

# Day10

java课程-李彦伯

## Day10

### HTTP

协议的组成和过程

抓包分析

http请求

get请求

post请求

Http响应

常用响应状态码

### Tomcat

服务器

web资源

web应用服务器

Tomcat下载安装

Tomcat目录结构

Tomcat启动

常见问题

Eclipse绑定Tomcat

创建Web工程

目录介绍

### Servlet

Servlet的内部实现原理

Servlet生命周期

Servlet的配置

url-pattern的配置方式

欢迎界面

HttpServlet

request和response内部原理

ServletContext对象

ServletContext的生命周期

如何得到ServletContext对象

ServletContext作用

HttpServletResponse

response运行流程

response设置响应行

response设置响应头

response重定向

重定向的过程分析

重定向特点

response设置响应体

# HTTP

HTTP，超文本传输协议（HyperText Transfer Protocol）是互联网上应用最为广泛的一种网络协议。

## 协议的组成和过程

http协议由http请求和http响应组成,当在浏览器中输入一个网址后敲回车,浏览器会将你的请求封装成一个http请求的格式发送给服务器,服务器收到请求以后会组织响应数据封装成一个http响应,返回给客户端.这就是http请求和响应的过程.也就是说没有请求就没有响应

## 抓包分析

使用chrome自带的工具我们可以看到数据相互之间传送的具体过程

### http请求

#### get请求

```
//请求行:请求的方法GET,请求的路径带提交的信息
HTTP的版本
GET /BigData/a.html?username=213123&password=3244234 HTTP/1.1
//本机的地址
Host: localhost:8080
//连接的方式 长连接
Connection: keep-alive
//则是告诉服务器,自己支持这种操作,也就是我能读懂你服务器发过来的上面这条信息,并且在以后发请求的时候不用http而用https
Upgrade-Insecure-Requests: 1
//客户端浏览器内核和响相应的操作系统的相关信息
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
//告诉服务器客户端浏览器能够接收哪种类型的数据
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
//浏览器通知服务器,当前请求来自何处。如果是直接访问,则不会有这个头。常用于:防盗链
Referer: http://localhost:8080/BigData/a/a.html
//浏览器通知服务器,浏览器支持的数据压缩格式
Accept-Encoding: gzip, deflate, sdch, br
//浏览器通知服务器,浏览器支持的语言
Accept-Language: zh-CN,zh;q=0.8
```

## post请求

```
POST /BigData/a.html HTTP/1.1
Host: localhost:8080
Connection: keep-alive
//内容的长度
Content-Length: 29
Cache-Control: max-age=0
Origin: http://localhost:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/58.0.3029.110 Safari/537.36
//如果使用的是post请求就会有这个头,表示请求体中
的数据使用的是url编码
Content-Type: application/x-www-form-urle
ncoded
Accept: text/html,application/xhtml+xml,a
pplication/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://localhost:8080/BigDat
a/a.html
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.8
//会有一个空行,下面的内容是请求体的内容
username=12123&password=34324
```

## Http响应

```
//响应行 http协议  状态码
HTTP/1.1 200
//相应头
//表明服务器是否支持指定范围请求及哪种类型的分段
请求
Accept-Ranges: bytes
//被请求变量的实体值”，ETag是一个可以与Web资源
关联的记号
ETag: W/"373-1499776861532"
//上次修改的时间
Last-Modified: Tue, 11 Jul 2017 12:41:01
GMT
//内容类型
Content-Type: text/html
//响应正文长度
Content-Length: 373
Date: Tue, 11 Jul 2017 12:41:10 GMT
//空行,响应体
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>

<style>
h1{
    color:yellow;
    font-style: italic;
}
```

```
</style>
</head>
<body>
<h1>好好学习,天天向上</h1>
    <form action="#" method="post">
        <input type="text" name="username">
        <input type="text" name="password">
        <input type="submit">
    </form>
</body>
</html>
```

## 常用响应状态码

常用状态码	意义
200	请求成功
302	重定向
304	读取本地缓存文件
404	请求的页面不存在
500	服务端程序错误

# Tomcat



# 服务器

服务器指一个管理资源并为用户提供服务的计算机软件,服务器的本质其实就是普通的电脑中装了相关的服务器软件,当我们的电脑装了mysql,就会成为一个数据库服务器,当我们的电脑装了Tomcat就成为了一个web应用服务器

## web资源

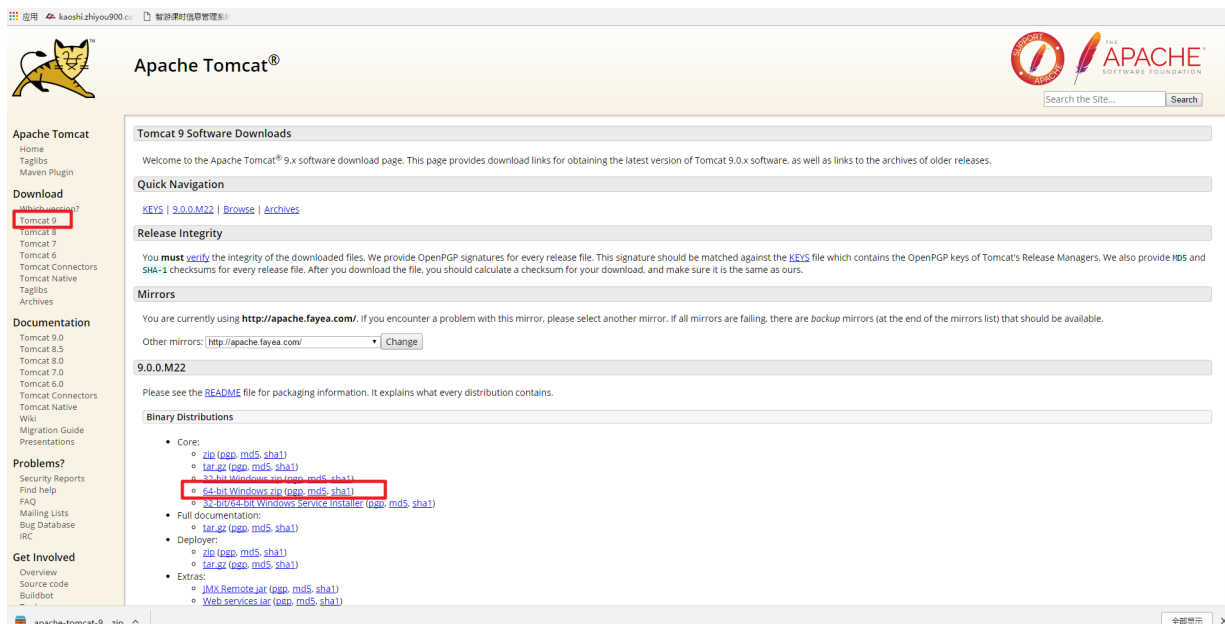
存在于web应用服务器内部,能够让外界进行访问的资源都叫做web资源,如图片,js,css,视频等

## web应用服务器

- weblogic : oracle公司的大型收费web服务器 支持全部javaEE规范
- websphere : IBM公司的大型收费web服务器 支持全部的javaEE规范
- Tomcat : Apache开源组织下的 开源免费的中小型的web应用服务器 支持 javaEE 中的 servlet 和 jsp规范
- Tomcat的实际作用就是通过HTTP协议规范让用户访问存在于服务器内部的资源

