重复造轮子---模拟IIs服务器

在写程序的时候，重复造轮子是程序员的一个大忌，很多人对重复造轮子持有反对的态度，但是我觉得这个造轮子的过程，是对于现有的知识的一个深入的探索的过程，虽然我们不可能把轮子造的那么的完善，对于现在有的东西是一个更好的理解和使用。下面我就用几篇文章说下造轮子的过程。

先说第一个问题，当你打开一个网页时，你会看到一个html页面的呈现，当然这是一个完整的Http的请求和响应的过程，无非是对HTML+CSS+JS一个综合的产物，在这个过程中浏览器请求数据的过程中会发出一个有一个格式的字符串（基于http协议生成的http请求报文）,服务器在接收这样的一段字符的时候会对会这个进行解析，拿到相应的信息，然后根据请求响应的文件结构生成响应数据，当然这个数据发送的格式也是基于Http协议的格式生成的响应报文。然后浏览器在接收到这个数据的时候进行转化和解析，最终生成html的界面展现给客户。

在.Net程序中，如果说浏览器充当的是一个客户端的话，则充当服务器的角色就是IIS了，当你在IIS上发布一个网站的时候，浏览器就能够直接访问你的网站，这个过程是浏览器和IIS服务器之间的一个通信，基于这个过程和上面的理解，我们可以试着去写一个IIS服务器。

对于数据的传输，我想应该都听说过socket，当前也可以基于TCP，但这个都不重要，重要的是我们怎么根据浏览器的请求，来解析出他想要的数据？下面我们用socket与浏览器进行统计。

private void btnStar\_Click(object sender, EventArgs e)

{

var socket=new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

var endPoint=new IPEndPoint(IPAddress.Parse("127.0.0.1"),50000);

socket.Bind(endPoint);

socket.Listen(10);

var socketProx = socket.Accept();//接收数据

var bytes=new byte[1024\*1024];

var len = socketProx.Receive(bytes, 0, bytes.Length, SocketFlags.None);

var httpHeader = Encoding.Default.GetString(bytes, 0, len);

this.textBox1.Text = httpHeader;

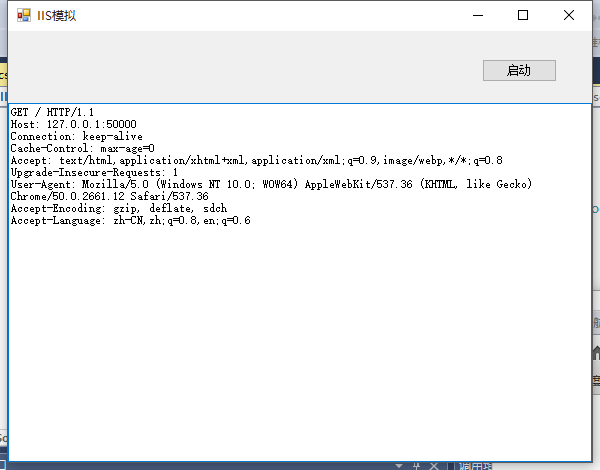
socketProx.Shutdown(SocketShutdown.Both);

socketProx.Close();

socket.Close();

}

当用浏览器访问这个地址和端口的时候，会出现如下的结果：



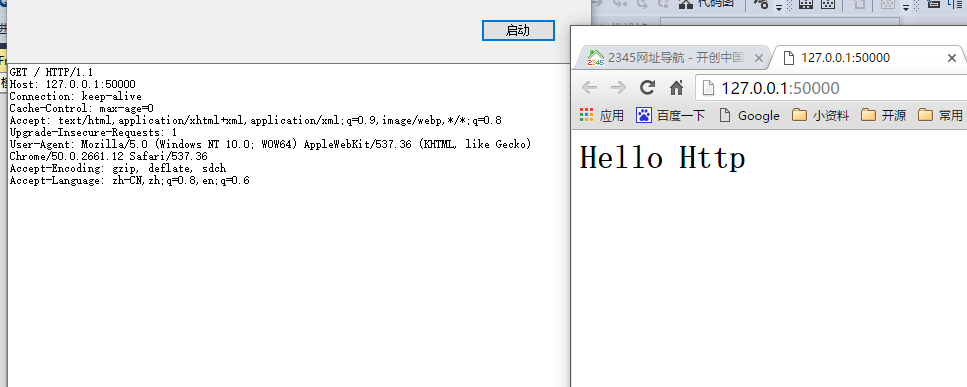
这个就是我们熟悉的请求报文，其实说白了也就是一段有特定格式的文本字符串。看到这里我们尝试会给浏览器发送一串文本试试，在关闭socket之前发送一个文本。

var httpResponse = "<h1>Hello Http</h1>";

var sentBytes = Encoding.Default.GetBytes(httpResponse);

socketProx.Send(sentBytes, 0, sentBytes.Length, SocketFlags.None);

