目录

1 大概思路

- Nginx 集群之 SSL 证书的 WebApi 令牌验证
- Openssl 生成 SSL 证书
- 编写.NET WebApi 的 OnAuthorization 身份验证
- 编写.NET WebApi 的 ActionFilterAttribute 令牌验证
- 编写.NET WebApi 的服务端
- 编写.NET WebApi 的客户端
- 部署 WebApi 到局域网内 3 台 PC 机
- Nginx 集群配置搭建
- 运行结果
- 总结

2 Nginx 集群之 SSL 证书的 WebApi 令牌验证

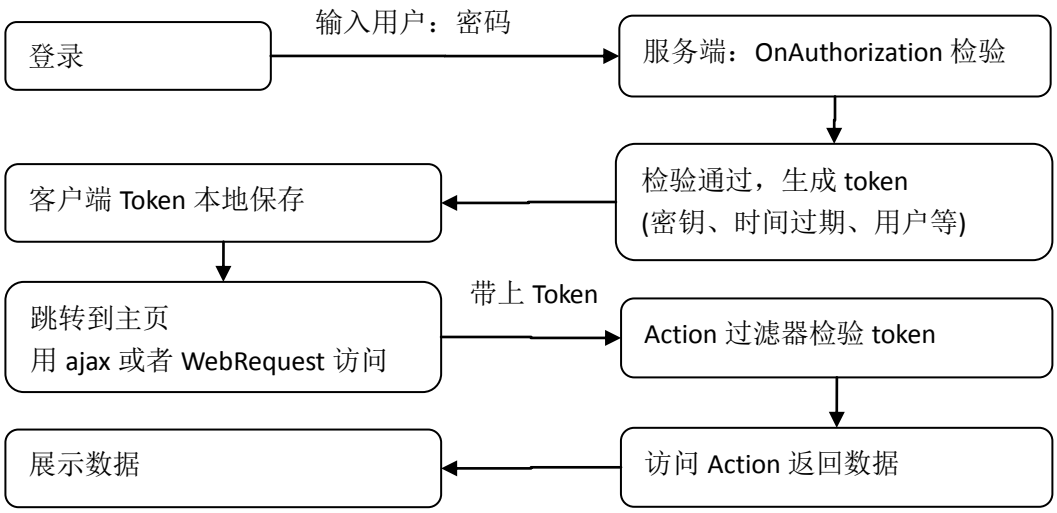
Nginx 在 WebApi 集群，除了 OAUTH 身份验证外，针对移动端的手机、平板电脑等，还经常使用 Token 令牌验证，通过服务器授权发出有效期的 Token，客户端通过此 Token 在当前有效期内，进行访问获取信息数据。

Token 验证在很多方面都广泛应用，举一个实际应用场景：A 客户想通过接收邮件或者短信网址打开一个 URL 的 PDF 报表，但是又不想安装 APP、或者访问我们的系统，连登录都不想登录。这时候，便可以使用一个有效期的 Token，然后结合 URL 发送给用户，过了有效期，当前 URL 就失效。便可以解决用户临时访问的问题。

