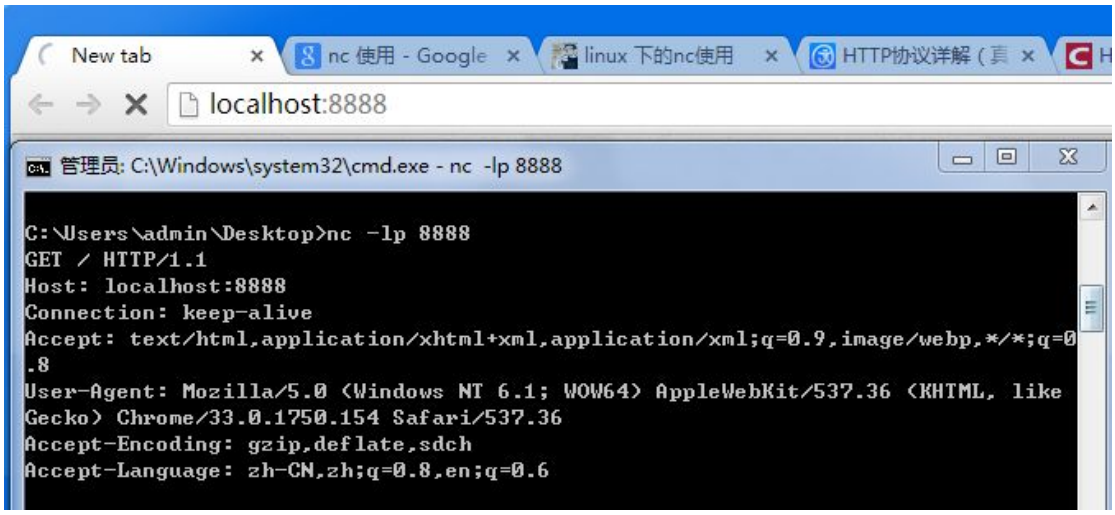
关于 http 协议的理论知识，我在这里就不详细说明了，具体下面给出的链接有。接下来都是用具体的操作显示的，各位可以结合起来看。

一、使用 nc 打开端口，并使用浏览器进行访问 (对应文章中的 HTTP 协议详解之请求篇)

nc -lp 8888　　#使用 nc 打开本地的 8888 端口

使用浏览器，在地址栏上输入 http://localhost:8888 进行访问（提出请求），此时 nc 界面上就会有得到一个请求的 HTTP 协议，具体的请求信息如下：

GET / HTTP/1.1
Host: localhost:8888
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6



请求后 nc 没有给出回应的话，浏览器会一直在该页面进行等待。如果手动结束 nc 程序的话，由于没有给出回应信息，浏览器会给出无法访问该页面。

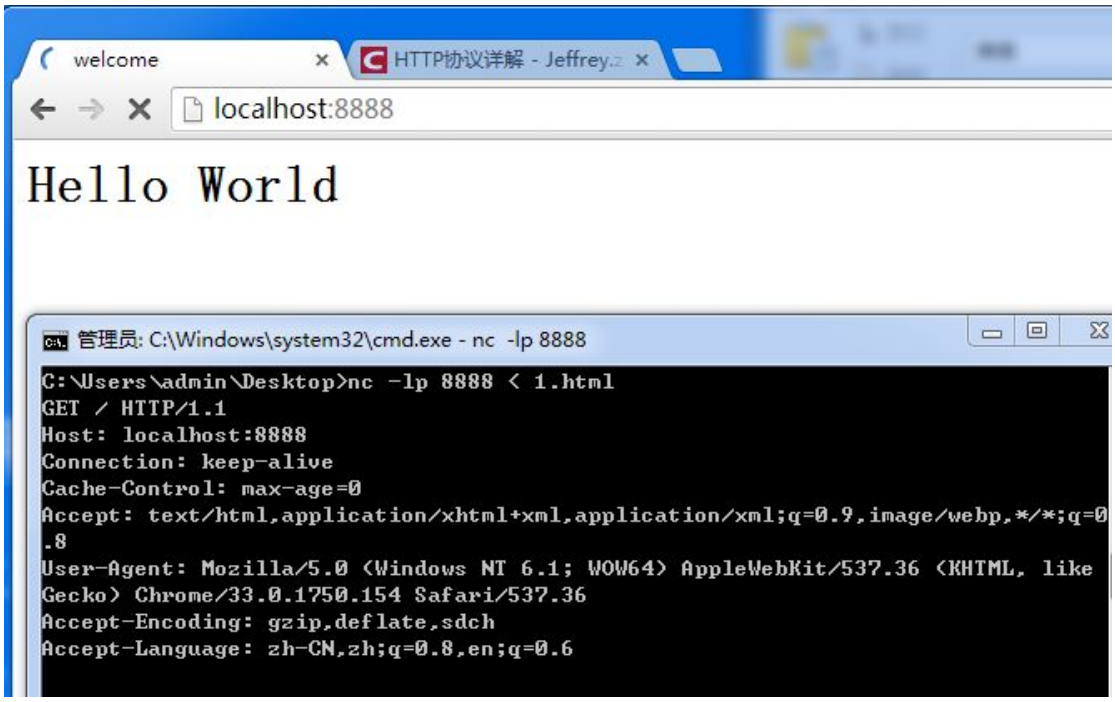
二、nc 后面接着一个资源文件(对应文章中的 HTTP 协议详解之响应篇)

