HTTP协议

什么是HTTP协议

HTTP 是 Hyper Text Transfer Protocol 超文本传输协议的简写。它是 TCP/IP协议的一个应用层协议。它定义了Web浏览器和Web服务器之间进行数据交互的规范。Web 客户端连接上Web服务器后，若想要获取Web服务器上的任意Web 资源需要提交请求（Request），这个请求必须要遵守一定的通讯格式，这个格式就在 HTTP 协议中定义。

OSI七层模型

> 了解就行

具体查看参考3

TCP 四层模型

> 了解就行

具体查看参考4

HTTP协议的版本

HTTP 协议目前有以下几个版本

- HTTP0.9
- HTTP1.0
- HTTP1.1
- HTTP2.0

HTTP多个版本之间的区别

HTTP版本	具备的特性
HTTP 0.9	只接收 GET 一种请求没有在通信中指定版本号且不支持请求头。因为该版本不支持 POST 因此无法向服务端发送太多的信息
HTTP 1.0	可以在通信中指定版本号。
HTTP 1.1	持久连接被默认采用，并能很好地配合代理服务器工作。还支持以管道方式在同时发送多个请求，以便降低线路负载，提高传输速度
HTTP 2.0	HTTP/2是HTTP协议自1999年HTTP 1.1发布后的首个更新，主要基于SPDY协议（是Google开发的基于TCP的应用层协议，用以最小化网络延迟，提升网络速度，优化用户的网络使用体验）。

HTTP2.0和 HTTP1.1的区别

1. HTTP/2采用二进制格式而非文本格式
2. HTTP/2是完全多路复用的，而非有序并阻塞的——只需一个连接即可实现并行
3. 使用报头压缩，HTTP/2降低了开销
4. HTTP/2让服务器可以将响应主动“推送”到客户端缓存中

HTTP1.0和 HTTP1.1的区别

1. persistent connection（持久连接）

HTTP/1.0中，每对请求/ 响应都使用一个新的连接。HTTP/1.1则支持持久连接（默认）。

2. Host域

HTTP/1.1在请求消息头多一个Host域；HTTP/1.0则没有这个域，建立TCP连接的时候已经指定了IP地址，而且默认一个IP地址只对应一个主机名，IP地址上只有一个host。

3. 带宽优化

HTTP/1.1中在请求消息中引入了range头域，它允许只请求资源的某个部分。在响应消息中Content-Range头域声明了返回的这部分对象的偏移值和长度。如果服务器相应地返回了对象所请求范围的内容，则响应码为206（Partial Content），它可以防止Cache将响应误以为是完整的一个对象。请求消息中如果包含比较大的实体内容，但不确定服务器是否能够接收该请求（如是否有权限），此时若贸然发出带实体的请求，如果被拒绝也会浪费带宽。HTTP/1.1加入了一个新的状态码100（Continue）。客户端事先发送一个只带头域的请求，如果服务器因为权限拒绝了请求，就回送响应码 401（Unauthorized）；如果服务器接收此请求就回送响应码100，客户端就可以继续发送带实体的完整请求了。注意，HTTP/1.0的客户 端不支持100响应码。

节省带宽资源的一个非常有效的做法就是压缩要传送的数据。Content-Encoding是对消息进行端到端（end-to-end）的编码，它可能是 资源在服务器上保存的固有格式（如jpeg图片格式）；在请求消息中加入Accept-Encoding头域，它可以告诉服务器客户端能够解码的编码方 式。而Transfer-Encoding是逐段式（hop-by-hop）的编码，如Chunked编码。在请求消息中加入TE头 域用来告诉服务器能够接收的transfer-coding方式。

4. 请求方法和状态码

HTTP1.1增加了OPTIONS, PUT, DELETE, TRACE, CONNECT这些Request方法

HTTP/1.0中只定义了16个状态响应码，对错误或警告的提示不够具体。HTTP/1.1引入了一个Warning头域，增加对错误或警告信息的描述。

在HTTP/1.1中新增了24个状态响应码，如409（Conflict）表示请求的资源与资源的当前状态发生冲突；410（Gone）表示服务器上的某个资源被永久性的删除。

5. 内容协商

为 了满足互联网使用不同母语和字符集的用户，一些网络资源有不同的语言版本（如中文版、英文版）。HTTP/1.0定义了内容协商 （content negotiation）的概念，也就是说客户端可以告诉服务器自己可以接收以何种语言（或字符集）表示的资源。例如如果服务器不能明确 客户端需要何种类型的资源，会返回300（Multiple Choices），并包含一个列表，用来声明该资源的不同可用版本，然后客户端在请求消息中包含Accept-Language和Accept- Charset头域指定需要的版本。

6. 状态码

100～199：信息状态码，表示成功接收请求，要求客户端继续提交下一次请求才能完成整个处理过程

100（continue）继续发送

200～299：成功状态码，表示成功接收请求并已完成整个处理过程，常用200（OK）成功接收

300～399：重定向状态码，例如，请求的资源已经移动一个新地址，常用302、307和304

400～499：客户端的请求有错误，常用404（Not Found），403（Fobidden）

500～599：服务器端出现错误，常用 500

简单来说 HTTP1.0客户端和服务端建立连接后只能获取一个资源，就会断开连接。而 HTTP1.1客户端和服务端建立连接后，引入了会话机制，在超时时间内只要不关闭网页（主动断开连接，或者长时间静默）可以获取多个资源。

HTTP和HTTPS的区别

HTTPS是在HTTP的基础上引入了SSL有验证.简单说 HTTPS 更安全些，现在很多网站都在采用 HTTPS 协议。

HTTP Message

HTTP Message有两部分组成

- 请求

> 由Client发送给Server的叫做请求

- 响应

> 由Server返回给Client的叫做响应.