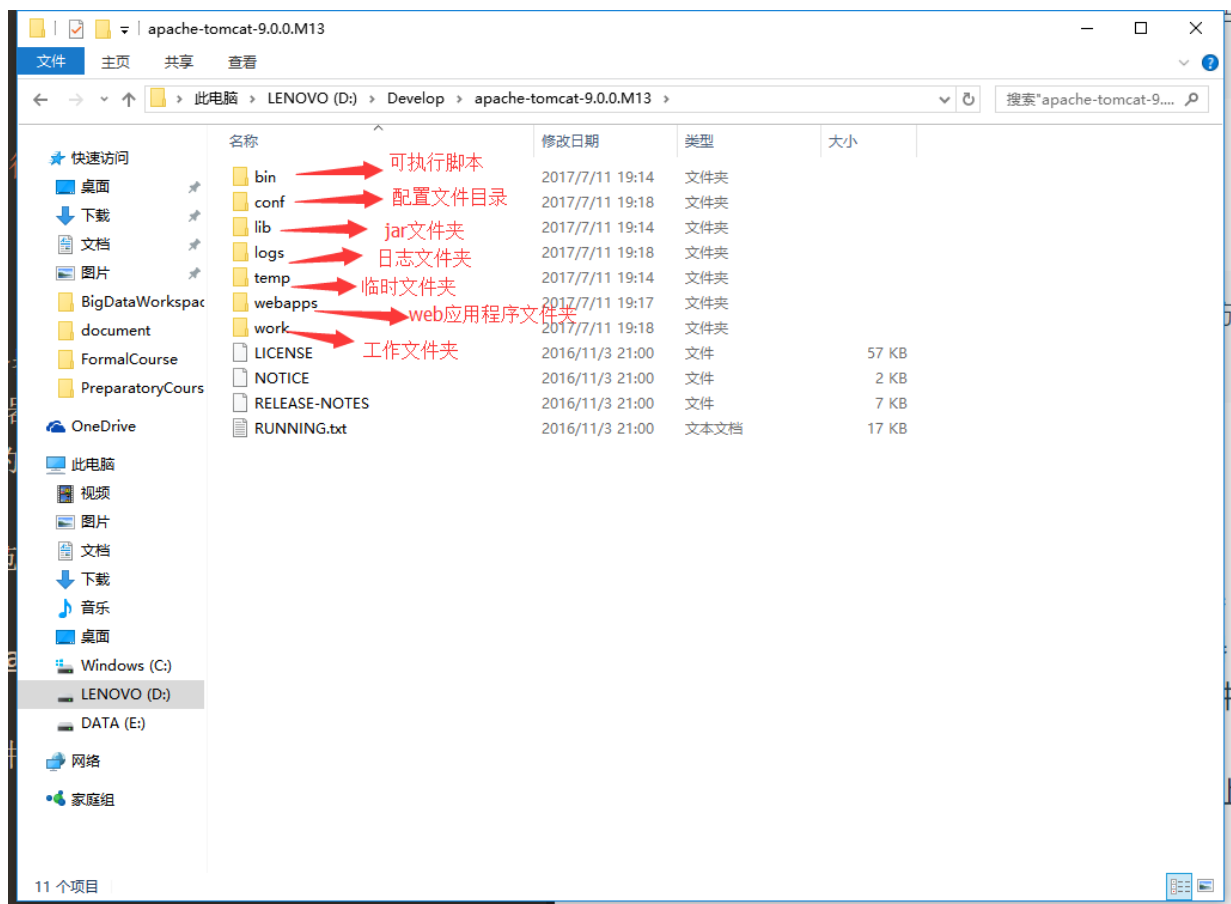
# Tomcat下载安装

- 下载地址<http://tomcat.apache.org/download-90.cgi>



- 解压即安装,建议解压在你们的develop文件夹中

# Tomcat目录结构



- bin中放的都是可执行的脚本程序
  - 如启动脚本 `startup.bat`
  - 停止脚本 `shutdown.bat`
- conf存放的是配置信息文件
  - `server.xml`核心配置文件:可以设置Tomcat端口,编码格式,web应用的发布信息
  - `tomcat-users.xml`用户权限配置文件:用于设置用户的分组和用户的密码
  - `web.xml` web项目默认配置文件,可以配置缺省路径,配置tomcat默认servlet
- lib 依赖库,tomcat和web项目中需要使用的jar包,如 `jsp-api.jar` 和 `servlet-api.jar`
- logs日志文件夹,如 `catalina.2017-07-11.log` 查看

## tomcat日志

- temp : 临时文件目录 , 其中内容可以任意删除
- webapps:存放发布的web应用的目录
- work:tomcat解析jsp文件的工作目录,会将jsp解析成的servlet就存在于这个目录

# Tomcat启动

- 需要jdk的支持,并且需要配置环境变量JAVA\_HOME
- 本机地址127.0.0.1或者localhost
- 端口配置,默认tomcat是8080端口
- 80端口:http协议默认的端口,我们可以通过修改servler.xml设置端口
- 配置用户名管理tomcat下所有的web应用,修改 tomcat-users.xml

```
<role rolename="tomcat"/>
<role rolename="role1"/>
<role rolename="manager-gui"/>
<user password="admin" roles="manager-gui,tomcat,role1" username="admin"/>
<user password="tomcat" roles="tomcat" username="tomcat"/>
<user password="tomcat" roles="tomcat,role1" username="both"/>
<user password="tomcat" roles="role1" username="role1"/>
<role rolename="admin-gui"/>
<user password="s3cret" roles="admin-gui,manager-gui" username="tomcat"/>
```