首先写一个 html 的 helloworld

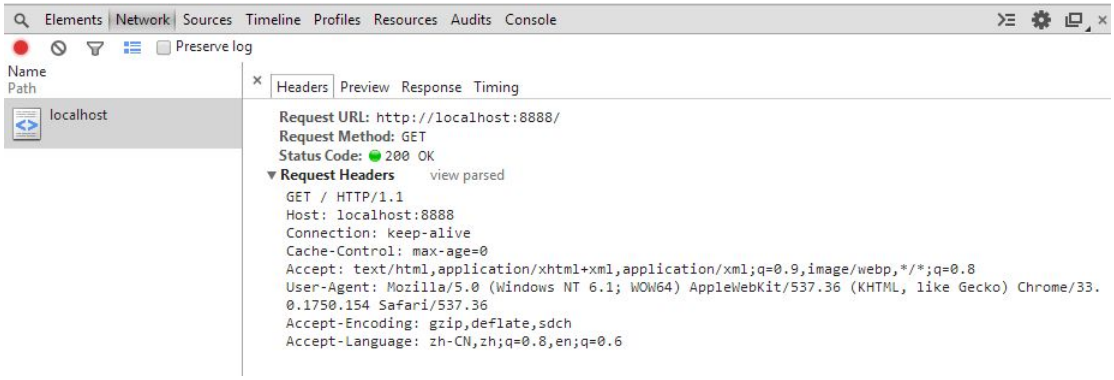
```
<html>
<head>
<title>welcome</title>
<head>
<body>
<h1>Hello World</h1>
</body>
</html>
```

然后再命令行中输入 nc -lp 8888 < helloworld.html

再浏览器中进行访问就可以得到一个页面了



我们打开浏览器的开发工具，chrome 浏览器的快捷键是 F12，再 network 中可以看到下面信息



可以看到里面有 200 OK 这个响应类别号

三、代码实现一个简单的服务器(这里给出一个网上的 java 实现)

