

day08

软件系统体系结构

1 常见软件系统体系结构 B/S、C/S

● 1.1 C/S

- C/S 结构即客户端/服务器 (Client/Server)，例如 QQ；
- 需要编写服务器端程序，以及客户端程序，例如我们安装的就是 QQ 的客户端程序；
- 缺点：软件更新时需要同时更新客户端和服务端两端，比较麻烦；
- 优点：安全性比较好。

1.2 B/S (*****)

- B/S 结构即浏览器/服务器 (Browser/Server)；
- 优点：只需要编写服务器端程序；
- 缺点：安全性较差。

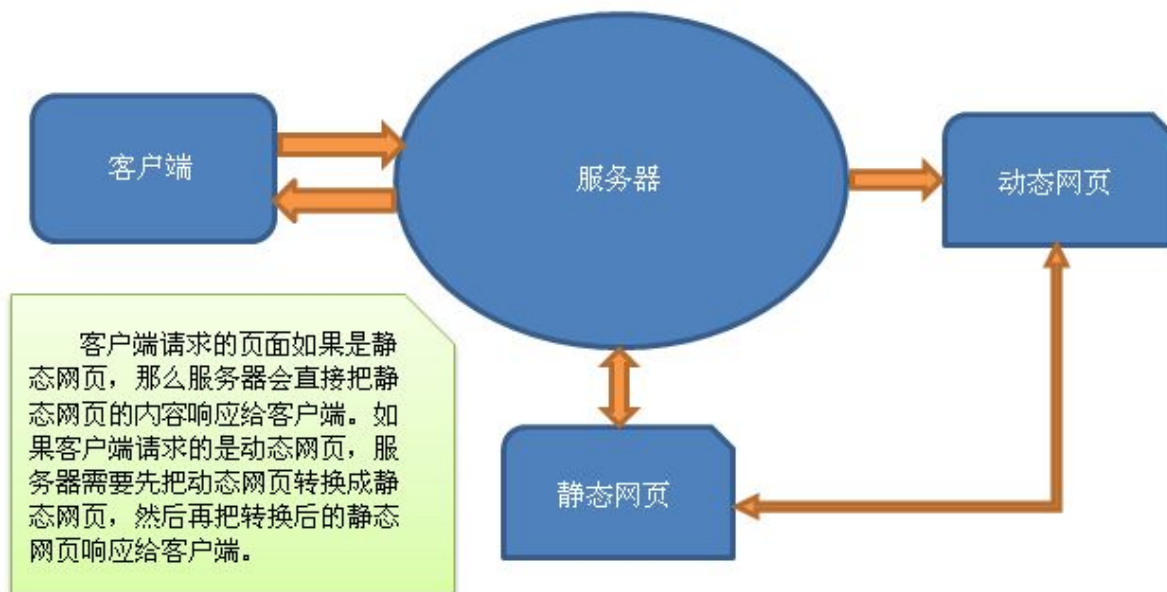
2 WEB 资源

2.1 Web 资源介绍

- **html**：静态资源；
- **JSP/Servlet**：动态资源。

当然，除了 JavaWeb 程序，还有其他 Web 程序，例如：ASP、PHP 等。

2.2 静态资源和静态资源区别



2.3 访问 Web 资源

打开浏览器，输入 URL：

- 协议名://域名:端口/路径，例如：`http://www.itcast.cn:80/index.html`

3 Web 服务器

Web 服务器的作用是接收客户端的请求，给客户端作出响应。

对于 JavaWeb 程序而已，还需要有 JSP/Servlet 容器，JSP/Servlet 容器的基本功能是把动态资源转换成静态资源，当然 JSP/Servlet 容器不只这些功能，我们会在后面一点一点学习。

我们需要使用的是 Web 服务器和 JSP/Servlet 容器，通常这两者会集于一身。下面是对 JavaWeb 服务器：

- Tomcat (Apache)：当前应用最广的 JavaWeb 服务器；
- JBoss (Redhat 红帽)：支持 JavaEE，应用比较广；EJB 容器
- GlassFish (Oracle)：Oracle 开发 JavaWeb 服务器，应用不是很广；
- Resin (Caucho)：支持 JavaEE，应用越来越广；
- Weblogic (Oracle)：要钱的！支持 JavaEE，适合大型项目；
- Websphere (IBM)：要钱的！支持 JavaEE，适合大型项目；

Tomcat (有重点)

1 Tomcat 概述

Tomcat 服务器由 Apache 提供，开源免费。由于 Sun 和其他公司参与到了 Tomcat 的开发中，所以最新的 JSP/Servlet 规范总是能在 Tomcat 中体现出来。当前最新版本是 Tomcat8，我们课程中使用 Tomcat7。Tomcat7 支持 Servlet3.0，而 Tomcat6 只支持 Servlet2.5！

2 安装、启动、配置 Tomcat

下载 Tomcat 可以到 <http://tomcat.apache.org> 下载。

Tomcat 分为安装版和解压版：

- 安装版：一台电脑上只能安装一个 Tomcat；
- 解压版：无需安装，解压即可用，解压多少份都可以，所以我们选择解压版。

2.1 Tomcat 目录结构

安装版 Tomcat 的安装过程请参考 day03_res/Tomcat 安装.doc 文件。

把解压版 Tomcat 解压到一个没有中文，没有空格的路径中即可，建议路径不要太深，因为我们经常需要进入 Tomcat 安装目录。例如：F:\apache-tomcat-7.0.42

2.2 启动和关闭 Tomcat

在启动 Tomcat 之前，我们必须配置环境变量：

- JAVA_HOME：必须先配置 JAVA_HOME，因为 Tomcat 启动需要使用 JDK；
- CATALINA_HOME：如果是安装版，那么还需要配置这个变量，这个变量用来指定 Tomcat 的安装路径，例如：F:\apache-tomcat-7.0.42。
- 启动：进入 %CATALINA_HOME%\bin 目录，找到 startup.bat，双击即可；
- 关闭：进入 %CATALINA_HOME%\bin 目录，找到 shutdown.bat，双击即可；

startup.bat 会调用 catalina.bat，而 catalina.bat 会调用 setclasspath.bat，setclasspath.bat 会使用 JAVA_HOME 环境变量，所以我们必须在启动 Tomcat 之前把 JAVA_HOME 配置正确。