## 常见问题

- 一启动就发生闪退

系统没有配置JAVA\_HOME,配置即可

- `java.net.BindException: Address already in use: JVM_Bind <null>:8080`

8080端口被占用,可以通过server.xml修改端口,也可以通过进程杀死占用程序

- 1. 命令行输入 `netstat -ano` 查询8080端口

```
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

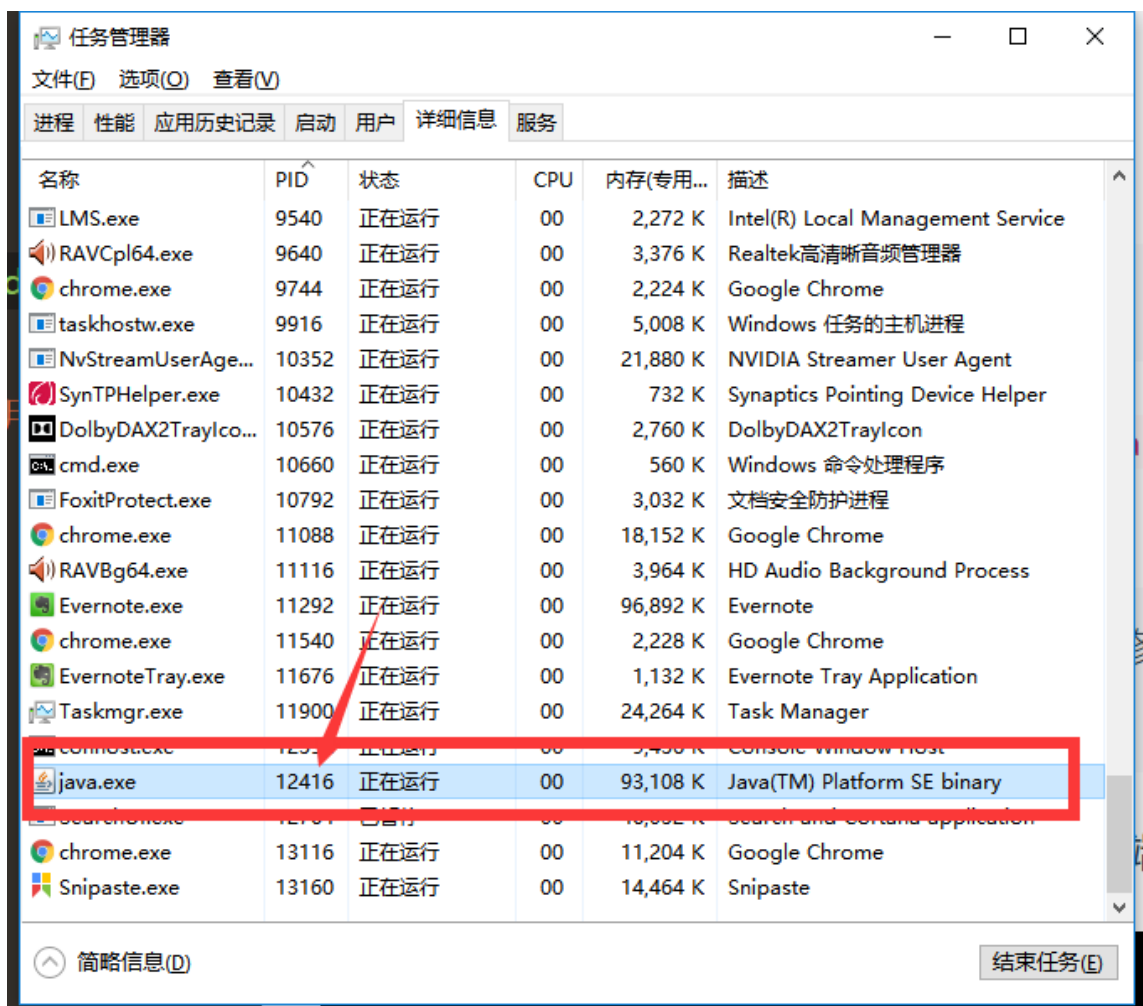
C:\Users\lyb>NETSTAT -ano

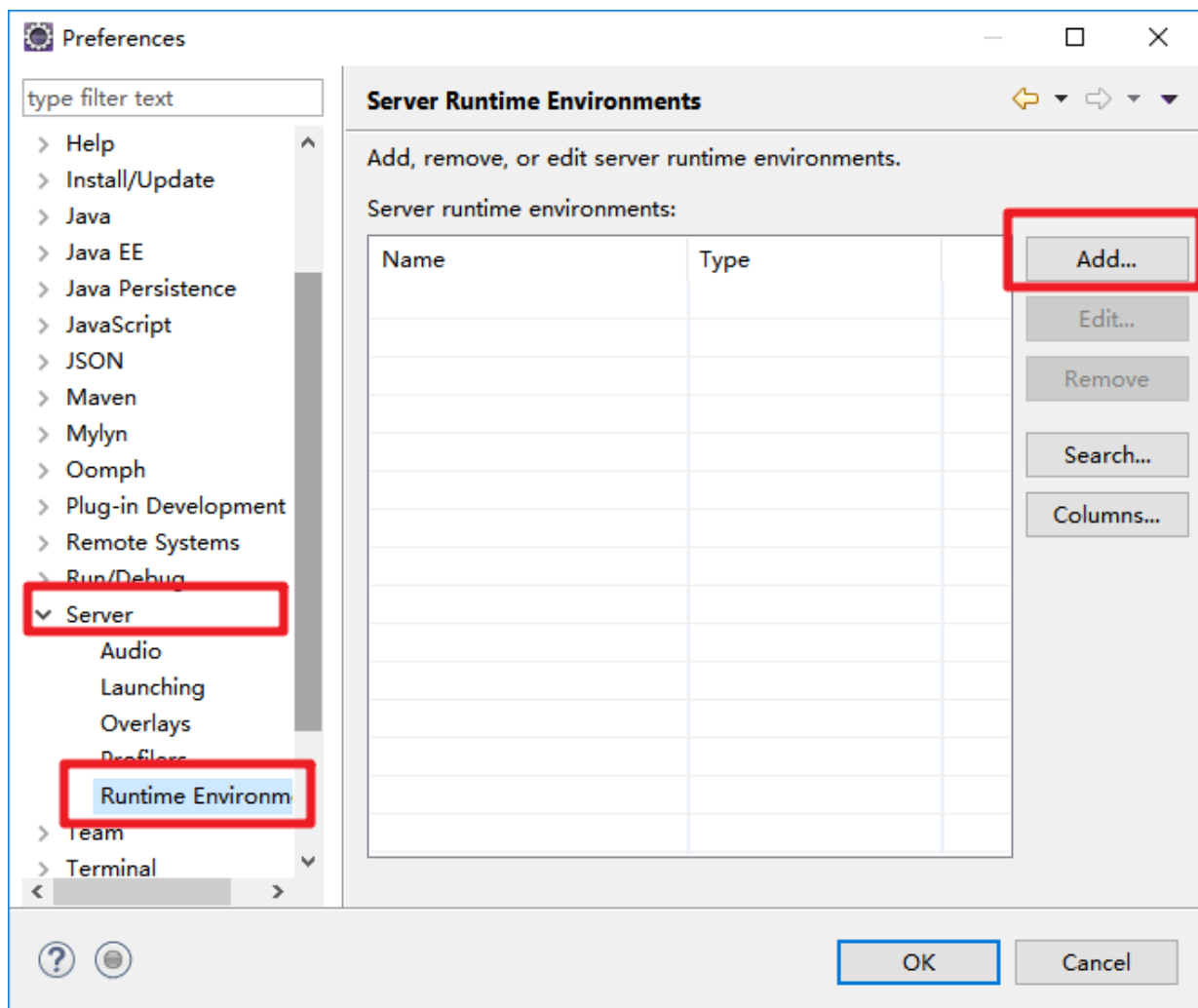
活动连接

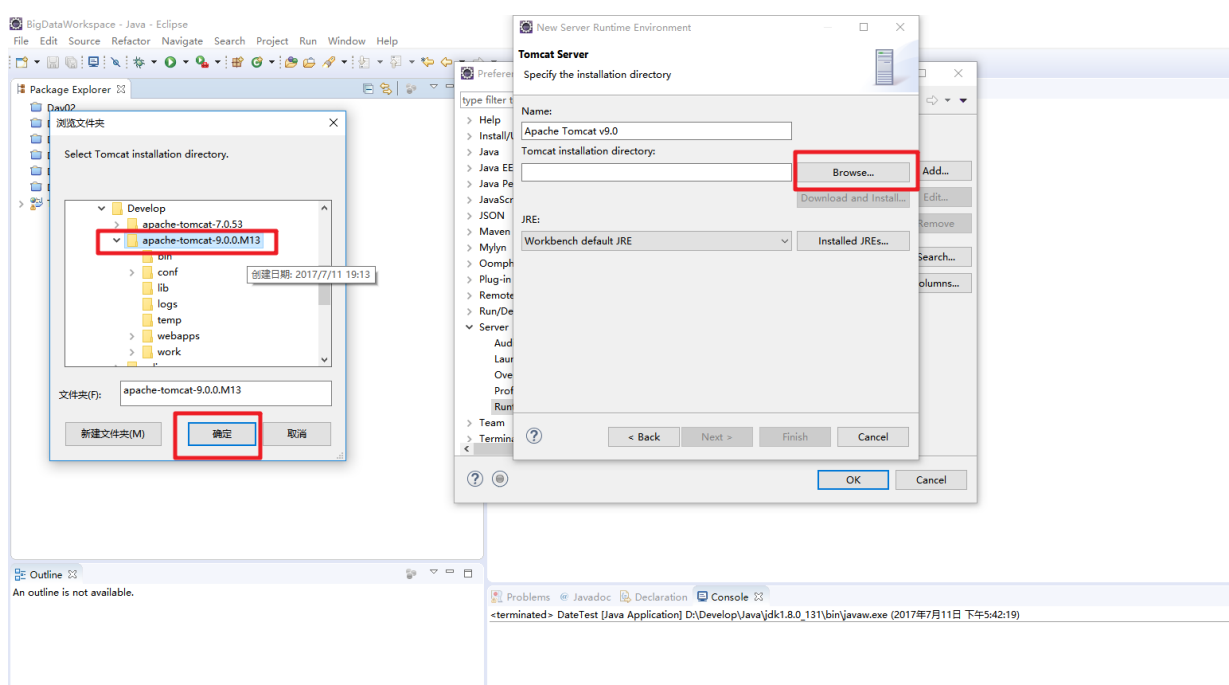
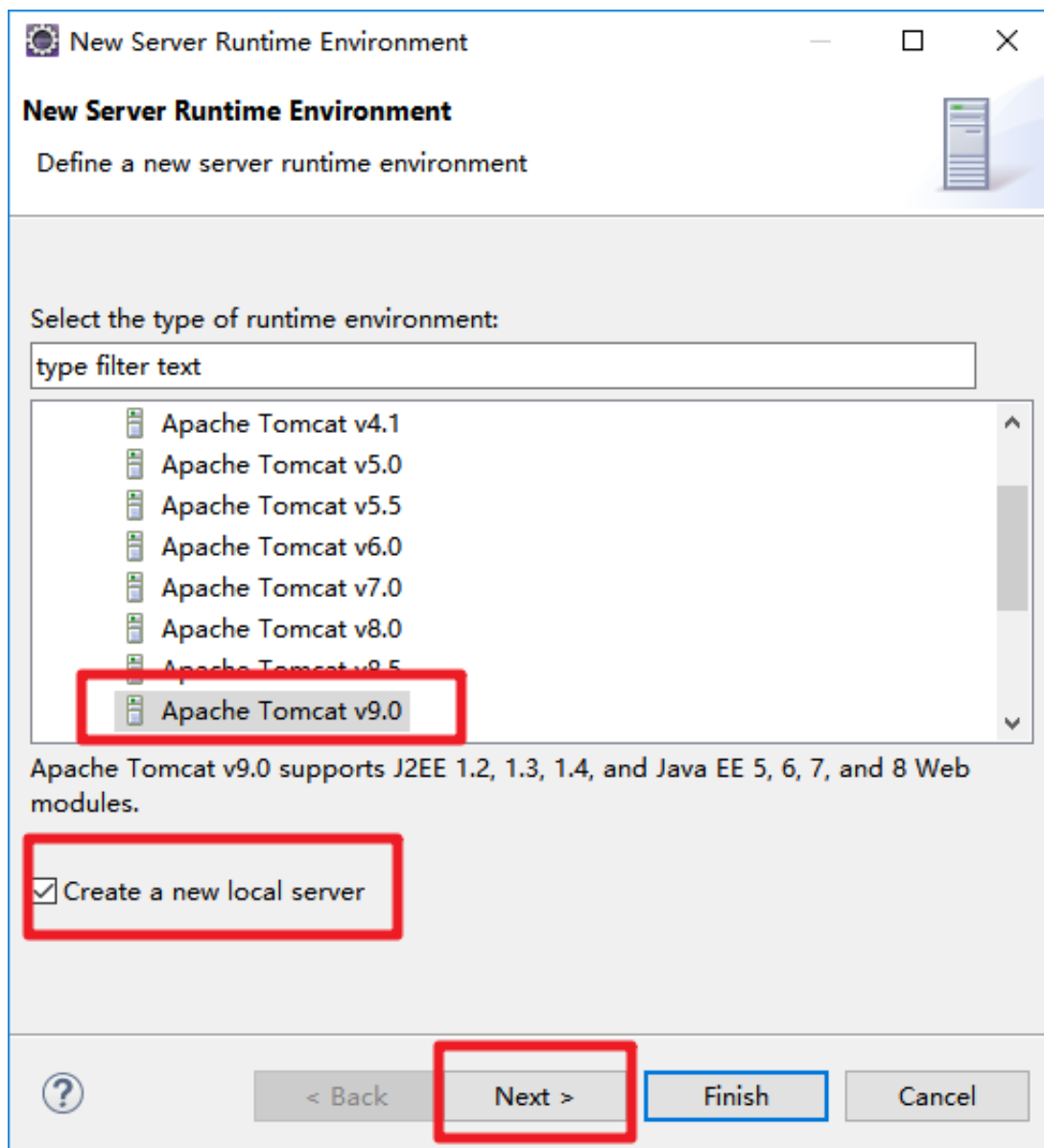
```