HTTP请求

当客户端连接 Web 服务器成功后，向服务器请求某个 Web 资源，称之为客户端向服务端发送了一个 HTTP 请求。一个完整的 HTTP 请求包含三个部分内容

- 请求头
- 请求行
- 请求体

GET 请求和 POST 请求的区别

HTTP请求行

HTTP请求头

请求头的参数

> 常用的几个

参数	含义
Accept	客户端向服务端表示,我能支持的数据类型有哪些.
Accept-Language	支持的语言格式
Accept-Encoding	支持的压缩格式
Connection	表示当前是否为长链接(默认是keep-alive)
Content-Length	内容长度(包含请求头请求体请求行)
Content-Type	提交的数据类型(经过urlencoding URL编码的Form表单数据)
Cookie	暂时不管.讲Cookie时再说
Host	主机地址
Referer	真正请求的地址路径. 全路径(防止倒链)
User-Agent	用户代理.向服务器表名来访者(客户端)的信息.

HTTP请求体

> 浏览器真正发送给服务器的数据.

浏览器发送给服务器的数据.数据以key=value的形式进行传输.如果存在多个key-value.者多个key-value之间使用&作为间隔

如果没有被重新排版.请求体会在请求头的下面.并且请求体和请求头之间会有一行空格.

HTTP响应

> 参考请求

HTTP响应行

响应行有三部分内容

- 协议版本
- 状态码
- 状态码解释.

```
HTTP1.1 200 OK
```

HTTP响应头

- Content-Type 服务端返回给客户端的内容类型
- Content-Length 返回的数据长度
- Date 通信的日期.响应的日期

HTTP响应体

就是返回内容.请求一个页面比如请求百度.那响应提就是百度首页整个HTML文档内容.下面是一个测试.在一个Web项目中没有提供任何静态页面或者JSP页面.只提供了一个Servlet代码如下

```java
package com.hwua.web.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "Servlet01", urlPatterns = { "/Servle01" })
public class Servlet01 extends HttpServlet {

    /**
     *
     */
    private static final long serialVersionUID = -7344023968316803106L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html;charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.write("<html>");
        out.write("<head>");
        out.write("<title>");
        out.write("这是标题");
        out.write("</title>");
        out.write("</head>");
        out.write("<body>");
        out.write("<h3>");
        out.write("这是一个三级标题");
        out.write("</h3>");
        out.write("</body>");
        out.write("</html>");
    }
}
```

当我们运行这个Servlet之后将会得到下面的页面

从这一点就可以发现.页面的内容完全是由响应对象利用输出流写出来的.

在服务器端设置响应头来控制客户端浏览器行为

设置Location响应头来实现请求重定向

```
@Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setStatus(302);
        //-- 通过设置响应头参数来实现页面跳转
        resp.setHeader("Location", "https://www.baidu.com");
    }
```

把客户请求的数据进行压缩

```java
/**
     * 实现压缩 指服务端向客户端发送数据时可以对原生数据进行压缩
     */
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        String str = "周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片
周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开
新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那
天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周
导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新
片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天
开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导
那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片周导那天开新片";
        System.out.println("数据的原始大小:\t" + str.length());

        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        GZIPOutputStream gos = new GZIPOutputStream(bos);
        gos.write(str.getBytes());
        gos.close();

        // -- 得到压缩后的数据
        byte[] target = bos.toByteArray();
        System.out.println("压缩后的数据大小:\t" + target.length);

        // 把压缩后的数据写个客户端
        resp.setHeader("Content-Encoding", "gzip");
        resp.setHeader("Content-Length", target.length + "");

        // -- 把数据写个客户端
        resp.getOutputStream().write(target);
```

输出的结果是

```
数据的原始大小:	413
压缩后的数据大小:	44
```

设置返回的数据类型

```
/**
     * ' 告诉客户端返回的数据是什么类型
     */
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // -- 把img/Koala.jpg发送给客户端
        InputStream is = this.getServletContext().getResourceAsStream("/img/Koala.jpg");
        byte[] buffer = new byte[1024];
        int length = 0;
        // -- 设置返回的数据类型
        resp.setHeader("content-type", "image/jpeg");
        OutputStream os = resp.getOutputStream();
        while (-1 != (length = is.read(buffer))) {
            os.write(buffer, 0, length);
        }
    }
```

设置 Refresh 响应头让浏览器定时刷新

```
/**
     * ' 设置刷新
     */
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //-- 让浏览器隔3秒后刷新.刷新到指定页面
        resp.setHeader("refresh", "3;https://www.baidu.com");
    }
```

参考

1. HTTP2.0介绍
2. HTTP和 HTTPS 的区别
3. OSI七层模型
4. TCP/IP四层模型