当我产再次用chrome访问的时候：

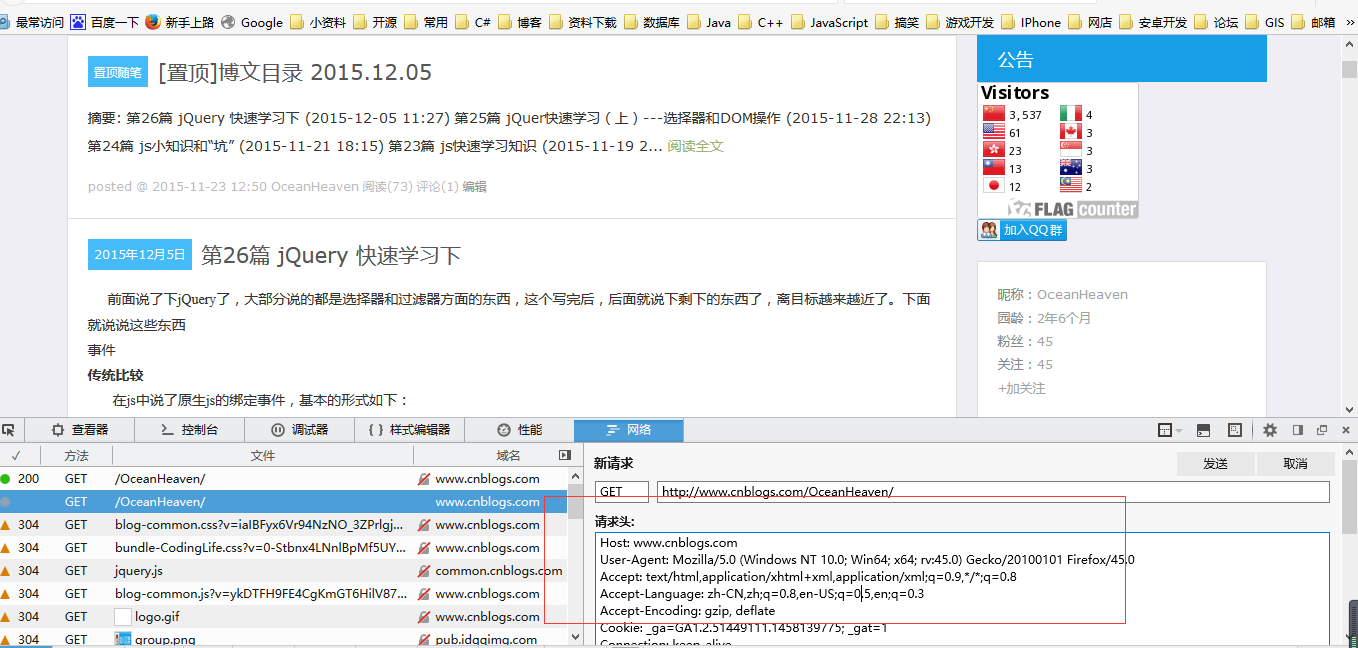


这个已经把我们的文本转义成了html代码处理了。

当我们用firefox去访问时：



我们可以发现这个firefox只是把我的数据当成了一个普通的文本并没有解释成html数据,这个是因为我们没有给浏览器一个响应的头信息，就是说你没有告诉浏览器你发送的内容是什么，所以他们才会用自己默认的方式去处理这个信息。下面我们就把响应的头信息给添加上去，看下效果,头信息的获得，可以随意打开一个网站找下。



我们找到如下的头信息：

这里我们不需要那么多的信息，只需要一部分，把我们得到后的头信息添加到代码里面，在发送内容前把头信息发送出去，

StringBuilder sb = new StringBuilder();

sb.AppendLine("HTTP/1.1 200 OK");

sb.AppendLine("Date:" + DateTime.Now);