协议	本地地址	外部地址	状态	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	964
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING	4568
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:902	0.0.0.0:0	LISTENING	3544
TCP	0.0.0.0:912	0.0.0.0:0	LISTENING	3544
TCP	0.0.0.0:3306	0.0.0.0:0	LISTENING	3468
TCP	0.0.0.0:5357	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:8080	0.0.0.0:0	LISTENING	12416
TCP	0.0.0.0:47984	0.0.0.0:0	LISTENING	5024
TCP	0.0.0.0:47989	0.0.0.0:0	LISTENING	5024
TCP	0.0.0.0:48010	0.0.0.0:0	LISTENING	5024
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING	668
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING	316
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING	480
TCP	0.0.0.0:49683	0.0.0.0:0	LISTENING	2872
TCP	0.0.0.0:49702	0.0.0.0:0	LISTENING	740
TCP	0.0.0.0:49750	0.0.0.0:0	LISTENING	748
TCP	127.0.0.1:4000	0.0.0.0:0	LISTENING	10792
TCP	127.0.0.1:4300	0.0.0.0:0	LISTENING	4296
TCP	127.0.0.1:4301	0.0.0.0:0	LISTENING	4296
TCP	127.0.0.1:8005	0.0.0.0:0	LISTENING	12416