启动问题：

- 点击 startup.bat 后窗口一闪即消失：检查 JAVA_HOME 环境变量配置是否正确；

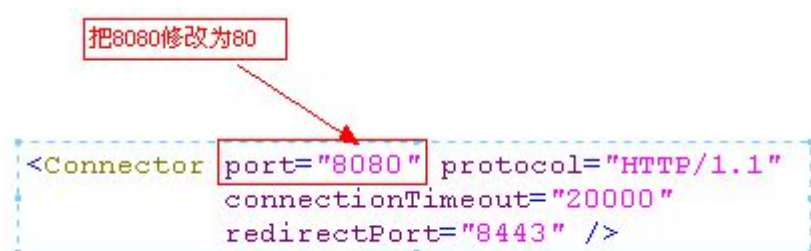
2.3 进入 Tomcat 主页

访问：本机上访问，输入本机 ip 地址和 tomcat 端口号，或者下面两个。非本机访问，就输入 Tomcat 所在电脑的 ip 和 tomcat 端口号访问

- 1、<http://localhost:8080>
- 2、<http://127.0.0.1:8080>

2.4 配置端口号

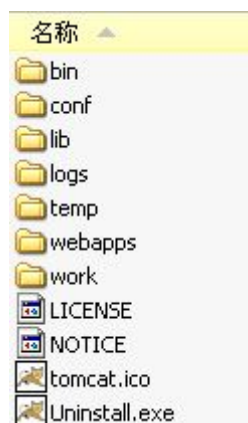
打开%CATALANA_HOME%\conf\server.xml 文件：



http 默认端口号为 80，也就是说在 URL 中不给出端口号时就表示使用 80 端口。当然你也可以修改为其它端口号。

当把端口号修改为 80 后，在浏览器中只需要输入：<http://localhost> 就可以访问 Tomcat 主页了。

2.5 Tomcat 的目录结构



- bin: 该目录下存放的是二进制可执行文件，如果是安装版，那么这个目录下会有两个 exe 文件：tomcat6.exe、tomcat6w.exe，前者是在控制台启动 Tomcat，后者是弹出 UGI 窗口启动 Tomcat；如果是解压版，那么会有 startup.bat 和 shutdown.bat 文件，startup.bat 用来启动 Tomcat，但需要先配置 JAVA_HOME 环境变量才能启动，shutdown.bat 用来停止 Tomcat；
- conf: 这是一个非常非常重要的目录，这个目录下有四个最为重要的文件：
 - server.xml: 配置整个服务器信息。例如修改端口号，添加虚拟主机等；下面会详细介绍这个文件；
 - tomcatusers.xml: 存储 tomcat 用户的文件，这里保存的是 tomcat 的用户名及密码，以及用户的角色信息。可以按着该文件中的注释信息添加 tomcat 用户，然后就可以在 Tomcat 主页中进入 Tomcat Manager 页面了；
 - web.xml: 部署描述符文件，这个文件中注册了很多 MIME 类型，即文档类型。这些 MIME 类型是客户端与服务器之间说明文档类型的，如用户请求一个 html 网页，那么服务器还会告诉客户端浏览器响应的文档是 text/html 类型的，这就是一个 MIME 类型。客户端浏览器通过这个 MIME 类型就知道如何处理它了。当然是在浏览器中显示这个 html 文件了。但如果服务器响应的是一个 exe 文件，那么浏览器就不可能显示它，而是应该弹出下载窗口才对。MIME 就是用来说明文档的内容是什么类型的！

- context.xml: 对所有应用的统一配置, 通常我们不会去配置它。
- lib: Tomcat 的类库, 里面是一大堆 jar 文件。如果需要添加 Tomcat 依赖的 jar 文件, 可以把它放到这个目录中, 当然也可以把应用依赖的 jar 文件放到这个目录中, 这个目录中的 jar 所有项目都可以共享之, 但这样你的应用放到其他 Tomcat 下时就不能再共享这个目录下的 Jar 包了, 所以建议只把 Tomcat 需要的 Jar 包放到这个目录下;
- logs: 这个目录中都是日志文件, 记录了 Tomcat 启动和关闭的信息, 如果启动 Tomcat 时有错误, 那么异常也会记录在日志文件中。
- temp: 存放 Tomcat 的临时文件, 这个目录下的东西可以在停止 Tomcat 后删除!
- webapps: 存放 web 项目的目录, 其中每个文件夹都是一个项目; 如果这个目录下已经存在了目录, 那么都是 tomcat 自带的。项目。其中 ROOT 是一个特殊的项目, 在地址栏中没有给出项目目录时, 对应的就是 ROOT 项目。<http://localhost:8080/examples>, 进入示例项目。其中 examples 就是项目名, 即文件夹的名字。
- work: 运行时生成的文件, 最终运行的文件都在这里。通过 webapps 中的项目生成的! 可以把这个目录下的内容删除, 再次运行时会重新生成 work 目录。当客户端用户访问一个 JSP 文件时, Tomcat 会通过 JSP 生成 Java 文件, 然后再编译 Java 文件生成 class 文件, 生成的 java 和 class 文件都会存放到这个目录下。
- LICENSE: 许可证。
- NOTICE: 说明文件。

Web 应用（重点）

静态网站:

- 在 webapps 目录下创建一个目录（命名必须不包含中文和空格），这个目录称之为项目目录;
- 在项目目录下创建一个 html 文件;

动态网站:

- 在 webapps 目录下创建一个项目目录;
- 在项目目录下创建如下内容:
 - WEB-INF 目录
 - ◆ 在 WEB-INF 目录下创建 web.xml 文件
 - 创建静态或动态页面
 - ◆ 创建动态页面 index.jsp

index.jsp

```
<%@page pageEncoding="utf-8"%><!--jsp 的内容-->
```

```
<html>
```

```
<head>
```

```
<title>hello2</title>
```

```
</head>
```

```
<body>
```

```
<h1>hello2</h1>
<h3>
    ${header['User-Agent']}<!--jsp 的内容，动态显示客户端的信息。暂时知道就行-->
</h3>
</body>
</html>
```

1 创建静态应用

- 在 webapps 下创建一个 hello 目录;
- 在 webapps\hello\下创建 index.html;
- 启动 tomcat;
- 打开浏览器访问 <http://localhost:8080/hello/index.html>

index.html

```
<html>
<head>
    <title>hello</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

2 创建动态应用

- 在 webapps 下创建 hello1 目录;
- 在 webapps\hello1\下创建 WEB-INF 目录;
- 在 webapps\hello1\WEB-INF\下创建 web.xml;
- 在 webapps\hello1\下创建 index.html。
- 打开浏览器访问 <http://localhost:8080/hello/index.html>

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
```

</web-app>

完整的 Web 应用还需要在 WEB-INF 目录下创建:

- classes;
- lib 目录;

webapps

```
| - hello
  | - index.html
  | - WEB-INF
    | - web.xml
    | - classes
    | - lib
```

- hello: 应用目录, hello 就是应用的名称;
- index.html: 应用资源。应用下可以有多个资源, 例如 css、js、html、jsp 等, 也可以把资源放到文件夹中, 例如: hello\html\index.html, 这时访问 URL 为: http://localhost:8080/hello/html/index.html;
- WEB-INF: 这个目录名称必须是大写, 这个目录下的东西是无法通过浏览器直接访问的, 也就是说放到这里的東西是安全的;
- web.xml: 应用程序的部署描述符文件, 可以在该文件中对应用进行配置, 例如配置应用的首页:

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

- classes: 存放 class 文件的目录;
- lib: 存放 jar 包的目录;

3 配置外部应用 (了解)

原来我们的项目放到 webapps 下, 现在我放到外面, 也希望 tomcat 可以找到它!

也可以把应用放到 Tomcat 之外, 这就是外部应用了。例如我们把上面写的 hello 应用从 webapps 目录中剪切到 C 盘下, 即 C:/hello。现在 hello 这个 Web 应用已经不在 Tomcat 中了, 这时我们需要在 tomcat 中配置外部应用的位置, 配置的方式一共有两种:

- conf/server.xml: 打开 server.xml 文件, 找到<Host>元素, 在其中添加<Context>元素, 代码如下:

server.xml

```
<Host name="localhost" appBase="webapps"
      unpackWARs="true" autoDeploy="true">
  <Context path="itcast_hello" docBase="C:/hello" />
</Host>
```

- 1) path: 指定当前应用的名称;
- 2) docBase: 指定应用的物理位置; 项目的真实地址

3) 浏览器访问路径: http://localhost:8080/itcast_hello/index.html。

- conf/catalana/localhost: 在该目录下创建 itcast_hello.xml 文件, 在该文件中编写<Context>元素, 代码如下:

```
1 <Context docBase="C:/hello"/>
```

- 1) 文件名: 指定当前应用的名称;
- 2) docBase: 指定应用的物理位置;
- 3) 浏览器访问路径: http://localhost:8080/itcast_hello/index.html。

4 理解 server.xml (了解)