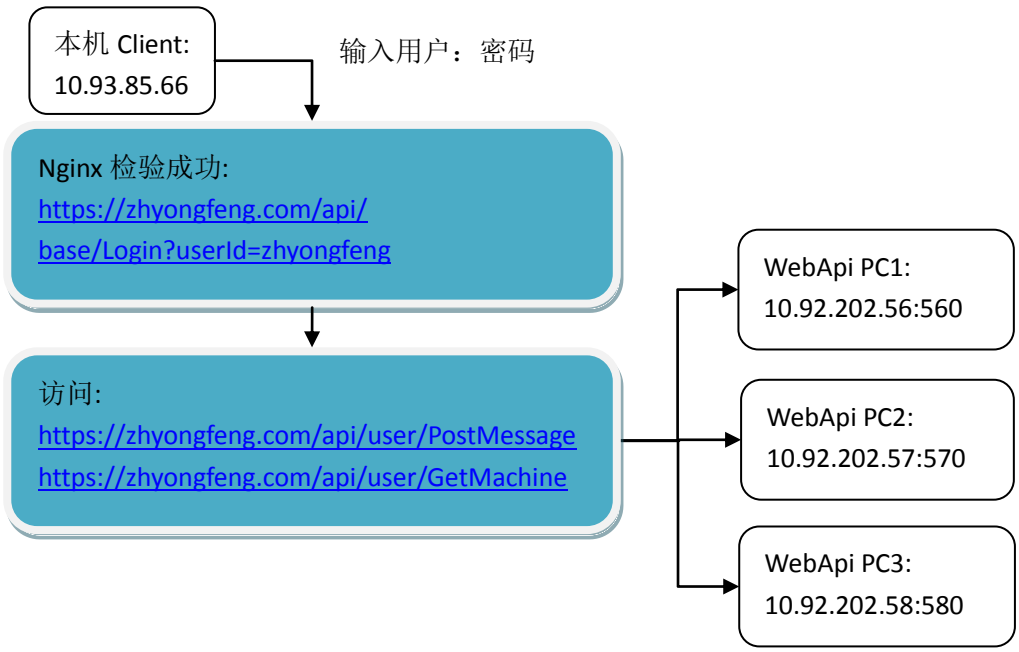
以下是本文讲述的主要结构图：

客户端输入用户名密码服务器，通过了用户名密码验证，其中一台 WebApi 服务器生成一个 Token 并返回 https 的响应。客户端收到 Token 保存在本地，带上 token 发出 ajax 请求、WebRequest 请求，经过 Action 过滤器的检验，访问 Action 并返回数据。

Token 令牌身份验证机制：



Nginx 集群之 SSL 证书的 WebApi 令牌验证，如下图所示：



3 Openssl 生成 SSL 证书

请参照《Nginx 集群之 SSL 证书的 WebApi 微服务》
http://www.cnblogs.com/yongfeng/p/7921905.html

4 编写.NET WebApi 的 OnAuthorization 身份验证

CustomAuthorizeAttribute.cs

```
using System.Web.Http;
using System.Web.Http.Controllers;

namespace SSLWebApi.Controllers
{
    public class CustomAuthorizeAttribute : AuthorizeAttribute
    {
        public override void OnAuthorization(HttpActionContext actionContext)
        {
            //判断用户是否登录
            if (actionContext.Request.Headers.Authorization != null)
            {
                string                                  userInfo                                  =
```

```
System.Text.Encoding.Default.GetString(System.Convert.FromBase64String(actionContext.Request.Headers.Authorization.Parameter));
                    //用户验证逻辑
                    if (string.Equals(userInfo, string.Format("{0}:{1}", "zhyongfeng", "123456")))
                    {
                        IsAuthorized(actionContext);
                    }
                    else
                    {
                        HandleUnauthorizedRequest(actionContext);
                    }
                }
                else
                {
                    HandleUnauthorizedRequest(actionContext);
                }
            }
            protected override void HandleUnauthorizedRequest(System.Web.Http.Controllers.HttpActionContext actionContext)
            {
                var challengeMessage = new System.Net.Http.HttpResponseMessage(System.Net.HttpStatusCode.Unauthorized);
                challengeMessage.Headers.Add("WWW-Authenticate", "Basic");
                throw new System.Web.Http.HttpResponseException(challengeMessage);
            }
        }
}
```

生成 Token

BaseController.cs

```
using SSLWebApi.Models;
using System;
using System.Web;
using System.Web.Http;

namespace SSLWebApi.Controllers
{
    /// <summary>
    /// BaseController 继承 BaseController 则需要身份验证
    /// </summary>
    [CustomAuthorize]
    [RoutePrefix("api/Base")]
    public class BaseController : ApiController
    {
        [HttpGet]
```

```
        [Route("Login")]
        public string Login(string userId)
        {
            if (HttpRuntime.Cache.Get(userId) == null)
            {
                return CreateToken(userId);
            }
            else
            {
                HttpRuntime.Cache.Remove(userId);
                return CreateToken(userId);
            }
        }

        /// <summary>
        ///  生成 token
        /// </summary>
        /// <param name="userId"></param>
        /// <returns></returns>
        private string CreateToken(string userId)
        {
            Token token = new Token();
            token.UserId = userId;
            token.SignToken = Guid.NewGuid();
            token.Seconds = 12;
            token.ExpireTime = DateTime.Now.AddSeconds(token.Seconds);
            HttpRuntime.Cache.Insert(token.UserId, token, null, token.ExpireTime,
TimeSpan.Zero);
            return token.SignToken.ToString();
        }
    }
}
```