- 1. 进入任务管理器
  - win7系统 进程 -> 查看->选择列->pid 找到对应关闭即可









 Edit Server Runtime Environment

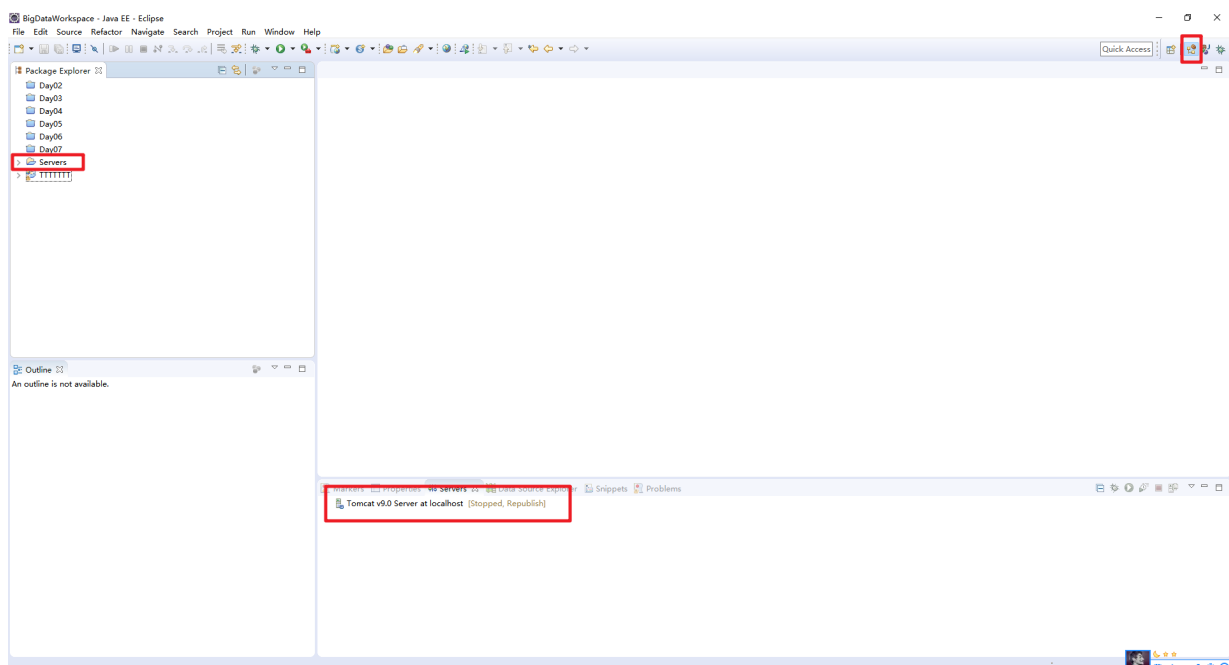
**Tomcat Server**

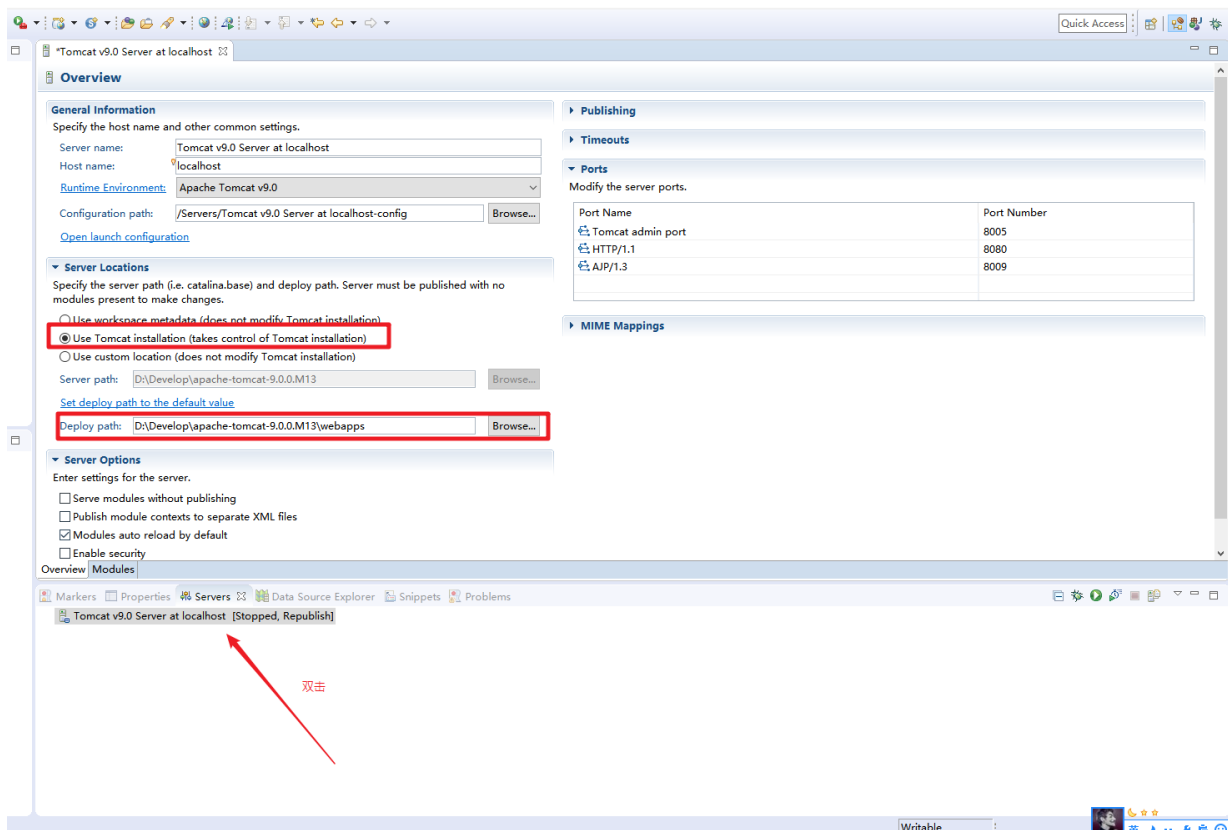
Specify the installation directory

Name:

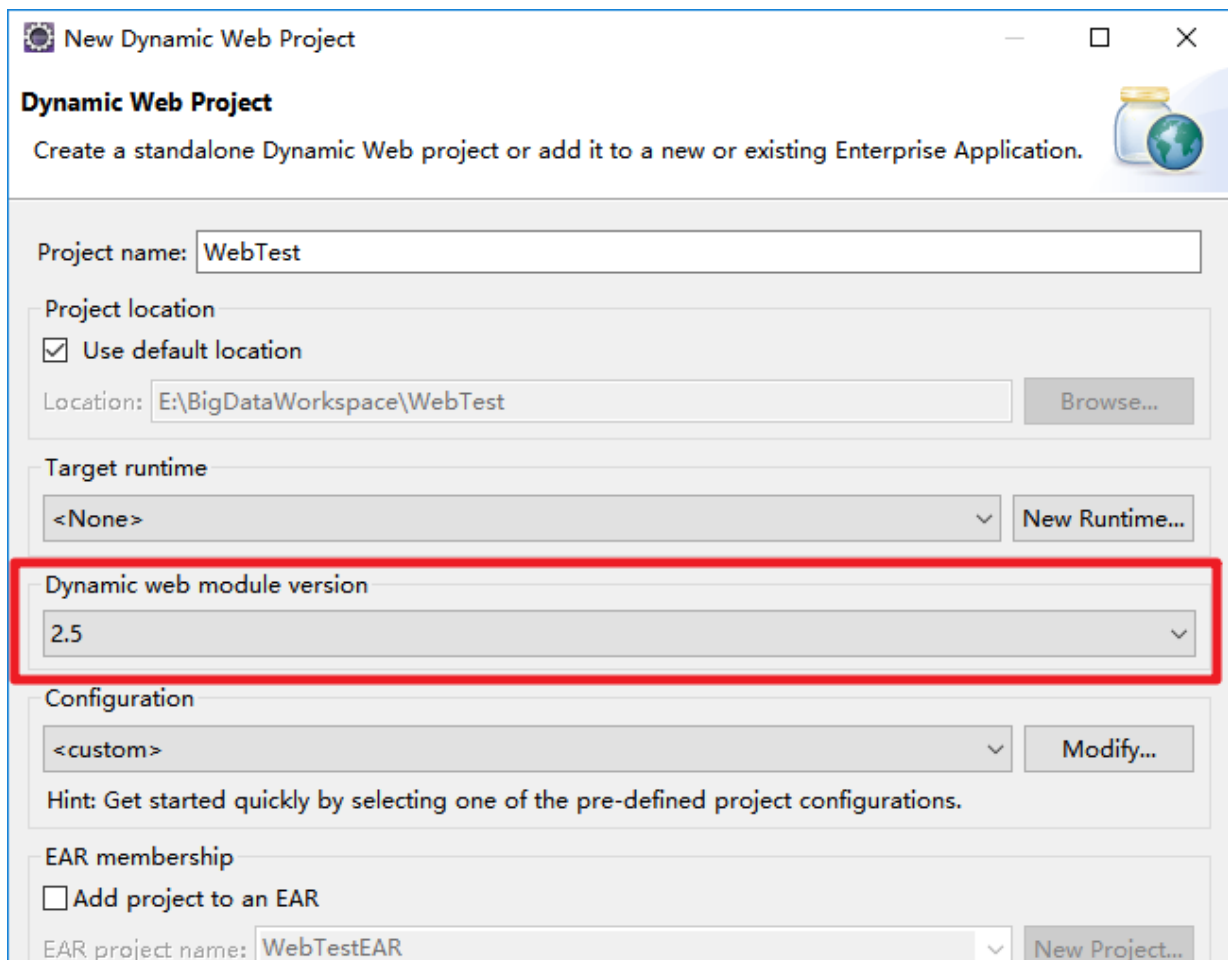
Tomcat installation directory:

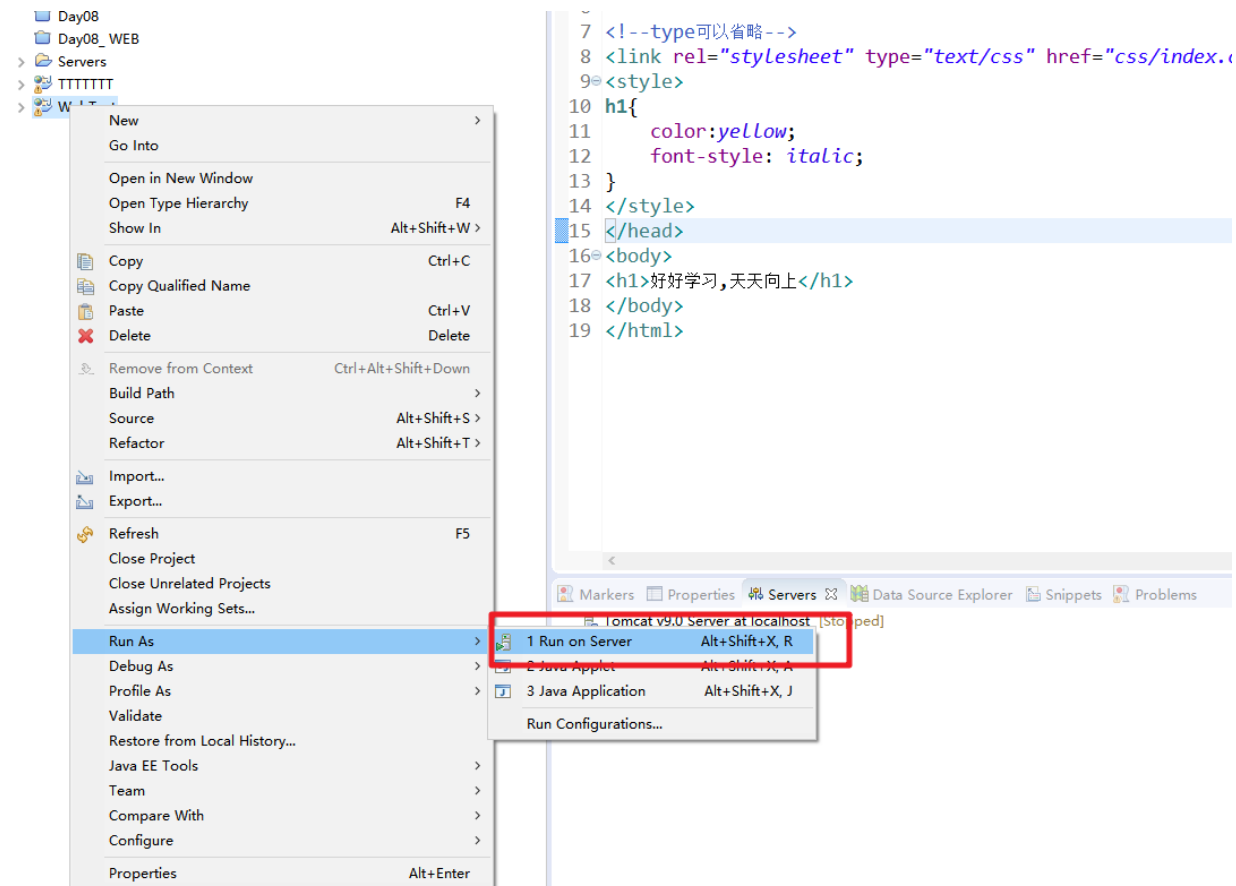
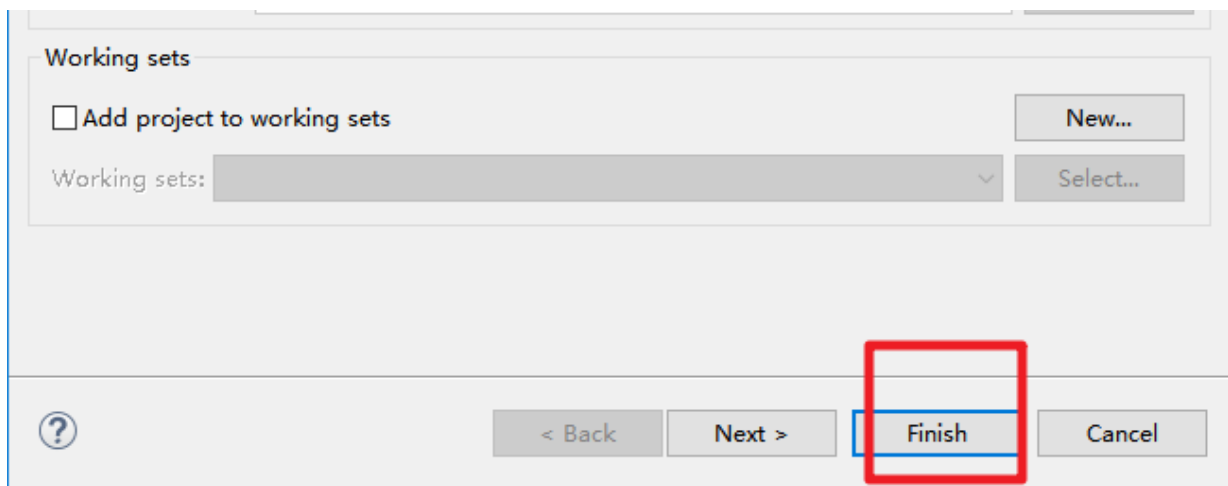
JRE:





# 创建Web工程



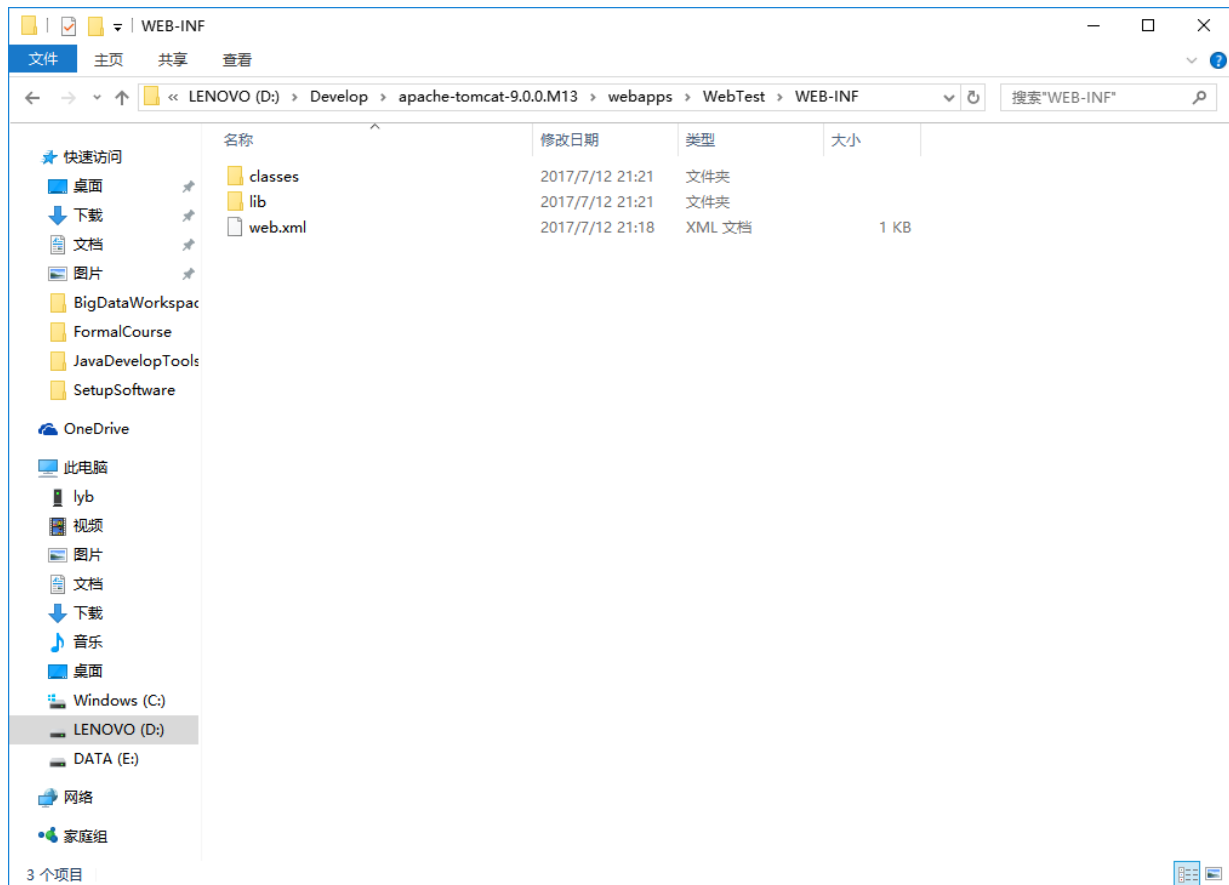


## 目录介绍

- eclipse中的WebContent目录中所有的内容会被发布到tomcat的webapp下
- 文件夹的名字和当前的工程名字相同



class文件夹存放的class文件,lib文件夹存放web应用需要的jar包还有负责配置web工程的web.xml文件



我们不能将静态资源如css,js,img等放在这个文件夹,因为一旦放入就无法访问,所以应该将这些资源统统放入webContent的根目录下

- 统一路径 `/web应用名称/.../...`

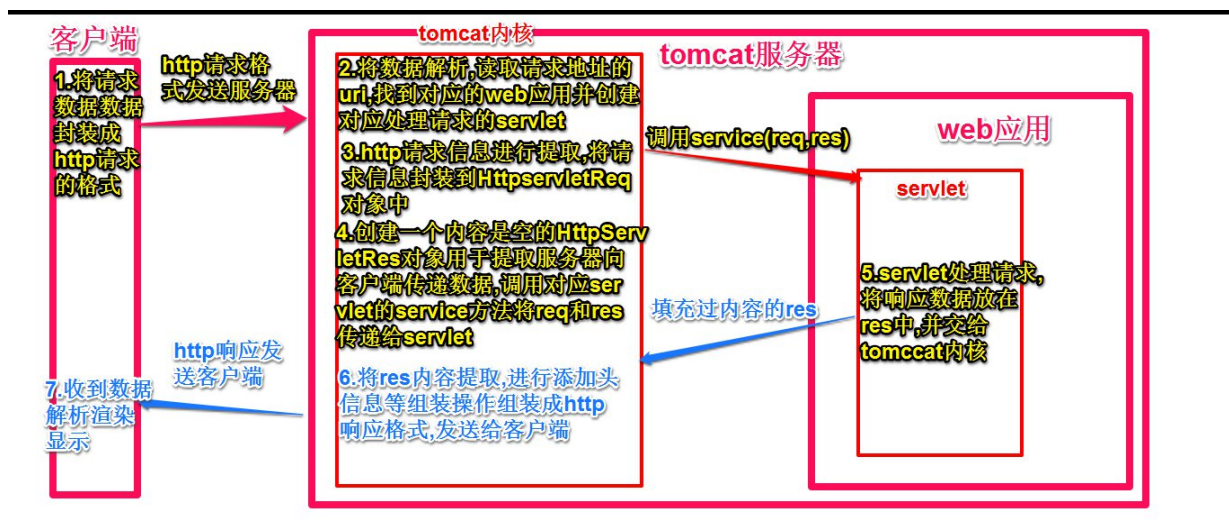
# Servlet

**Servlet** 运行在服务端的Java小程序，是sun公司提供一套规范（接口），用来处理客户端请求、响应给浏览器的动态资源。但servlet的实质就是java代码，通过java的API 动态的向客户端输出内容

## Servlet的内部实现原理

我们需要写java程序来处理客户端发送过来的请求,请思考一个问题,如果我们能否随便的去定义一个任何的类,都能够处理客户端的请求呢?很明显是不能的,显然我们要遵守一个规范,所谓的规范就是sun公司提供的接口.这个接口即是**Servlet**接口.

- servlet规范：包含三个技术点:servlet技术,filter(过滤器)技术,listener(监听器)技术
- 当客户端发送过来一个请求的时候,tomcat会先解析请求的路径,在web.xml的配置文件中去找找到对应匹配的路径,如果找到,就会通过配置文件中servlet-name找到对应的类,然后去创建对应类的对象,此时因为接口规范中规定了,init方法,所以tomcat会去调用相应对象的init方法,然后再去调用service方法,由我们service方法去处理响应的请求



## Servlet生命周期

- init方法:servlet创建的时候执行
- service方法:每次发送请求的时候执行
  - ServletRequest 代表请求 认为ServletRequest 内部封装的是http请求的信息
  - ServletResponse 代表响应 认为要封装的是响应的信息
- destory方法:服务器关闭的时候执行

## Servlet的配置

当我们创建了类并且实现了Servlet接口,需要告知tomcat当有对应请求的时候创建我们的Servlet对象,并调用相应的init,service,destory方法,那么就需要在web.xml中配置相关的Servlet信息.