<Server>

<Servier>

<Connector>

<Engine>

<Host>

<Context>

- <Server>: 根元素, 表示整个服务器的配置信息;
- <Service>: <Server>的子元素, 在<Server>中只能有一个<Service>元素, 它表示服务;
- <Connector>: <Service>的子元素, 在<Service>中可以有 N 个<Connector>元素, 它表示连接。
- <Engine>: <Service>的子元素, 在<Service>中只能有一<Engine>元素, 该元素表示引擎, 它是<Service>组件的核心。
- <Host>: <Engine>的子元素, 在<Engine>中可以有 N 个<Host>元素, 每个<Host>元素表示一个虚拟主机。所谓虚拟主机就像是真的主机一样, 每个主机都有自己的主机名和项目目录。例如<Host name="localhost" appBase="webapps">表示主机名为 localhost, 这个主机的项目存放在 webapps 目录中。访问这个项目下的主机时, 需要使用 localhost 主机名, 项目都存放在 webapps 目录下。
- <Context>: <Host>元素的子元素, 在<Host>中可以有 N 个<Context>元素, 每个<Context>元素表示一个应用。如果应用在<Host>的 appBase 指定的目录下, 那么可以不配置<Context>元素, 如果是外部应用, 那么就必须配置<Context>。如果要为应用指定资源, 也需要配置<Context>元素。

我们可以把<Server>看作是一个大酒店:

- <Service>: 酒店的服务部门;
- <Connector>: 服务员;
- <Engine>: 后厨;
- <Host>: 后厨中的一个区, 例如川菜区是一个<Host>、粤菜区是一个<Host>;
- <Context>: 后厨的一个厨师。

用户发出一个请求: <http://localhost:8080/hello/index.jsp>。发现是 http/1.1 协议, 而且还是 8080 端口, 所以就交给了处理这一请求的“服务员 (处理 HTTP 请求的<Connector>)”, “服务员”再把请求交给了“后厨 (<Engine>)”, 因为请求是要一盘水煮鱼, 所以由“川菜区 (<Host>)”负责, 因为

“大老王师傅<Context>”做水煮鱼最地道，所以由它完成。

- <Connector>: 关心请求中的 http、和 8080;
- <Host>: 关心 localhost;
- <Context>: 关心 hello

5 映射虚拟主机（了解）

我们的目标是，在浏览器中输出：<http://www.itcast.cn> 就可以访问我们的项目。

完成这一目标，我们需要做三件事：

- 修改端口号为 80，这一点应该没有问题吧；
- 在本机上可以解析域名为 127.0.0.1，这需要修改 C:\WINDOWS\system32\drivers\etc\hosts 文件，添加对 <http://www.itcast.cn> 和 127.0.01 的绑定关系；
- 在 server.xml 文件中添加一个<Host>（主机）。

1) 修改端口号为 80

```
<Connector port="80" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

2) 绑定 <http://www.itcast.cn> 和 127.0.01 的绑定关系