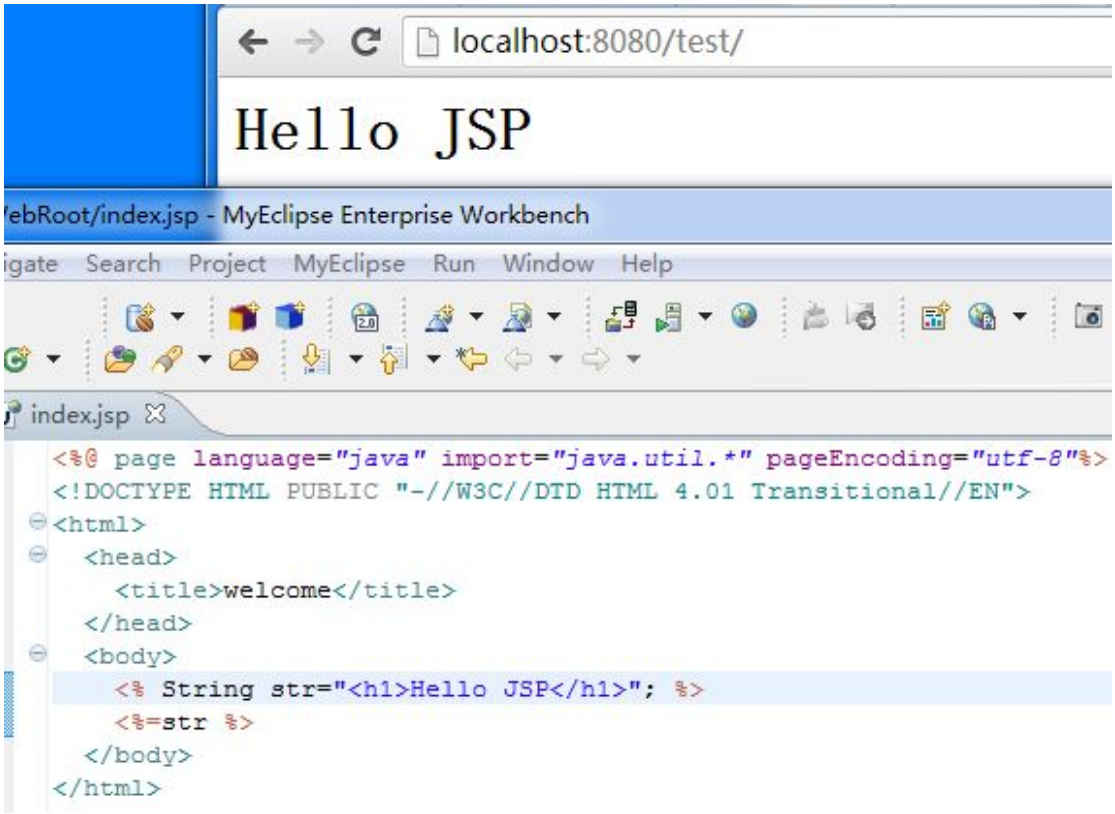
没有办法，http 服务器，就要用到 Socket 编程，而 c++在这一方面又没有具体的标准。所以会有 linux 和 windows 下的不同，不过 java 在这方面就没有问题了，先给出个 java 版的尝尝鲜。代码虽然多，但是具体不难理解。

