链滴

C# 开发 Window Service 实现 WEB 与客户端之间的交互

作者：hi-asan

原文链接：https://ld246.com/article/1520243373889

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

背景

上一篇文章 **C# 开发一个简单的 Window Service 入门** 简单的写了一个Demo

文章末尾也提及了WEB与客户端之间的障碍。

其实这个问题很简单，只要给他们搭一座沟通的桥梁即可。

沟通 = 通讯。

在浏览器上我们第一想到的通讯协议是**HTTP（超文本传输协议）**、**WebSocket**通讯协议

也是比较常用的协议。

根据具体情况选择不同的办法。

那接下来的事情就很简单了。我们只需要在Window Service里面写一个通讯服务器就OK了。

开始

HTTPService

在上一篇文章Demo的基础上，我们添加一个类。来实现HTTPService。

新建类 **HttpService.cs**

```
public class HttpService
{
  HttpListener listerner = null;
  public void Start(){
    listerner = new HttpListener();
    try
    {
        listerner.AuthenticationSchemes = AuthenticationSchemes.Anonymous;
        listerner.Prefixes.Add("http://127.0.0.1:8888/");
        listerner.Start();
    }
    catch (Exception ex)
    {
      LogHelper.Info("端口监听打开失败：" + ex.Message);
      break;
    }
    int min, max, port;
    ThreadPool.GetMinThreads(out min, out port);
    ThreadPool.GetMaxThreads(out max, out port);
    while (true)
    {
      HttpListenerContext ctx = listerner.GetContext();//等待请求
      ThreadPool.QueueUserWorkItem(new WaitCallback(Callback), ctx);
    }
  }
```

```
  static void Callback(Object o)
  {
    HttpListenerContext ctx = (HttpListenerContext)o;
    Stream stream = ctx.Request.InputStream;
    StreamReader reader = new StreamReader(stream, Encoding.UTF8);
    String body = reader.ReadToEnd();

    ctx.Response.AddHeader("Access-Control-Allow-Origin", "*");//打开跨域权限
    ctx.Response.AddHeader("Access-Control-Allow-Methods", "POST");//限制请求方式

    LogHelper.Info("request body : " + body);

    ctx.Response.StatusCode = 200;//Http返回状态码
    using (StreamWriter writer = new StreamWriter(ctx.Response.OutputStream, Encoding.UTF
))
    {
      writer.Write("success");//简单的输出一个字符串，个人喜欢返回json
      writer.Close();
      ctx.Response.Close();
    }
  }

  public void Stop()
  {
    try
    {
      listerner.Close();
      LogHelper.Info("停止监听9313端口");
    }
    catch (Exception e)
    {
      LogHelper.Info("停止监听9313端口失败" + e.Message);
    }
  }
}
```

OK ,一个简单的HTTP服务器就完成了。端口8888。按照程序逻辑来，只要收到请求就会响应succes，并在日志文件中写入请求参数。

也算是Web和客户端进行交互了。

服务的启动和关闭

我们把启动服务的代码卸载WindowService OnStart事件中。让Window Service一启动。Http服务紧接着启动。

为了不在事件中堵塞，我们另开一个线程进行。

```
  http = new HttpService();//全局实例
  new Thread(http.Start).Start();
```

在OnStop事件中停止Http服务

```
  http.Close();
```

重新编译下安装至系统中，即可调用。

```
$.ajax({
  url : 'http://127.0.0.1:8888',
  type : 'POST',
  timeout: 3000,
  data : {cmd : 'check'},
  dataType : 'json',
  success : function(result){
    console.log(result);
  },
  error : function(XMLHttpRequest,textStatus,errorThrown){
    console.log("请求超时，未检测到服务");
  }
});
```

上面例子简单的完成一个检查插件是否安装或启动的一个小功能。在代码中我们可以根据请求参数的同而做出不同的动作。

一个简单的交互小例子也算是完成了。Demo只是起到一个抛砖引玉的作用。

Window Session 机制

我也只是大概知道一点点。(我是Java程序员ORZ)

Window Service 运行在Window的 Session0层,而GUI是运行在Session0层之上。
他们两是安全隔离的，也就是说Session0层访问不到GUI层。

可以尝试一下在收到请求后。打开C盘目录

```
System.Diagnostics.Process.Start("explorer.exe", "C://");
```

是没有任何反应的。

网上有一段代码是可以解决这个问题的。

解决了这个问题，Web相当于是可以通过这个WindowService完成大部分的电脑常规操作。

前提是你得安装这个服务，并且只能在Window环境下安装。

有空我会整理出来如何解决这个问题，给自己留一个记录。备忘。