```
23 127.0.0.1      www.itcast.cn
```

3) server.xml 文件中添加一个<Host>

```
<Host name="www.itcast.cn" appBase="F:/itcastapps"
      unpackWARs="true" autoDeploy="true">
</Host>
```

- name="www.itcast.cn": 指定虚拟主机名为 www.itcast.cn;
- appBase="F:/itcastapps": 指定当前虚拟主机的应用程序存放目录为 F:/itcastapps。
- 在 itcastapps 目录下创建名为 ROOT 的应用，因为一个主机只可以有一个名为 ROOT 的应用，名为 ROOT 的应用在浏览器中访问是可以不给出应用名称。



现在访问：<http://www.itcast.cn> 看看是什么页面！

请注意，只有本机可以通过 <http://www.itcast.cn> 来访问，而其他电脑不可以！


6 MyEclipse 创建 JavaWeb 应用

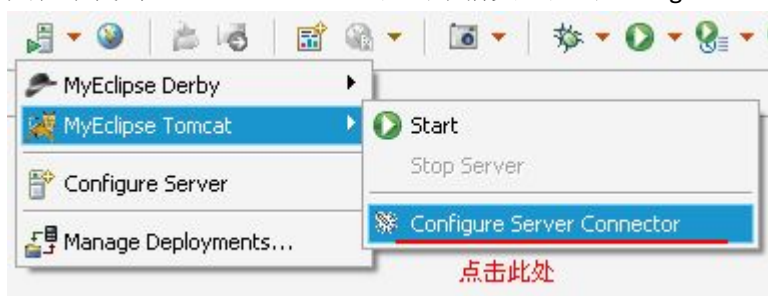
6.1 配置 Tomcat

使用 MyEclipse 配置服务器后，就可以使用 MyEclipse 来启动和停止服务器了。当然，你需要先安装好服务器（Tomcat），才能配置。

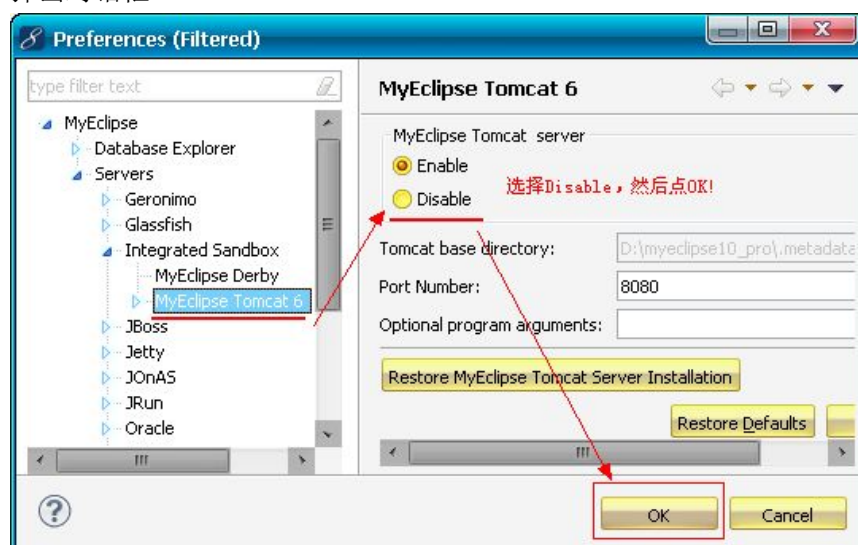
MyEclipse 自带了一个 Tomcat，强烈建议不要使用它。所以，我们需要先把 MyEclipse 自带的 Tomcat 关闭，然后再来配置我们自己的 Tomcat。

- 关闭 MyEclipse 自带 Tomcat。

在工具栏中找到 ，点击下箭头，点击 Configure Server Connector。

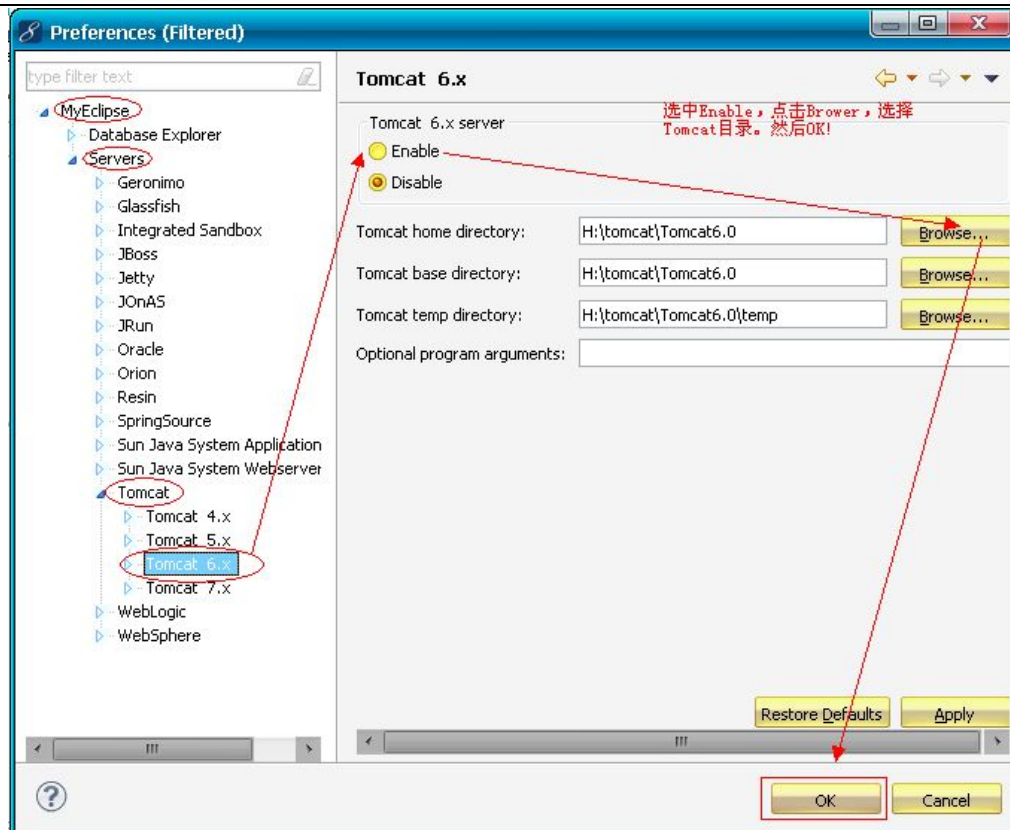


弹出对话框



- 配置我们自己的 Tomcat



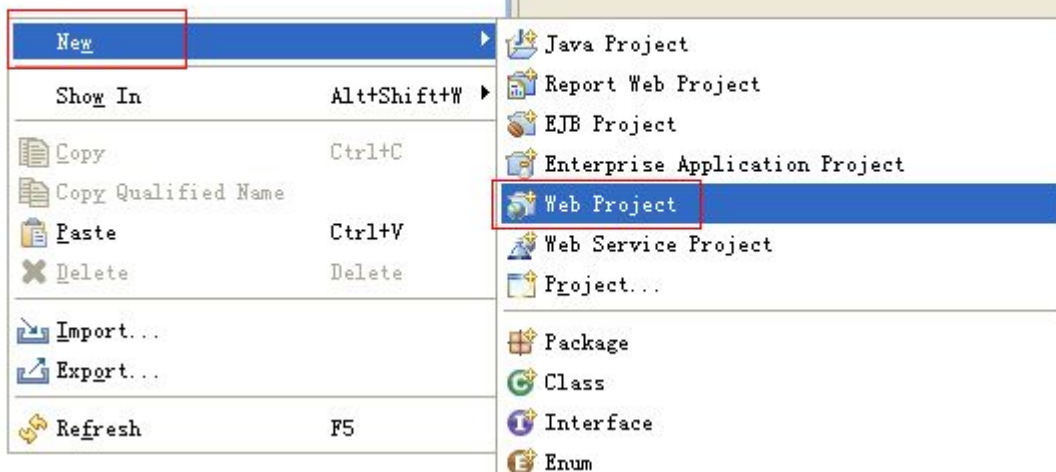


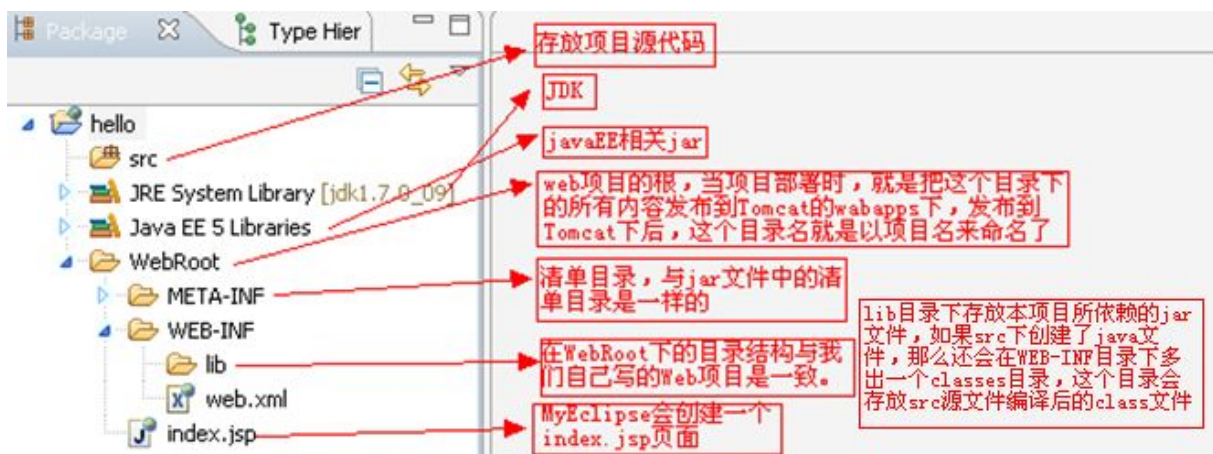
使用 MyEclipse 启动 Tomcat



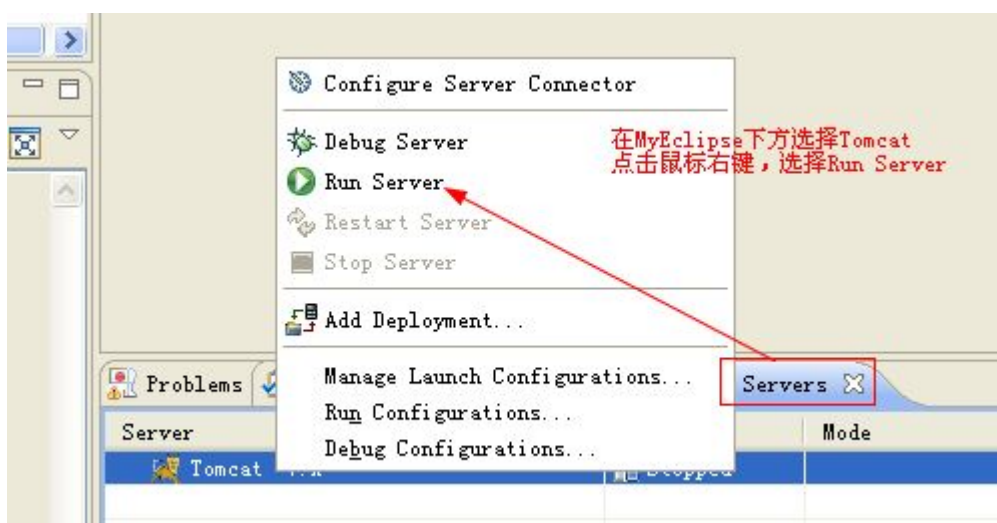
6.2 创建 JavaWeb 应用

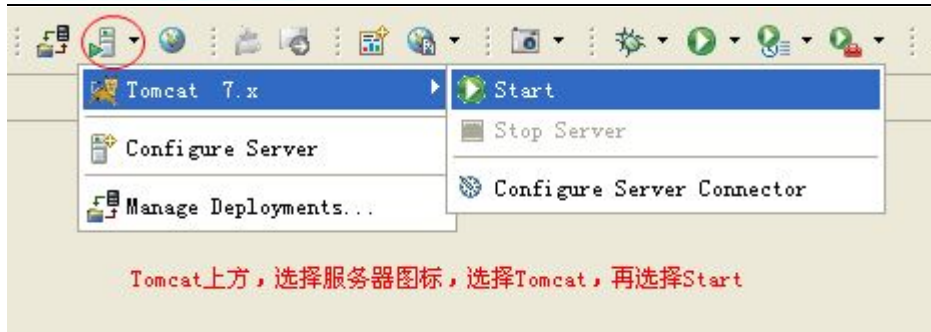
在左侧Package Explorer的空白处点击鼠标右键
选择New，再选择Web Project





6.3 启动 Tomcat



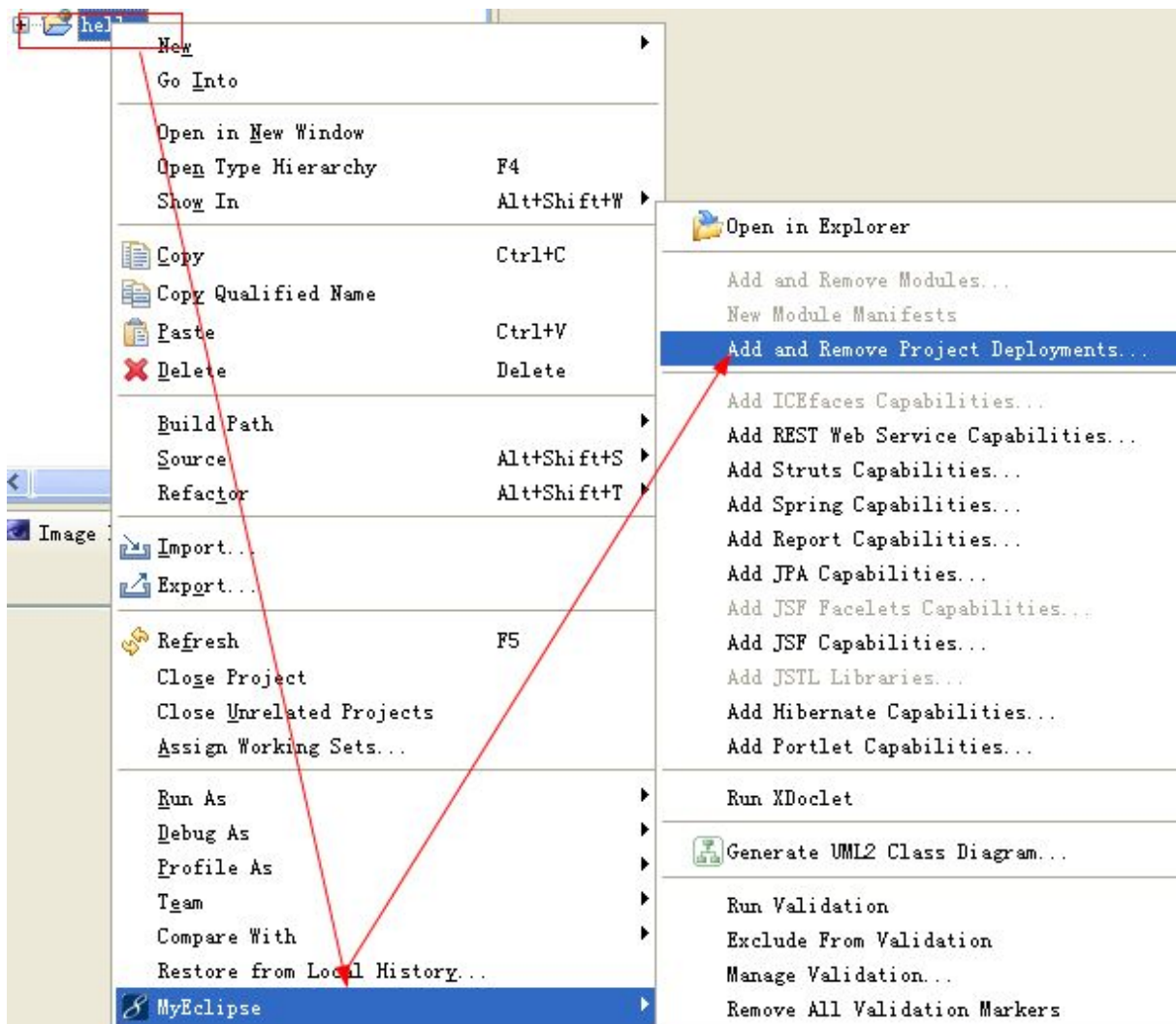


