# day01

JAVA分成三种类型：

JavaSE:数据类型 流程控制语句 常用使用类集合、string IO

JavaEE：12组件（举例：jsp servlet listener filter）

JaveME:手持设备 弱化

## 软件系统体系结构

### 常见软件系统体系结构Browser/Server、C/S

Tomcat:JAVAEE使用服务器

#### Client/Server：客户端一定是先安装软件，然后才能使用。

Windows系统：c/s, 打补丁，更新

特点：需要先安装，才能使用的软件，一般都是C/S软件

* 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器端两端，比较麻烦；
* 优点：安全性比较好。

#### B/S：基于浏览器访问的程序。主流的一种软件开发结构

* B/S结构即浏览器/服务器（Browser/Server）；例如：淘宝网、百度、新浪、京东…
* 优点：只需要编写服务器端程序；不需要安装
* 缺点：安全性较差。防范注入：SQL注入 js注入 暴力破解
  + 举例：12306设置验证码
  + 暴力登录

### 2　WEB资源

#### 2.1　Web资源介绍

* html：静态资源；JavaScript不就可以与用户互动了？？？
  + 特点：不能连接数据库，即不能提供动态数据变化
  + 举例：mysite.html🡺img table 文字

新浪：新闻 实时

* JSP/Servlet：动态资源。

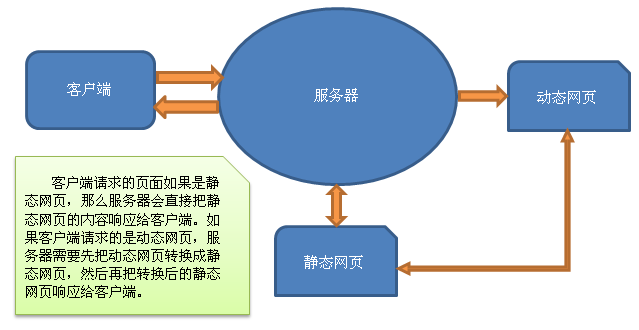
当然，除了JavaWeb程序，还有其他Web程序，例如：ASPX、PHP等。

#### 2.2　静态资源和动态资源区别

客户端：B 浏览器就是客户端.

S服务器：是一台安装了mysql服务器软件的电脑

Web服务器：电脑安装web服务器的软件



#### 2.3　访问Web资源

打开浏览器，输入URL：标识万维网上面唯一的一台电脑通过IP地址

* 协议名://域名:端口/路径，例如：http://www.baidu.com: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程序而言，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）：免费开源 当前应用最广的JavaWeb服务器，支持JSP以及Servlet规范；
* JBoss（Redhat红帽）：支持JavaEE，应用比较广；EJB容器
* GlassFish（Orcale）：Oracle开发JavaWeb服务器，应用不是很广；
* Weblogic（Orcale）：要钱的！支持JavaEE，适合大型项目；
* Websphere（IBM）：要钱的！支持JavaEE规范，适合大型项目；

备注：JavaEE规范包含13种技术：JDBC、JNDI、EJB、RMI、JSP、Java Servlet、XML、JMS、JTA、JavaMail…..

## Tomcat（有重点）

### Tomcat概述

1. 解压路径不能出现中文
2. path必须有java\_home变量，如果没有java\_home，tomcat启动，一闪而过

就是一种Web应用程序服务器软件。

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Tomcat9，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；=== 很少见
* 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。
  + 实际应用其实就一个

#### 2.1　Tomcat目录结构

安装版Tomcat的安装过程请参考素材附件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，因为我们需要经常进入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，我们必须要配置环境变量：

* 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。
  + 缺点：确定一个唯一的tomcat，版本较多，混淆
  + 特点：安装版tomcat一台电脑只有一个。快速的通过startup.bat启动服务器
* 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：

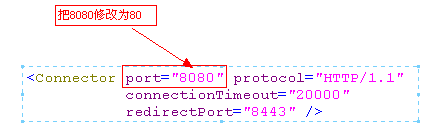
* 点击startup.bat后窗口一闪即消失：检查JAVA\_HOME环境变量配置是否正确；

#### 2.3　进入Tomcat主页

　　访问：http://localhost:8080

#### 2.4　配置端口号

打开%CATALANA\_HOME%\conf\server.xml文件：



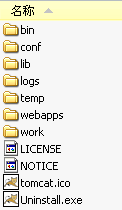
http默认端口号为80，也就是说在URL中不给出端口号时就表示使用80端口。当然你也可以修改为其它端口号。

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。

*备注：*

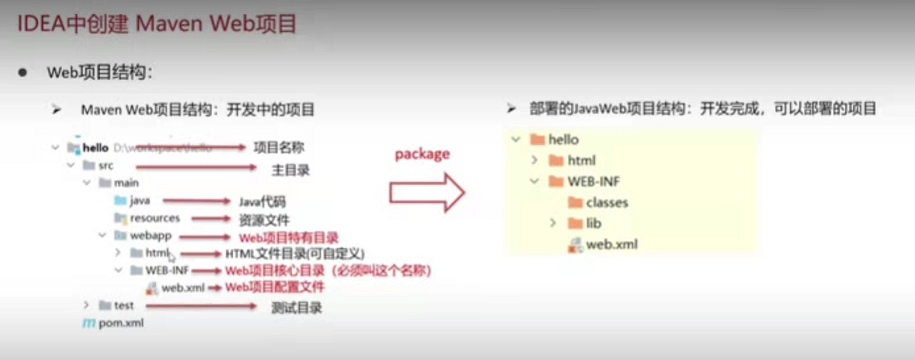
*1、工作中常见的协议端口：http:80 ftp:23 smtp:25 pop3:110 https:443 mysql:3306*

#### 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，前者是在控制台下启动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，shutdawn.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，再次运行时会生再次生成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）:一定要安装一个web服务器



如何使用浏览器访问tomcat上面的项目：路径写法！！！

静态网站：

网页不能跟数据库交互 html+css+js+jquery

* 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：

Jdbc+jsp 动态项目

* 在webpass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

### 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> |

### 创建动态应用程序

提示：动态项目中才能创建动态资源（）

* 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

|- hello2

|-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东西是安全的；
* 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）====侧重后期的部署

　　原来我们的项目放到webapps下，现在我放到其他盘符中，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G盘下，即G:/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：不推荐使用原因---每次修改server.xml,必须重启tomcat的服务器 本地测试方式可以使用，正式开发中不推荐使用。

server.xml

|  |
| --- |
|  |

1. path：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/hello/index.html>。

* \conf\Catalina\localhost：在该目录下创建hello.xml文件，在该文件中编写<Context>元素，代码如下：

实现步骤：

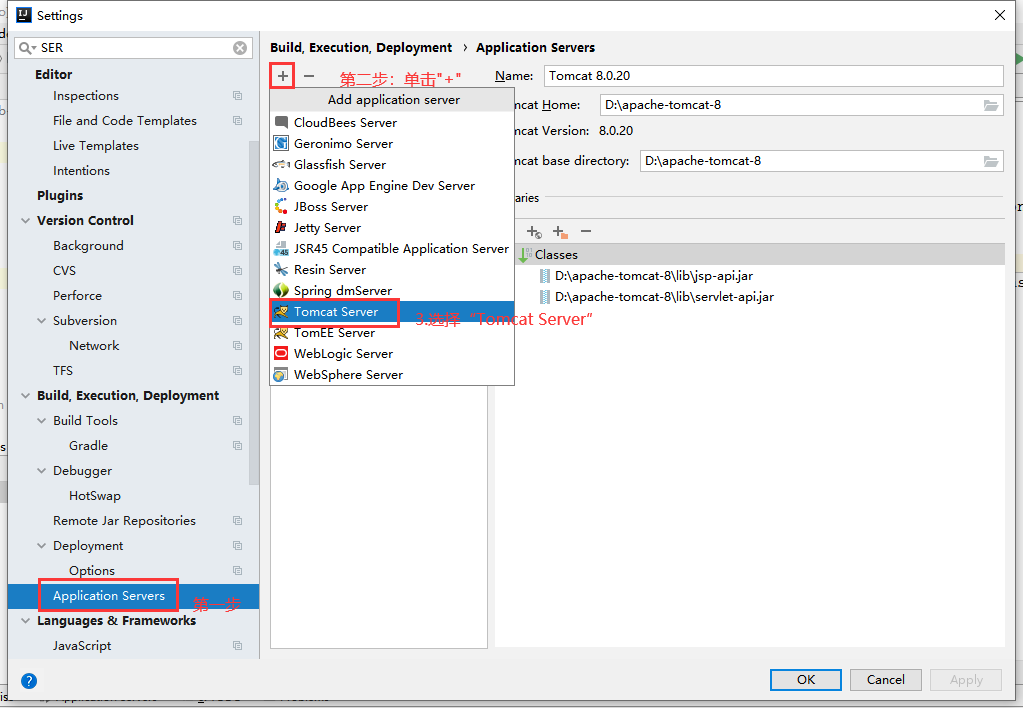
* + - 1. 新建xml文件，文件名称就是你的虚拟路径，类似第一种配置中path属性
      2. Xml文件写<Context docBase=”web项目的实际路径”/>

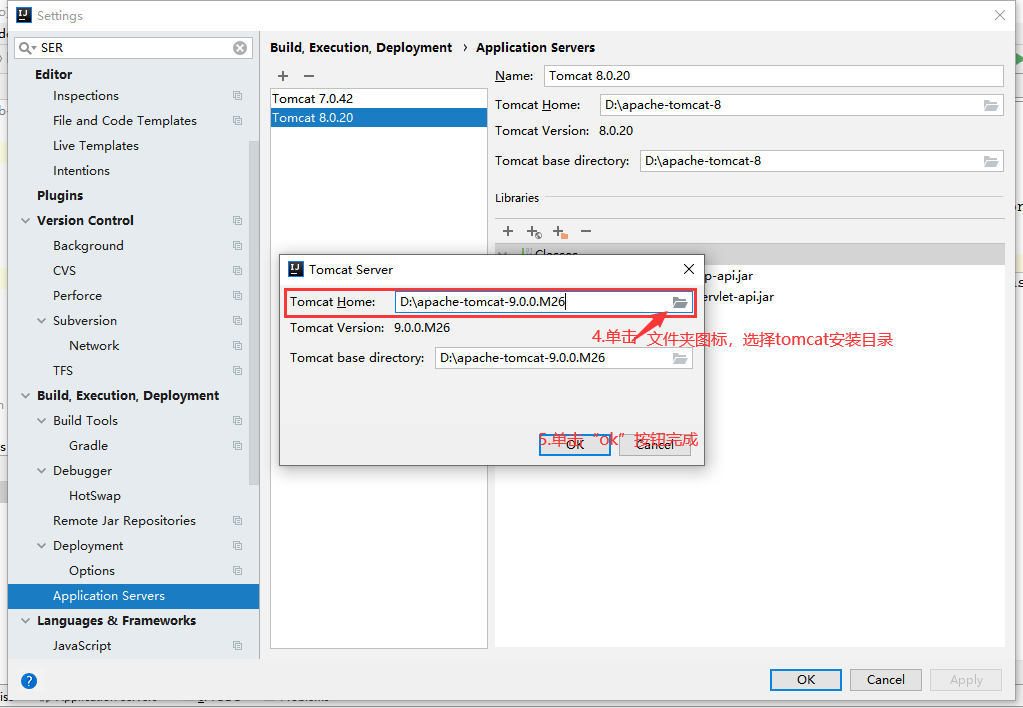
|  |
| --- |
|  |

1. 文件名：指定当前应用的名称；如果希望多级路径访问，文件名称中使用#分割级别
   1. 例如aa#bb#cc.xml，那么访问路径就是：http://localhost:8080/aa/bb/cc/index.html
2. docBase：指定应用的物理位置；

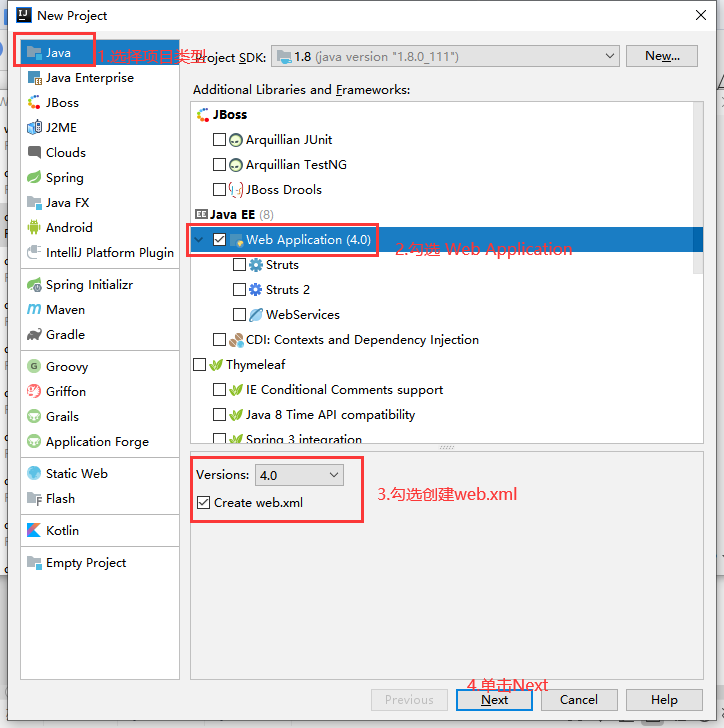
### IDEA创建JavaWeb应用

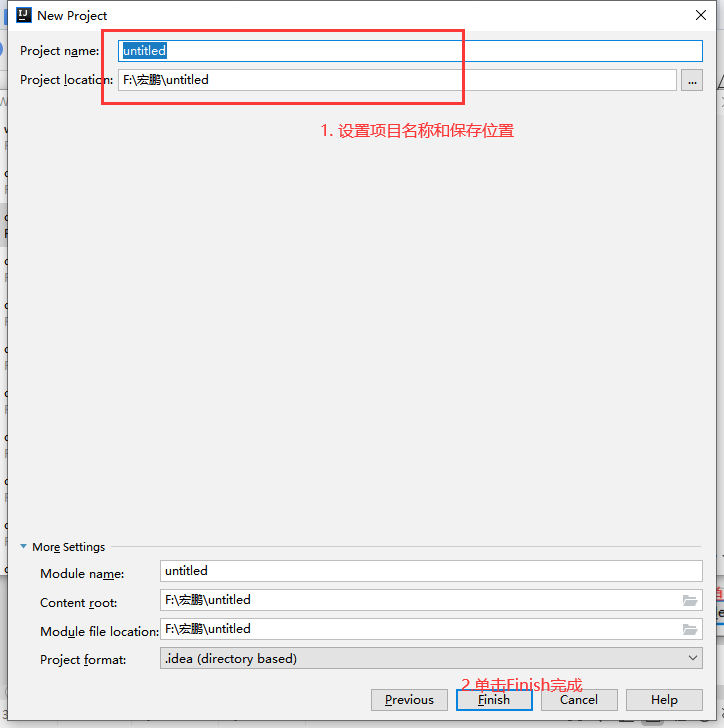
#### 4.1　配置Tomcat

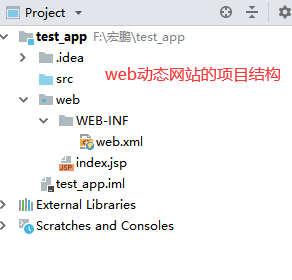




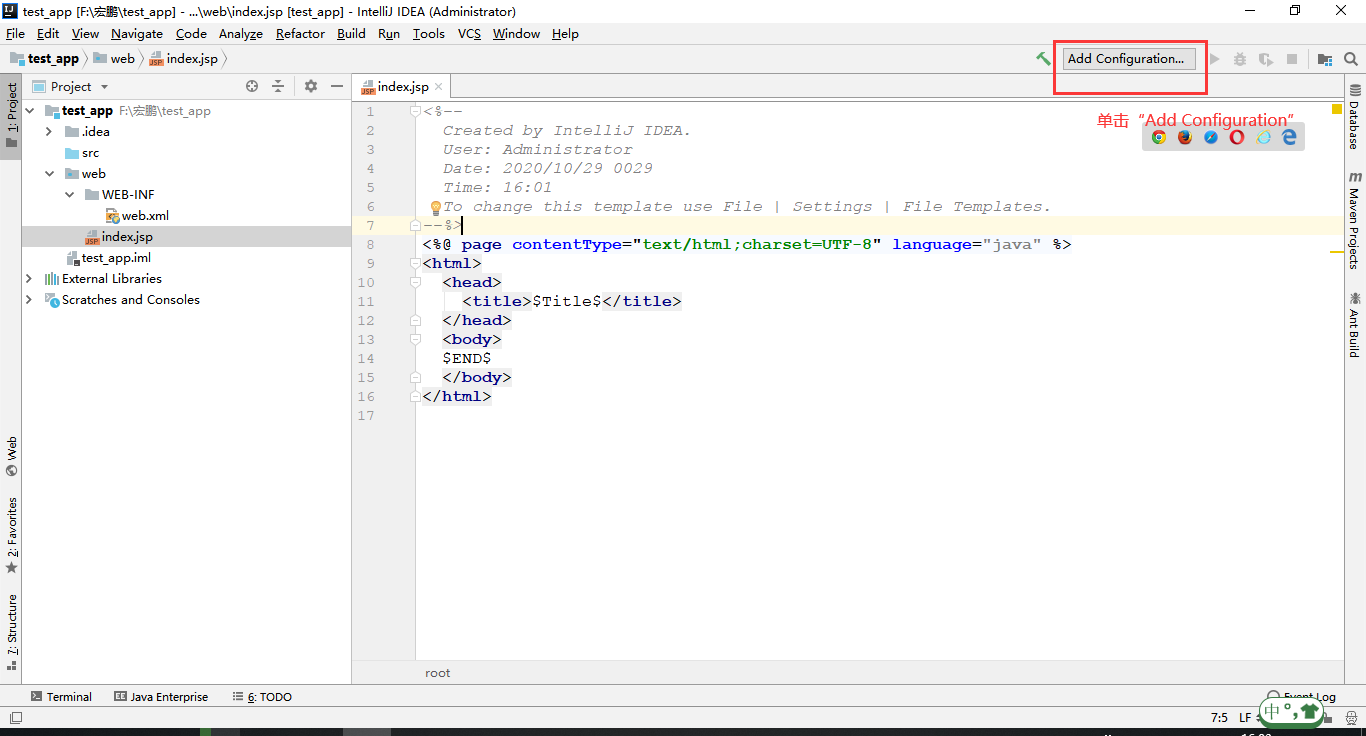
#### 4.2　IDEA创建JavaWeb应用

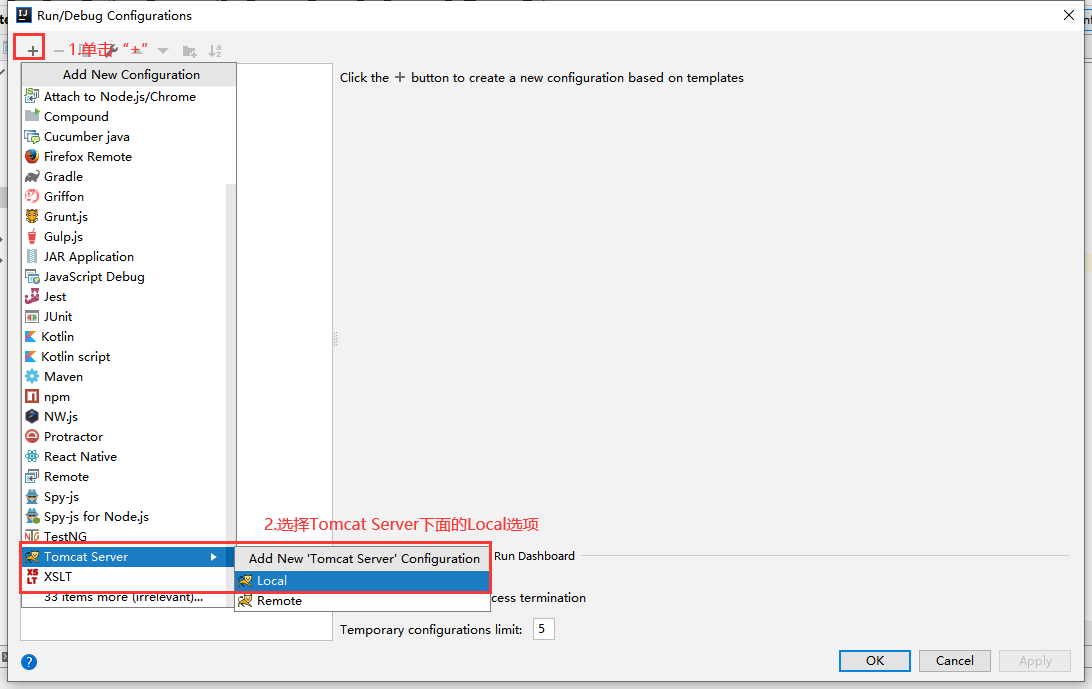


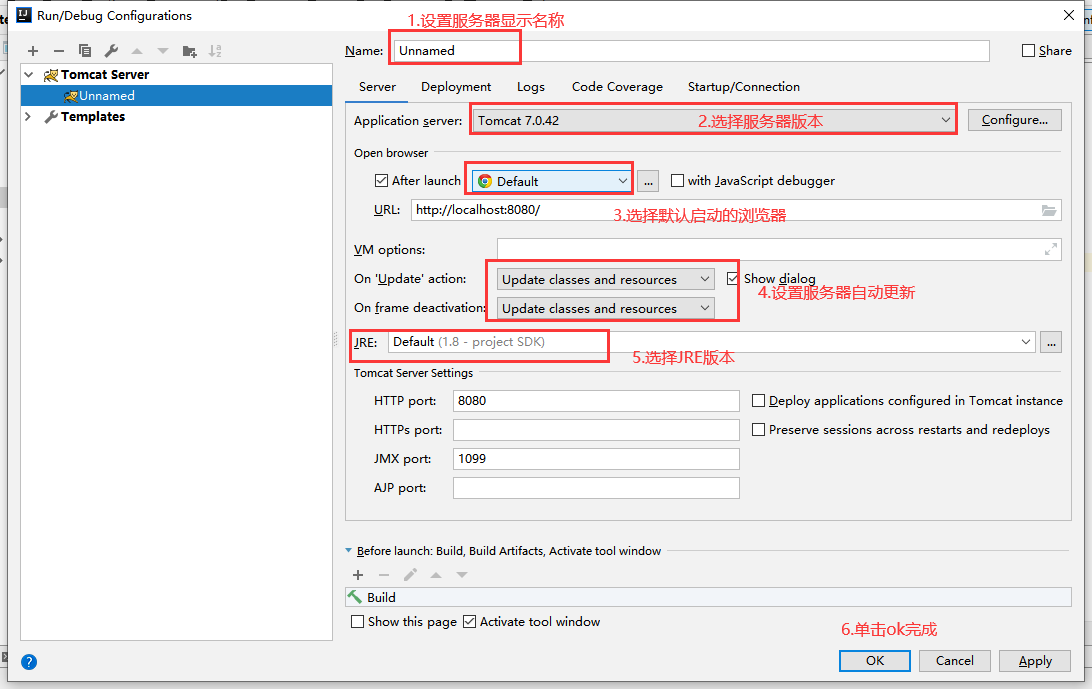




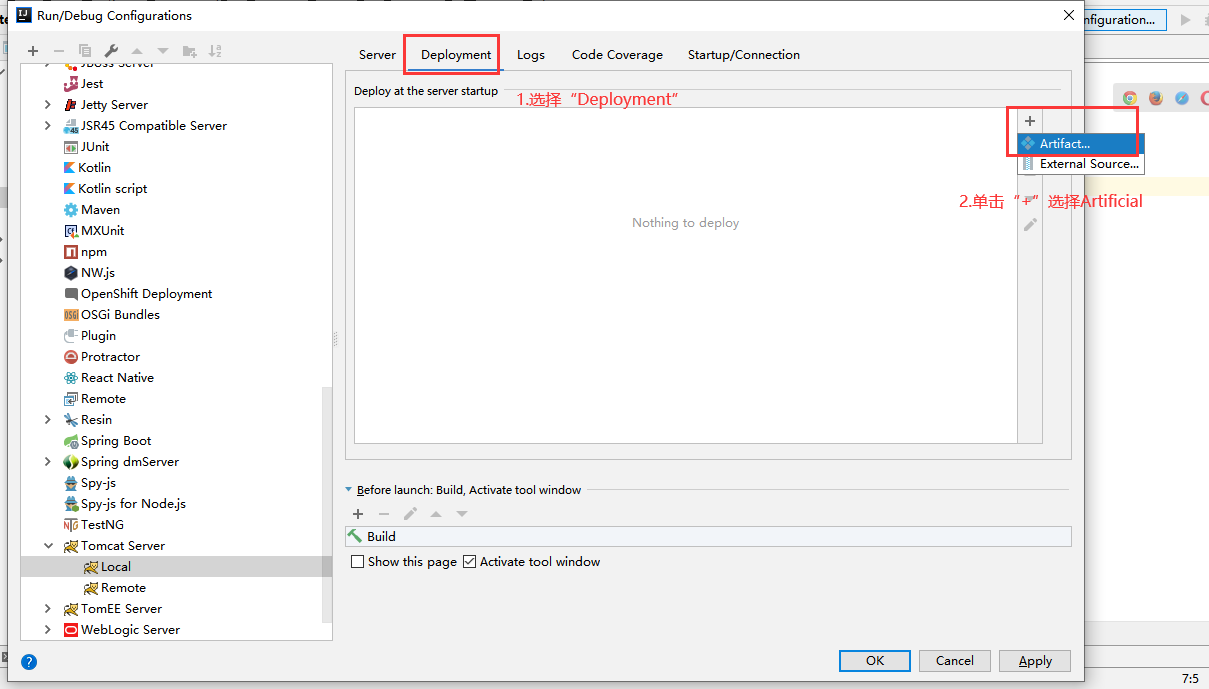
#### 4.3　在项目中配置Tomcat

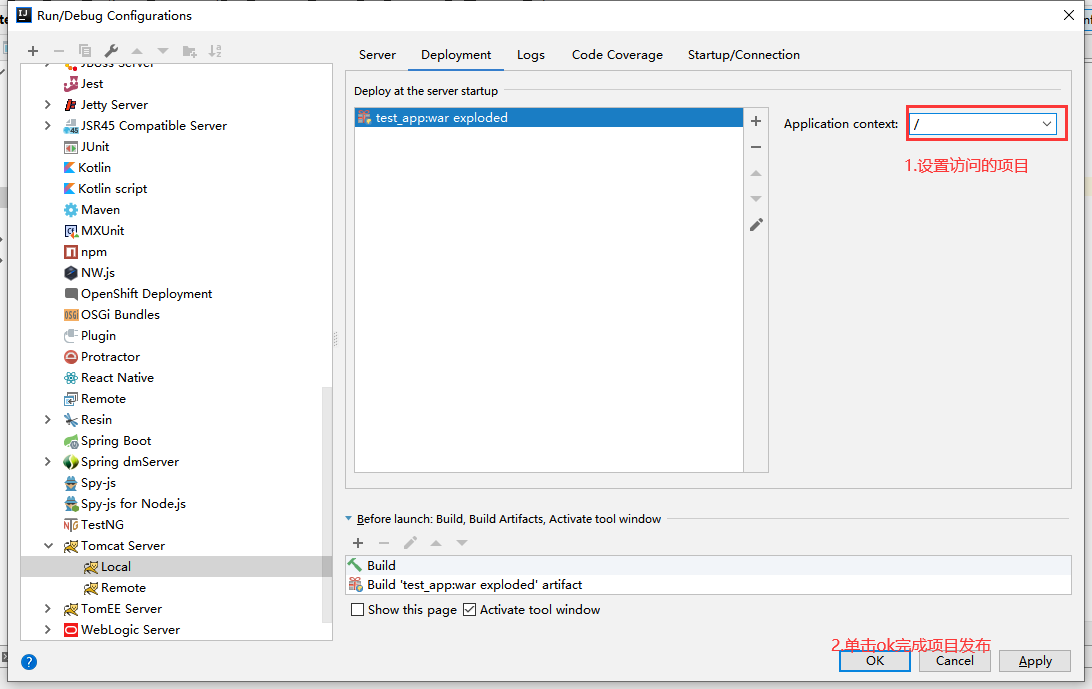




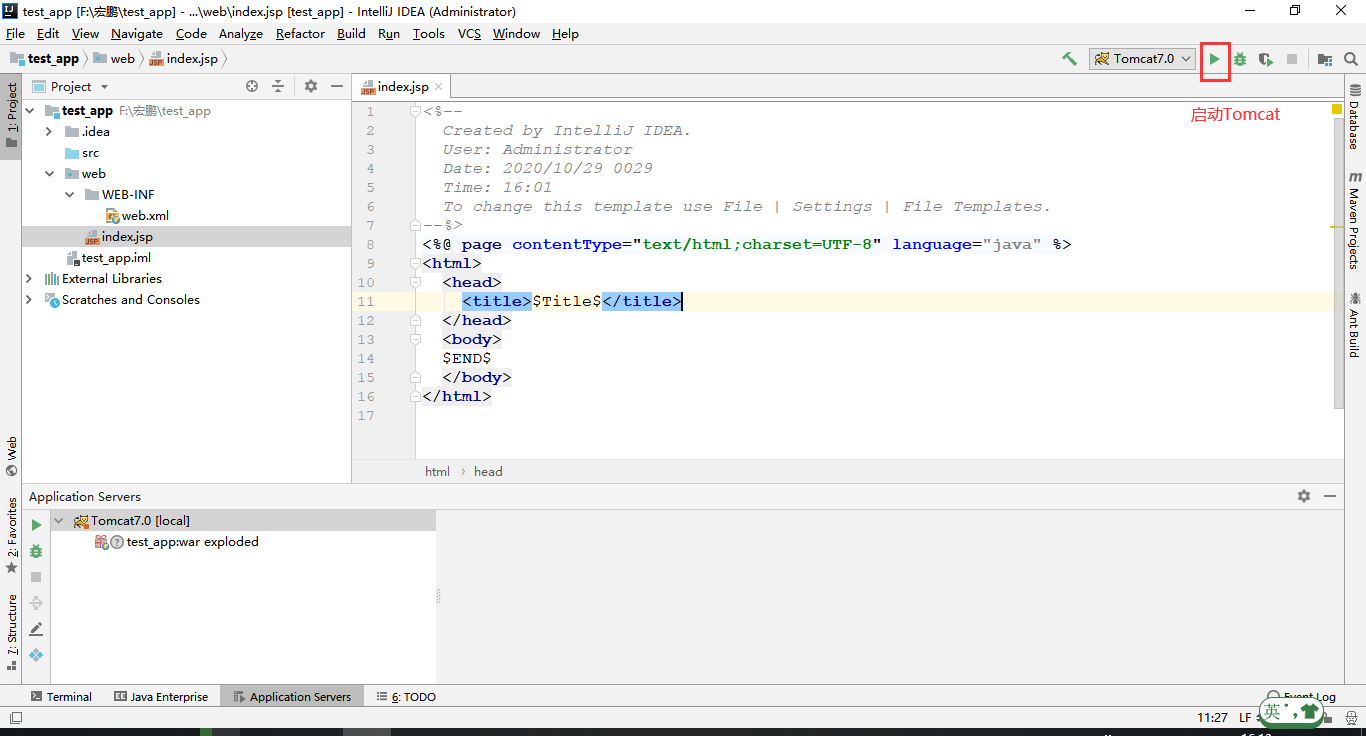


#### 4.4　IDEA发布web项目

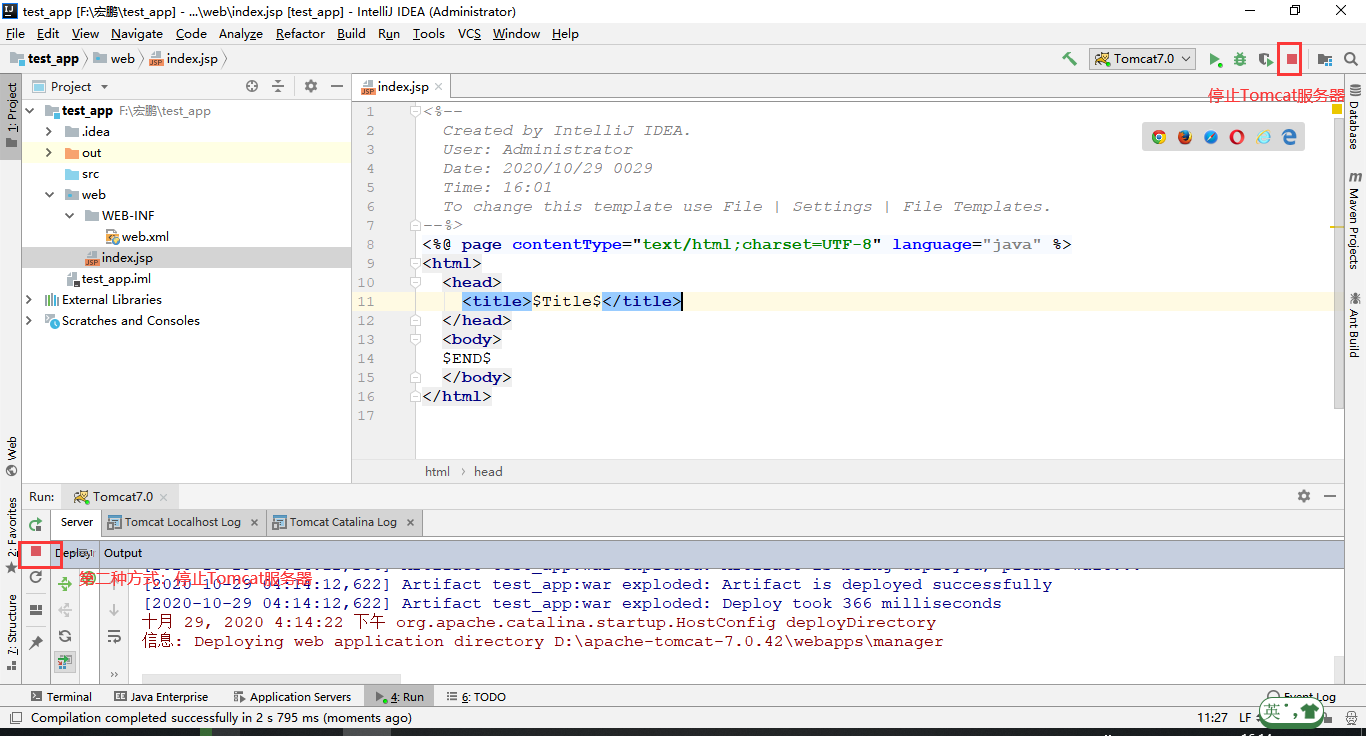




#### 4.5 启动Tomcat



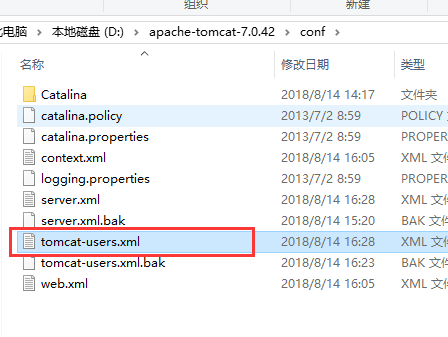
#### 4.5 关闭tomcat

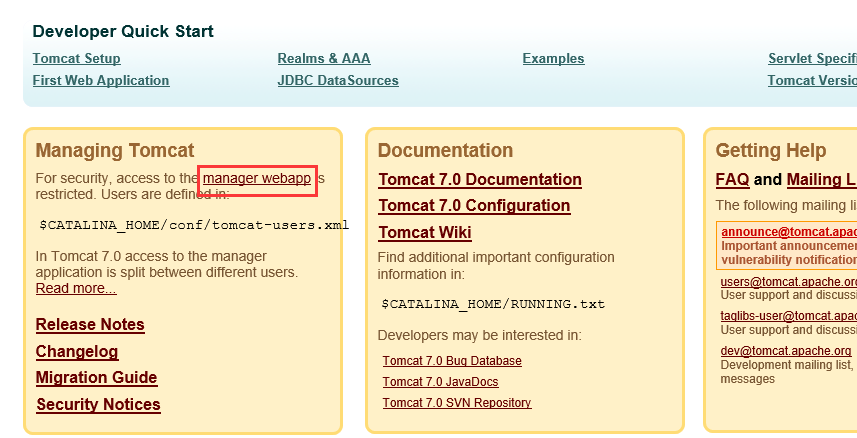


扩展：通过tomcat的主页发布WEB应用。（了解）

需要启用用户

<user username="tomcat" password="tomcat" roles="manager-gui"/>





使用账号和密码登录即可

## HTTP协议（理解）

协议：协议的甲乙双方，就是客户端（浏览器）和服务器！

HTTP协议：理解浏览器和服务器进行交流一种文本格式。  
　　理解成双方通信的格式！

* 请求协议；===》浏览器写给服务器的信要求格式
  + 读懂浏览器给服务器写的信 Request对象
* 响应协议；===》服务器回复浏览器的信要求格式
  + 整理回送浏览器看的内容 Response对象

### 1　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”。

HTTP协议规定两种协议类型：

请求协议：浏览器发给服务器的数据，称为请求协议

响应协议：服务器发给浏览器的数据，称为响应协议

**HTTP协议特点：**

1. 基于TCP协议：面向连接，安全
2. 基于请求-响应模型的：一次请求对应一次响应
3. HTTP协议是无状态的协议：对于事务处理没有记忆能力。每次请求和响应都是独立的
   1. 缺点：多次请求之间不能共享数据
      1. 会话技术：Session Cookie
   2. 优点：速度快

### 2　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。

HTTP协议版本：

http1.1:keep alive 保存连接一小会儿，常用http1.1

http1.0:no-connection 一旦连接成功，立马断开

#### 2.1　GET请求

　　打开火狐的开发者工具，在访问hello项目的index.jsp，并点击“Record”按钮。然后访问index.jsp页面。查看请求内容如下：

GET和POST 区别

Get:数据显示地址栏， key=value&key=value 不安全

固定格式：

请求行

多个请求头

一个空行

**没有请求体**

Post：请求体形式发送服务器

固定格式：

请求行

多个请求头

一个空行

**请求体**

|  |
| --- |
| 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 压缩格式 clob大文本字符2g🡪压缩—>返回浏览器  Accept-Charset: GB2312,utf-8;q=0.7,\*;q=0.7  Connection: keep-alive  Cookie: JSESSIONID=369766FDF6220F7803433C0B2DE36D98 |

* 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在请求中一并发送去过；这个Cookie的名字为JSESSIONID，然后在讲会话是讲它！



#### 2.2　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> |



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“你好”这两个字使用UTF-8的url编码用为“%E4%BD%A0%E5%A5%BD”；
* 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是表单字段的名字。

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。

### 响应协议



#### 3.1　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> |

* 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=C97E2B4C55553EAB46079A4F263435A4; 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；

#### 3.2　响应码==============重点

##### 常见状态码



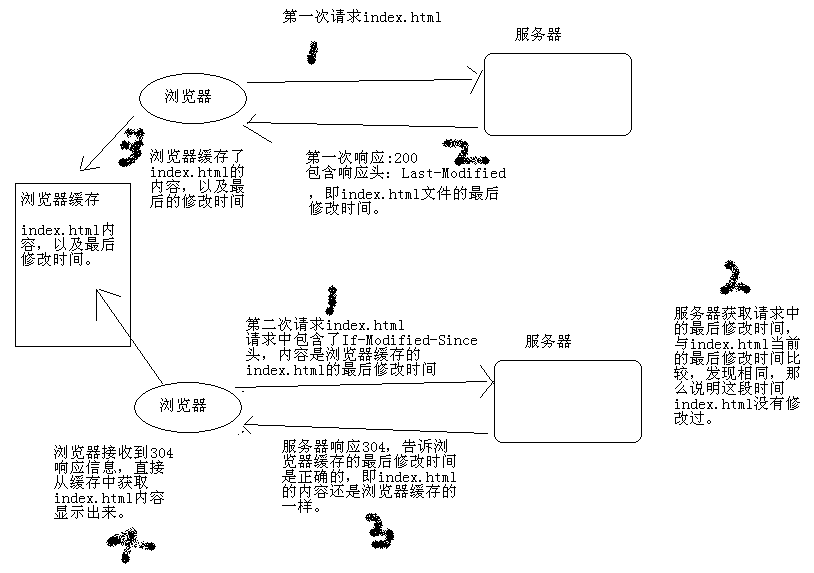
|  |  |
| --- | --- |
| 状态码分类 | 说明 |
| 1XX | 响应中：临时状态码，通常表示请求已经接受，告诉客户端可以继续请求或者如果它已经完成则可以忽略它 |
| 2XX | 成功响应：表示请求已经被成功接收，处理已经完成 |
| 3XX | 重定向：重定向到其他地方：它让客户端再发起一个请求用来完成整个处理 |
| 4XX | 客户端错误：处理发生异常，责任在客户端，如：404客户端请求一个不存在的资源；403客户端未被授权，禁止访问等 |
| 5XX | 服务器端错误：处理发生错误，责任在服务器端，如：500服务器端程序抛出异常；路由出错；HTTP版本不支持等 |

http://tools.jb51.net/table/http\_status\_code



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；=========通常与代码没有关系
* 500：请求资源找到了，但服务器内部出现了错误；java程序员代码问题=======一定是代码错误
* 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！

#### 3.3　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.baidu.com：

* Refresh: 3;url=http://www.baidu.com

#### 3.4　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.baidu.com">，表示浏览器只会显示index.html页面3秒，然后自动跳转到http://www.baidu.com。

预习：

Servlet的应用