

day27-tomcat和servlet入门

学习目标

- 1.能够理解软件的架构
- 2.能够理解WEB资源概念
- 3.能够理解WEB服务器
- 4.能够启动关闭Tomcat服务器
- 5.能够解决Tomcat服务器启动时遇到的问题
- 6.能够运用Tomcat服务器部署WEB项目
- 7.能够使用idea编写servlet
- 8.能够使用idea配置tomcat方式发布项目
- 9.能够使用注解开发servlet

一,WEB开发介绍

1.WEB资源分类

1.1静态资源

- web页面中供人们浏览的数据始终是不变。
- html,css,js....

1.2动态资源

- 指web页面中供人们浏览的数据是由程序产生的，不同的用户,不同时间点访问web页面看到的内容各不相同。
- servlet
- jsp,asp,php

2.软件架构

1.1C/S架构

Client / Server,客户端和服务端，用户需要安装专门客户端程序。

1.2B/S架构

Browser / Server,浏览器和服务端，不需要安装专门客户端程序，浏览器是操作系统内置。

1.3 B/S 和C/S交互模型的比较

- 相同点

都是基于请求-响应交互模型:即浏览器（客户端）向服务器发送一个请求。服务器向浏览器（客户端）回送一个响应。

必须先有请求 再有响应

请求和响应成对出现

- 不同点

实现C/S模型需要用户在自己的操作系统安装各种客户端软件（百度网盘、腾讯QQ等）；实现B/S模型，只需要用户在操作系统中安装浏览器即可。

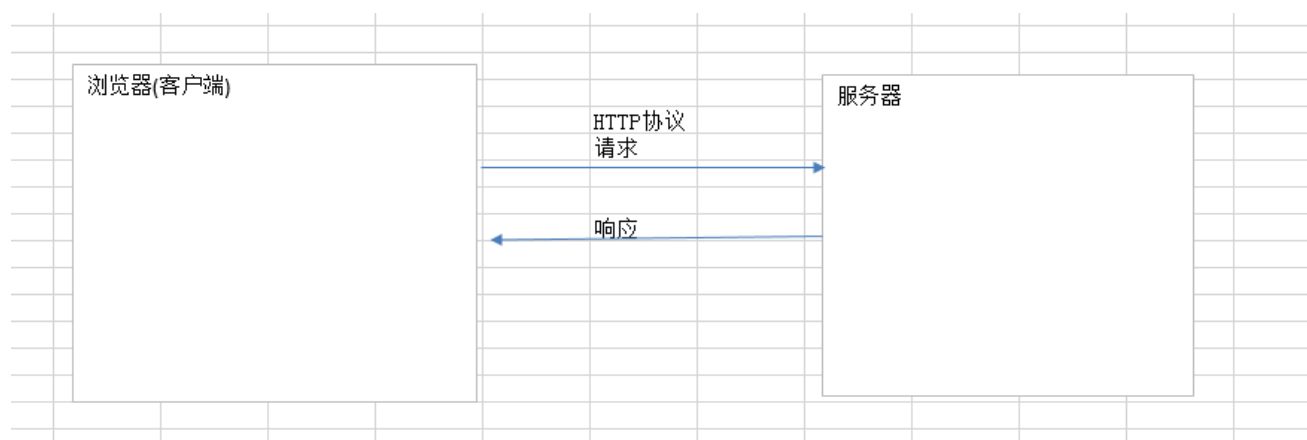
注：B/S模型可以理解作为一种特殊C/S模型。

3.web通信【重点】

基于http协议,请求响应的机制

请求一次响应一次

先有请求后有响应



二,服务器

1.服务器介绍

1.1 什么是服务器

服务器就是一个软件，任何电脑只需要安装上了服务器软件，我们的电脑就可以当做一台服务器了. 然后该电脑的指定目录下的资源就能提供对外访问的资源。

1.2常见服务器

- WebLogic

Oracle公司的产品，是目前应用最广泛的Web服务器，支持J2EE规范。WebLogic是用于开发、集成、部署和管理大型分布式Web应用、网络应用和数据库应用的Java应用服务器。

- WebSphere

IBM公司的WebSphere，支持JavaEE规范。WebSphere 是随需应变的电子商务时代的最主要的软件平台，可用于企业开发、部署和整合新一代的电子商务应用。

- Glass Fish

最早是Sun公司的产品，后来被Oracle收购，开源，中型服务器。

- JBoss

JBoss公司产品，开源，支持JavaEE规范，占用内存、硬盘小，安全性和性能高。

- Tomcat

中小型的应用系统，免费开源，支持JSP和Servlet。

2. tomcat服务器软件安装和介绍

2.1概述

Tomcat服务器是一个免费的开放源代码的Web应用服务器。Tomcat是Apache软件基金会（Apache Software Foundation）的Jakarta项目中的一个核心项目，由Apache、Sun和其他一些公司及个人共同开发而成。由于有了Sun的参与和支持，最新的Servlet和JSP规范总是能在Tomcat中得到体现。

因为Tomcat技术先进、性能稳定，而且免费，因而深受Java爱好者的喜爱并得到了部分软件开发商的认可，是目前比较流行的Web应用服务器。

2.2 tomcat的下载

1. 先去官网下载：<http://tomcat.apache.org/>，选择tomcat8版本（红框所示）：

2. 选择要下载的文件：

tar.gz 文件 是linux操作系统下的安装版本

exe文件是window操作系统下的安装版本

zip文件是window操作系统下压缩版本（我们选择zip文件）

2.3 tomcat服务器软件安装

1. 直接解压当前这个tomcat压缩包：

2. 配置环境变量：

tomcat运行依赖于java环境：

2.4启动与关闭tomcat服务器

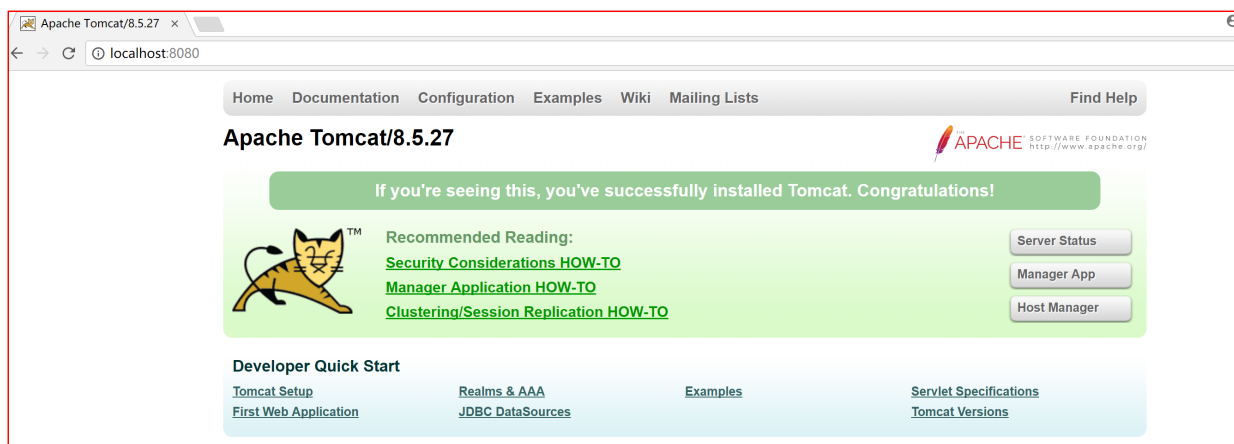
1. 启动tomcat服务器

查找tomcat目录下bin目录，查找其中的startup.bat命令，双击启动服务器：

2. 测试访问tomcat服务器

打开浏览器在，在浏览器的地址栏中输入：

<http://localhost:8080>



注：Localhost相当于127.0.0.1

3. 关闭tomcat服务器

查找tomcat目录下bin目录，查找其中的shutdown.bat命令，双击关闭服务器：

3.Tomcat常见问题解决

3.1端口号冲突

报如下异常: java.net.BindException: Address already in use: JVM_Bind

解决办法:

修改Tomcat的端口号

卷 (E:) > worksoft > tomcat > apache-tomcat-8.5.27-43 > apache-tomcat-8.5.27 > conf				
名称	修改日期	类型	大小	
Catalina	2018/8/5 9:16	文件夹		
catalina.policy	2018/1/18 20:13	POLICY 文件	14 KB	
catalina.properties	2018/1/18 20:13	PROPERTIES 文件	8 KB	
context.xml	2018/1/18 20:13	XML 文件	2 KB	
jaspic-providers.xml	2018/1/18 20:13	XML 文件	2 KB	
jaspic-providers.xsd	2018/1/18 20:13	XSD 文件	3 KB	
logging.properties	2018/1/18 20:13	PROPERTIES 文件	4 KB	
server.xml	2018/1/18 20:13	XML 文件	8 KB	
tomcat-users.xml	2018/1/18 20:13	XML 文件	3 KB	
tomcat-users.xsd	2018/1/18 20:13	XSD 文件	3 KB	
web.xml	2018/1/18 20:13	XML 文件	170 KB	












修改conf/server.xml，第70行左右

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1"
<!-- A "Connector" using the shared thread pool -->
<!--
```

3.2JAVA_HOME没有配置

- 会出现闪退

4.Tomcat的目录结构

	bin	可执行文件
	conf	配置文件
	lib	tomcat运行时需要用的jar文件
	logs	tomcat运行产生的日志文件
	temp	运行时产生临时文件
	webapps	*** 存放我们编写的应用
	work	tomcat的工作目录
	LICENSE	
	NOTICE	
	RELEASE-NOTES	
	RUNNING.txt	

5.标准的JavaWeb应用目录结构【重点】

WebAPP

- | --- 静态资源: html,css,js
- | --- WEB-INF 固定写法。此目录下的文件不能被外部(浏览器)直接访问
 - | --- lib: jar包存放的目录
 - | --- web.xml: 配置文件(3.0规范之后可以省略)
 - | --- classes: 自定义类编译后生成class文件存放的路径

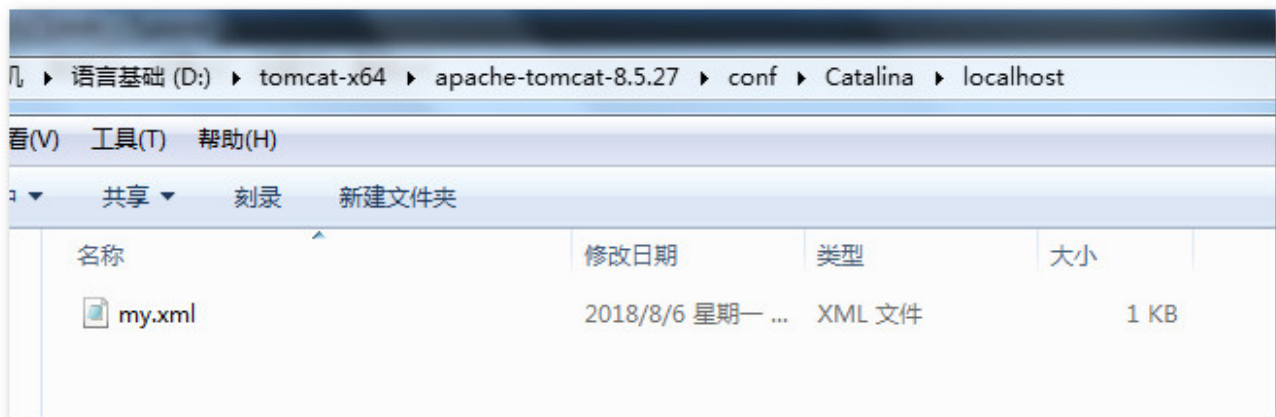
6.发布项目到tomcat

6.1 方式一:直接发布

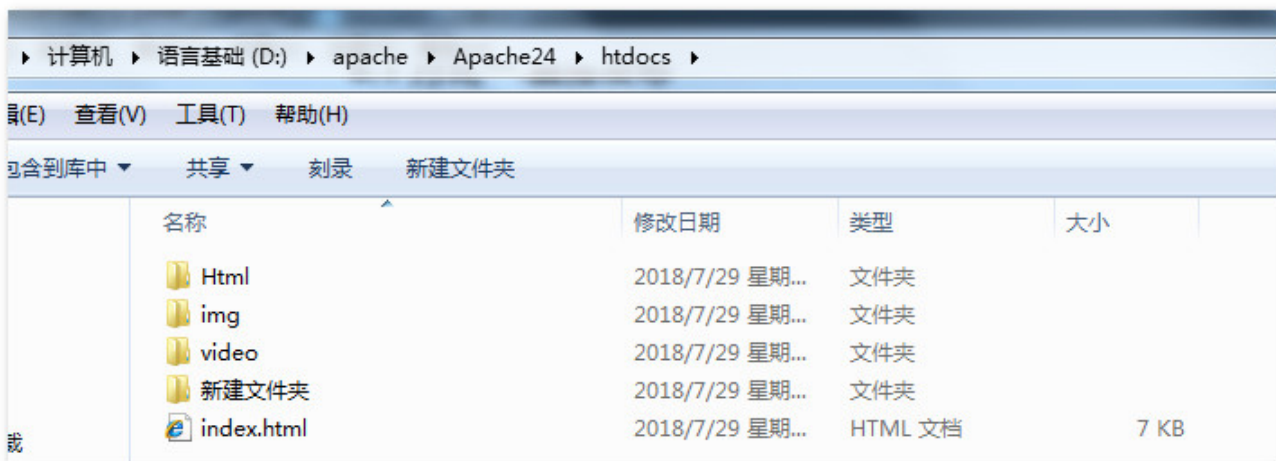
只要将准备好的web资源直接复制到tomcat/webapps文件夹下,就可以通过浏览器使用http协议访问获取

6.2虚拟路径的方式发布项目

第一步: 在tomcat/conf/Catalina/localhost目录下创建xml配置文件



第二步：添加my.xml文件的内容为：docBase就是你需要作为虚拟路径的项目的路径



```
<?xml version = "1.0" encoding = "utf-8"?>
<Context docBase="D:/apache/Apache24/htdocs" />
```

第三步：打开服务器,直接访问(通过写配置文件的路径来访问):



<http://localhost/my/index.html> (my就是配置文件的名字, 映射成了htdocs文件夹)

7. tomcat与nginx的区别

- 设计目的

Tomcat是一个免费的开源的Servlet容器，实现了JAVAE规范，遵循http协议的服务器

Nginx是一款轻量级的电子邮件（电子邮件遵循IMAP/POP3协议）代理服务器，后来又发展成可以部署web应用程序和进行反向代理的服务器

- 存放内容

tomcat可以存放静态和动态资源

nginx只可以存放静态资源

- 应用场景

tomcat用来开发和测试javaweb应用程序

nginx用来做负载均衡服务器

三, http协议

1.HTTP协议概述

HTTP是HyperText Transfer Protocol(超文本传输协议)的简写，传输HTML文件。

HTTP作用：用于定义WEB浏览器与WEB服务器之间**交换数据**的过程和数据本身的**格式**。

浏览器和服务交互过程: 浏览器请求, 服务请求响应

请求:

请求行

请求头

请求体

响应:

响应行

响应头

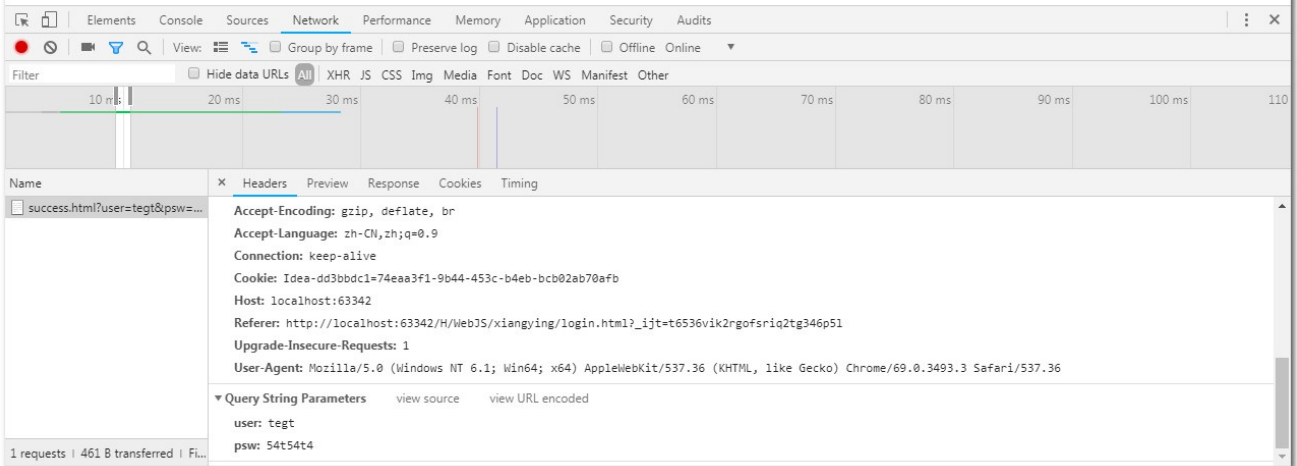
响应体

2.请求部分

使用谷歌浏览器抓包

注册成功

按F12后看Network



POST /day12 myApp/login.html HTTP/1.1 请求消息行

Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/p
Referer: http://localhost:8080/day12_myApp/login.html
Accept-Language: zh-CN,en-US;q=0.5
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: localhost:8080
Content-Length: 20
Connection: Keep-Alive
Cache-Control: no-cache

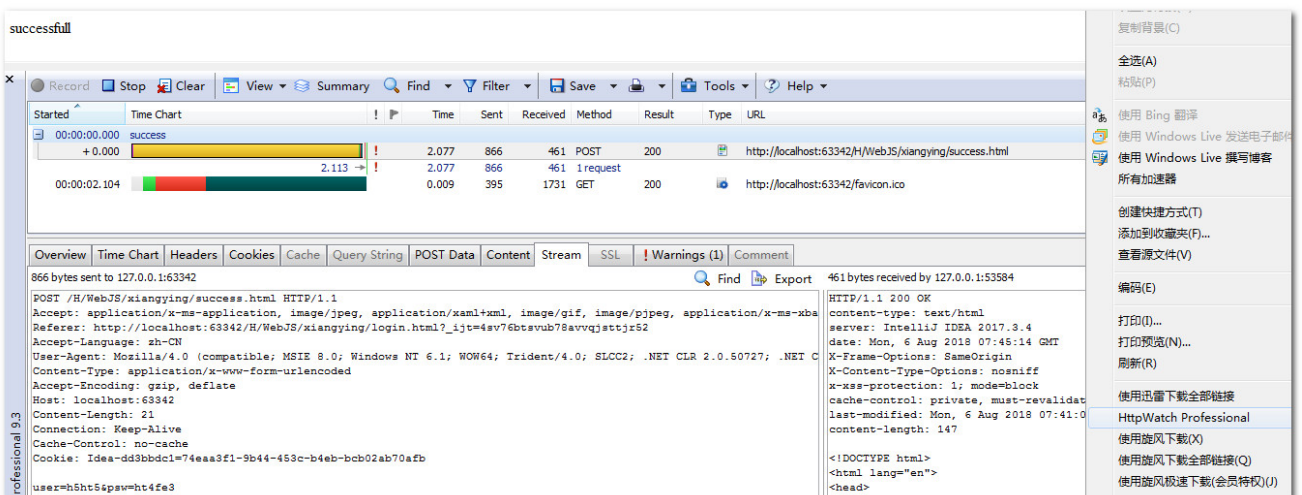
请求消息头

username=tom&pwd=333

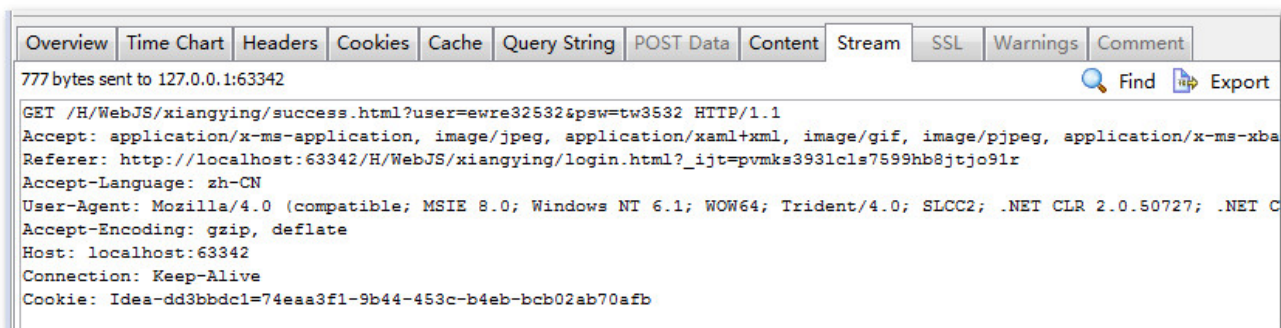
请求正文

注：客户端浏览器向服务器传递消息（悄悄话）

- post请求 使用IE浏览器HttpWatch插件抓包



- get请求



2.1请求行

- 请求行

```
POST /H/WebJS/xiangying/success.html HTTP/1.1
GET /H/WebJS/xiangying/success.html?user=ewre32532&psw=tw3532 HTTP/1.1
```

- 请求方式(7种,put,delete等)
GET:明文传输, 不安全,参数跟在请求路径后面,对请求参数大小有限制,
POST: 暗文传输,安全一些,请求参数在请求体里,对请求参数大小没有有限制,
- URI:统一资源标识符 (即 : 去掉协议和IP地址部分)
- 协议版本:HTTP/1.1

2.2请求头

从第2行到空行处 , 都叫请求头,以键值对的形式存在,但存在一个key对应多个值的请求头.

作用:告诉服务器,浏览器相关的设置.

- Accept:浏览器可接受的MIME类型 ,告诉服务器客户端能接收什么样类型的文件。
- **User-Agent**:浏览器信息.
- Accept-Charset: 浏览器通过这个头告诉服务器 , 它支持哪种字符集
- Content-Length:表示请求参数的长度
- Host:初始URL中的主机和端口
- **Referrer**:来自哪个页面、防盗链
- **Content-Type**:内容类型,告诉服务器,浏览器传输数据的MIME类型 , 文件传输的类型,application/x-www-form-urlencoded .
- Accept-Encoding:浏览器能够进行解码的数据编码方式 , 比如gzip
- Connection:表示是否需要持久连接。如果服务器看到这里的值为“Keep -Alive” , 或者看到请求使用的是HTTP 1.1 (HTTP 1.1默认进行持久连接)
- **Cookie**:这是最重要的请求头信息之一(会话技术, 后面会有专门的时间来讲的)
- Date : Date: Mon, 22Aug 2011 01:55:39 GMT请求时间GMT

2.3请求体

只有请求方式是post的时候,才有请求体. 即post请求时,请求参数所在的位置

3.响应部分

HTTP/1.1 200 OK 响应消息行

Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"556-1460798378245" 响应消息头
Last-Modified: Sat, 16 Apr 2016 09:19:38 GMT
Content-Type: text/html
Content-Length: 556
Date: Sat, 16 Apr 2016 09:26:13 GMT

<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="Generator" content="EditPlus">
<meta name="Author" content="">
<meta name="Keywords" content="">
<meta name="Description" content="">

响应正文

服务器向客户端发送的消息（悄悄话）

• 响应部分



谷歌



3.1 响应行

HTTP/1.1 200 OK

- 协议/版本
- 响应状态码（记住）

状态码	含义
100~199	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
200~299	表示成功接收请求并已完成整个处理过程，常用200
300~399	为完成请求，客户需进一步细化请求。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
500~599	服务器端出现错误，常用 500

200:正常,成功

302:重定向

304:表示客户机缓存的版本是最新的，客户机可以继续使用它，无需到服务器请求. 读取缓存

404:客户端错误(一般是路径写错了,没有这个资源)

500:服务器内部错误:

- 响应码的描述

3.2响应头

- **Location:** <http://www.it315.org/index.jsp>指示新的资源的位置,通常和状态码302一起使用，完成请求重定向
- **Content-Type:** text/html; charset=UTF-8; 设置服务器发送的内容的MIME类型,文件下载时候
- **Refresh:** 5;url=<http://www.it315.org>指示客户端刷新频率。单位是秒
- **Content-Disposition:** attachment; filename=a.jpg 指示客户端(浏览器)下载文件
- **Content-Length:**80 告诉浏览器正文的长度
- **Server:**apachetomcat 服务器的类型
- **Content-Encoding:** gzip服务器发送的数据采用的编码类型
- **Set-Cookie:**SS=Q0=5Lb_nQ;path=/search服务器端发送的Cookie
- **Cache-Control:** no-cache (1.1)
- **Pragma:** no-cache (1.0) 表示告诉客户端不要使用缓存
- **Connection:**close/Keep-Alive
- **Date:**Tue, 11 Jul 2000 18:23:51 GMT

3.3响应体

页面展示内容, 和网页右键查看的源码一样

四,servlet入门

1.Servlet概述

1.1什么是Servlet

Servlet 运行在服务端的Java小程序，是sun公司提供一套规范，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

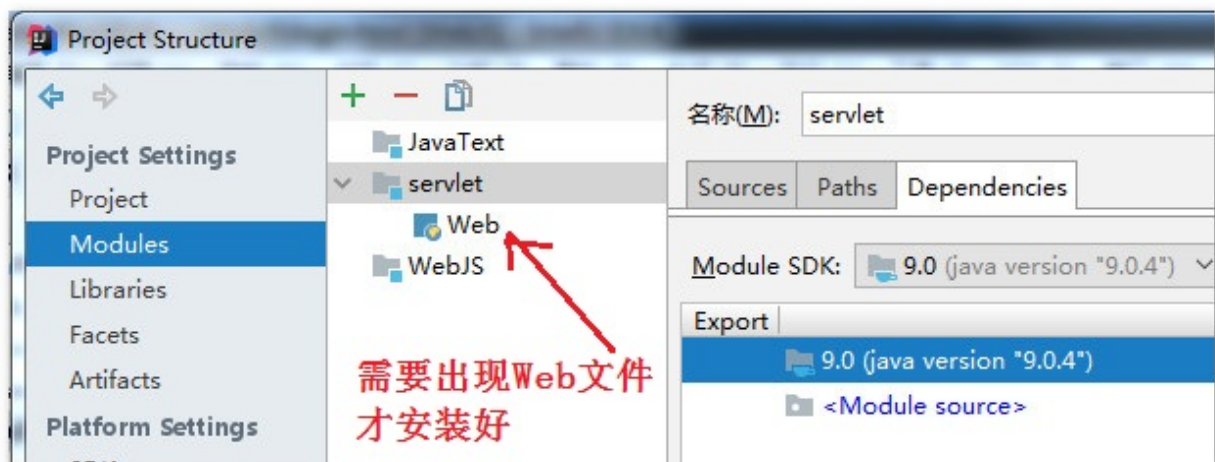
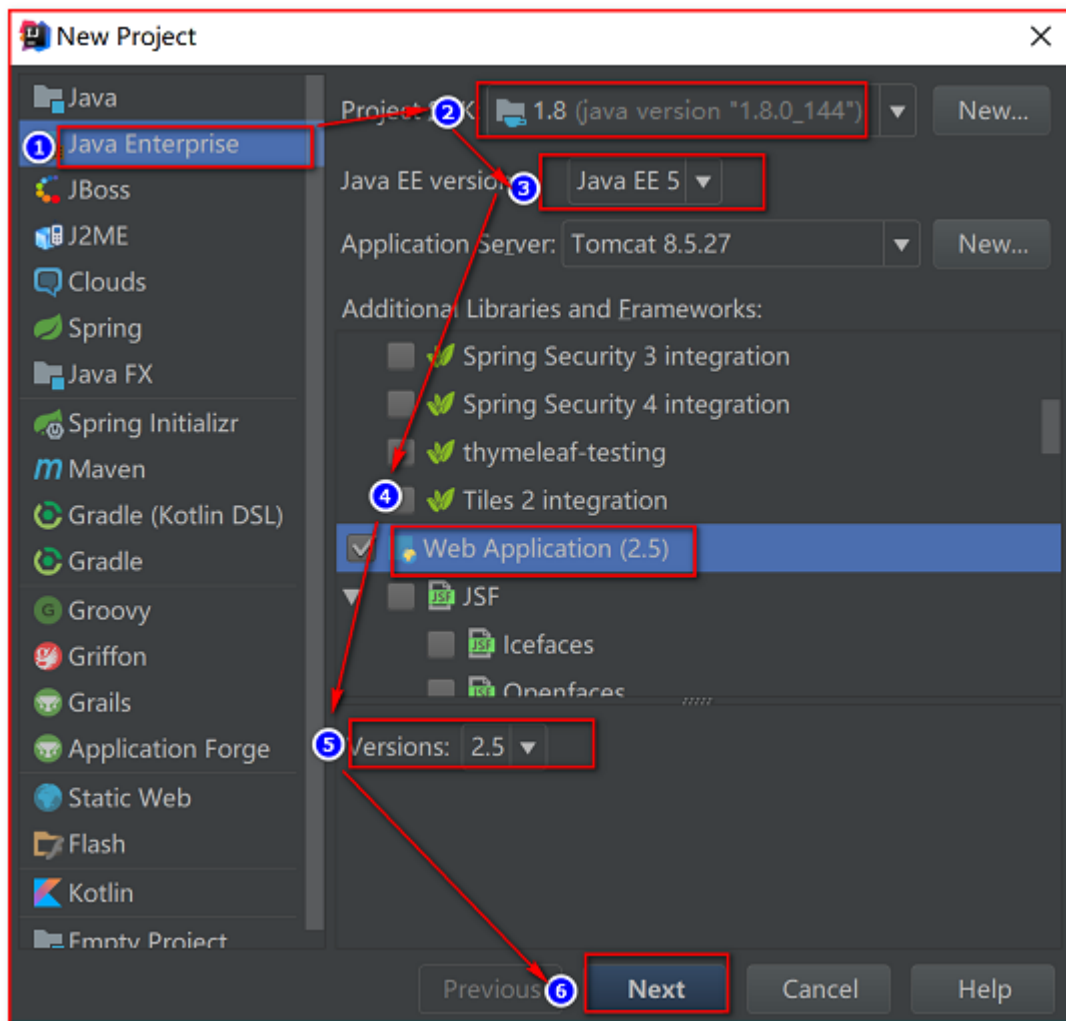
1.2 servlet与普通的java程序的区别

1. 必须实现servlet接口
2. 必须在servlet容器（服务器）中运行
3. servlet程序可以接收用户请求参数以及向浏览器输出数据

2.Servlet2.5实现HelloWorld

2.1实现步骤

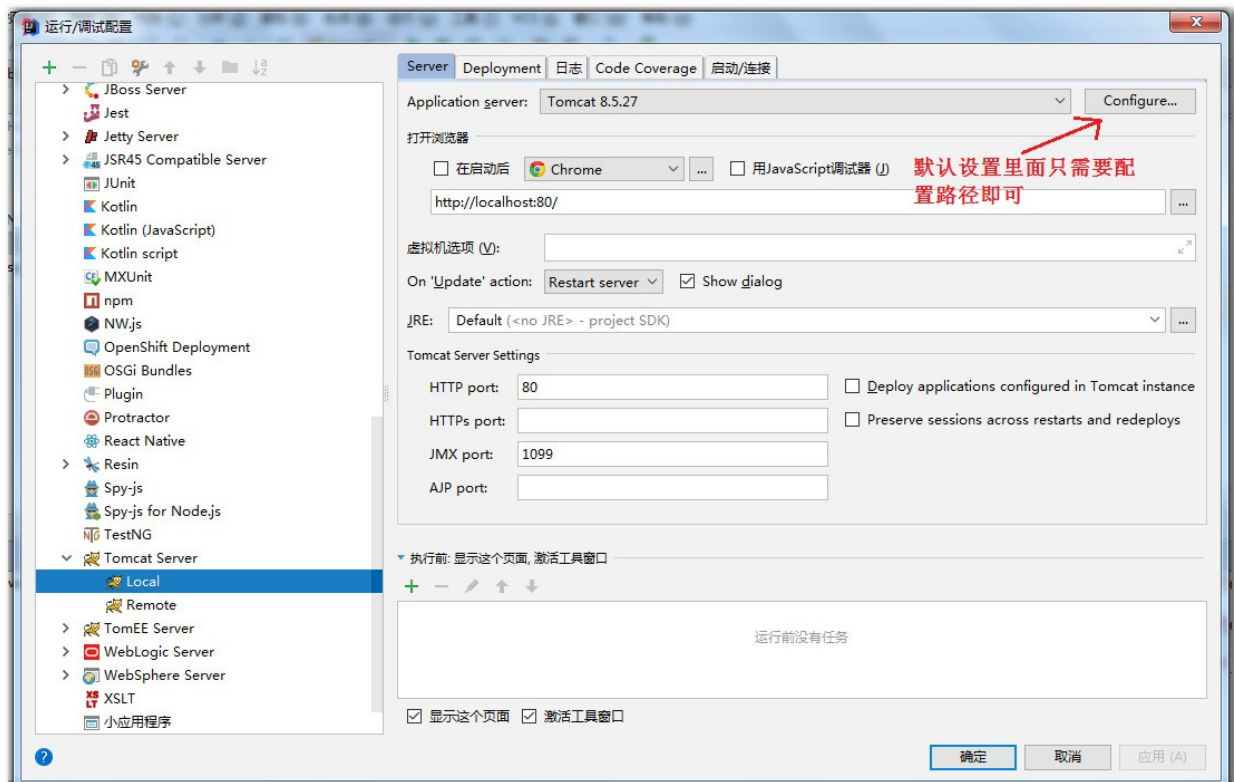
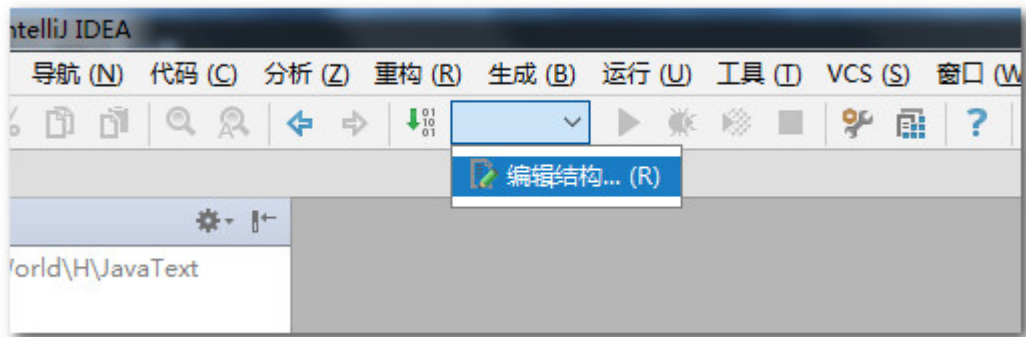
1. 创建web工程



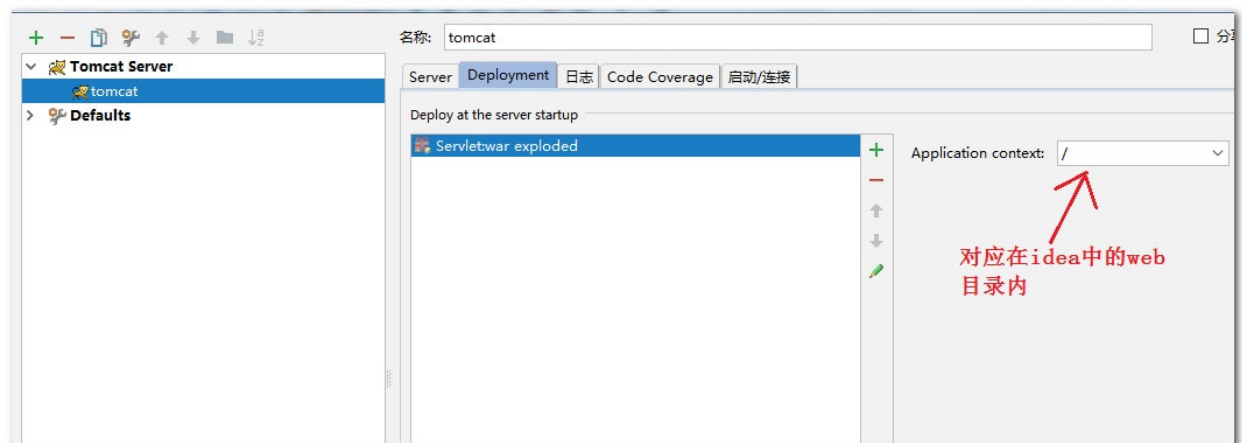
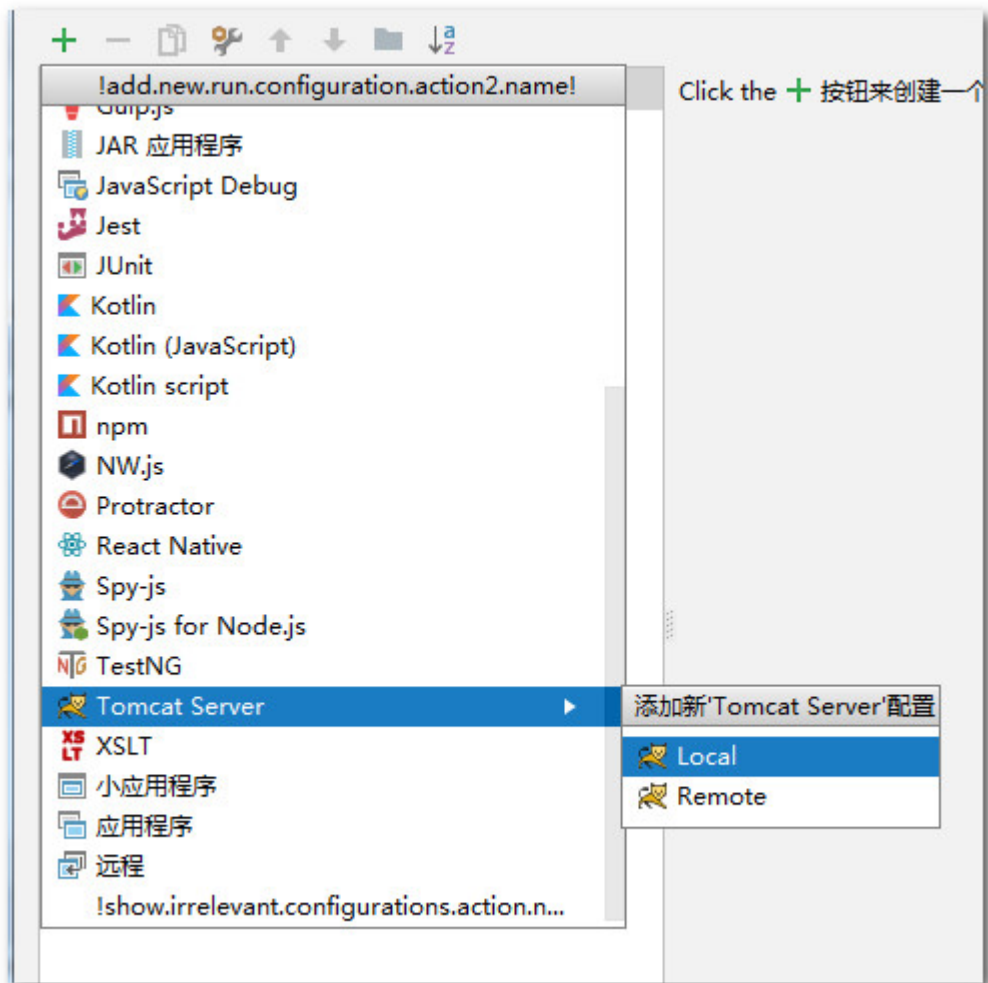
2.2 idea配置tomcat方式发布

Servlet已经书写完成，接下来，我们要将书写好的servlet发布到tomcat上去。接下来我们要将idea和tomcat集成到一起，可以通过idea就控制tomcat的启动和关闭：

1. 添加tomcat服务器部署项目



增加外部选项



点击OK设置完成！

2. 启动服务器：

3. 浏览器测试访问：

3. Servlet2.5和Servlet3.0实现Hello world例子

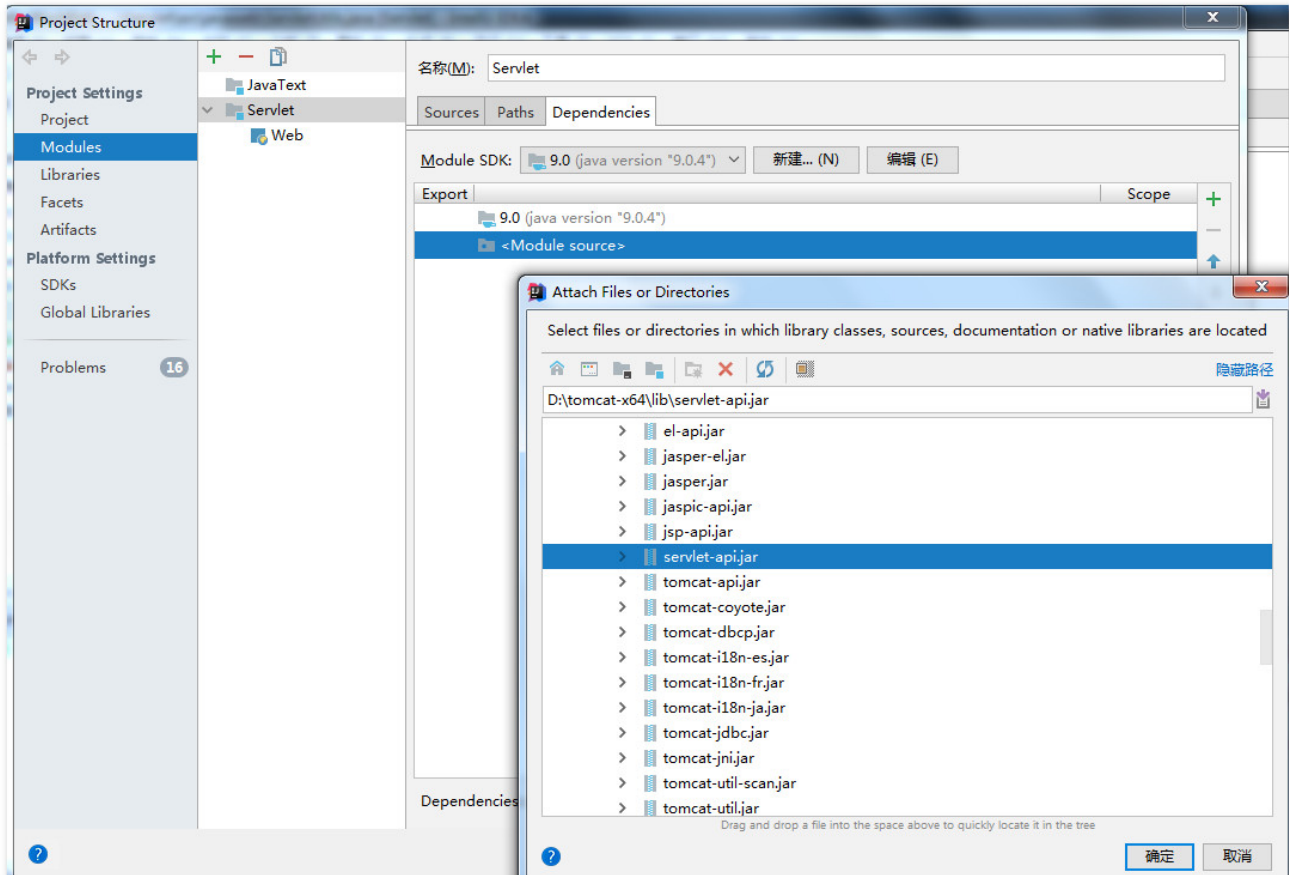
3.1 Servlet2.5与Servlet3.0的区别

- 新增了一些注解，简化的javaweb代码开发，可以省略web.xml配置文件

- 支持异步处理（多线程技术）
- 支持可插性特性（书写的代码编译后生成的class文件可以直接部署到其他项目的，自动加载执行）

3.2 Servlet2.5实现步骤

1,创建java类继承Servlet接口,引用该接口需要在Tomcat里面需要添加jar包依赖



2,java代码演示：

```
public class ServletUtils implements Servlet {  
    @Override  
    public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws  
ServletException, IOException {  
        System.out.println("HelloWorld");  
    }  
    @Override  
    public void init(ServletConfig servletConfig) throws ServletException {  
    }  
    @Override  
    public ServletConfig getServletConfig() {  
        return null;  
    }  
    @Override  
    public String getServletInfo() {  
        return null;  
    }  
    @Override  
    public void destroy() {  

```



```
}  
}
```

3.web.xml配置访问路径



```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app 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"  
  version="2.5">  
  <servlet>  
    <servlet-name>Servlet</servlet-name>  
    <servlet-class>javax.servlet.ServletUtils</servlet-class>  
  </servlet>  
  <servlet-mapping>  
    <servlet-name>Servlet</servlet-name>  
    <url-pattern>/servlet</url-pattern>  
  </servlet-mapping>  
</web-app>
```

4.启动服务器

5.在浏览器后缀输入访问路径/servlet,观察控制台有无响应

3.3Servlet3.0实现步骤

@WebServlet注解替换掉了xml文件,创建java类,主类上增加WebServlet注解,直接在注解内配置映射名和访问路径

```
@WebServlet(name = "se", urlPatterns = "/ser")  
public class ServletU implements Servlet {  
  @Override  
  public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws  
    ServletException, IOException {  
    System.out.println("Hello 3.0");  
  }  
}
```

```

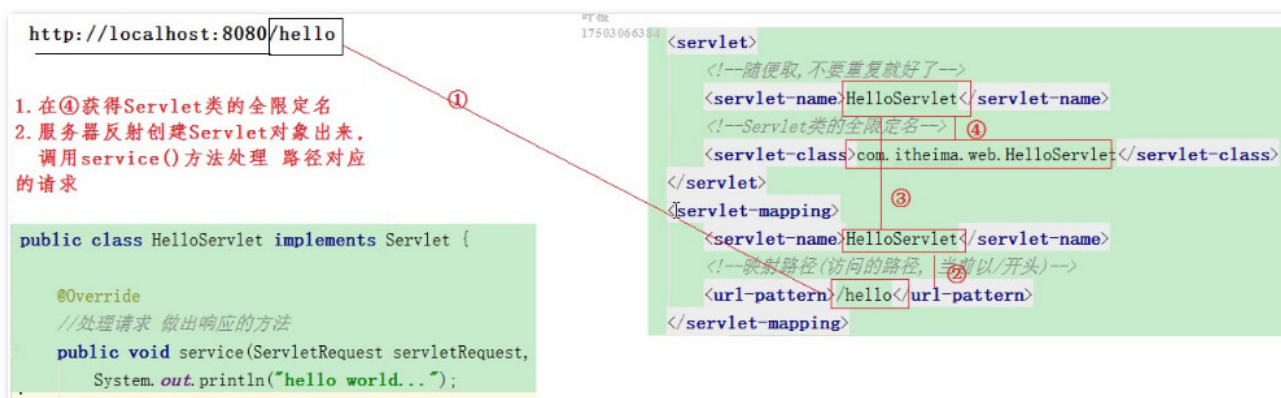
@Override
public void init(ServletConfig servletConfig) throws ServletException {
}
@Override
public ServletConfig getServletConfig() {
    return null;
}
@Override
public String getServletInfo() {
    return null;
}
@Override
public void destroy() {
}
}

```

启动服务器

在浏览器后输入访问路径/ser,观察控制台有无响应

4.Servlet执行原理



通过上述流程图我们重点需要掌握如下几个点：

- Servlet对象是由服务器创建
- request与response对象也是由tomcat服务器创建
- request对象封装了浏览器过来的所有请求信息，response对象代表了服务器的响应信息。
- 当配置的访问路径和从浏览器传来的请求路径一致时服务器内部才会调用service方法,这个路径相当于标记

5.Servlet路径的配置url-pattern

url-pattern配置方式共有三种:

- 完全路径匹配: 以 / 开始。 注: 访问的路径不能多一个字母也不能少一个

例如: /hello02 访问: 必须 /hello02

- 目录匹配"以 / 开始需要以 * 结束. 注: Servlet里面用的 不多, 但是过滤器里面通常就使用目录匹配

例如: /aa/* 访问: /aa/demo01, /aa/bb
url-pattern配置的是 /aa/* 则浏览器地址栏可以/aa/随便写

- 扩展名匹配不能以 / 开始, 以 * 开始的. 注框架里面用的 SpringMVC框架里面会用

例如: *.action; 访问: aa.action, bb.action, c.action; 错误写法: /*.do, 不可以写*.jsp, *.html
url-pattern配置的是 *.action 则浏览器地址栏可以 a.action b.action ...

注意的地方:

- 一个路径只能对应一个servlet, 但是一个servlet可以有多个路径
- tomcat获得匹配路径时, 优先级顺序: 完全路径匹配> 目录匹配 > 扩展名匹配