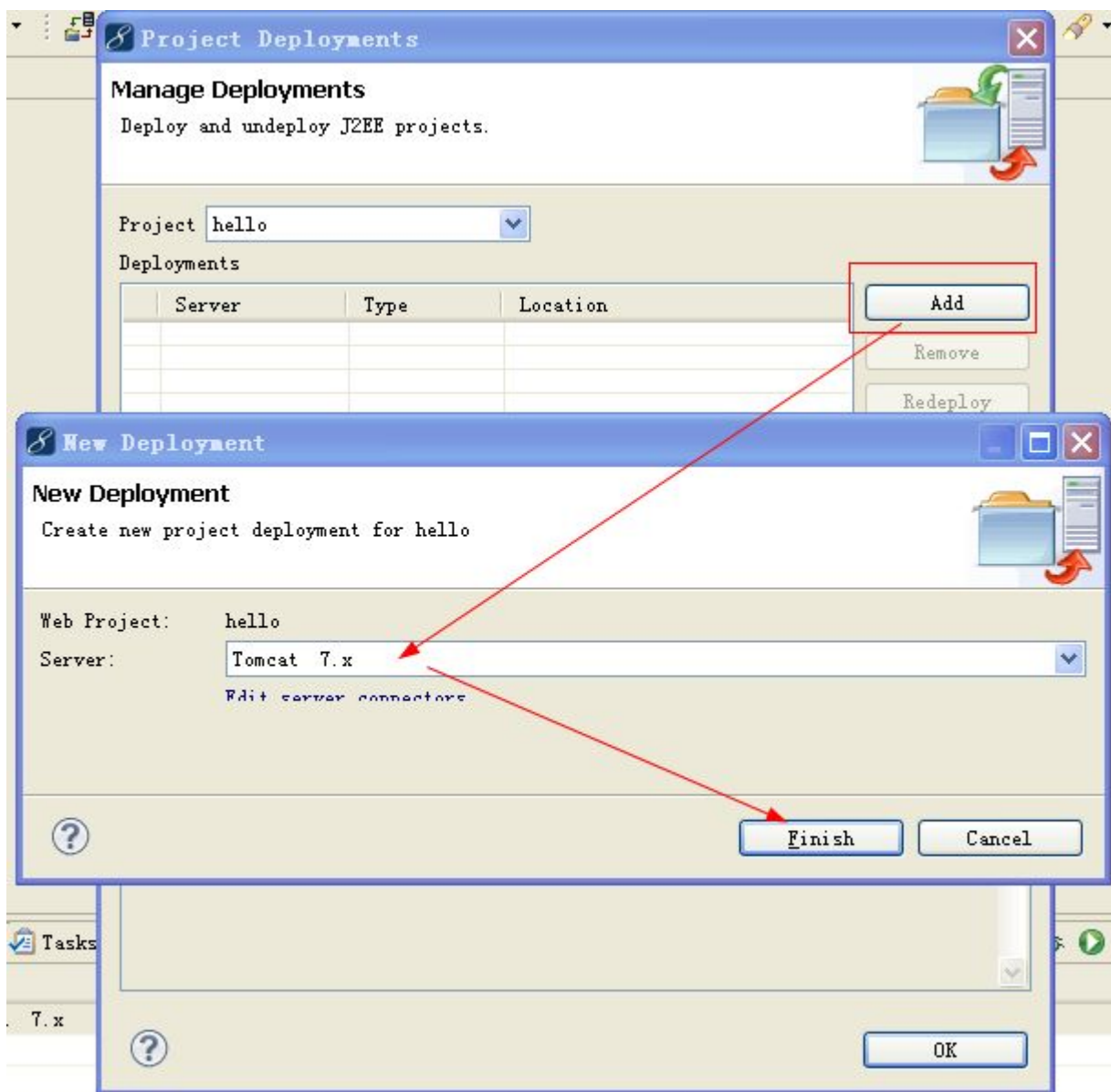
上面两种方式都可以启动 tomcat

6.4 关闭 tomcat

与启动 Tomcat 相同位置下方就是 Stop Server，即可关闭 tomcat 了。

6.5 发布项目到 tomcat 的 webapps 目录

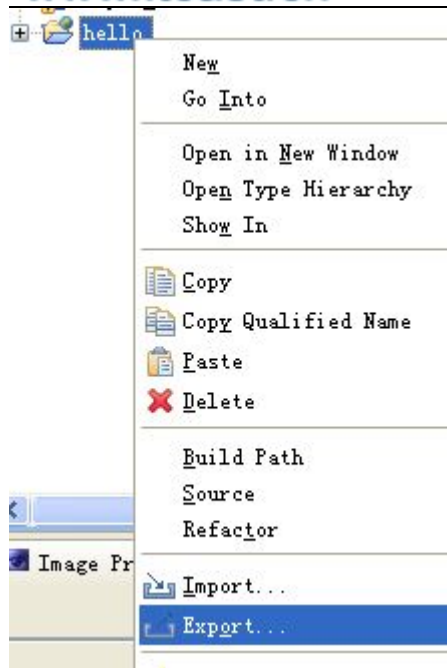


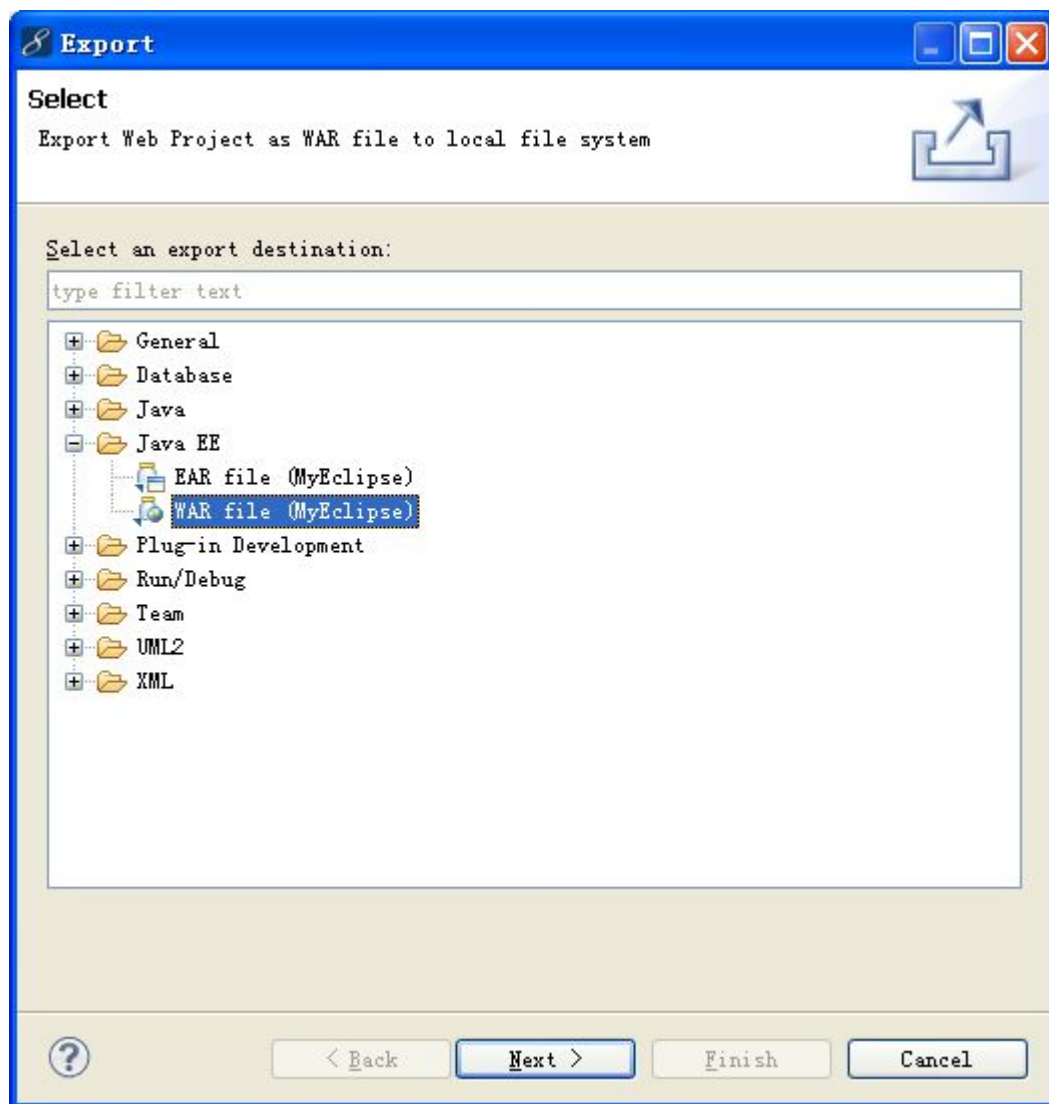


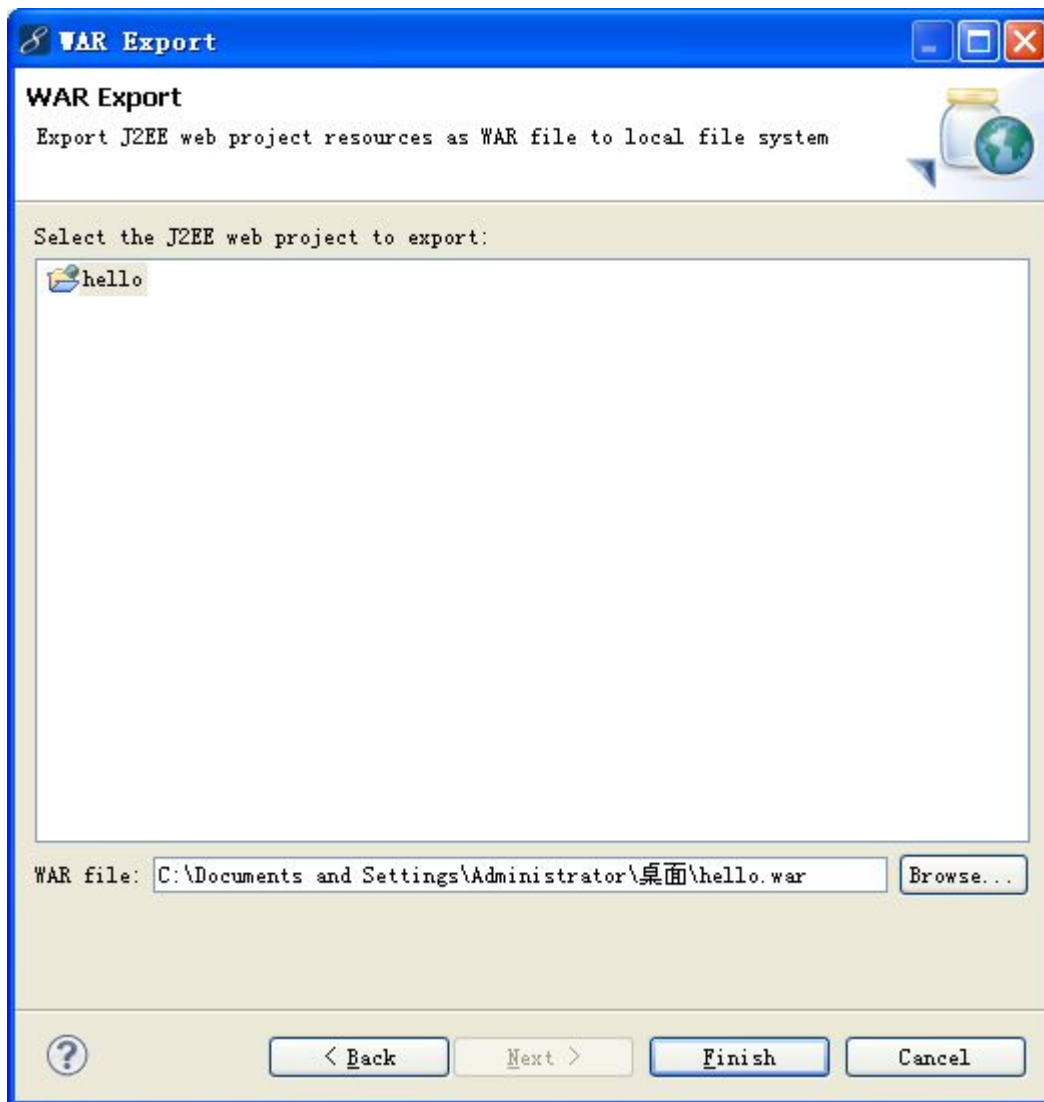
项目发布后，就是把项目的 WebRoot 目录 copy 到 Tomcat 的 webapps 目录，并把 WebRoot 重命名为项目名称，即 hello。所以在 Tomcat 的 webapps 下会多出一个文件夹 hello。

6.6 打 war 包

JavaSE 程序可以打包成 Jar 包，而 JavaWeb 程序可以打包成 war 包。然后把 war 发布到 Tomcat 的 webapps 目录下，Tomcat 会在启动时自动解压 war 包。







HTTP 协议（重点）

协议：协议的甲乙双方，就是客户端（浏览器）和服务端！

理解成双方通信的格式！

- 请求协议；
- 响应协议；

1 安装 HttpWatch

HttpWatch 是专门为 IE 浏览器提供的，用来查看 HTTP 请求和响应内容的工具。而 FireFox 上需要安装 FireBug 软件。如果你使用的是 Chrome，那么就不用自行安装什么工具了，因为它自身就有查看请求和响应内容的功能！

HttpWatch 和 FireBug 这些工具对浏览器而言不是必须的，但对我们开发者是很有帮助的，通过查看 HTTP 请求响应内容，可以使我们更好的学习 HTTP 协议。

2 HTTP 概述

HTTP (hypertext transport protocol)，即超文本传输协议。这个协议详细规定了浏览器和万维网服务器之间互相通信的规则。

HTTP 就是一个通信规则，通信规则规定了客户端发送给服务器的内容格式，也规定了服务器发送给客户端的内容格式。其实我们要学习的就是这个两个格式！客户端发送给服务器的格式叫“请求协议”；服务器发送给客户端的格式叫“响应协议”。

3 请求协议

请求协议的格式如下：

请求首行；
请求头信息；
空行；
请求体。

浏览器发送给服务器的内容就是这个格式的，如果不是这个格式服务器将无法解读！在 HTTP 协议中，请求有很多请求方法，其中最为常用的就是 GET 和 POST。不同的请求方法之间的区别，后面会一点一点的介绍。

3.1 GET 请求

打开 IE，在访问 hello 项目的 index.jsp 之前打开 HttpWatch，并点击“Record”按钮。然后访问 index.jsp 页面。查看请求内容如下：

