# 一、Javaweb

## 1、定义

在Sun的Java Servlet规范中，对Java Web应用作了这释：

定义：“Java Web应用由一组Servet、HTML页、类、以及其它可以被绑定的资源构成。它可以在各种供应商提供的实现Servlet规范的Serlet容器中运行。”

## 2、内容

Servlet

JSP

实用类

静态的文档如HTML、图片等

描述web应用的信息（web.xml）

## 软件的架构：

1、C/S（客户端/服务器）(client/server)

1、一般我们使用的软件都是C/S结构

2、比如系统的中软件qq、360、office

3、C表示客户端，用户通过客户端来使用软件

4、S表示服务器，服务器负责软件处理软件的业务逻辑

5、特点：1）软件使用前必须的安装，软件更新时，服务器和客户端同时更新。

2）C/S架构的软件不能跨平台使用。

3）C/S架构的软件客户端和服务器通信采用的是自由协议，相对来说比较安全。

2、B/S 浏览器/服务器

1、B/S本质上也是C/S，只不过B/S架构的软件，使用浏览器作为软件的客户端。

2、B/S架构软件通过使用浏览器访问网页的形式，来使用软件。

3、比如：京东，淘宝、12306.

4、特点：软件不需要安装，直接使用浏览器访问指定的网页。

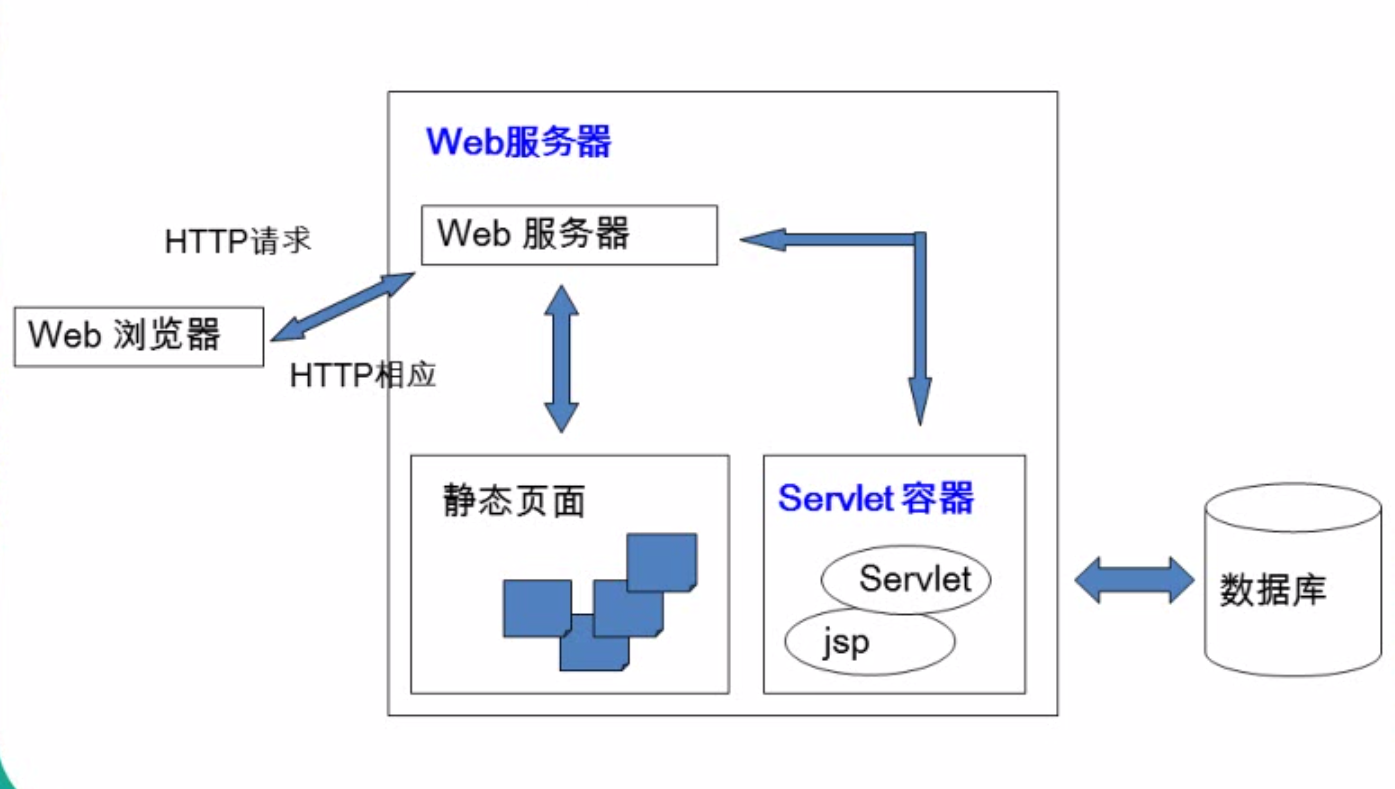
2）软件更新时，客户端不跟新

3）软件可以跨平台。只要系统有浏览器，就可以使用。

4）B/S架构的软件，客户端和服务器之间通信采用通用的Http协议。

5)相对来说不安全。可http +s 加了密

## 3、web资源介绍运行原。



Web浏览器发送一个请求到web服务器，web服务器会解析这个请求是请求动态资源还是静态资源。如果是静态页面，会直接找到并并返回。如果是动态资源，则由servlet容器中的servlet进行解析，解析的过程可能为与数据库发生关系，然后将servlet返回的结果由jsp处理，返回一个html形式返回。Servlet类访问数据库采用JDBC。

重点：Servlet容器和在Servlet容器中写servlet和jsp，可以被servlet调用。

### web浏览器

浏览器是指 Web 服务的客户端浏览程序。 它可以向服务器发送各种请求， 并对从服务器中返回的各种信息（包括文本、超文本、音频、视频等各种数据）进行解释、显示和播放。

### 2、web服务器

浏览器与服务器的关系可谓是 “唇齿相依飞浏览器发送请求， 服务器处理请求井将结果返回给浏览器显示。Web 服务器的种类繁多，目前比较流行的有 WebSphere、WebLogic、Tomcat 等。它们的配置、启动方式各不相同，也各自有其优缺点。

Web 服务器和 JSP/Servlet 容器，通常这两者会集于一身。

#### 1、html:静态资源

#### 2、Servlet容器

Servlet容器：也称为JavaWeb应用容器，或者Servlet/JSP容器。

作用：为JavaWeb应用提供运行时环境，它负责管理Servlet和JSP的生命周期，以及管理它们的共享数据。JSP/Servlet 容器的基本功能是把动态资源 转换成静态资源

1. 作用：Web 服务器的作用是接收客户端的请求，给客户端作出响应
2. Java Servlet（Java服务器小程序）是一个基于Java技术的Web组件，运行在服务器端，它由Servlet容器所管理，用于生成动态的内容。 Servlet是平台独立的Java类，编写一个Servlet，实际上就是按照Servlet规范编写一个Java类。Servlet被编译为平台独立 的字节码，可以被动态地加载到支持Java技术的Web服务器中运行。   
   Servlet容器也叫做Servlet引擎，是Web服务器或应用程序服务器的一部分，用于在发送的请求和响应之上提供网络服务，解码基于 MIME的请求，格式化基于MIME的响应。Servlet没有main方法，不能独立运行，它必须被部署到Servlet容器中，由容器来实例化和调用 Servlet的方法（如doGet()和doPost()），Servlet容器在Servlet的生命周期内包容和管理Servlet。在JSP技术 推出后，管理和运行Servlet/JSP的容器也称为Web容器。

### 3、数据库

### 4、静态资源和动态资源的区别:

静态资源：一般客户端发送请求到web服务器，web服务器从内存在取到相应的文件，返回给客户端，客户端解析并渲染显示出来。

静态网站特点：

　 1. 静态网站是最初的建站方式，浏览者所看到的每个页面是建站者上传到服务器上的一个 html （ htm ）文件，这种网站每增加、删除、修改一个页面，都必须重新对服务器的文件进行一次下载上传。网页内容一经发布到网站服务器上，无论是否有用户访问，每个静态网页的内容都是保存在网站服务器上的，也就是说，静态网页是实实在在保存在服务器上的文件，每个网页都是一个独立的文件；

　 2. 静态网页的内容相对稳定，因此容易被搜索引擎检索；

　 3. 静态网页没有数据库的支持，在网站制作和维护方面工作量较大，因此当网站信息量很大时完全依靠静态网页制作方式比较困难；

　 4. 静态网页的交互性较差，在功能方面有较大的限制。

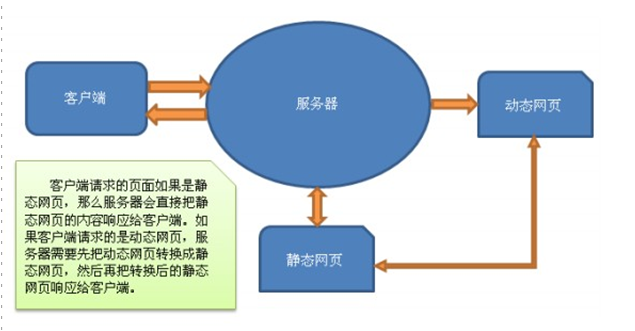
动态资源：一般客户端请求的动态资源，先将请求交于web容器，web容器连接数据库，数据库处理数据之后，将内容交给web服务器，web服务器返回给客户端解析渲染理。

动态网页中的变化内容大部分是来自于数据库中数据的变化。它是通过增加、删除、修改、 查找数据库中存储的数据来显示内容的变化。例如在微博中发布一条微博后，查看微博时，会将所 发的微博即时显示出来，这在静态网页中是无法完成的。

目前比较流行的动态网页技术主要包含：ASP、PHP以及JSP。

1. 交互性：网页会根据用户的要求和选择而动态地改变和响应，浏览器作为客户端，成为一个动态交流的桥梁，动态网页的交互性也是今后 Web 发展的潮流。

　 2. 自动更新：即无须手动更新 HTML 文档，便会自动生成新页面，可以大大节省工作量。

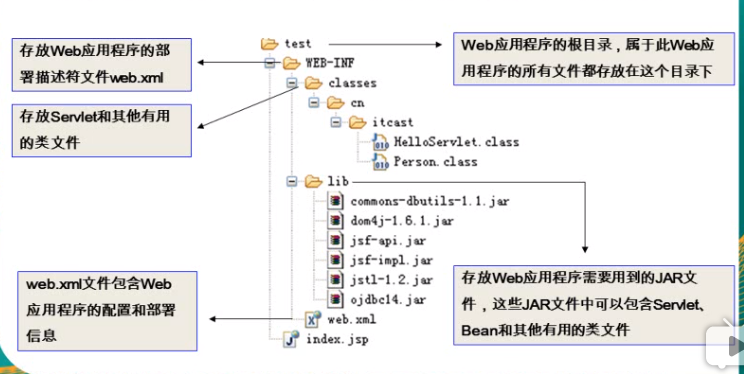
　 3. 因时因人而变：即当不同时间、不同用户访问同一网址时会出现不同页面。

## 4、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

## 5、Web程序结构



一个web应用程序是由一组Servlet，HTML页面，类，以及其它的资源组成的运行在web服务器上的完整的应用程序，以一种结构化的有层次的目录形式存在组成web应用程序的这些文件要部署在相应的目录层次中，根目录代表整个web应用程序的”根”通常将web应用程序的目录放在webapps目录下，在webapps目录下的每一个子目录都是一个独立的web应用程序，子目录的名字就是web应用程序的名字，也就是web应用程序的“根”。用户通过web应用程序的”根”来访问web应用程序中的资源

## 6、web容器（Servlet 引擎）

1、可以创建servlet，并调用servlet的相关声明周期方法。

2、JSP，filter，listener，tag…..都是在servlet容器中运行，并被调永相应的生命周期方法。

主流的servlet容器：

Tomcat（ Apache）：当前应用最广的JavaWeb 服务器；JSP/Servlet容器

注意：Tomcat 不支持JavaEE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；

Resin（ Caucho）：支持 JavaEE，应用越来越广；

Weblogic （ Orcale）：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；

## 7、第一个web程序

第一个WEB应用程序：开发，并部署到Tomcat服务器下运行

(1)在Eclipse新建一个Java Project

(2)在Java项目下创建WEB开发的目录结构

-WebContent（应用名称）

-WEB-INF

-classes（编译后的.class文件必须放在该目录下.）

-1ib

-web.xm1

htm1页面

JSP页面

图片…

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 ，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。

lib：存放jar包的目录。

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.

(3)在src下新建一个person类。

（4）手动的将person类对应的class文件（含包）复制到classes目录中。

注：可以通过修改的默认输出目录达到把编译好的class放到classes目录下：

右键-build path-configure build path-browse

（5）在WebContent目录下新建一个JSP文件。

（6）把WebContent目录复制到Tomcat的webapps目录下，并修改web项目的名称。

（7）、配置外部应用：项目不在tomcat的webapps文件夹下。

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：

# 二、HTTP

## 1、HTTP简介：

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。它是TCP/IP协议集中的一个应用层协议，用于定义WEB浏览器与WEB服务器之间交换数据的过程以及数据本身的格式。

## 2、请求协议和响应协议

格式：

请求首行；

请求头信息；

空行；

请求体。

## 3、GET请求

1、在浏览器地址栏中输入某个URL地址或单击网页上的一个超链接时，浏览器发出的HTTP请求消息的请求方式为GET。

2、如果网页中的<form>表单元素的method 属性被设置为了“GET”，浏览器提交这个FORM表单时生成的HTTP请求消息的请求方式也为GET。

3、使用GET请求方式给WEB服务器传递参数的格式：

<http://www.atguigu.com/counter.jsp?name=lc&password=123>

4、使用GET传递数据一遍在1KB以下。

例如: GET /hello/index.jsp HTTP/1.1 （请求首行，get没有请求体）

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(请求头类似 map)

解释：GET hello/index.jsp HTTP/1.1 ：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。

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-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然后在讲会话是讲究它！

## 4、POST请求

1、POST 请求方式主要用于向WEB服务器端程序提交FORM表单中的数据：form 表单的 method 置POST。

2、POST方式将各个表单字段元素及其数据作为HTTP消息的实体内容发送给WEB服务器，传送的数据量要比使用GET方式传送的数据量大得多。

例如：

表单：

< form action= ""method ="post">

关键字： <input type = "text" name ="keyword" />

< input type ="submit" value = " 提交 " />

</ form >

POST:

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

解释：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“传智”这两个字使用UTF-8 的 url 编码用为“ %E4%BC%A0%E6%99%B”A；

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。

## 5、refer请求头

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。

# 二、Tomcat

## 1、Tomcat简介

Tomcat 服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache 服务器，可利用它响应HTML（标准通用标记语言下的一个应用）页面的访问请求。实际上Tomcat 部分是Apache 服务器的扩展，但它是独立运行的，所以当你运行tomcat 时，它实际上作为一个与Apache 独立的进程单独运行的。

诀窍是，当配置正确时，Apache 为HTML页面服务，而Tomcat 实际上运行JSP 页面和Servlet。另外，Tomcat和IIS等Web服务器一样，具有处理HTML页面的功能，另外它还是一个Servlet和JSP容器，独立的Servlet容器是Tomcat的默认模式。不过，Tomcat处理静态HTML的能力不如Apache服务器。目前Tomcat最新版本为9.0

## 2、Tomcat安装和文件介绍

1.官方下载地址：http://tomcat.apache.org

2.Tomcat分为安装版和解压版：（目录的名称不能是中文）

A.安装版：一台电脑上只能安装一个Tomcat；

B.解压版：无需安装，解压即可用，解压份数无限制，使用比较方便，我们选择解压版。

3、启动Tomcat服务器：

A.启动前必须配置JAVA\_HOME环境变量。因为Tomcat程序本身是用java写的，需要用到JDK。

B. 2、配置Tomcat环境变量:

右击【计算机】---【属性】-----【高级系统设置】---【环境变量】

1，新建用户变量名：CATALINA\_BASE，变量值: D:\apache-tomcat-8.0.39 (可选）

1，新建用户变量名：TOMCAT\_HOME，变量值:D:\apache-tomcat-8.0.39 （可选

2，新建用户变量名：CATALINA\_HOME，变量值：D:\apache-tomcat-8.0.39 （必须）

3.置环境系统变量Path，添加变量值：%CATALINA\_HOME%\lib;%CATALINA\_HOME%\bin

C.进入到Tomcat解压目录下的bin\startup.bat 双击可启动Tomcat服务器

D.进入到Tomcat解压目录下的bin\shutdown.bat 双击即可关闭Tomcat服务器

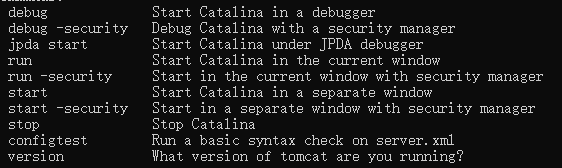
E.启动完成后可以在浏览器地址栏输入：http://localhost:8080 来检查Tomcat是否启动成功，如果看到如下页面。则恭喜你成功了。

F. 若已经启动了一个Tomcat 应用，若再启动同一个Tomcat应用，会抛出异常：

java.net.BindException:Address already in use:JVM\_Bind:8080因为端口已经被占用了。

G：.阅读startup.bat 文件，发现启动tomcat 服务器的是catalina.bat 文件

在命令行窗口直接输入catalina，提示如下：



Run：在catalina同一个命令行窗口下启动服务器。

Start：开启一个新窗口启动服务器

Stop：关闭服务器

4、修改配置

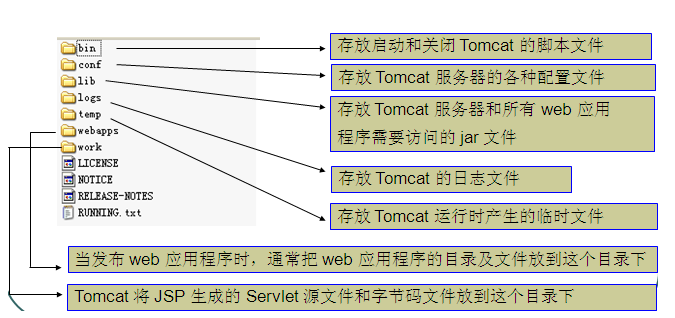
server.xml 中可以 更改服务端口和改变 Web 默认的访问目录。

（1）修改端号

<Connector port=“8080” protocol＝”HTTP/1.1” connectionTimeout=”20000”

reclirectPort=”8443” URIEncoding=”UTF-8”／〉

文件介绍：



1)bin：该目录下存放的是二进制可执行文件会有startup.bat和shutdown.bat文件（其余文件无需理会），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；

2)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；以后会详细介绍这个文件；

tomcat-users.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。

3)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，各个应用需要的jar各个应用自己管理。

4)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。

5)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！

6)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。

7)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。

8)LICENSE：许可证。

9)NOTICE：说明文件。

# 四、Servlet

## 1、servlet简介

Java Servlet是和平台无关的服务器端组件，它运行在Servlet容器中。Servlet容器负责Servlet和客户的通信以及调用Servlet的方法，它是作为来自 Web 浏览器或其他 HTTP 客户端的请求和 HTTP 服务器上的数据库或应用程序之间的中间层。 使用 Servlet，您可以收集来自网页表单的用户输入，呈现来自数据库或者其他源的记录，还可以动态创建网页

## 2、主要功能

1、创建并返回基于客户请求的动态HTML页面。

2、创建可嵌入到现有HTML页面中的部分HTML页面（HTML片段）。

3、与其它服务器资源（如数据库或基于Java的应用程序）进行通信。

## 3、servlet的创建过程

### 1、配置servlet

Servlet程序必须在WEB应用程序的web.xml文件中进行注册和映射其访问路径，才可以被Servlet引擎加载和被外界访问。

一个<servlet>元素用于注册一个Servlet，它包含有两个主要的子元素：<servlet-name>和<servlet-class>，分别用于设置Servlet的注册名称中和Servlet的完整类名。

一个<servlet-mapping>元素用于映射一个已注册的Servlet的一个对外访问路径，它包含有两个子元素：<servlet-name>和<url-pattern>，分别用于指定Servlet的注册名称和Servlet的对外访问路径。

< servlet >

<servlet-name> hello </ servlet-name>注册名

<servlet-class > cn.itcast.servlet.HelloServlet</servlet-class>servlet的完整类名字

<load-on-startup> n</load-on-startup>

</ servlet>

< servlet-mapping>

<servlet-name> hello </ servlet-name>注册名

<url-pattern>/helloworld</url-pattern>对外访问路径

<url-pattern>/AServlet </url-pattern >

<url-pattern>/BServlet </url-pattern >

</ servlet-mapping>

1、配置绑定：

目的在 web.xml 中配置Servlet 的目的其实只有一个，就是把访问路径与一个Servlet 绑定到一起，上面配置是把访问路径：“ /helloworld ”与“ cn.itcast.servlet.HelloServlet ”绑定到一起。

<servlet> ：指定HelloServlet这个Servlet 的名称为hello；

<servlet-mapping> ：指定 /helloworld访问路径所以访问的Servlet 名为 hello 。

<servlet> 和<servlet-mapping> 通过 <servlet-name> 这个元素关联在一起。

注：1、同一个Servlet可以被映射到多个URL上，即多个<servlet-mapping>元素的<serlet-name>子元素的设置值可以是同一个Servlet的注册名。

<url-pattern>n<url-pattern>:<url-pattern> 是<servlet-mapping>的子元素，用来指定 Servlet的访问路径 ，即URL。它必须是以 “ / ”开头！

1、可以在<servlet-mapping>中给出多个<url-pattern>，Servlet 绑定了三个 URL，无论访问 /helloworld 、/AServlet 还是 /BServlet ，访问的都是cn.itcast.servlet.HelloServlet。

2、通配符“\*” 匹配任何URL前缀或后缀

第一种格式：已斜杠开头，并已/\*结尾。

<url-pattern>/servlet/\*<url-patter>

/servlet/a 、 /servlet/b ，都匹配 /servlet/\* ；路径匹配

<url-pattern>\*.do</url-pattern>

/abc/def/ghi.do、/a.do ，都匹配 \*.do ；扩展名匹配

第二种格式：\*.扩展名。

<url-pattern>/\*<url-pattern>

匹配所有URL；

匹配原则：优先级，匹配的越少，优先级越高

请注意，通配符要么为前缀，要么为后缀，不能出现在URL 中间位置，也不能只有通配符。例 如： /\*.do就是错误的，因为星号出现在URL 的中间位置上了。\*.\* 也是不对的，因为一个URL 中最多只能出现一个通配符。

通配符是一种模糊匹配URL的方式，如果存在更具体的<url-pattern> ，那么访问路径会去匹配具体的 <url-pattern>。

1. 配置启动优先级

<load-on-startup> n<load-on-startup>

在 <servlet> 元素中配置 <load-on-startup> 元素可以让服务器在启动时就创建该Servlet ，其中<load-on-startup> 元素的值必须是大于等于的整数，它的使用是服务器启动时创建Servlet 的顺序。

N为负数：则在第一次请求时被创建。

N为0或者整数：则哎当前web应用被servlet容器加载时创建实例，且数越小越早创建。

3、

### 2、servlet生命周期方法：以下方法由servlet容器负责调用

构造器：只被调用一次，只有第一次请求Servlet时，创建Servlet，调用构造器。标明servlet为单实例。

void init(ServletConfig) ；只被调用一次，在创建好实例后立即被调用，用于初始化当前的servlet。

void service(ServletRequest , ServletResponse) ；被多次调用，每次请求都会调用service方法，实际用于响应请求。

void destroy()；只被调用一次，在当前servlet所造的web应用卸载前被调用，用于释放当前servlet所占用的资源。

4、响应请求的过程

(1)Servlet引擎检查是已经装载并创建了该Serviet的实例对象。如果是，则直接执行第（4）步，否则，执行第(2)步。

(2)装载并创建该Servlet的一个实例对象：调用该Servlet的构造器.

(3)调用Servlet实例对象的init()方法。

(4)创建一个用于封装请求的SerlvetRequest对象和一个代表响应消息的ServletResponse对象，然后调用Servlet的service()方法并捋请求和响应对象作为参数传递进去。

(5)WEB应用程序被停止或重新启动之前，Servlet引擎捋卸载Servlet，并在卸载之前调用Servlet的destroy(）方法。

### 3、ServletConfig接口

String getServletName() ：获取 Servlet 在 web.xml 文件中的配置名称，即<servlet-name> 指定的名称；(不怎么用)

ServletContext getServletContext() ：用来获取ServletContext对象，ServletContext 会在后面讲解；（重点）

String getInitParameter(String name) ：用来获取在web.xml 中配置的初始化参数，通过参数名来获取参数值；

Enumeration getInitParameterNames() ：用来获取在 web.xml 中配置的所有初始化参数名称；

1. 配置初始化参数：(该参数在<servlet>标签里面，是servlet的初始化参数)

<servlet>

<servlet-name> hello </ servlet-name>

<servlet-class > cn.itcast.servlet.HelloServlet</servlet-class>

<l--配置 Serlvet的初始化参数。且节点必须在1oad-on-startup节点的前面-->

<init-param>

<！--参数名-->

<param-name>user</param-name>

<！--参数值-->

<param-value>root</param-value>

</init-param>

<init-param>

<param-name>pas sword</param-name>

<param-value>1230</param-value>

</init-param>

<1oad-on-startup>-1</1oad-on-startup>

</servlet>

（2）获取初始化参数：

String getInitParameter(String name)

string user=servletConfig.getInitParameter("user")；

System.out.println(user)；

Enumeration getInitParameterNames()

Enumeration<String>names=servletConfig.getInitParameterNames()；

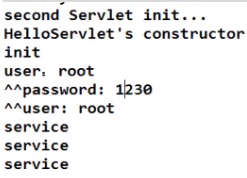
while(names.hasMoreElements()){

String name=names.nextElement()；

String value = servletConfig.getInitParameter(name);

System.out.println("name:"+name+“value”+value)；

}



（3）ServletContext：Servlet擎为每个WEB应用程序都创建一个对应的ServletContext对象，SerletContext对象被包含在SerletConfic对象中，调用ServletConfig.getServletContext方法可以返回ServletContext对象的引用。

功能： 1、该对象可由ServletConfig 通过getServletContext()获取。

2、该对象代表当前WEB应用：可以认为ServletContext是当前WEB应用的一个大管家，可以从中获取当前应用的各个方面的信息。

例如：1、获取当前WEB应用的参数。（可以为所有的servlet所获取，而sevlet的参数只能是servlet获取）

<！--配置当前WEB应用的初始化参数-->

<context-param>

<param-name>driver</param-name>

<param-value>com.mysq1.jdbc.Driver</param-value>

</context-param>

<context-param>

<param-name>jdbcUr1</param-name>

<param-value> jdbc:mysq1:///atguigu </param-value)

</context-param>

获取方法：getInitParameter()

GetInitParameterNames() |

//获取 Servletcontext 对象

Servletcontext servletContext=servletConfig.getservletContext()；

string driver =servletContext.getInitParameter("driver")；

System.out.println("driver:"+driver)；

Enumeration<string>names2=servletcontext.getInitParameterNames()；

while(names2.hasMoreElements()){

String name=names2.nextlement()；

System.out.println(\"-->\"+name)；|

}

2、获取当前web应用的某一个文件相对于服务器的绝对路径

getRealpath（String path）

3、获取当前web应用的名称 getContextPath（）

String contextPath=servletcontext.getContextPath()；

System.out.println(contextPath)；

4、获取当前WEB引用的某一个文件对应输入流。

getResourceAsStream（String path）:path的/为当前web的根目录。

## 4、ServletRequest和ServletResponse

在Servlet中获取请求信息

Public void service(ServletRequest,ServletResponse)

ServletRequest:封装了请求信息。可以从中获取到任何的请求信息。

ServletResponse:封装了响应信息，如果想给用户什么响应，具体可以使用该接口的方法实现。

### 1、ServletRequest接口（查看api中的方法）

request 是Servlet.service()方法的一个参数，类型为javax.servlet.http.HttpServletRequest。在客户端发出每个请求时，服务器都会创建一个request 对象，并把请求数据封装到 request 中，然后在调用 Servlet.service() 方法时传递给 service()方法，这说明在service()方法中可以通过 request 对象来获取请求数据。

1、获取请求参数（servletrequest封装了请求信息）

String getParameter(String name):根据请求参数的名字，返回参数值。请求参数有多个值（例如 checkbox），该方法只能获取到第一个提交的值。

String[]getParameterValues(String name):根据请求参数的名字，返回请求参数对应的字符串数组。

Enumeration getParameterNames():返回参数名对应的Enumeration对象，似于ServletConfig(或ServletContext)的getInitParameterNames()方法。

例如：

Enumeration<String> names = request.getParameterNames();

while（names.hasMoreElements）(){

String name=names.nextElement()；

String val=request.getParameter(name)；

system.out.print1n(name+":"+val)；

}

Map getParameterMap():返回请求参数的键值对：key：参数名，value：参数值，String 数组类型。

例如：

Map<String,String[]>map=request.getParameterMap();

for(Map.Entry<String,String[]>entry:map.entrySet()){

System.out.printin(entry.getkey()+\":\"+Arrays.aslist(entry.getValue()))；

2、获取请求的URL(与http协议相关的要转换为httpSevletRequest)

HttpServletRequest httpServletRequest=(HttpServletRequest)request；

string requestURI=httpServletRequest.getRequestURI()；

System.out.println(requestURI)；

3、获取请求方式

String method=httpServletRequest.getMethod()；

System.out.print1n(method)；

4、若是一个GET请求，获取请求参数对应的那个字符串，即？后的那个字符串。

string queryString=httpServletRequest.getQueryString();

System.out.println(queryString)；

注：如果是POST，返回值为null。

5、和attribute相关的几个方法

### 2、HttpServletRequest

HttpServletRequest：是SerlvetRequest的子接口。针对于HTTP请求所定义。里边包含了大量获取HTTP请求相关的方法。

### 3、ServletResponse（接口方法）

response是Servlet.service 方法的一个参数，类型为 javax.servlet.http.HttpServletResponse。在客户端发出每个请求时，服务器都会创建一个response 对象并传入给 Servlet.service()方法。response 对象是用来对客户端进行响应的，这说明在service()方法中使用response 对象可以完成对客户端的响应工作。

1、响应正文：response是响应对象，向客户端输出响应正文响应体可以使用 response的响应流，repsonse一共提供了两个响应流对象。

getWriter():返回Printwriter对象。调用该对象的print()方法，将把print()中的参数直接打印到客户的浏览器上。

PrintWriter out = response.getWriter() ：获取字符流

ServletOutputStream out = response.getOutputStream()：获取字节流 ；

注：在一个请求中，不能同时使用这两个流！也就是说，要么你使用repsonse.getWriter() ，要么使用 response.getOutputStream() ，但不能同时使用这两个流。不然会抛出IllegalStateException异常。

2、设置响应内容类型 ：

设置响应的内容类型：response.setContentType("application/msword")；

例如：返回一个word文档

response.setContentType("application/msword")；//word的格式

PrintWriter out=response.geturiter()；

out.print1n("helloworld...")；//返回一个word，word的内容是“helloworld”

## 5、genericServlet（Servlet的实现类）

1. GenericServlet是Servlet接口和ServletConfig接口的实现类，但是一个抽象类。其中的service方法为抽象方法。
2. 如果新建的servlet程序直接继承GenericServlet开发会更加简便。
3. 在GenericServlet中声明的ServletCionfigl类型的成员变量，在init（ServletConfig）方法中对其进行初始化。
4. 还定义了一个init（）方法，在init（Serlvetconfig)方法中对其进行调用，子类可以直接覆盖init（）在其中实现对Servlet的初始化。
5. 不建议直接覆盖init（Servletconfig），因为如果忘记编写 super（Servletconfig)，而还是用了Serlvetconfig接口的方法，则会出现空指针异常。（第二种解决方法）
6. 新建的init（）并非servlet的生命周期方法，而init（ServletConfig）是生命周期的相关方法。

|GenericServlet.java

Public abstract class GenericServlet implements Servlet, ServletConfig, java.io.Serializabl

//该类可以被序列化和反序列化

private static final long serialVersionUID = 1L;

private transient ServletConfig config ;//

public GenericServlet() {}

@Override

public void destroy() {}

@Override

public String getInitParameter(String name) {

return getServletConfig().getInitParameter(name);

}

@Override

public Enumeration<String> getInitParameterNames() { return getServletConfig().getInitParameterNames();

}

@Override

public ServletConfig getServletConfig() {

return config ;

}

@Override

public ServletContext getServletContext() {

return getServletConfig().getServletContext();

}

@Override

public String getServletInfo() {

return "" ;

}

@Override

public void init(ServletConfig config) throws ServletException {

this . config = config;//将服务器创建的servletconfig对象传给实现累的config

this .init();

}

public void init() throws ServletException {} public void log(String msg) {

getServletContext().log(getServletName() + ": " + msg);

//该方法为新建的方法，并非servlet的生命周期方法。

}

public void log(String message, Throwable t) { getServletContext().log(getServletName() + ": " + message, t);

}

@Override

public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException;

@Override

public String getServletName() {

return config .getServletName();

}

}

## 6、**HttpServlet**

1、是一个Servlet，继承自Genericservlet.针对于HTTP协议所定制。

2、.在service（）方法中直接把 ServletReuqest 和ServletResponse转为HttpServletRequest 和HttpServletResponse并调用了重载的 service(HttpServletRequest,HttpservletResponse)。

3、在service(HttpServletRequest,HttpServletResponse)获取了请求方式：request.getMethod().根据请求方式有创建了doxxx（)方法（xxx为具体的请求方式，比如doGet，doPost)

4、实际开发中，直接继承HttpServlet，并根据请求方式复写doXxx（）方法接口。

5、好处：直接由针对性的覆盖doXXX（）；直接使用HttpServletRequest和HttpServletReponse，不需要强转。

### 1、request转为httprequest

public void service(ServletRequest req,ServletResponse res)

throws ServletException,IOException{

HttpServletRequest request；

HttpServletResponse response;

try{

request=(HttpServletRequest)req;

response=(HttpServletResponse)res；

}catch(ClassCastException e){

throw new ServletException(\"non-HTTP request or response\")；

service(request,response)；

public void service(HttpServletRequest request,HttpServletResponse response)

throws ServletException,IOException{

//1.获取请求方式。

String method=rehuest.getMethod()；

//2.根据请求方式再调用对应的处理方法

if(\"GET\".equalsIgnoreCase(method)){

doGet(request,response)；

}else if(\"poST\".equalsIgnorecase(method)){

doPost(request,response)；

}

public void doPost(HttpServletRequest request,HttpServletResponse response)

throws ServletException,IOException{

public void doGet(HttpServletRequest request,HttpServletResponse response)

throws ServletException,IOException{

}

# 五、JSP

## 1、Jsp简介（java server pages）

JSP是简 Servlet编写的一种技术，它将Java代码和HTML语句混合在同一个文件中编写，只对网页中的要动态产生的内容采用Java代码来编写，而对固定不变的静态内容采用普通静态HTML页面的方式编写。

1、JSP可以放置在WEB应用程序中的除了WEB-INF及其子目录外的其他任何目录中，

JSP页面的访间路径与普通HTML页面的访间路径形式也完全一样。

2、JSP本质上是已个Servlet。WEB容器（Servlet引擎）接收到以.jsp为扩展名的URL的访问请求时，它将把该访问请求交给JSP引擎去处理。每个JSP页面在第一次被访问时，JSP引擎将它翻译成一个Servlet源程序，接着再把这个Servlet源程序编译成Servlet的cdass类文件，然后再由WEB容器（Serlet引擎）像调用普通Servlet程序一样的方式来装载和解

释执行这个由JSP页面翻译成的Serlet程序。

3、JSP规范也没有明确要求JSP中的脚本程序代码必须采用Java语言，JSP中的脚本程序代码可以采用Java语言之外的其他脚本语言来编写，但是JSP页面最终必须转换成Java Servlet程序。

4、JSP规范也没有明确要求JSP中的脚本程序代码必须采用Java语言，JSP中的脚本程序代码可以采用Java语言之外的其他脚本语言来编写，但是，JSP页面最终必须转换成Java Servlet程序。

## 2、JSP的helloworld

新建一个Jsp页面，在body节点楼内<% %>即可编写jav代码

<body>

<%

Date date = **new** Date();

System.out.print(date);

%>

</body>

## 3、Jsp页面的隐含变量

隐含对象：.JSP页面的隐含变量：没有声明就可以使用的对象。JSP页面一共有9个隐含对象.

public void \_jspService(HttpServletRequest request,HttpServletResponse response)

throws java.io.IOException,ServletException{

PageContext pagecontext=null；

HttpSession session=null；

Servletcontext application=null;

ServletConfig config=null；

JspWriter out=null；

Obiect page=this:

//………(其他代码)

//使用<%%>编写的代码在此位置。可以用到request,response,pageContext,session

//application,config,out,page 这8个隐含对象。（实际上还可以使用一个叫exception的隐含对象）

1、request:HttpServletRequest 的一个对象。（常用）

2、.response:HttpServletResponse的一个对象（在JSP页面中几乎不会调用response的任何方法.)（几乎不用）

3、pageContext:页面的上下文，是PageContext的一个对象。可以从该对象中获取到其他8个隐含对象。也可以从中获取到当前页面的其他信息。（学习自定义标签时使用它。（很少使用）

4、session:代表浏览器和服务器的一次会话，是HttpSession的一个对象。后面详细学习。常用

5、application:代表当前WEB应用。是Servletcontext对象。常用

6、config：当前JSP对应的 Servlet的 Servletconfig 对象（几乎不使用）。若需要获取JSP配置的初始化参数，需要通过映射的地址才可以。几乎不用

Web.xml中的代码：

<servlet>

<servlet-name>hellojsp</servlet-name>

<jsp-file>/hel1o.jsp</jsp-file>

<init-param>

<param-name>test</param-name>

<param-value>testValue</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>hellojsp</servlet-name>

<ur1-pattern>/hellojsp</url-pattern>

</servlet-mapping>

1. out:JspWriter 对象。调用out.print1n（“<br/>”)可以直接把字符串打印到浏览器上。常用
2. page:只想当前JSP
3. exception：特殊的情况下使用。

<% page isErrorPage=”true”%> 指定为true是才能够使用。

pageContex t,request, session ,appocation(对属性的作用于范围从小到大)

## 2、jsp的基本语法

1、jsp模板元院

模板元素：JSP页面中国的静态Html内容称之为JSP模板元素。

2、‘<%=变量 %>

.JSP表达式（expression)提供了将一个java变量或表达式的计算结果输出到客户端的简化方式，它将要输出的变量或表达式直接封装在<%=和%>之中。

<%

Date date=new Date()；

out.print(date)；

%>

<%=date%>//效果一样

3、JSP的脚本片段

JSP脚本片断（scriptlet）是指嵌套在<%和%>之中的一条或多条Java程序代码。在JSP脚本片断中，可以定义变量、执行基本的程序运算、调用其他Java类、访问数据库、访问文件系统等普通Java程序所能实现的功能。在JSP脚本片断可以直接使用JSP提供的隐式对象来完成WEB应用程序特有的功能。JSP脚本片断中的Java代码将被原封不动地搬移进由JSP页面所翻译成的Serlet的ispService方法中，所以，JSP脚本片断之中只能是符合Java语法要求的程序代码，除此之外的任何文本、HTML标记、其他JSP元素都必须在脚本片断之外编写。JSP脚本片断中的Java代码必须严格遵循Java语法，例如，每条命执行语句后面必须用分号（；）结束。在一个JSP页面中可以有多个脚本片断（每个脚本片断代码嵌套在各自独立的一对<%和%>之间），在两个或多个脚本片断之间可以嵌入文本、HTML标记和其他JSP元素。

<%

String ageStr=request.getParameter("age")；

Integer age=Integer.parseInt(ageStr);

if(age>=18){

%>

成人...

<%

}else{

%>

未成人...

<%

4、JSP声明（JSP代码中基本不这样使用）

JSP声明：JSP声明将Java代码封装在<%！和%>之中，它里面的代码将被插入进Servlet的\_jspService方法的外面。

5、Jsp的注释

<%-- JSP注释--> <!— Html 的注释-->

区别：Jsp的注释可以阻止java代码的执行。

# 六、域对象

## 1、简介

## 2、方法

void setAttribute(String name, Object value)：用来存储一个对象，也可以称之为存储一个域属性，例如： servletContext.setAttribute(“xxx”, “XXX”)，在 ServletContext 中保存了一个域属性， 域属性名称为 xxx，域属性的值为 XXX。请注意，如果多次调用该方法， 并且使用相同的name ， 那么会覆盖上一次的值，这一特性与Map 相同；

Object getAttribute(String name) ：用来获取 ServletContext 中的数据， 当前在获取之前需要先 去存储才行，例如： String value =(String)servletContext.getAttribute( “xxx”);，获取名为 xxx 的 域属性；

void removeAttribute(String name)：用来移除 ServletContext中的域属性，如果参数name 指 定的域属性不存在，那么本方法什么都不做；

Enumeration getAttributeNames()：获取所有域属性的名称

pagecontext, request, session, application 对象都有这些方法，四个对象为域对象。

pageContext:属性的作用范围仅限于当前JSP页面

request:属性的作用范围仅限于同一个请求。

session:属性的作用范围限于一次会话：浏览器打开直到关闭称之为一次会话（在此期间会话不失效）

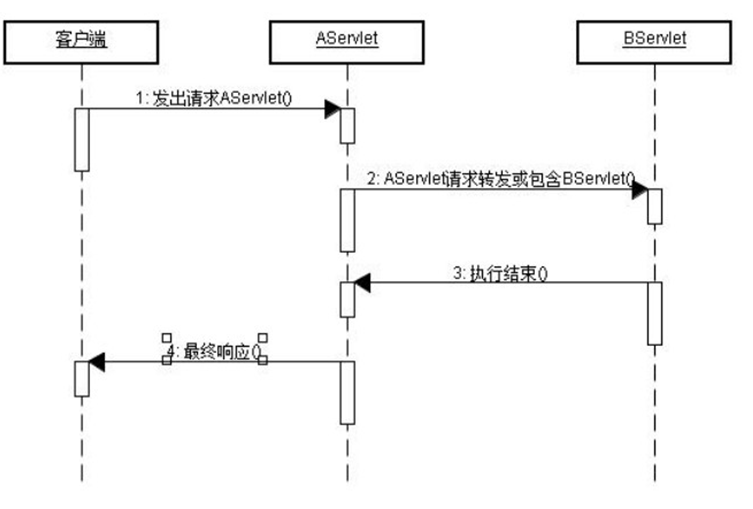
Application：属性的作用范围限于当前的web应用

# 七、请求重定向和转发

1、请求转发：

多个Servlet 共同来处理一个请求

请求转发过程：浏览器向服务器发送一个请求（发送到web容器），服务器第一次访问AServlet时创建AServlet，并创建ServletRequest和ServletRsponse，然后传递给Aservlet，Aservlet读取请求信息和写入响应信息，发出forward指令给了web容器，然后服务器继续给访问BServlet，重复访问操作，最后服务器将响应传递给客户端。



（1）调用HttpServletRequest的getRequestDispatcher()方法获取RequestDispatcher 对象调用 getRequestDispatcher（)需要传入要转发的地址。

String path = “testServlet”;

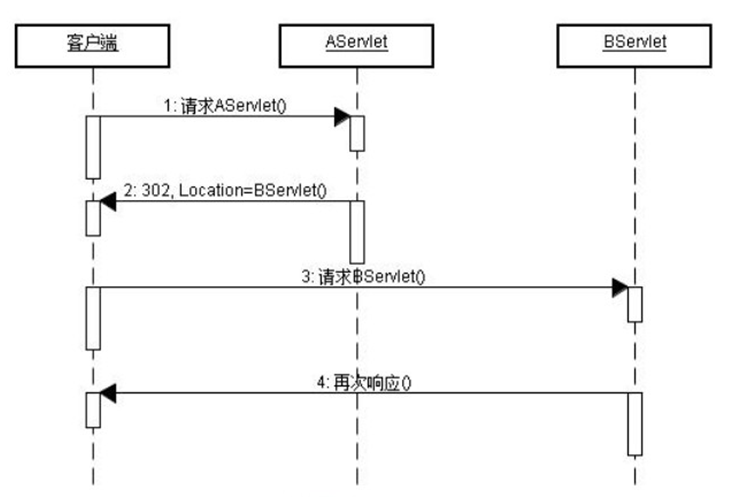
RequestDispatcher requestDispatcher=request.getRequestDispatcher(path);

(2)调用HttpServletRequest的forward(request,response)进行请求的转发。

requestDispatcher.forward(request,response);|

## 2、请求重定向

浏览器向服务器发送一个请求（发送到web容器），服务器第一次访问AServlet时创建AServlet，并创建ServletRequest和ServletRsponse，然后传递给Aservlet，Aservlet读取请求信息和写入响应信息返回服务器，服务器将Http响应传递给客户端。客户端继续发出请求响应Bservlet。



执行请求的重定向，直接调用response.sendRedirect(path)方法，

path 为要重定向的地址

string path="testservlet"；

response.sendRedirect(location)；

3、请求转发和请求重定向的区别：

1、请求的转发只发出了一次请求，重定向发出了两次请求

2、请求的转发：地址栏是初次发出请求的地址。

请求的冲定向:地址栏不再是初次发出的请求地址，地址栏为最后响应的地址。

3、请求转发：在最终的Servlet中，request 对象和中转的那个request是同一个对象。

请求的重定向：在最终的 Servlet中，request 对象和中转的那个request不是同一个对象。

4、请求的转发：只能转发给当前WEB应用的的资源

请求的重定向：可以重定向到任何资源。

5、请求的转发：/代表的是当前WEB应用的根目录

请求的重定向：/代表的是当前WEB站点的根目录。

# 八、JSP指令

JSP指合（directive)是为JSP引擎而设计的，它们并不直接产生任何可见输出，而只是告诉引拿如何处理JSP页面中的其余部分。

## 1、指令格式;

JSP指合的基本语法格式：

<%@指合属性名=“值”%>

举例：<%@page contentType=text/html;charset=gb2312“%>

注意：属性名部分是大小写敏感的

## 2、结合方式

如果要在一个JSP页面中设置同一条指合的多个属性，可以使用多条指合语句单独设置每个属性，也可以使用同一条指合语句设该指合的多个属性。

第一种方式：

<%@page contentType="text/htmlkcharset=gb2312"%>

<%@ page import=java.util.Date"%>

第二种方式：

<%@page contentType="text/html;charset=gb2312"

import=java.uti.Date"%>

## 3、page指令

page指合用于定义JSP页面的各种属性，无论page指合出现在JSP页面中的什么地方，它作用的都是整个JSP页面，为了保持程序的可读性和遵循良好的编程习惯，page指令最好是放在整个JSP页面的起始位置。

<%@page

[language="java"] //当前JSP页面使用的语言是java，但合法的语言只有java一个。

[extends="package.class"] //Servlet被编译成源文件是能够继承哪些类。（没什么用）

[import="{package.class I package.},.…"]导入包

import javax.servlet.\*; //以下为jsp默认导入的包

import javax.servlet.http.\*;

import javax.servlet.jsp.\*;

[ session="trueI false"] //session 是true或者是false表示是否能够与使用隐藏变量session

errorPage和isErrorPage

>errorPage 指定若当前页面出现错误的实际响应页面时什么。其中/表示的是当前WEB应用的根目录。

[%@page errorPage="/WEB-INF/error.jsp”%](mailto:%25@page%20prrorPage=\%22/error.jsp)

isErrorPage 指定当前页面是否为错误处理页面，可以说明当前页面是否可以使用exception隐藏变量。需要注意的是：若指定isErrorPage="true",并使用exception的方法了，一般不建议能够直接访问该页面。

在响应error.jsp时，JSP引擎使用的请求转发的方式。

>如何使客户不能直接访间某一个页面呢？对于Tomcat 服务器而言，WEB-INF下的文件是不能通过在浏览器中直接输入地址来访问的。但通过请求的转发是可以的！

contentType:指定当前JSP页面的响应类型。（可以在tomcat/conf/web.xml中查看文本类型。）

实际调用的是response.setContentType("text/html；charset=UTF-8")；

通常情况下，对于JSP页面而言其取值均为text/html；charset=UTF-8，charset返回指定的编码是什么。

pageEncoding：指定当前JSP页面的字符编码。通常情况下该值和 contentType中的 charset一致。

isELIgnored:指定当前JSP页面是否可以使用EL表达式。

4、include指令

include指合用于通知JSP引擎在翻译当前JSP页面时将其他文件中的内容合并进当前JSP页面转换成的Servlet源文件中，这种在源文件级别进行引入的方式称之为静态引入，当前JSP页面与静态引入的页面紧密结合为一个Servlet。

2、格式

<%@include file=\"relativeURL\"%>

其中的file属性用于指定被引入文件的相对路径。

3、.如果以/开头，表示相对于当前WEB应用程序的根目录（注意不是站点根目录），否则，表示相对于当前文件

# 九、jsp标签

JSP还提供了一种称之为Action的元素，在JSP页面中使用Action元秦可以完成各种通用的JSP页面功能，也可以实现一些处理复杂业务逻辑的专用功能。

1、<isp:include>标签用于把另外一个资源的输出内容插入进当前JSP页面的输出内容之中，这种在JSP页面执行时的引入方式称之为动态引入。

Action元素采用XML元素的语法格式，即每个Action元素在JSP页面中都以XML标签的形式出现。JSP规范中定义了一些标准的Action元素，这些元素的标签名都以isp作为前缀，并且全部采用小写，例如，<p:include>、<jsp:forward>等等。

1、.jsp:incluce标签：

<jsp:include page="b.jsp"><jsp:include>

2）.动态引入：并不是像include 指令生成一个Servlet 源文件，而是生成两个Servlet源文件，然后通过一个方法的方式把目标页面包含进来。

org.apache.jasper.runtime.JspRuntimelibrary.include(request,response,\"b.jsp\",out,false);

10、中文乱码问题

1、1）.在JSP页面上输入中文，请求页面后不出现乱码：保证 contentType="text/html；charset=UTF-8"pageEncoding="UTF-8"，charset和pageEcoding的编码一致，且都支持中文，通常建议取值UTF-8。还需要保证浏览器的显示的字符编码也和请求的JSP页面编码一致。

2、获取中文参数值

2）.获取中文参数值：默认参数在传输过程中使用的编码为ISO-8859-1

对于POST请求：只要在获取请求信息之前，调用request.setcharacterEncoding（\"UTF-8\")即可。（在调用request.getParameter 或者是request.getReader等）

对于GET请求：前面的方式对于GET无效。可以通过修改Tomcat的 server.xml文件的方式。

参照http://1ocalhost:8989/docs/config/index.html文档的 useBodyEncodingForURI 属性。

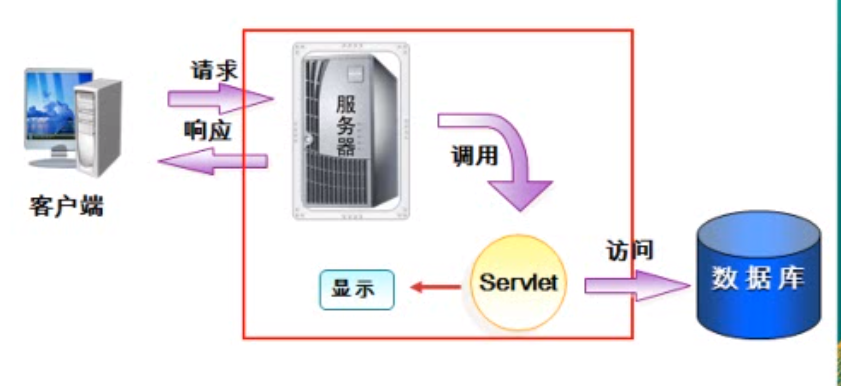
为Connector节点添加 useBodyEncodingForURI="true"属性即可。

# 十、javaweb的MVC设计模式

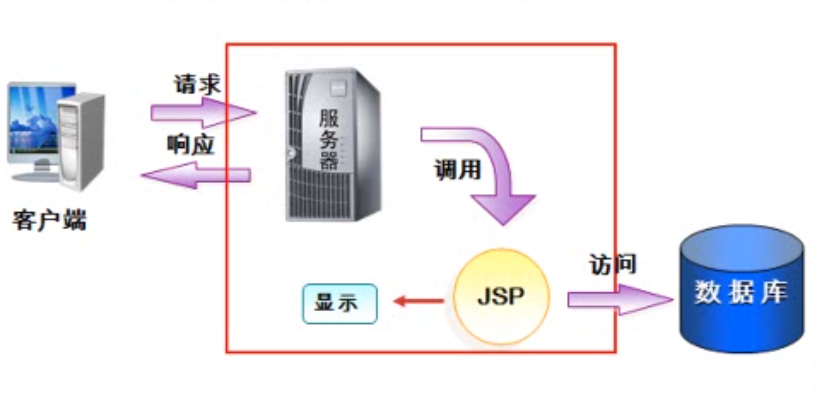
1、JAVAEE的开发思想

2、JAVAEE的开发流程

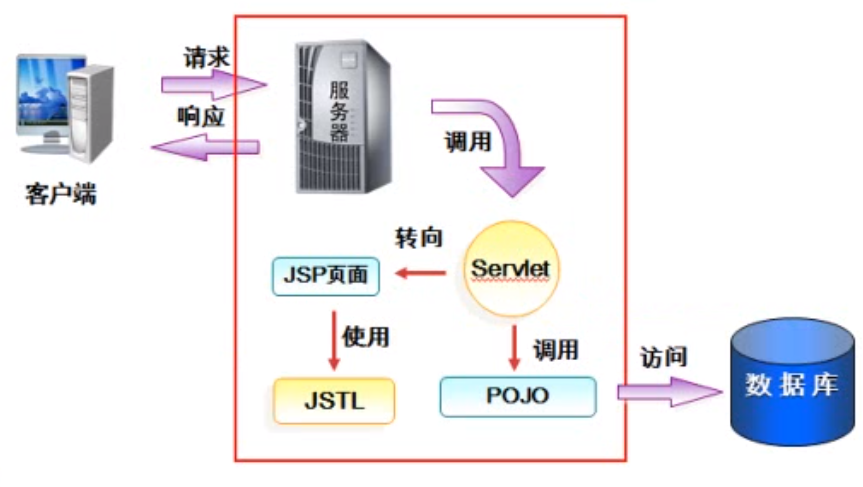
1、sevlet



2、jsp



3mvc设计模式（servlet和jsp）



3、MVC的概念

MVC是Model-View-Controller的简称，即模型-视图-控制器。MVC是一种设计模式，它把应用程序分成三个核心模块：模型、视图、控制器，它们各自处理自己的任务。

1、model（POJO和数据库）

·模型是应用程序的主体部分，模型表示业务数据和业务逻辑。

模型是应用程序的主体部分，模型表示业务数据和业务逻辑。

一个模型能为多个视图提供数据。可以为不同系统提供收数据，如安卓、ios等。

由于应用于模型的代码只需写一次就可以被多个视图重用，所以提高了代码的可重用性。

2、视图

视图是用户看到并与之交互的界面，作用如下：

-视图向用户显示相关的数据。

-接受用户的输入。

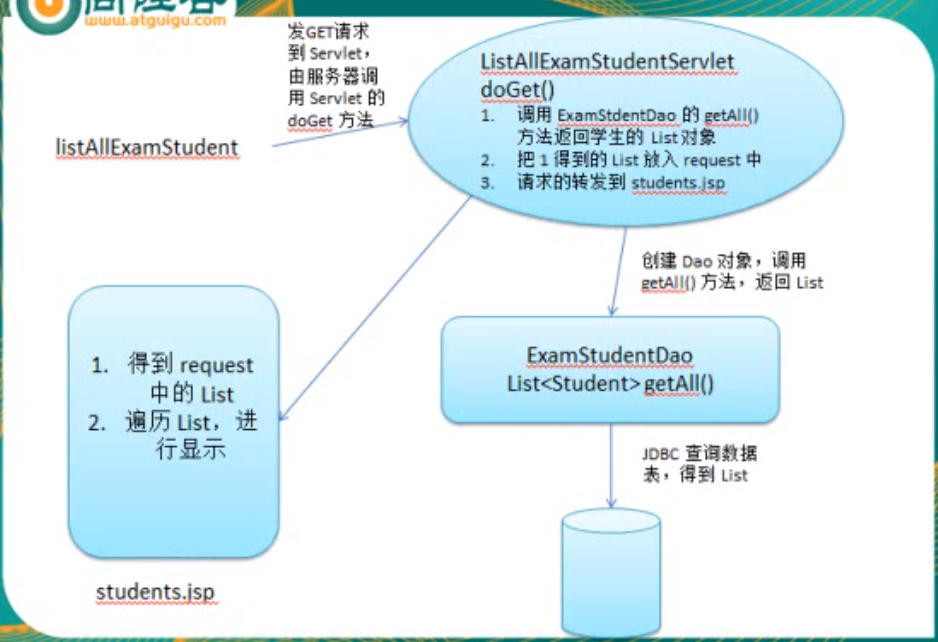
-不进行任何实际的业务处理。

3、控制器

控制器接受用户的输入并调用模型和视图去完成用户的需求。

控制器接收请求并决定调用哪个模型组件去处理请求，然后决定调用哪个视图来显示模型处理返回的数据。

3、mvc查询



请求页面test.jsp

<a href="ListAllStudents">List All Students </a>

Servlet页面：

request.setAttribute("Students", "aa");

request.getRequestDispatcher("/students.jsp").forward(request, response);

转发给student.jsp

<%= request.getAttribute("students") %>

<%

List<String> names= (List)request.getAttribute("students");

for(String name:names){

%>

<%= name %>

<%

}

%>

Student.java