5 编写.NET WebApi 的 ActionFilterAttribute 令牌验证

WebApiSecurityFilter.cs

```
using SSLWebApi.Models;
using System;
using System.Linq;
using System.Net.Http;
```

```
using System.Web;
using System.Web.Http.Controllers;
using System.Web.Http.Filters;

namespace SSLWebApi.Filter
{
    public class WebApiSecurityFilter : ActionFilterAttribute
    {
        public override void OnActionExecuting(HttpActionContext actionContext)
        {

            if (actionContext.ActionDescriptor.ActionName == "Login")
            {
                //登录成功则生成 token
                base.OnActionExecuting(actionContext);
                return;
            }
            else
            {
                //判断 token 令牌
                HttpRequestMessage request = actionContext.Request;
                string staffid = request.Headers.Contains("userid") ? HttpUtility.UrlDecode(request.Headers.GetValues("userid").FirstOrDefault()) : string.Empty;
                string timestamp = request.Headers.Contains("timestamp") ? HttpUtility.UrlDecode(request.Headers.GetValues("timestamp").FirstOrDefault()) : string.Empty;
                string nonce = request.Headers.Contains("nonce") ? HttpUtility.UrlDecode(request.Headers.GetValues("nonce").FirstOrDefault()) : string.Empty;
                string signature = request.Headers.Contains("signature") ? HttpUtility.UrlDecode(request.Headers.GetValues("signature").FirstOrDefault()) : string.Empty;

                if (String.IsNullOrEmpty(staffid) || String.IsNullOrEmpty(timestamp) || String.IsNullOrEmpty(nonce) || String.IsNullOrEmpty(signature))
                {
                    //令牌检验不通过
                    actionContext.Response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
                    return;
                }
                else
                {
                    //令牌检验 token 失效
                    if (HttpRuntime.Cache.Get(staffid) == null)
                    {
                        actionContext.Response = new
```

```
HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
                    return;
                }
                else
                {
                    //停牌检验 2：判断 timespan 是否失效
                    Token token = (Token)HttpRuntime.Cache.Get(staffid);
                    double ts1 = 0;
                    bool timespanvalidate = double.TryParse(timestamp, out ts1);
                    double ts2 = (token.ExpireTime - new DateTime(1970, 1, 1, 0, 0, 0,
0)).TotalSeconds;
                    bool flag = (ts2 - ts1) > token.Seconds ? true : false;
                    bool tokenFlag = (token.SignToken.ToString() == signature) ? true :
false;
                    if (timespanvalidate && (!flag) && tokenFlag)
                    {
                        //时间转换成功、时间有效、token 值相等
                        //令牌通过
                        base.OnActionExecuting(actionContext);
                        return;
                    }
                    else
                    {
                        actionContext.Response = new
HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);
                        return;
                    }
                }
            }
        }
    }
}
```

6 编写.NET WebApi 的服务端

UserController.cs

```
using System;
using System.Net;
using System.Web.Http;

namespace SSLWebApi.Controllers
```

```
{
    [RoutePrefix("api/User")]
    public class UserController : ApiController
    {
        /// <summary>
        /// 获取当前用户信息
        /// </summary>
        /// <param name="msg"></param>
        /// <returns></returns>
        [HttpPost]
        [Route("PostMessage")]
        public string PostMessage(dynamic obj)
        {
            return string.Format("当前输入的消息是:{0}", Convert.ToString(obj.msg));
        }

        [Route("GetMachine")]
        public string GetMachine()
        {
            string AddressIP = string.Empty;
            foreach (IPAddress _IPAddress in Dns.GetHostEntry(Dns.GetHostName()).AddressList)
            {
                if (_IPAddress.AddressFamily.ToString() == "InterNetwork")
                {
                    AddressIP = _IPAddress.ToString();
                }
            }
            return string.Format("当前 WebApi 部署的 IP 是：{0}", AddressIP);
        }
    }
}
```