- `<servlet>` 中的 `<servlet-name>` 中的名字可以任意起,但是需要和 `<servlet-mapping>` 中的 `<servlet-name>` 相同
- `<servlet>` 中的 `<servlet-class>` 指定是对应servlet的类的全名
- `<servlet-mapping>` 中的 `<url-pattern>` 指的是当浏览器中输入什么路径的时候去匹配我们对应的servlet

```
<servlet>
    <servlet-name>myServlet</servlet-name>
    <servlet-class>com.zhiyou100.web.servlet.MyServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>myServlet</servlet-name>
    <url-pattern>/myServlet</url-pattern>
</servlet-mapping>
```

## url-pattern的配置方式

- 完全匹配如 `/myServlet`
- 目录匹配如 `/a/b/c/*`
- 扩展名匹配 `*.do`
- 缺省配置如 `/` 当你访问资源地址所有的servlet都不匹配时,缺省的servlet负责处理其实,web应用中所有的资源的响应都是servlet负责,包括静态资源,对于静态资源其



实是由tomcat的默认servlet进行处理的

- /和/\*区别
  - /不会处理后缀名是.jsp的资源
  - /\*会处理后缀名是.jsp的资源
- 启动服务器的时候创建servlet配置 `<load-on-startup>1</load-on-startup>` 只要不是负数都会随着服务器的启动而创建,值越小优先级越高

```
<servlet>
    <servlet-name>myServlet</servlet-name>
    <servlet-class>com.zhiyou100.web.servlet.MyServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>myServlet</servlet-name>
    <url-pattern>/myServlet</url-pattern>
</servlet-mapping>
```

## 欢迎界面

当我们向浏览器中输入我们的服务器地址,路径只写到项目名称如: `localhost:8080/WebTest/` 此时tomcat会自动按照以下列表从上向下寻找当前web应用根目录下的对应的文件,如果没有才会返回404,如果我们需要可以将我们写的网页配置如web.xml的欢迎界面

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
```

## HttpServlet

在实际的开发过程中,我们不会去创建一个类实现Servlet接口,我们会直接创建一个类去继承HttpServlet,这个是Tomcat提供的一个类,并且实现了Servlet接口

```
public class TTT extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        response.getWriter().write("12133");  
    }  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        doGet(request, response);  
    }  
}
```

- 通过查阅源码,可以看到,HttpServlet的doGet和doPost就是Service方法中进行调用的,所以我们在doGet和doPost方法中写的代码实际上就是在service内部进行调用执行的
- 方法中的两个参数request和response就是请求的对象和响应对象

## request和response内部原理

- 参照客户端和服务端发送数据的具体原理

## ServletContext对象

ServletContext代表的是一个web应用的环境（上下文）对象，ServletContext对象 内部封装是该web应用的信息一个web应用只有一个ServletContext对象

## ServletContext的生命周期

- 创建:当前的web应用创建的时候,一般web应用会随着服务器的启动而创建(或者发布的时候)
- 销毁:web应用被卸载(服务器关闭,或者删除当前的web应用)

## 如何得到ServletContext对象

在doGet或者doPost方法中调用 `this.getServletContext();`,注意在当前web应用中所有的servlet中调用此方法获取的servletContext对象是全局唯一的

## ServletContext作用

- 因为ServletContext对象随着服务器的启动而启动,所以可以通过ServletContext获得web应用全局的初始化参数,在我们日后学习的spring中,就是将Spring文件的路径配置在初始化参数中

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
</context-param>
```

```
public void init() throws ServletException {
    super.init();
    String str= this.getServletContext().getInitParameter("contextConfigLocation");
    System.out.println(str);
}
```

- ServletContext是一个域对象,因为它随着服务器的启动而创建,服务器的关闭而销毁,所以存放在ServletContext中的数据是整个web应用所共享的
  - 向ServletContext域中放入数据 `setAttribute(String name, Object obj);`
  - 从ServletContext域中获取数

据 `getAttribute(String name);`

- 从ServletContext域中删除数

据 `removeAttribute(String name);`

## HttpServletResponse

service方法中的response的类型是ServletResponse，而doGet/doPost方法的response的类型是HttpServletResponse，HttpServletResponse是ServletResponse的子接口

### response运行流程

tomcat内核发送一个空内容的response对象,供我们去将需要的内容放入

### response设置响应行

- 如果使用response设置了状态码,那么tomcat就不会再去设置状态码了

```
response.setStatus(302);
```

### response设置响应头

设置不同的头,客户端收到消息后会做响应的操作

```
response.setStatus(302);  
response.setHeader("Location", "/WebTest/index.html");
```

- 以上功能能够实现重定向,原理就是当客户端收到响应后,响应头信息中含有 `Location` 是 `/WebTest/index.html`,所以客户端就会自动跳转,当然我们如果要去实现重定向,可以直接使用封装好的方法,但是其内部实现还是上述内容.

## response重定向

重定向相当于客户端发送第二次请求

```
response.sendRedirect("/WebTest/index.html");
```

## 重定向的过程分析



## 重定向特点

- 重定向过程是客户端收到消息后,再做的请求,所以浏览器的地址会发生改变
- 服务器收到的请求是两次请求

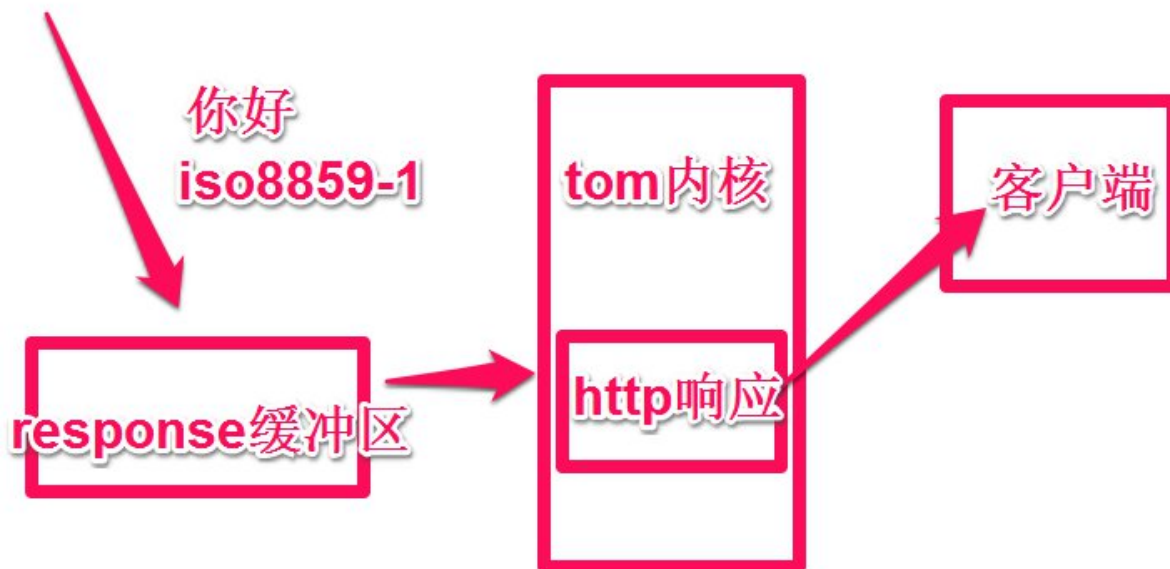
## response设置响应体

通过response可以将客户端需要的数据放入响应体中,其过程是先将返回的信息放在response的缓冲区中,然后由tomcat读取缓冲区的内容,封装成http的响应内容发送给客户端

```
response.getWriter().write("大家好");//将数据写入response缓冲区
```

- 乱码原因分析





- 此过程会出现乱码,原因是将字符串写入缓冲区的时候,使用的是默认的ISO8859-1码表,所以出现乱码,所以我们要设置response查询的码表

```
response.setCharacterEncoding("UTF-8");
```

- 但是客户端进行显示的时候有可能还会出现乱码,原因是因为客户端浏览器可能解码的时候不是使用UTF-8,所以我们需要设置一个响应头,通知客户端使用响应的码表进行解码

```
response.setHeader("Content-Type", "text/html;charset=UTF-8");
```

- 但是如果 we 设置了

```
response.setHeader("Content-Type",
```

`"text/html; charset=UTF-8");` tomcat会自动为我们  
将response缓冲区的编码表设置为utf-8,所以我们只设  
置这一个就行了