```
GET /hello/index.jsp HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: GB2312,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Cookie: JSESSIONID=369766FDF6220F7803433C0B2DE36D98
```

加***的要记起来

- ***GET /hello/index.jsp HTTP/1.1: GET 请求，请求服务器路径为/hello/index.jsp，协议为 1.1；
- *** Host:localhost: 请求的主机名为 localhost；
- ***User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0: 与浏览器和

OS 相关的信息。有些网站会显示用户的系统版本和浏览器版本信息，这都是通过获取 User-Agent 头信息而来的；

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8: 告诉服务器，当前客户端可以接收的文档类型，其实这里包含了*/*, 就表示什么都可以接收；
- ***Accept-Language: zh-cn,zh;q=0.5: 当前客户端支持的语言，可以在浏览器的工具→选项中找到语言相关信息；
- Accept-Encoding: gzip, deflate: 支持的压缩格式。数据在网络上传递时，可能服务器会把数据压缩后再发送；
- Accept-Charset: GB2312,utf-8;q=0.7,*;q=0.7: 客户端支持的编码；
- Connection: keep-alive: 客户端支持的链接方式，保持一段时间链接，默认为 3000ms；
- Cookie: JSESSIONID=369766FDF6220F7803433C0B2DE36D98: 因为不是第一次访问这个地址，所以会在请求中把上一次服务器响应中发送过来的 Cookie 在请求中一并发送过去；这个 Cookie 的名字为 JSESSIONID，然后在讲会话是讲究它！

3.2 POST 请求

为了演示 POST 请求，我们需要修改 index.jsp 页面，即添加一个表单：