7 编写.NET WebApi 的客户端

WebApiHelper.cs

```
using Newtonsoft.Json;
using System;
using System.IO;
using System.Net;
using System.Text;

namespace SSLWebApiClient.Common
```

```
{
    public class WebApiHelper
    {
        /// <summary>
        /// Post 请求
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="url">url</param>
        /// <param name="data">数据</param>
        /// <param name="userid">帐户</param>
        /// <param name="signature">数字签名</param>
        /// <returns></returns>
        public static string Post(string url, string data, string userid, string signature)
        {
            return PostData(url, data, userid, signature);
        }

        /// <summary>
        /// Post 请求
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="url">url</param>
        /// <param name="data">数据</param>
        /// <param name="userid">帐户</param>
        /// <param name="signature">数字签名</param>
        /// <returns></returns>
        public static T Post<T>(string url, string data, string userid, string signature)
        {
            return JsonConvert.DeserializeObject<T>(Post(url, data, userid, signature));
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="webApi"></param>
        /// <param name="queryStr"></param>
        /// <param name="userid"></param>
        /// <param name="signature"></param>
        /// <returns></returns>
        public static string Get(string webApi, string queryStr, string userid, string signature)
        {
            return GetData(webApi, queryStr, userid, signature);
        }
```

```
        /// <summary>
        /// Get 请求
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="webApi"></param>
        /// <param name="query"></param>
        /// <param name="queryStr"></param>
        /// <param name="userid"></param>
        /// <param name="signature"></param>
        /// <returns></returns>
        public static T Get<T>(string webApi, string queryStr, string userid, string signature)
        {
            return JsonConvert.DeserializeObject<T>(GetData(webApi, queryStr, userid,
signature));
        }

        /// <summary>
        /// 获取时间戳
        /// </summary>
        /// <returns></returns>
        private static string GetTimeStamp()
        {
            TimeSpan ts = DateTime.Now - new DateTime(1970, 1, 1, 0, 0, 0, 0);
            return ts.TotalSeconds.ToString();
        }



        /// <summary>
        /// 获取随机数
        /// </summary>
        /// <returns></returns>
        private static string GetRandom()
        {
            Random rd = new Random(DateTime.Now.Millisecond);
            int i = rd.Next(0, int.MaxValue);
            return i.ToString();
        }
        /// <summary>
        /// Post 请求
        /// </summary>
        /// <param name="url"></param>
        /// <param name="data"></param>
        /// <param name="userid">用户名称</param>
        /// <param name="signature">数字签名</param>
```

```
        /// <returns></returns>
        private static string PostData(string url, string data, string userid, string signature)
        {
            try
            {
                byte[] bytes = Encoding.UTF8.GetBytes(data);
                HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

                string timeStamp = GetTimeStamp();
                string nonce = GetRandom();
                //加入头信息
                //当前请求用户
                request.Headers.Add("userid", userid);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("timestamp", timeStamp);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("nonce", nonce);
                //当前请求内容的数字签名
                request.Headers.Add("signature", signature);

                //写数据
                request.Method = "POST";
                request.ContentLength = bytes.Length;
                request.ContentType = "application/json";
                request.GetRequestStream().Write(bytes, 0, bytes.Length);
                //读数据
                request.Timeout = 300000;
                request.Headers.Set("Pragma", "no-cache");
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                Stream streamReceive = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(streamReceive,
Encoding.UTF8);
                string strResult = streamReader.ReadToEnd();

                //关闭流
                //reqstream.Close();
                streamReader.Close();
                streamReceive.Close();
                request.Abort();
                response.Close();
                return strResult;
            }
            catch (Exception ex)
            {
```

```
                return ex.Message;
            }
        }

        /// <summary>
        /// Get 请求
        /// </summary>
        /// <param name="webApi"></param>
        /// <param name="queryStr"></param>
        /// <param name="userid"></param>
        /// <param name="signature"></param>
        /// <returns></returns>
        private static string GetData(string webApi, string queryStr, string userid, string signature)
        {
            try
            {
                HttpWebRequest request = (HttpWebRequest)WebRequest.Create(webApi + "?" + queryStr);
                string timeStamp = GetTimeStamp();
                string nonce = GetRandom();
                //加入头信息
                //当前请求用户
                request.Headers.Add("userid", userid);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("timestamp", timeStamp);
                //发起请求时的时间戳（单位：秒）
                request.Headers.Add("nonce", nonce);
                //当前请求内容的数字签名
                request.Headers.Add("signature", signature);

                request.Method = "GET";
                request.ContentType = "application/json";
                request.Timeout = 90000;
                request.Headers.Set("Pragma", "no-cache");
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                Stream streamReceive = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(streamReceive, Encoding.UTF8);
                string strResult = streamReader.ReadToEnd();

                streamReader.Close();
                streamReceive.Close();
                request.Abort();
```

```
                response.Close();
                return strResult;
            }
            catch (Exception ex)
            {
                return ex.Message;
            }
        }

    }
}
```