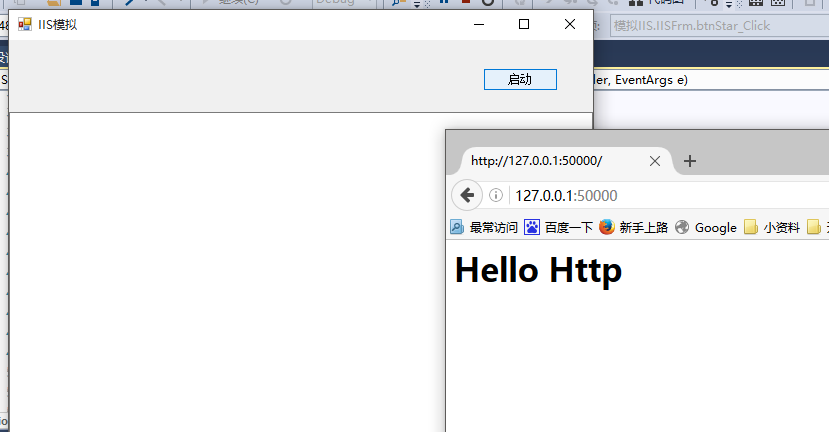
sb.AppendLine("Content-Type:" + "text/html");

sb.AppendLine();//很重要

byte[] headerBytes = Encoding.Default.GetBytes(sb.ToString());

socketProx.Send(headerBytes);

在完成后的我们修改的后的代码运行后的结果如下：



这样我们就写一个简单的IIS服务器，我们可以下内容的信息转成文件的格式存储在一个路径下，根据当前请求url取得这个文件,例如我们请求127.0.0.1：50000/1.html，那就找到1.html，读取文件信息然后发送头信息，然后在把内容发送给浏览器。

下面我们对上面的程序进行改写，改成读取文件信息，改写如下：

private byte[] ResovleHttp(string httpHead)

{

string[] requestLines = httpHead.Split(new string[] { "\r\n" }, StringSplitOptions.None);

var requestMethod = requestLines[0].Split(' ')[0];

var requestUrl = requestLines[0].Split(' ')[1];

var filePath = Application.StartupPath+Path.Combine(requestUrl);

if (File.Exists(filePath))

{

var fileBytes = File.ReadAllBytes(filePath);

return fileBytes;

}

return NotFound();

}

private byte[] NotFound()

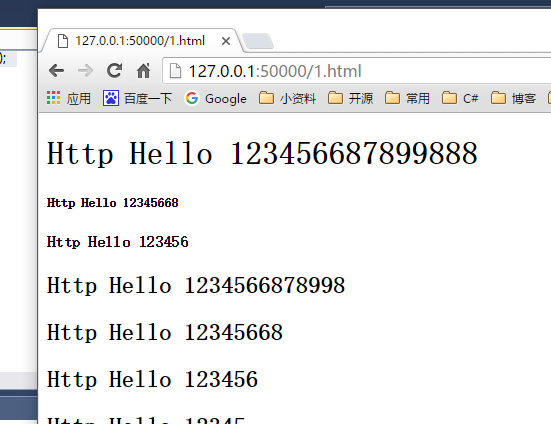
{

var strHtml = "<h1>Not Found404</h1>";

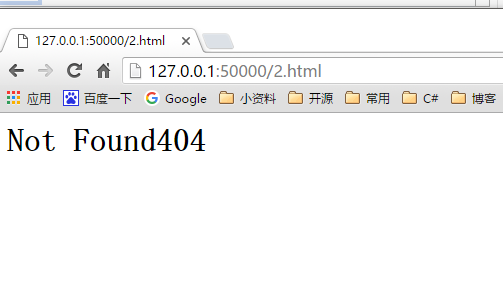
return Encoding.Default.GetBytes(strHtml);

}

当我们访问127.0.0.1：50000/1.html的时候我们能够看到我们html里面内容：



当我们没有找到页面的时候，我添加一个未找到的提示信息。



好了，到现在我们已经把基本的原理已经完成了，下面我们就要对现在的程序进行改造了。下面是我根据现在的框架改写的一个，源码在。。。。。