```
<form action="" method="post">
  关键字: <input type="text" name="keyword"/>
  <input type="submit" value="提交"/>
</form>
```

关键字:

打开 HttpWatch，输入 hello 后点击提交，查看请求内容如下：

```
POST /hello/index.jsp HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/msword, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/x-ms-application, application/x-ms-xbap,
application/vnd.ms-xpsdocument, application/xaml+xml, */*
Referer: http://localhost:8080/hello/index.jsp
Accept-Language: zh-cn,en-US;q=0.5
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; InfoPath.2; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: localhost:8080
Content-Length: 13
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=E365D980343B9307023A1D271CC48E7D

keyword=hello
```

POST 请求是可以有体的，而 GET 请求不能有请求体。

- Referer: http://localhost:8080/hello/index.jsp: 请求来自哪个页面，例如你在百度上点击链接接到了这里，那么 Referer:http://www.baidu.com; 如果你是在浏览器的地址栏中直接输入的地址，那么就没有 Referer 这个请求头了；
- Content-Type: application/x-www-form-urlencoded: 表单的数据类型，说明会使用 url 格式编码数据；url 编码的数据都是以 “%” 为前缀，后面跟随两位的 16 进制，例如 “传智” 这两个字使用 UTF-8 的 url 编码用为 “%E4%BC%A0%E6%99%BA”；
- Content-Length:13: 请求体的长度，这里表示 13 个字节。
- keyword=hello: 请求体内容！hello 是在表单中输入的数据，keyword 是表单字段的名称。

Referer 请求头是比较有用的一个请求头，它可以用来做统计工作，也可以用来做防盗链。

统计工作：我公司网站在百度上做了广告，但不知道在百度上做广告对我们网站的访问量是否有影响，那么可以对每个请求中的 Referer 进行分析，如果 Referer 为百度的很多，那么说明用户都是通过百度找到我们公司网站的。

防盗链：我公司网站上有一个下载链接，而其他网站盗链了这个地址，例如在我网站上的 index.html 页面中有一个链接，点击即可下载 JDK7.0，但有某个人的微博中盗链了这个资源，它也有一个链接指向我们网站的 JDK7.0，也就是说登录它的微博，点击链接就可以从我网站上下载 JDK7.0，这导致我们网站的广告没有看，但下载的却是我网站的资源。这时可以使用 Referer 进行防盗链，在资源被下载之前，我们对 Referer 进行判断，如果请求来自本网站，那么允许下载，如果非本网站，先跳转到本网站看广告，然后再允许下载。

5 响应协议

5.1 响应内容

响应协议的格式如下：

响应首行；
响应头信息；
空行；
响应体。

响应内容是由服务器发送给浏览器的内容，浏览器会根据响应内容来显示。

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Content-Length: 724
Set-Cookie: JSESSIONID=C97E2B4C55553EAB46079A4F263435A4; Path=/hello
Date: Wed, 25 Sep 2012 04:15:03 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
```

```
<base href="http://localhost:8080/hello/">

<title>My JSP 'index.jsp' starting page</title>
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->
</head>

<body>
<form action="" method="post">
  关键字: <input type="text" name="keyword"/>
  <input type="submit" value="提交"/>
</form>
</body>
</html>
```