Program.cs

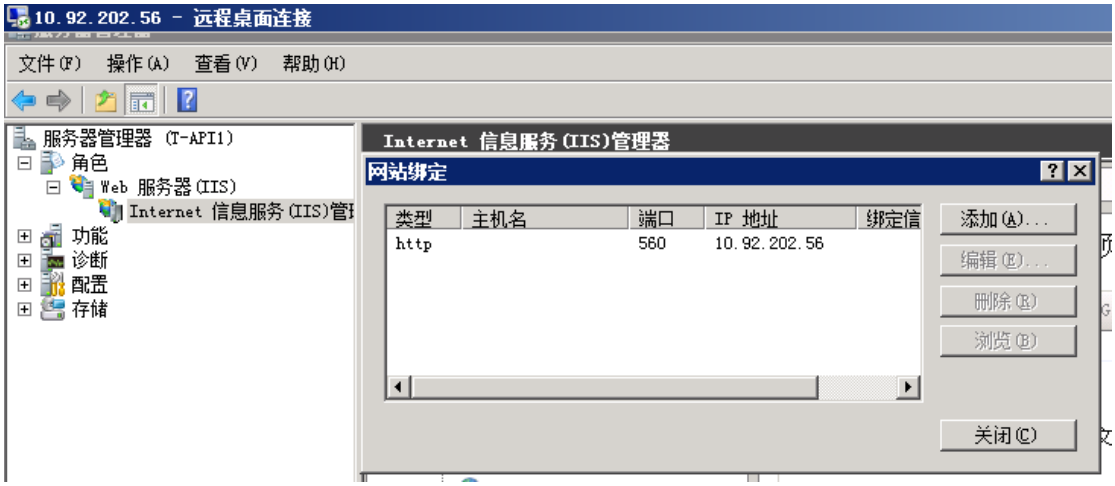
```
using System;
using System.IO;
using System.Net;
using System.Text;
using Newtonsoft.Json;
namespace SSLWebApiClient
{
    class Program
    {
        static void Main(string[] args)
        {
            string basicUrl = "http://localhost:20107";
            string html = string.Empty;
            for (int i = 0; i < 1; i++)
            {
                //https 协议基本认证  Authorization
                string url = basicUrl + "/api/base/Login?userId=zhyongfeng";
                ServicePointManager.ServerCertificateValidationCallback = delegate { return
true; };
                HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
                NetworkCredential credential = new NetworkCredential("zhyongfeng",
"123456");
                req.Credentials = credential;
                HttpWebResponse response = (HttpWebResponse)req.GetResponse();
                Stream responseStream = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(responseStream,
Encoding.UTF8);
                html = streamReader.ReadToEnd().Replace("\"", "");
                Console.WriteLine("Token 服务器保存时间为 12s");
                Console.WriteLine(String.Format("服务器返回的 Token 值为:{0}", html));
            }
            //token 设置了 12s 有效期
```

```
                for (int j = 0; j < 5; j++)
                {
                    System.Threading.Thread.Sleep(1000);
                    string url = basicUrl + "/api/user/PostMessage";
                    Console.WriteLine(Common.WebApiHelper.Post(url,
JsonConvert.SerializeObject(new { msg = "hello" }), "zhyongfeng", html));
                }
                for (int j = 0; j < 10; j++)
                {
                    System.Threading.Thread.Sleep(1000);
                    string url = basicUrl + "/api/user/GetMachine";
                    Console.WriteLine(Common.WebApiHelper.Get(url,      null,      "zhyongfeng",
html));
                }
                Console.Read();
            }

        }
}
```

8 部署 WebApi 到局域网内 3 台 PC 机

将 WebApi 部署到以下 10.92.202.56 的 3 台 PC 机



9 Nginx 集群配置搭建

通过自主义域名 zhyongfeng.com：80 端口进行负载均衡集群访问，则访问 C:\Windows\System32\drivers\etc\hosts，添加下列“本机 IP 自定义的域名”：

```
10.93.85.66    zhyongfeng.com
```

Nginx 的集群配置：

```
#user    nobody;
worker_processes    1;
events {
      worker_connections    1024;
}

http {
      include             mime.types;
      default_type    application/octet-stream;
      sendfile              on;
      keepalive_timeout    65;
      #server {
      #       listen             80;
      #       server_name    localhost;
      #       location / {
      #             root      html;
      #             index    index.html index.htm;
      #       }
      #       error_page      500 502 503 504    /50x.html;
      #       location = /50x.html {
      #             root      html;
      #       }
      #}

      upstream zhyongfeng.com {
            server     10.92.202.56:560;
            server     10.92.202.57:570;
            server     10.92.202.58:580;
      }
      server {
            listen             80;
            server_name    zhyongfeng.com;
            rewrite ^(.*)$    https://$host$1 permanent;
      }
      # HTTPS server
      #
      server {
            listen             443 ssl;
            server_name    zhyongfeng.com;
            ssl_certificate           server.crt;
            ssl_certificate_key    server_nopass.key;
      #       ssl_session_cache        shared:SSL:1m;
```

```
    #       ssl_session_timeout    5m;
    #       ssl_ciphers    HIGH:!aNULL:!MD5;
    #       ssl_prefer_server_ciphers    on;
            location / {
                proxy_pass      http://zhyongfeng.com;
            }
    }
}
```