四、jsp 等动态网站的问题

在 myeclipse 中创建一个 web 工程，然后创建一个 index.jsp 文件

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>welcome</title>
  </head>
  <body>
    <% String str="<h1>Hello JSP</h1>"; %>
    <% out.println(str); %>
  </body>
    </html>
```

然后部署在 tomcat 中，在浏览器中访问。



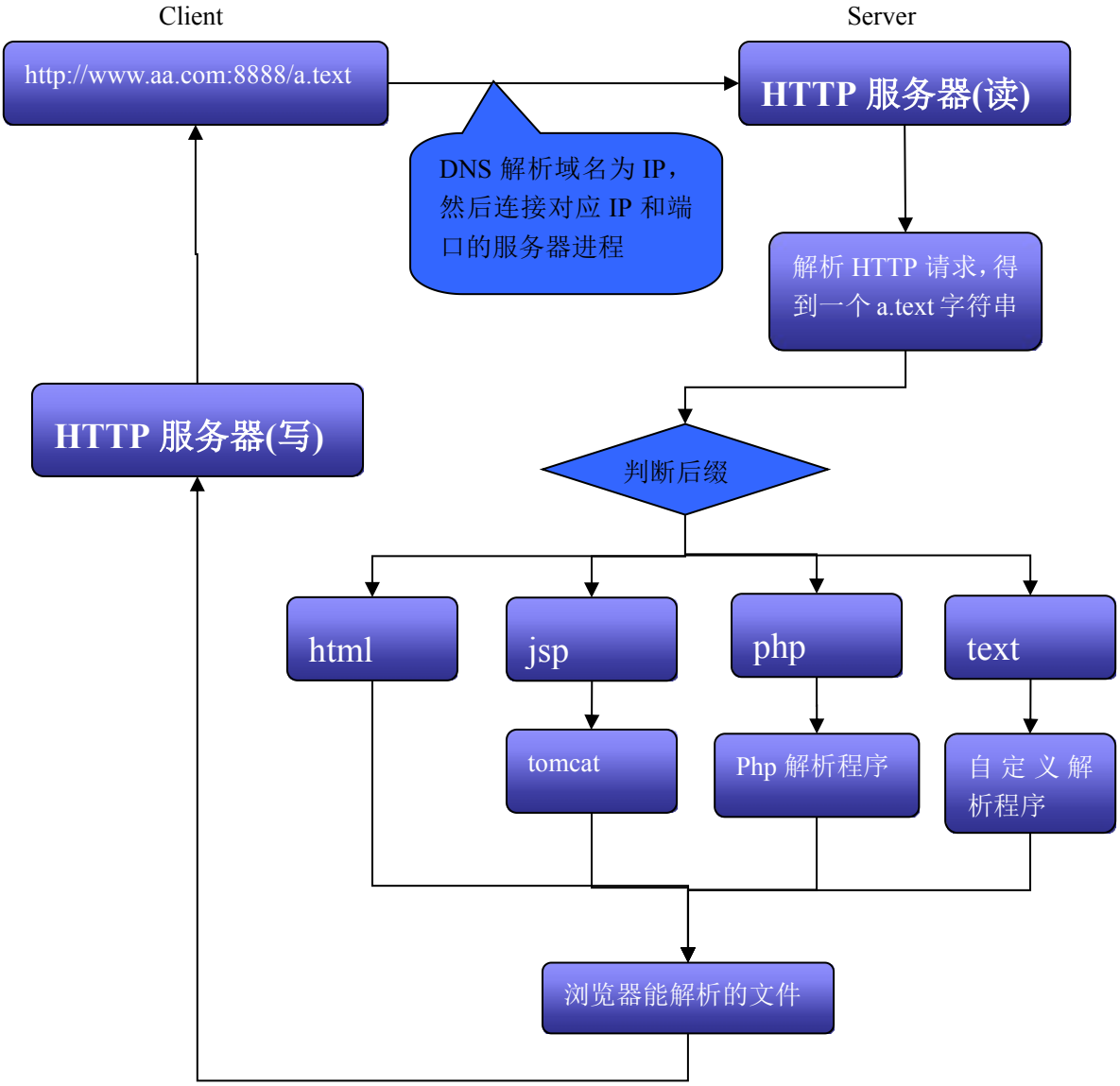
我们可以在 tomcat 的 work 目录(D:\tomcat-6.0.18\work\Catalina\localhost\test\org\apache\jsp)下找到一个名字为 index_jsp.java 的文件。

