HttpMngr.GetSingleton().PostWithUrl(url, data, InterfaceCallback);
    }

    private void InterfaceCallback(Hashtable args)
    {
        Debug.Log(args["message"].ToString());
    }
```

说明：和 Post 方法相同，区别：不使用 HttpManager 设置的 Url，使用自定义的接口地址

5、发送二进制的 Post 请求

定义：public void PostBytes(string interfaceName, byte[] postData, Action<Hashtable> callBack = null);

参数：

interfaceName：接口名

postData：接口参数

callBack：接口回调，返回一个 Hashtable 类型的回调数据

示范：

```
    private void Awake()
    {
        byte[] bytes = new byte[1000];
        HttpMngr.GetSingleton().PostBytes("testInterface", bytes, InterfaceCallback);
    }

    private void InterfaceCallback(Hashtable args)
    {
        Debug.Log(args["message"].ToString());
    }
```

说明：用于上传图片、视频到服务器

6、发送二进制的 Post 请求（自带 Url）

定义：public void PostBytesWithUrl(string url, byte[] postData, Action<Hashtable> callBack = null);

参数：

url：接口地址

postData：接口参数

callBack：接口回调，返回一个 Hashtable 类型的回调数据

示范：

```
private void Awake()
    {
        byte[] bytes = new byte[1000];
```

```
        HttpMngr.GetSingleton().PostBytesWithUrl(url,                bytes,
InterfaceCallback);
    }

    private void InterfaceCallback(Hashtable args)
    {
        Debug.Log(args["message"].ToString());
    }
```

说明：和 PostBytesWithUrl 方法相同，区别：不使用 HttpManager 设置的 Url，使用自定义的接口地址