运行 CMD：

```
D:\DTLDownLoads\nginx-1.10.2>start nginx

D:\DTLDownLoads\nginx-1.10.2>nginx -s reload
```

10 运行结果

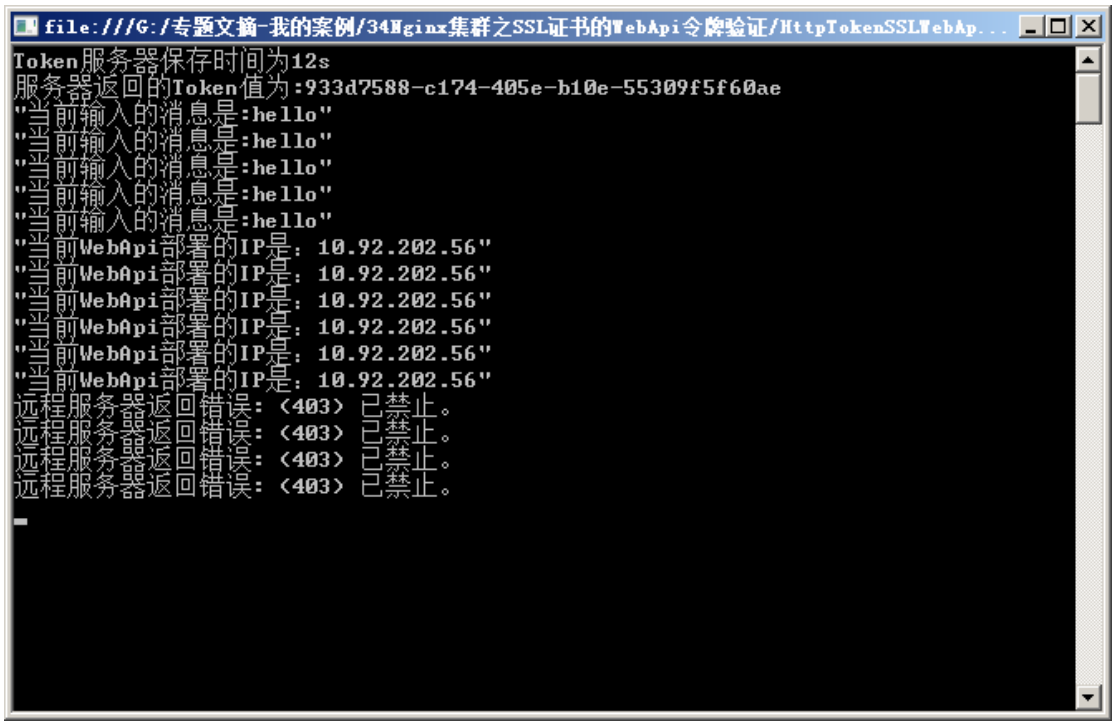
- 访问集群：https://zhyongfeng.com

因只有其中一台计算生成了相应的 Token 值（这里只做其中一台，如果要三台都能够响应，可以用 redis 做相应的 Token 值存储）

其中一台 10.92.202.58 生成了 Token 值，运行结果如下：

- 访问指定的 http://10.92.202.56:560

因 Token 值设定了 12s 后失效，则返回“远程服务器返回错误：<403>已禁止”，运行效果如下：



11 总结

Nginx 基于 SSL 协议下，客户端利用 http basic 身份验证，访问 WebApi 获得 Token，通过 Token 值获取相应的权限数据，使系统的安全性有了保障，同时灵活运用 Token 身份令牌，可以实现时效性的数据访问。基于 Token 的身份验证，针对前后端分离有着很大的作用。例如手机移动端、平板电脑。

源代码下载：
http://download.csdn.net/download/ruby_matlab/10146669

PDF 下载：
Nginx 集群之 SSL 证书的 WebApi 令牌验证.pdf