```
out.write("\r\n");
out.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">\r\n");
out.write("<html>\r\n");
out.write("  <head>\r\n");
out.write("  \t<title>welcome</title>\r\n");
out.write("  </head>\r\n");
out.write("  <body>\r\n");
out.write("    ");
String str="<h1>Hello JSP</h1>";
out.write("\r\n");
out.write("    ");
out.print(str );
out.write("\r\n");
out.write("  </body>\r\n");
out.write("</html>\r\n");
```

使用 servlet，对 jsp 文件进行打印输出。具体的原理我也不是很懂，可以看《How Tomcat Works》。不过我想最后还是输出成一个 html 文件不然在客户端查看源代码怎么会是 html 呢。

其他的动态语言应该也是这个思路了。

五、图解服务器-客户端连接过程(单服务器)

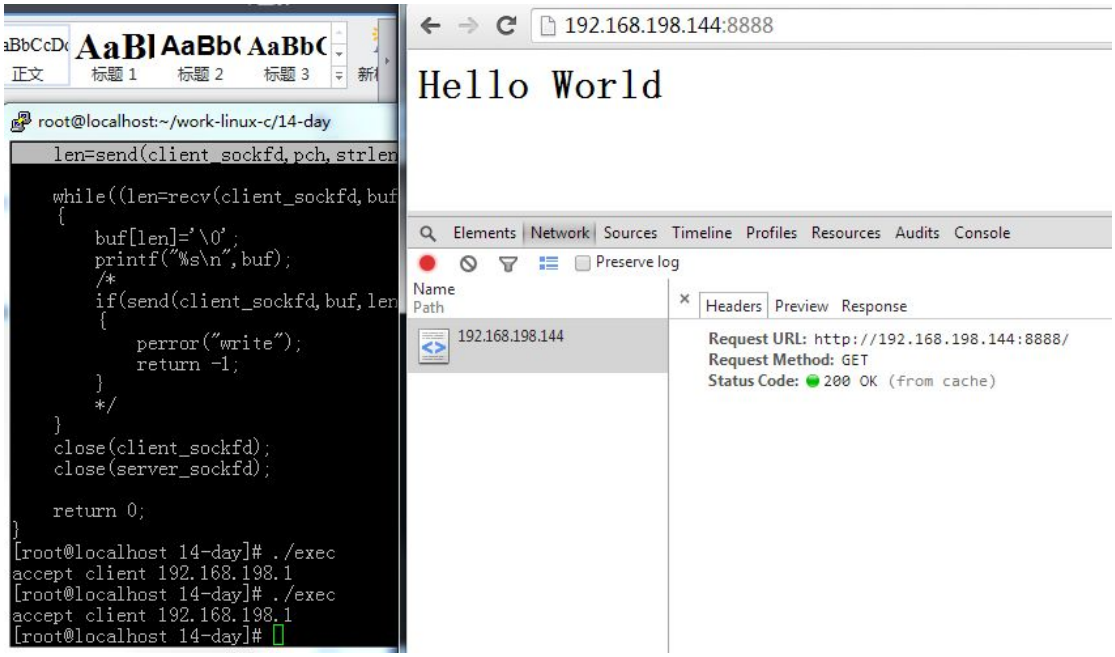


六、C++版的 Http 服务器(Linux Socket)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc,char * argv[])
{
```

```
int server_sockfd;
int client_sockfd;
int len;
struct sockaddr_in my_addr;
struct sockaddr_in remote_addr;
socklen_t sin_size;
char buf[BUFSIZ];
memset(&my_addr,0,sizeof(my_addr));
my_addr.sin_family=AF_INET;
my_addr.sin_addr.s_addr=INADDR_ANY;
my_addr.sin_port=htons(8888);

if((server_sockfd=socket(PF_INET,SOCK_STREAM,0))<0)
{
    perror("socket");
    return -1;
}
if(bind(server_sockfd,(struct sockaddr *)&my_addr,sizeof(struct sockaddr))<0)
{
    perror("bind");
    return -1;
}

listen(server_sockfd,5);
sin_size=sizeof(struct sockaddr_in);

if((client_sockfd=accept(server_sockfd,(struct                    sockaddr
*)&remote_addr,&sin_size))<0)
{
    perror("accept");
    return -1;
}

printf("accept client %s\n",inet_ntoa(remote_addr.sin_addr));
char *pch="<html><h1>Hello World</h1></html>";
len=send(client_sockfd,pch,strlen(pch),0);

while((len=recv(client_sockfd,buf,BUFSIZ,0))>0)
{
    buf[len]='\0';
    printf("%s\n",buf);
    /*
    if(send(client_sockfd,buf,len,0)<0)
    {
```

```
                perror("write");
                return -1;
            }
            */
        }
        close(client_sockfd);
        close(server_sockfd);

        return 0;
}
```



七、

八、

九、

十、

Nc 下载: http://joncraton.org/blog/46/netcat-for-windows/
Nc 的使用: http://freetstar.com/use-nc-in-the-linux/
Http 协议详解: http://blog.csdn.net/gueter/article/details/1524447
Java 版的 http 服务器: http://blog.csdn.net/yanghua_kobe/article/details/7296156
C++版的 Http 服务器: http://www.cppblog.com/kevinlynx/archive/2008/07/30/57521.html
TCP Socket linux: http://blog.csdn.net/wangyf101/article/details/9790807
本文链接: