1. JAVA WEB前奏(了解)
   1. C/S结构

C/S结构即客户端/服务器（Client/Server），例如QQ就是C/S结构的。我们每个人电脑上都需要安装QQ的客户端，客户端软件向服务器端软件发送请求，服务器端给客户端返回响应。

但是，因为客户端需要不断的更新，用户使用起来就比较麻烦。现在的C/S结构的客户端已经好多了，可以提示用户更新，用户只需要点击更新就可以了。最早时需要用户自己去官网上下载最新客户端，然后把老版本卸载，再去安装新版本。



* 1. B/S结构

B/S结构即浏览器/服务器（Browser/Server），例如网站都是B/S结构的。当然，网站也只是B/S结构体系软件中的一种而已。网上银行也都是B/S结构的！

B/S结构的好处在于不需要用户更新客户端，客户端只需要有一个浏览器就OK。当软件需要更新时，开发人员只需要关心服务器端就行。就像你在浏览器中访问百度时，发现百度的主页发生了变化，但你也不需要更新什么。



* 1. 静态网页和动态网页

早期在Internet上都是静态网页，即html页面。静态网页的数据都是不能自动变化的，例如网页上的当前用户信息：“欢迎XXX登录本系统”，这说明当前用户名为XXX。如果有其他用户登录系统就应该显示其他用户的名字，而且无需修改页面！说白一点，用户名应该是变量，而不是常量！这是静态网页做不到的！

动态网页可以中包含变量，数据的变化就不是问题了。但动态网页只能在服务器端使用，客户端浏览器只能识别静态网页。所以，如果用户请求的是动态网页，那么就需要服务器先把动态网页转换成静态网页发送给客户端浏览器！

动态网页必须有服务器来处理。

* 1. 常见的网页
* 静态网页：

htm、html

* 动态网页：

php、asp、aspx

jsp、.do、.action

我们要学习的是JSP（Java server pages），与其相关的是do（struts1）、action（struts2）

* 1. 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。

知名Java Web服务器

* **Tomcat（Apache）**：用来开发学习使用；免费，开源
* JBoss（Redhat红帽）：；
* Weblogic（Orcale）：要钱的！使用Tomcat开发，项目写完了，放到Weblogic上运行；
* Websphere（IBM）：要钱的！与Weblogic相似。

1. Tomcat安装(熟练)
   1. Tomcat概述

Tomcat服务器由Apache提供，开源免费。开发学习使用最佳选择。安装Tomcat之前需要先安装JDK，其实无论哪一种JavaWeb服务器都需要先**安装JDK**。我们现在使用的Tomcat版本可以是Tomcat8或Tomcat9。

Tomcat6支持Servlet2.5；

Tomcat7,8,9支持Servlet3.0；

* 1. Tomcat的作用

当用户通过发送过来一个请求后，Tomcat来接收请求，然后给用户返回响应。通常用户都是通常浏览器发出请求，然后浏览器的响应都是html代码。当然，这也不是全部，有时用户可能会通过其他手段向服务器发送请求，而Tomcat返回的响应也可能不是html，也可能是一个文件，一张图片等。



* 1. Tomcat安装

解压版Tomcat 只需要解压就能使用，无需安装，这样一台电脑上可以安装多个Tomcat。

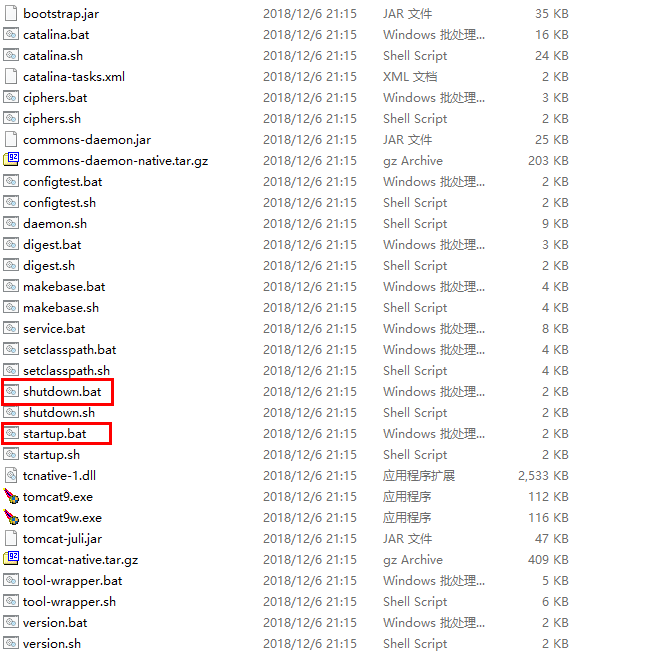
启动解压版的Tomcat有点不同，你需要先去配置环境变量：

JAVA\_HOME：没什么可多说的，如果你这都不会。

启动：在Tomcat安装目录中的bin目录中找到startup.bat文件，然后双击之。

停止：在Tomcat安装目录中的bin目录中找到shutdown.bat文件，然后双击之。

建议使用解压版！！！

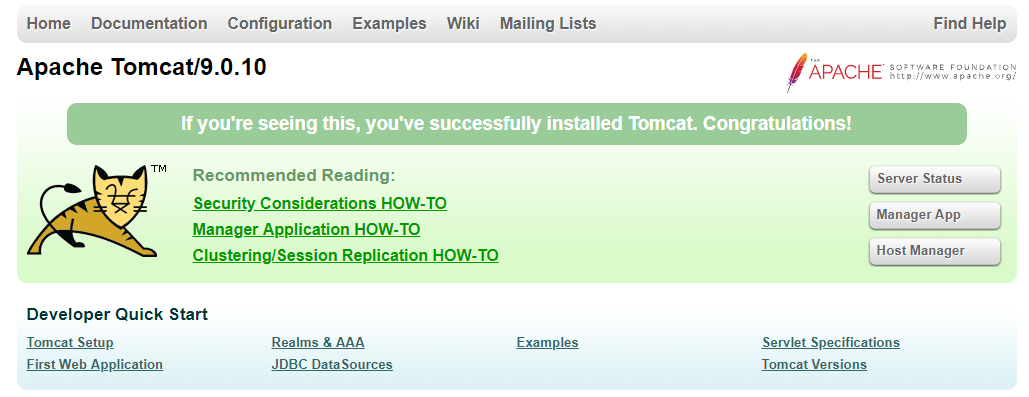


* 1. Tomcat测试

测试Tomcat需要在浏览器中输入：http://localhost:8080，或者是http://127.0.0.1:8080。127.0.0.1表示本机的IP地址，localhost表示本机的主机名。

如果其他机器需要访问，那么要把localhost使用本机IP地址替换。

假如我的IP是192.168.1.100，你们要访问我机器上的Tomcat，那么需要在你们的浏览器中访问：<http://192.168.1.100:800>



1. Tomcat分析(熟练)
   1. Tomcat目录结构



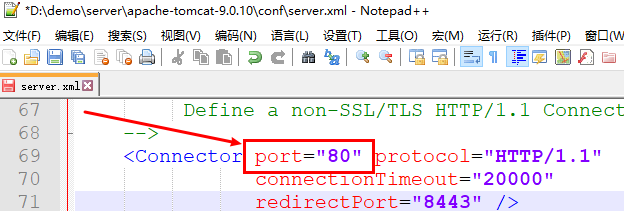
我们有必要去了解一下Tomcat目录结构，这有助与我们更好的学习Tomcat。

* bin：该目录下存放的是可执行文件，有startup.bat和shutdown.bat文件，startup.bat用来启动Tomcat，但需要先配置JAVA\_HOME和shutdown.bat用来停止Tomcat；
* conf：这是一个非常非常重要的目录，这个目录下有三个最为重要的文件：
  + server.xml：配置整个服务器信息。例如修改端口号，下面会详细介绍这个文件；
  + 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！
* 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，当然也可以把项目依赖的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；
* 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。
* 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。

　　你应该已经发现，访问ROOT项目时无需给出项目名，其实localhost是一个主机名，每个主机都会对应一个项目空间，localhost对应的就是Tomcat目录下的webapps，每个项目空间都可以有一个名为ROOT项目，这个ROOT项目被访问时无需给出项目名称。

* 1. Tomcat修改端口号

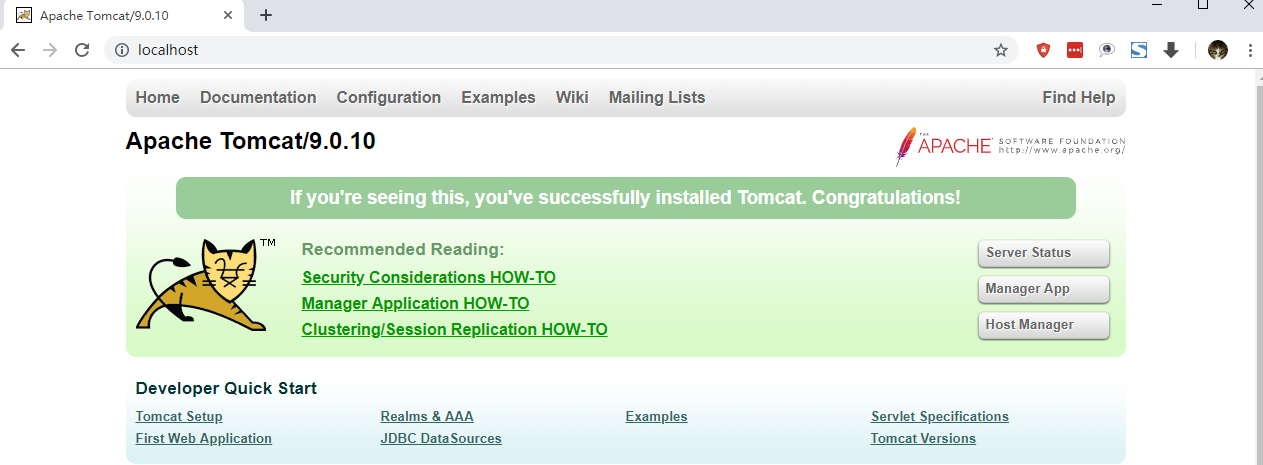
打开conf\server.xml文件



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。修改之后必须重新启动服务器

80端口是默认的端口，如果使用80端口访问时就可以省略端口

如果我们改成80端口，访问时就可以不加端口号。<http://localhost/>



* 1. Tomcat结构重点

**bin、conf、webapps、work。**

bin目录中东西我们只关心启动和停止Tomcat，现在大家也应该掌握了！

webapps目录下存放的是项目，每个文件夹都对应一个项目，而且在访问项目时，其中包含了项目名，而项目名往往都对应项目目录名。这个也应该可以掌握了！还有就是JavaEE Web项目的目录结构，这个必须必须背下来！

work：这个东西有点小难度，现在只需要知道它存放的文件是通过webapps目录下的项目在运行时产生的，最终运行的都是work目录下的文件。这个目录下的东西可以删除，然后再次运行时还会产生！

1. WEB项目(精通)
   1. WEB项目部署后目录结构

webapps目录是用来存放Java项目的，每个文件夹都是一个项目。默认这个目录下已经有了四个项目，都是tomcat自带的。其中ROOT就是我们测试Tomcat时访问的Tomcat主项目。

我们要创建自己的项目，需要遵循JavaEE目录结构规范，下面是hello项目的目录结构

Hello (必须出现的)

|------ 存放web资源。(jsp, xml, HTML CSS JS 图片...)

|------ **WEB-INF**  (必须出现的)

|------ **web.xml** (必须出现的)

|------ classes Java类的编译路径（Servlet、自定义类 ）

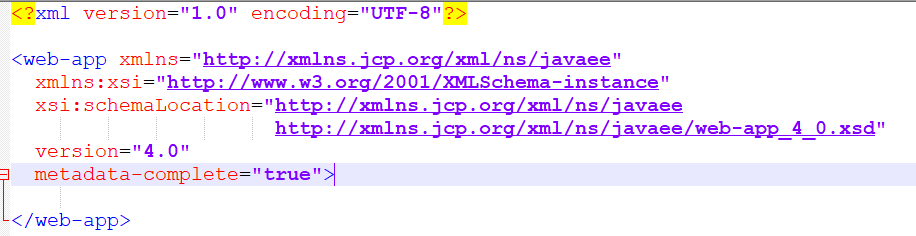
|------ lib 需要引入的第三方的jar包.

* hello是项目目录，项目目录下必须有一个名为WEB-INF的目录（必须大写），在WEB-INF目录下必须有一个web.xml文件。
* WEB-INF目录是受保护目录，这下面的东西用户是不能直接访问的，但动态页面是可以访问的。通常WEB-INF下还会存在lib和classes，lib下存放项目所需的jar文件，classes目录下存在项目所需的class文件！

如果直接访问WEB-INF就会找不到路径，



* web.xml文件中只需要给出XML文档声明，以及根元素。当然，根元素中包含很多与名称空间相关属性，背下来是有难度的，不过我们可以去webapps下的其他项目中去“借”。



* 1. Tomcat发布项目
     1. webapps文件夹下

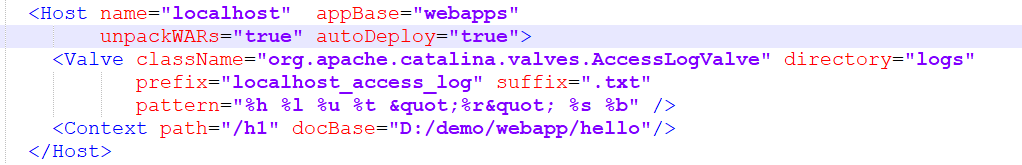
就是把项目放到webapps目录下，即localhost主机名对应的项目空间目录中。

访问http://localhost/hello/hello.html

* + 1. 任意目录下

可以把项目放到任何目录，然后通过添加<Context>元素来指定项目的真实位置。在<Host>元素中添加<Context>元素，然后指定<Context>元素的path和docBase属性，其中path指定的是项目虚拟路径，可以随意给出，例如abc。docBase指定你的项目的真实存放的路径。

例如，项目的真实目录在D:\demo\webapp\hello。你找到server.xml中的<Host>元素，给其添加子元素<Context>，内容如下：

<Context path="/自己编写的项目虚拟路径" docBase="项目的真实目录"/>

Tomcat7要设置通过设置服务器配置来让get请求支持中文,例如我们使用tomcat服务器,则设置/conf/server.xml:

<Connector port="8080" protocol="HTTP/1.1"

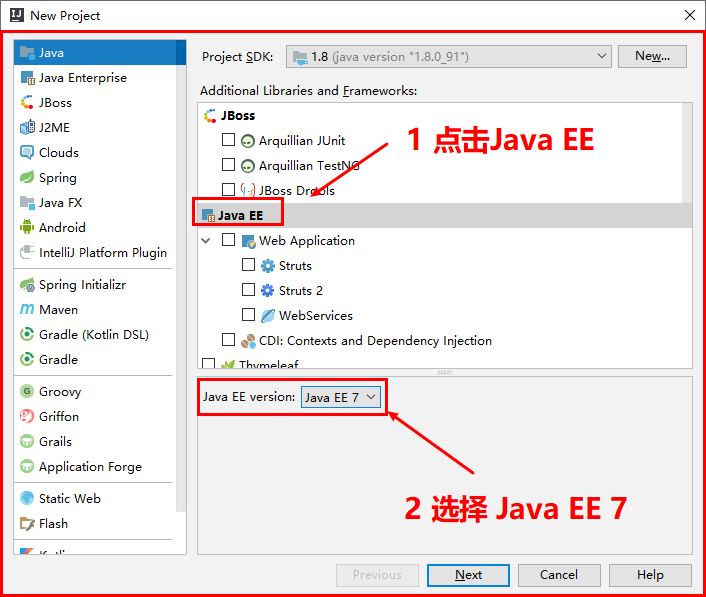
connectionTimeout="20000"

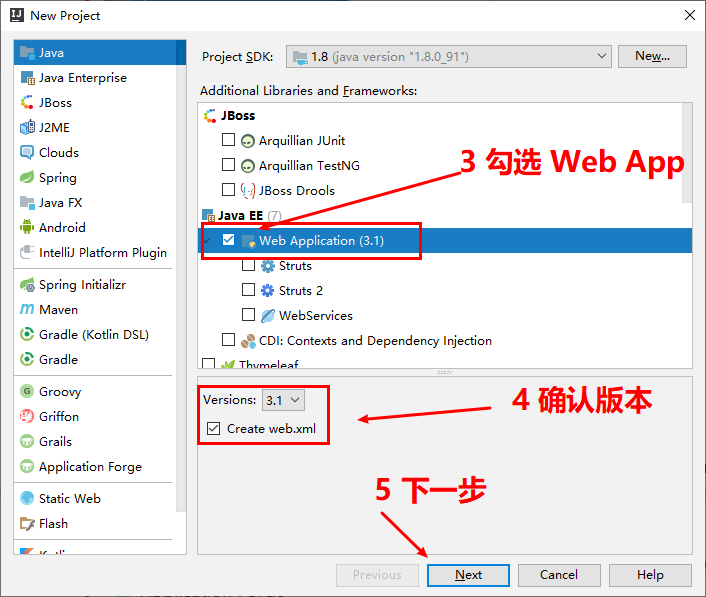
redirectPort="8443" URIEncoding="UTF-8"/>

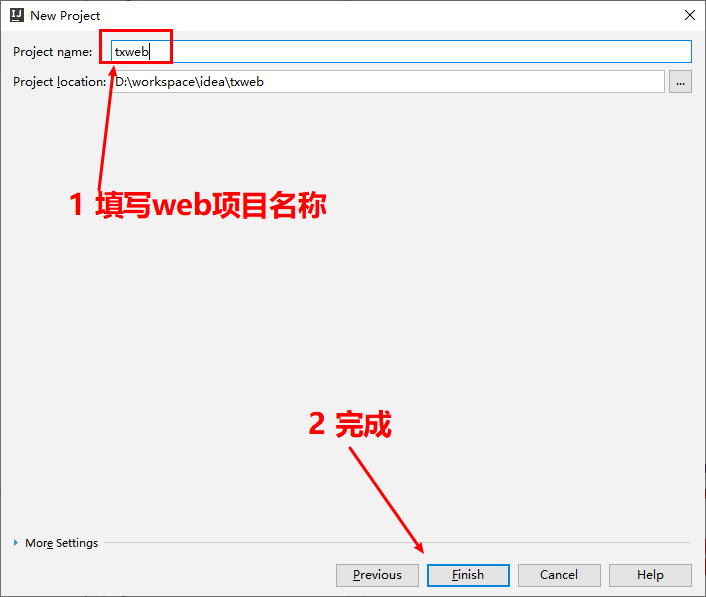
注意，一旦给出了docBase属性的值后，访问localhost的abc项目时，就不会再去webapps目录下去查找项目，而是通过docBase的值来锁定项目位置，即F:\hello1。

注意，访问项目的URL：http://localhost/h1/index.jsp，而不是http://localhost/hello1/index.jsp，你可能会说，我们的项目目录名叫hello1，但因为指定了<Context>元素的path属性，那么访问项目时就不会再使用项目目录名来作为访问路径了。

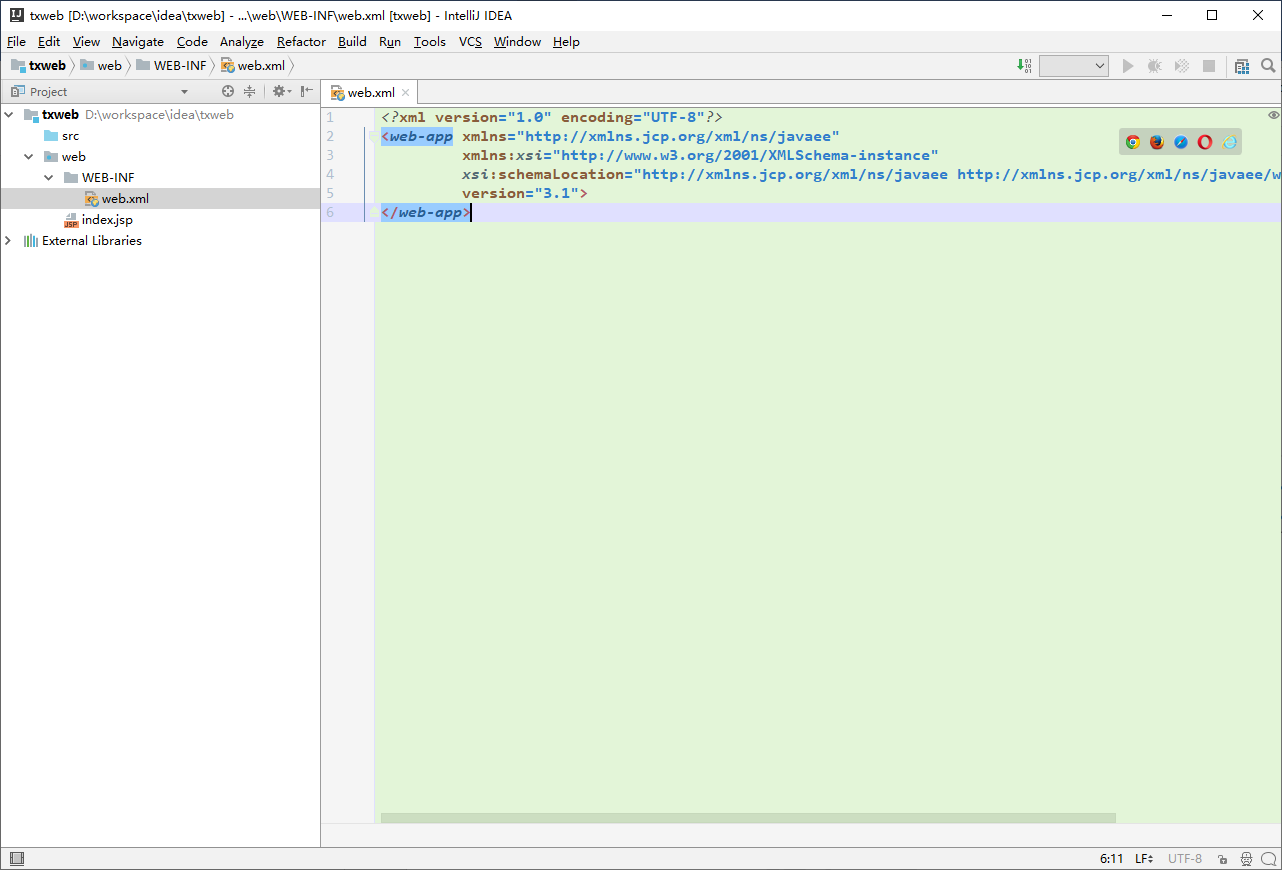
1. IDEA整合Tomcat(精通)
   1. IDEA创建Java web项目

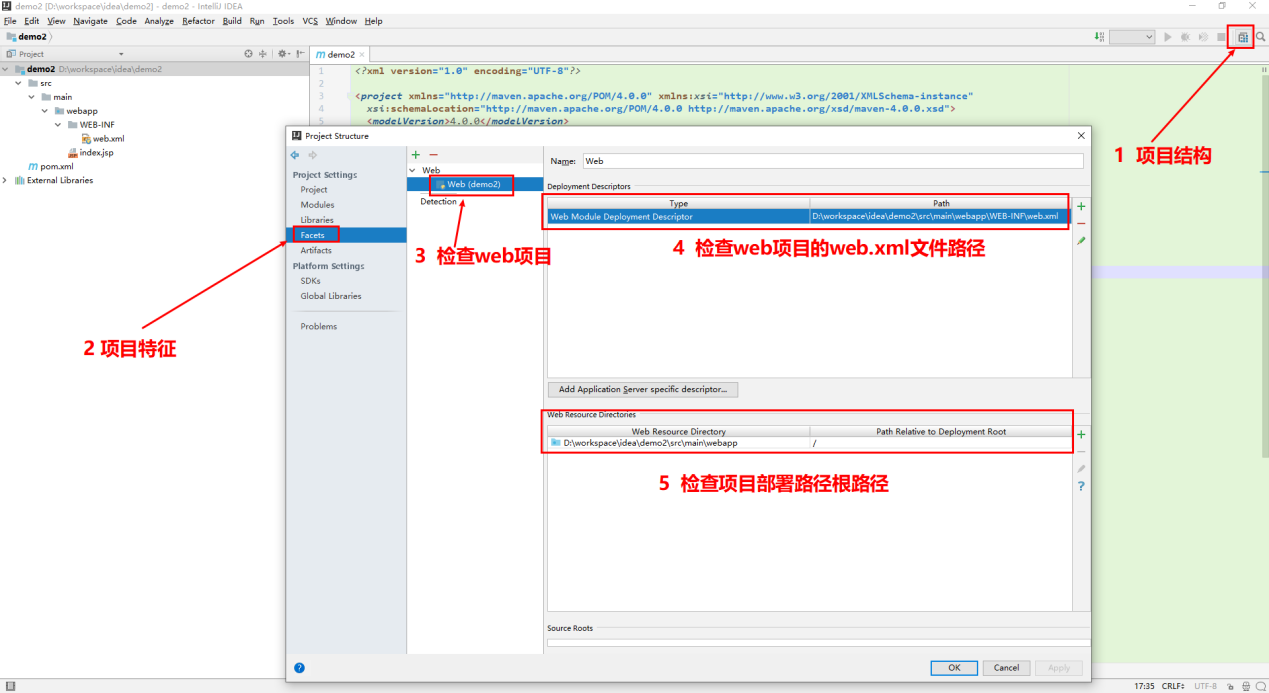


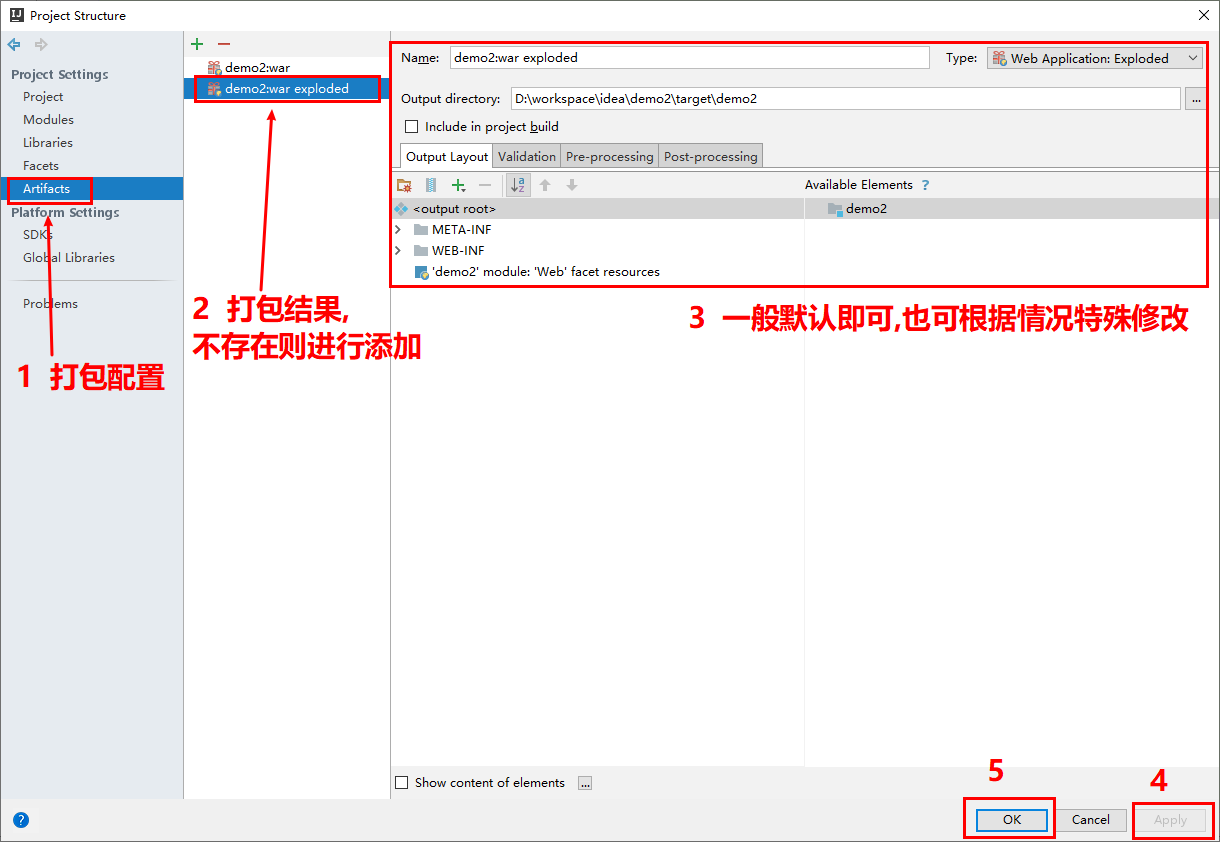


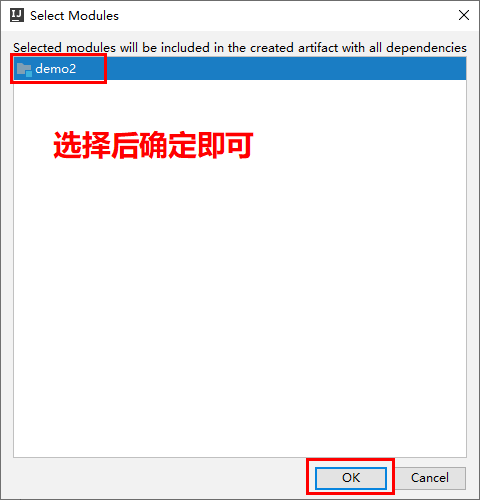
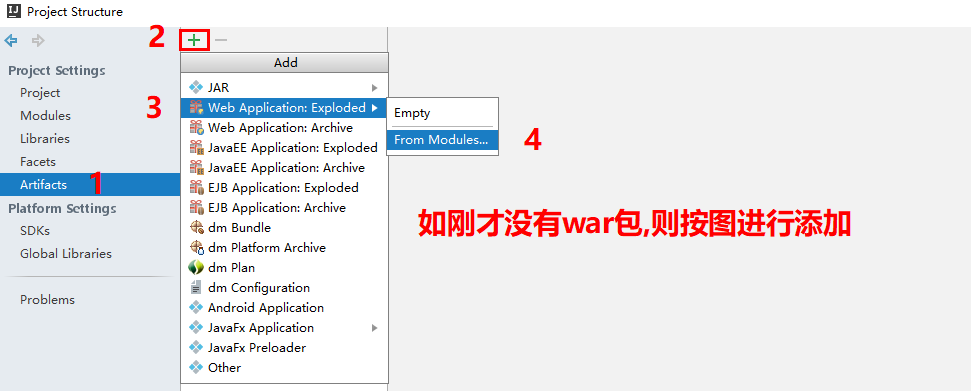


创建成功



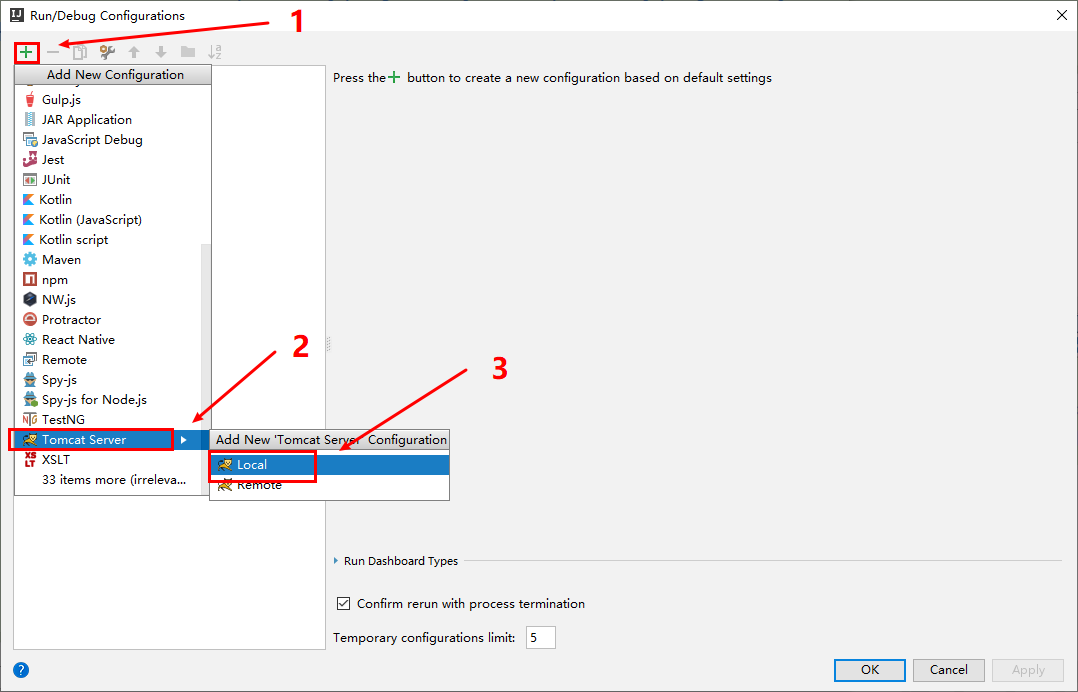


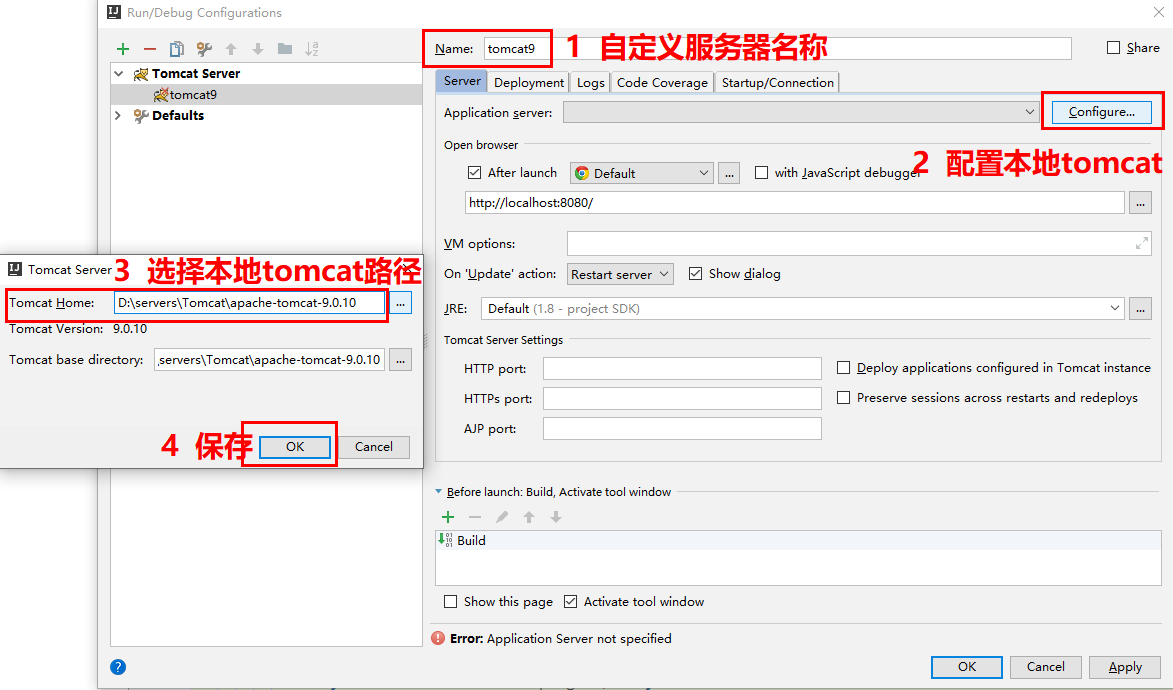


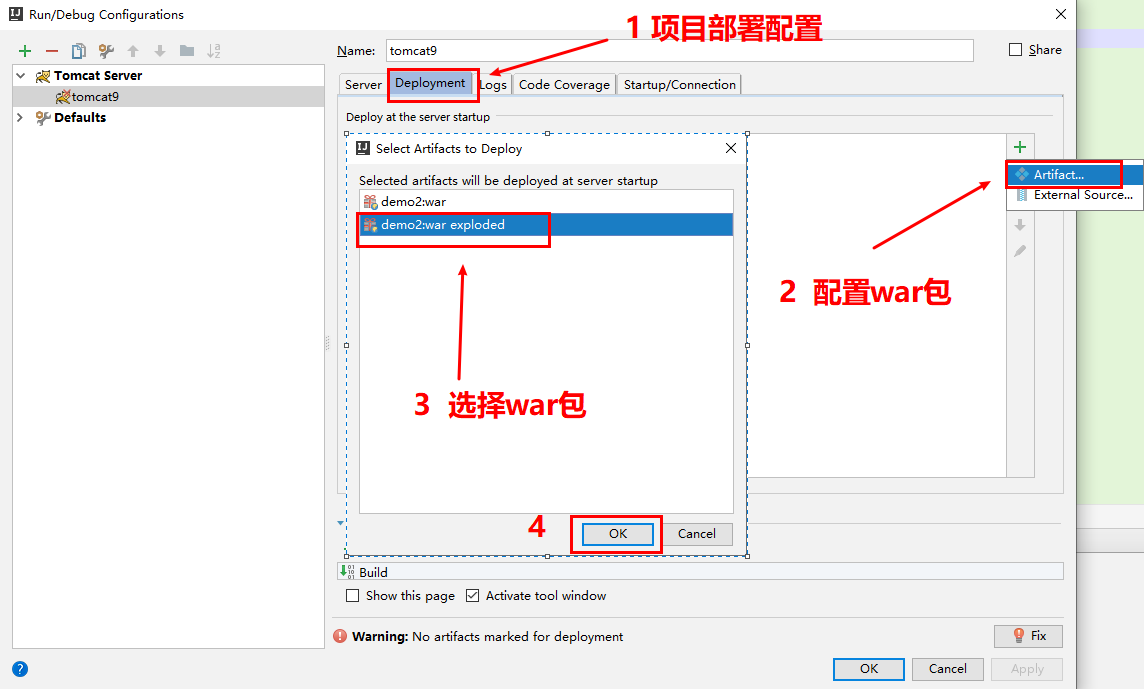


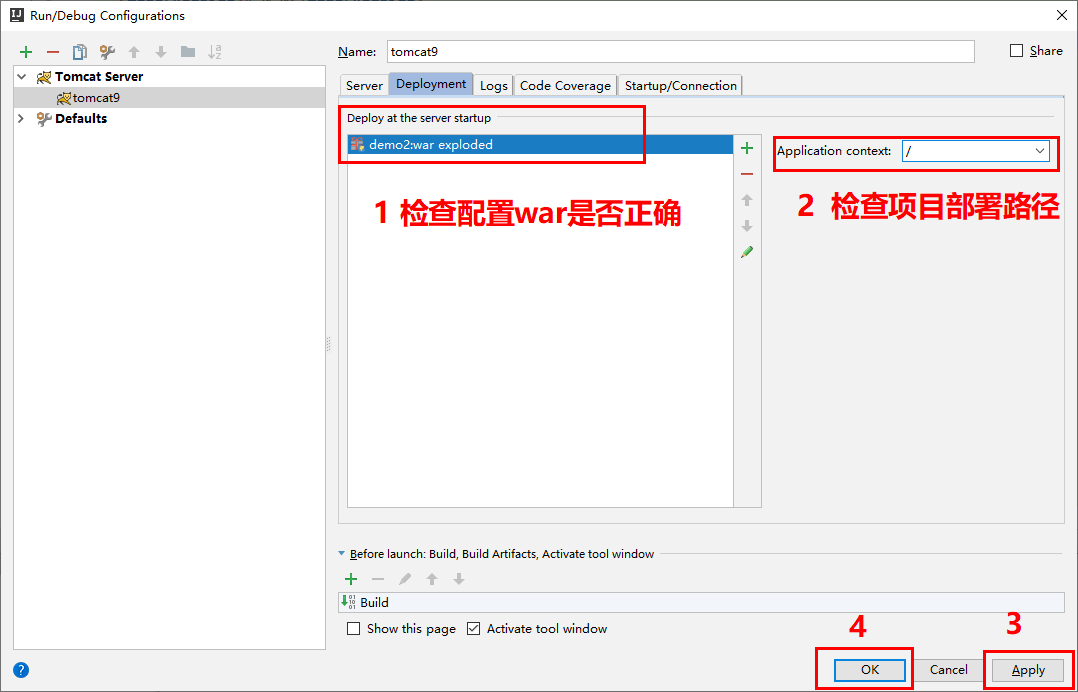
* 1. IDEA 部署Java web项目

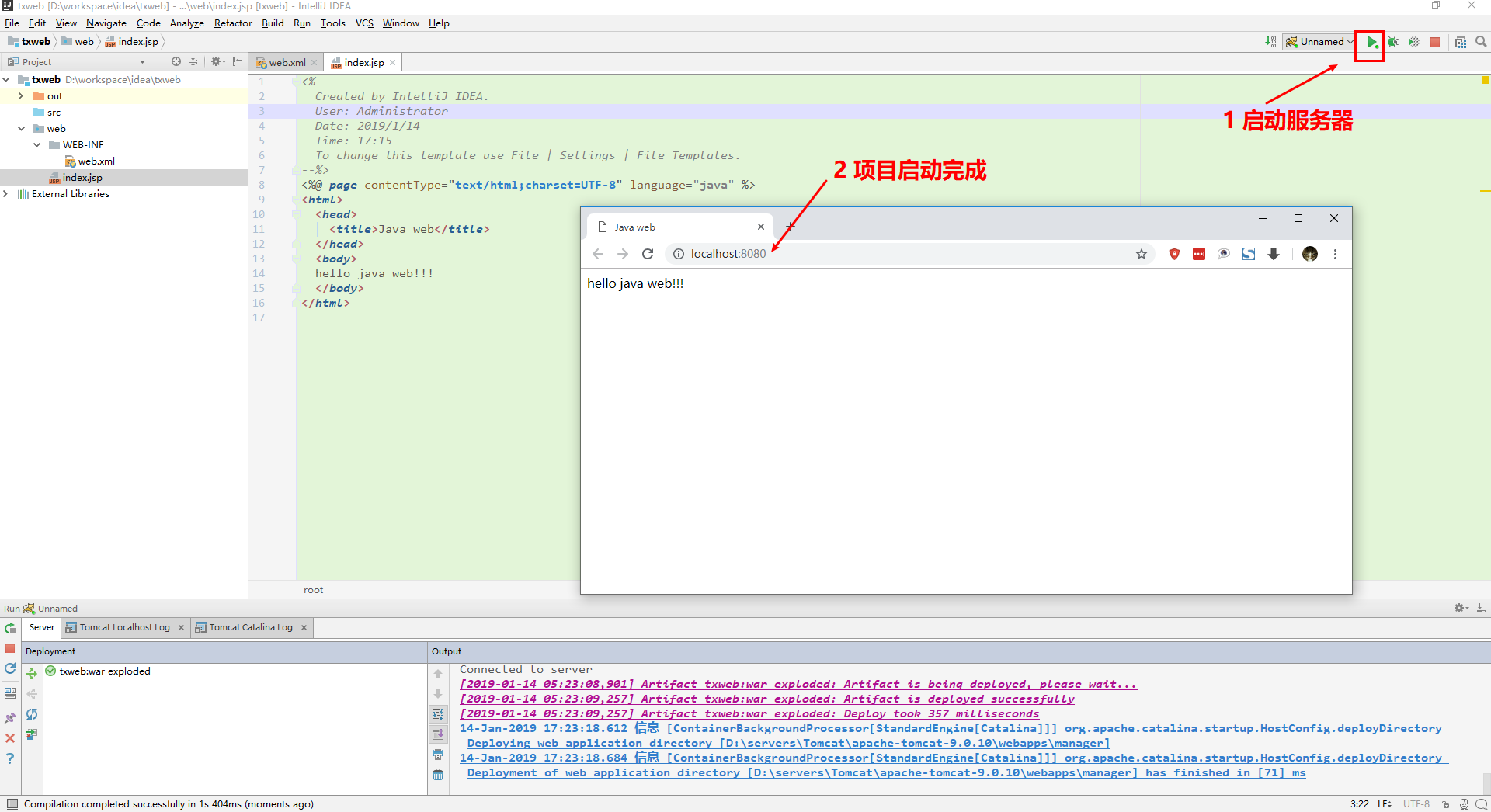












1. HTTP协议(熟练)
   1. HTTP协议概念

HTTP，即**超文本传输协议**。所有浏览器都可以发出http协议。

协议：就是规定的文本格式！例如写本书也要有格式，写总结也有格式。当然客户端发送请求也要有格式，这个格式是已经约定的格式，所以服务器端按请求格式来获取请求信息，然后服务器端再按照响应格式来响应，当然客户端还要按响应的格式来解析。

* 1. 无状态协议

HTTP是无状态协议，必须必须理解这一点！！！

所谓无状态，就是客户端发送一个请求，服务器返回一个响应，然后就断开连接了。

无状态协议的好处，当然是**效率高**了，用户连接一下就断开，当用户再次发送请求时，再连接，然后在响应之后再次断开。如果用户在服务器上找到了自己想看的小说，那么服务器只是把小说的内容（html）发送给用户的浏览器，然后用户的浏览器显示已经发送到本地的html，其实这时已经与服务器断开连接了。用户可能会看很久，但这不会占用服务器的任何资源！

* 1. HTTP1.0和HTTP1.1

HTTP现在有两个版本，HTTP1.0和HTTP1.1。

HTTP1.0：响应结束后会马上断开；

HTTP1.1：必须在请求中包含Host（主机）头信息，响应结束后不会马上断开，会有一个超时时间，可能会在超时之前完成多个请求/响应，当超时的时候就断开。

GET /web\_demo/index.jsp HTTP/1.0

Host: localhost:8080

* 1. 请求协议和响应协议

无论是请求协议还是响应协议，都是由如下部分构成的：

* 请求（或响应）首行；
* 请求（或响应）头信息；
* 空行；
* 请求（或响应）正文。
  1. 请求协议

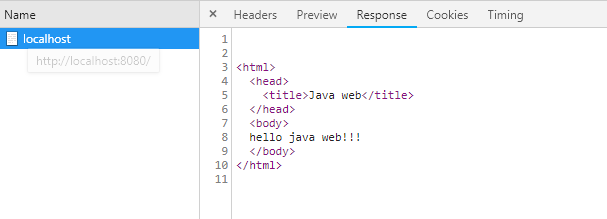
我们可以通过浏览器来监视请求和响应，获取完整的请求和响应信息。

* Fire Fox：
* Google：内置了这功能，也无需安装。

请求

GET /hello/index.jsp HTTP/1.1

响应



* + 1. 常见的请求协议
    2. GET请求

下面是通过FireFox获取到的请求信息，这就是浏览器发送给服务器的全部请求信息内容。请求的地址是：http://localhost/hello/index.jsp

|  |
| --- |
| //请求首行  GET /hello/index.jsp HTTP/1.1  //下面全部是请求头信息，因为GET请求没有正文  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  //空行，因为GET没有正文，所以下面没有东西了 |

* 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；
* 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，然后在讲会话是讲究它！
* Get请求没有正文，只有头信息，请求参数是在请求的首行的url中传递的。请求的参数的容量是有限的，不能超过1024kb
  + 1. POST请求

下面是通过POST请求获取到的请求信息

|  |
| --- |
| // 请求首行  POST /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  Referer: http://localhost/hello/index.jsp  Cookie: JSESSIONID=369766FDF6220F7803433C0B2DE36D98  Content-Type: application/x-www-form-urlencoded  Content-Length: 14  // 这里是空行  //POST有请求正文  username=hello |

* Referer: http://localhost/hello/index.jsp：请求来自哪个页面。这个值说明请求是从index.jsp页面发送出去的！这个东东还是比较有用的！如果这个表单的作用是下载，那么在给用户提供资源之前可以去判断请求是从哪个页面发送过来的，如果不是从本网站发送过来的，那么就可以拒绝下载，例如盗链！
* Content-Type: application/x-www-form-urlencoded：表单的数据类型，说明会使用url格式的UTF-8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；
* Content-Length:14：表示请求数据的长度，这里表示14个字节。
* username=hello：这是请求正文！hello是在表单中输入的数据，username是文本框的名字。
  + 1. 响应协议

响应信息:

|  |
| --- |
| // 响应首行  HTTP/1.1 200 OK  // 响应头信息  Server: Apache-Coyote/1.1  Content-Type: text/html;charset=UTF-8  Content-Length: 777  Date: Sat, 09 Feb 2012 18:30:52 GMT  // 空行，下面是响应正文，即HTML代码  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="http://localhost:80/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>  This is my JSP page. <br>    <form method="post" action="index.jsp">  <input type="text" name="username" /><br/>  <input type="submit" value="SUBMIT"/>  </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：这是服务器的版本信息；
* Content-Type: text/html;charset=UTF-8：请求正文使用的编码为UTF-8；
* Content-Length: 777：响应的内容为777字节；
* Set-Cookie: JSESSIONID=C97E2B4C55553EAB46079A4F263435A4; Path=/hello：响应给客户端的Cookie；
* Date: Sat, 09 Feb 2012 18:30:52 GMT：响应的时间，这可能会有8小时的时区差；

常见状态码：

* 404 not found：没有找到；
* 500 error：找到了，但内部出错；
* 200 ok：成功！

请求：GET/POST

只有表单才可能是POST，其他都是GET

POST有正文，即表单内容：username=zhangSan&password=123

GET没有正文，但是空行可是有的！

请求首行

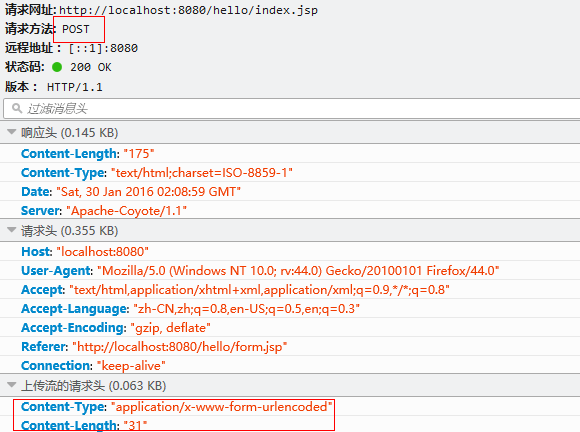
(GET/POST) URL HTTP/1.1

响应首行

HTTP/1.1 状态码 状态码的解释

响应的正文就是HTML！

Post请求：有正文，而且正文内容不受限制，通过表单的方式提交（除了表单方式的提交时post请求方式，其余的都是get请求）。



1. 学习目标

* 了解Javaweb相关知识
* 能够进行tomcat安装启动及端口号配置
* 能够手动创建web项目并部署在tomcat下启动
* 能够使用idea创建Java web项目并成功启动访问
* 了解http协议相关知识