加***号的记起来

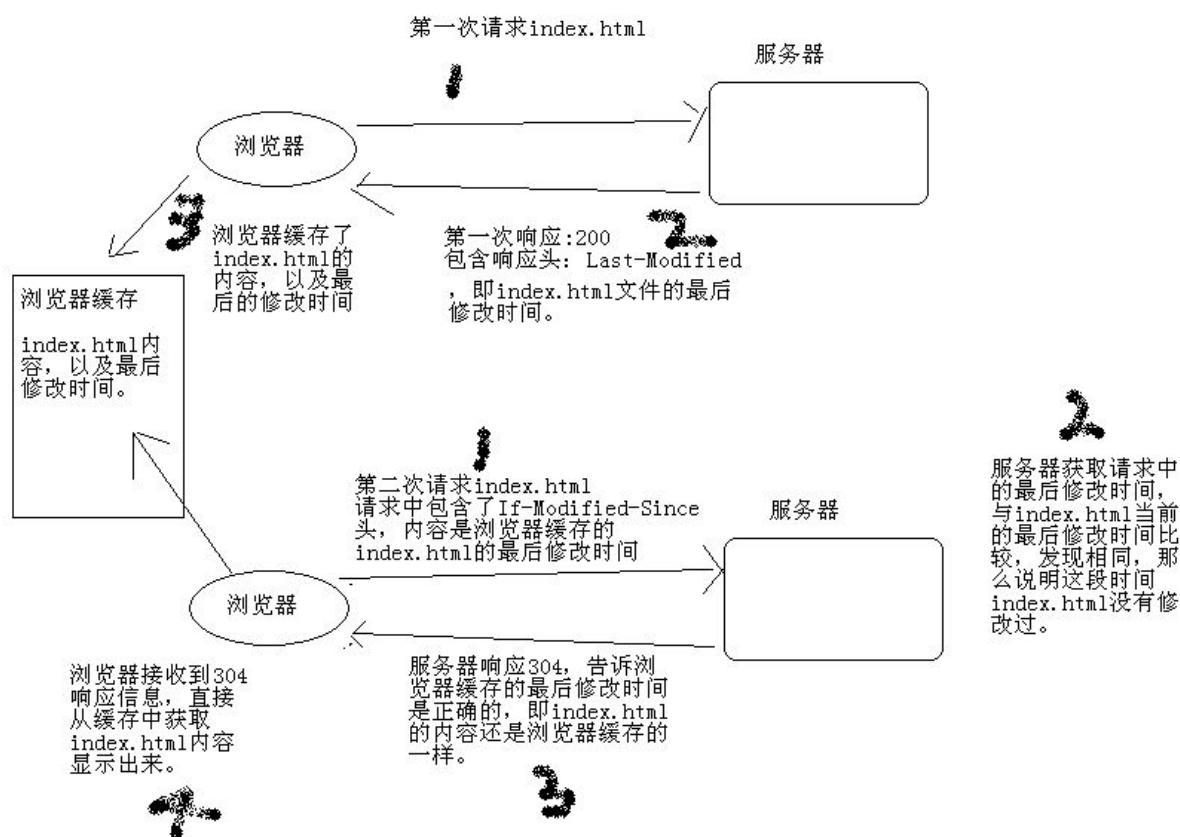
- ***HTTP/1.1 200 OK: 响应协议为 HTTP1.1, 状态码为 200, 表示请求成功, OK 是对状态码的解释;
- Server: Apache-Coyote/1.1: 服务器的版本信息;
- *** Content-Type: text/html;charset=UTF-8: 响应体使用的编码为 UTF-8;
- Content-Length: 724: 响应体为 724 字节;
- Set-Cookie: JSESSIONID=C97E2B4C5553EAB46079A4F263435A4; Path=/hello: 响应给客户端的 Cookie;
- Date: Wed, 25 Sep 2012 04:15:03 GMT: 响应的时间, 这可能会有 8 小时的时区差;

5.2 响应码

响应头对浏览器来说很重要, 它说明了响应的真正含义。例如 200 表示响应成功了, 302 表示重定向, 这说明浏览器需要再发一个新的请求。

- 200: 请求成功, 浏览器会把响应体内容 (通常是 html) 显示在浏览器中;
- 404: 请求的资源没有找到, 说明客户端错误的请求了不存在的资源;
- 500: 请求资源找到了, 但服务器内部出现了错误;
- 302: 重定向, 当响应码为 302 时, 表示服务器要求浏览器重新再发一个请求, 服务器会发送一个响应头 Location, 它指定了新请求的 URL 地址;
- 304: 当用户第一次请求 index.html 时, 服务器会添加一个名为 Last-Modified 响应头, 这个头说明了 index.html 的最后修改时间, 浏览器会把 index.html 内容, 以及最后响应时间缓存下来。当用户第二次请求 index.html 时, 在请求中包含一个名为 If-Modified-Since 请求头, 它的值就是第一次请求时服务器通过 Last-Modified 响应头发送给浏览器的值, 即 index.html

- 最后的修改时间，If-Modified-Since 请求头就是在告诉服务器，我这里浏览器缓存的 index.html 最后修改时间是这个，您看看现在的 index.html 最后修改时间是不是这个，如果还是，那么您就不用再响应这个 index.html 内容了，我会把缓存的内容直接显示出来。而服务器端会获取 If-Modified-Since 值，与 index.html 的当前最后修改时间比对，如果相同，服务器会发响应码 304，表示 index.html 与浏览器上次缓存的相同，无需再次发送，浏览器可以显示自己的缓存页面，如果比对不同，那么说明 index.html 已经做了修改，服务器会响应 200。



响应头:

- Last-Modified: 最后的修改时间;

请求头:

- If-Modified-Since: 把上次请求的 index.html 的最后修改时间还给服务器;

状态码: 304, 比较 If-Modified-Since 的时间与文件真实的时间一样时, 服务器会响应 304, 而且不会有响正文, 表示浏览器缓存的就是最新版本!

5.3 其他响应头

告诉浏览器不要缓存的响应头:

- Expires: -1;
- Cache-Control: no-cache;
- Pragma: no-cache;

自动刷新响应头, 浏览器会在 3 秒之后请求 <http://www.itcast.cn>:

- Refresh: 3;url=<http://www.itcast.cn>

5.4 HTML 中指定响应头

在 HTML 页面中可以使用 `<meta http-equiv="" content="">` 来指定响应头, 例如在 `index.html` 页面中给出 `<meta http-equiv="Refresh" content="3;url=http://www.itcast.cn">`, 表示浏览器只会显示 `index.html` 页面 3 秒, 然后自动跳转到 <http://www.itcast.cn>。