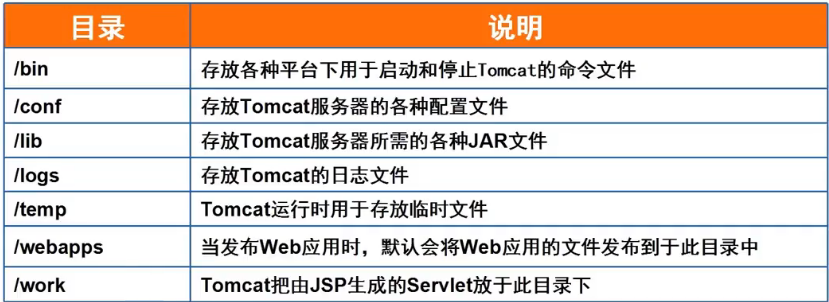
**第1章 JAVA WEB简介**

·Tomcat服务器的目录结构



·WEB-INF目录结构

1. WEB-INF是Java的WEB应用的安全目录。所谓安全就是客户端无法访问，只有服务端可以访问的目录
2. web.xml，项目部署问阿金。
3. classes文件夹，用以放置\*.class文件
4. lib文件夹，用于存放需要的jar包

·默认打开index.jsp

修改默认打开网页为/haha.jsp：

<welcome-file-list>

<welcome-file>/haha.jsp</welcome-file>

</welcome-file-list>

·修改默认端口

修改conf/server.xml文件<Connector port="8080" ...............>

**第2章 JSP基础语法**

·Jsp页面元素构成：

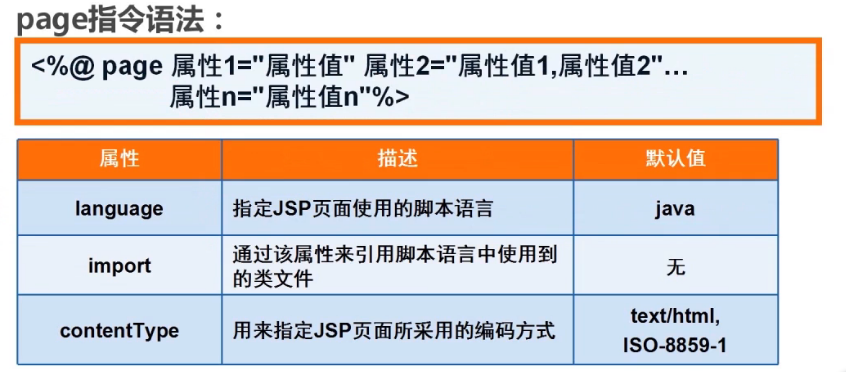


·Jsp指令

page指令：通常位于jsp页面的顶端，同一个页面可以由多个page指令

include指令：将一个外部文件嵌入到当前JSP文件中，同时解析这个页面中的JSP语句

taglib指令：使用标签库定义新的自定义标签，在JSP页面中启用定制行为



<%@ page language="java" import="java.util.\*" contentType="text/html; charset=utf-8"%>

·pageEncoding是jsp文件本身的编码；contentType的charset是指服务器发给客户端时的内容的编码

·Jsp注释

·HTML的注释：

<!--html注释--> //客户端可见

·在JSP页面的注释：

<%--html注释--%> //客户端不可见

·JSP脚本注释（在<% %>中间有效）：

// 单行注释

/\*\*/ 多行注释

·Jsp脚本

在JSP页面中执行的java代码。

语法：

<%Java代码%>

·Jsp声明

在JSP页面中定义变量或者方法。

语法：

<%! JAVA代码 %>

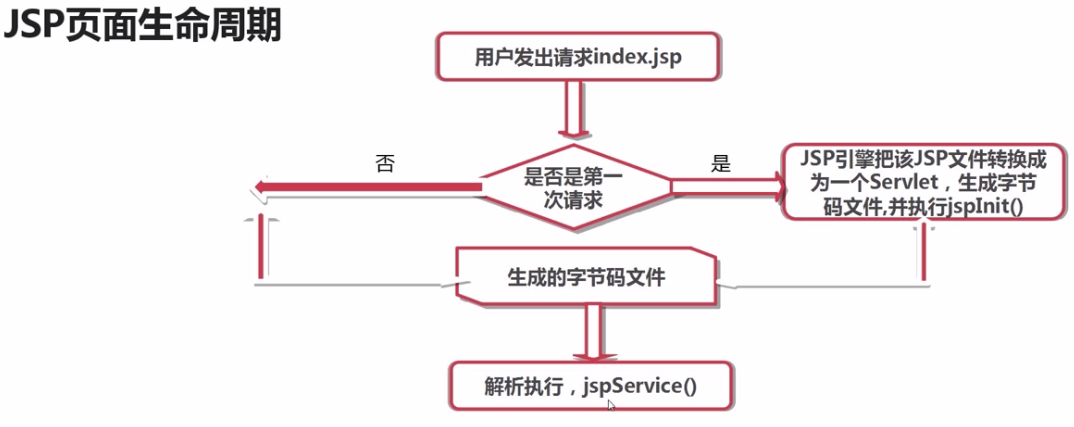
·Jsp表达式

在JSP页面中执行的表达式。

语法：

<%=表达式>//注意：表达式不以分号结束

·Jsp页面生命周期

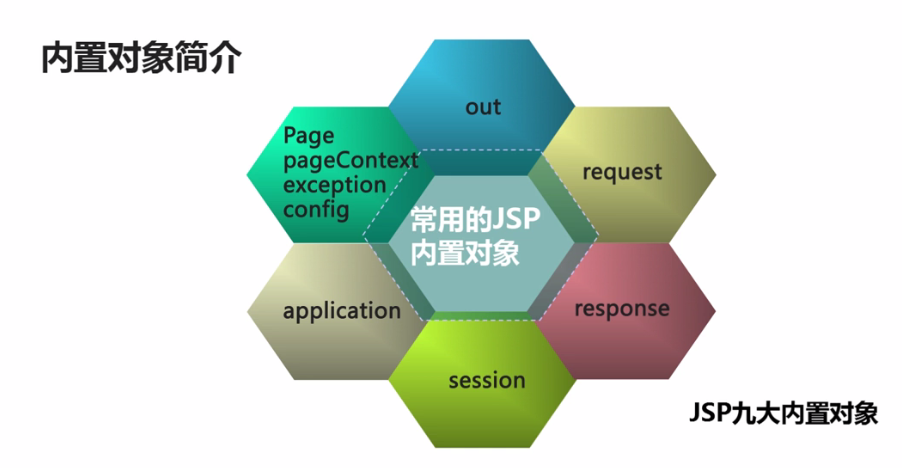


jspService()方法被调用来处理客户端的请求。对每一个请求，Jsp引擎创建一个新的线程来处理该请求。如果有多个客户端同时请求该JSP文件，则JSP引擎会创建多个线程。每个客户端请求对应一个线程。以多线程方式执行可以大大降低对系统的资源需求，提高系统的并发量及响应时间。但也要注意到多线程的编程带来的同步问题，由于该Servlet始终驻于内存，所以响应是非常快的。

**第3、4章 JSP内置对象**

·JSP内置对象

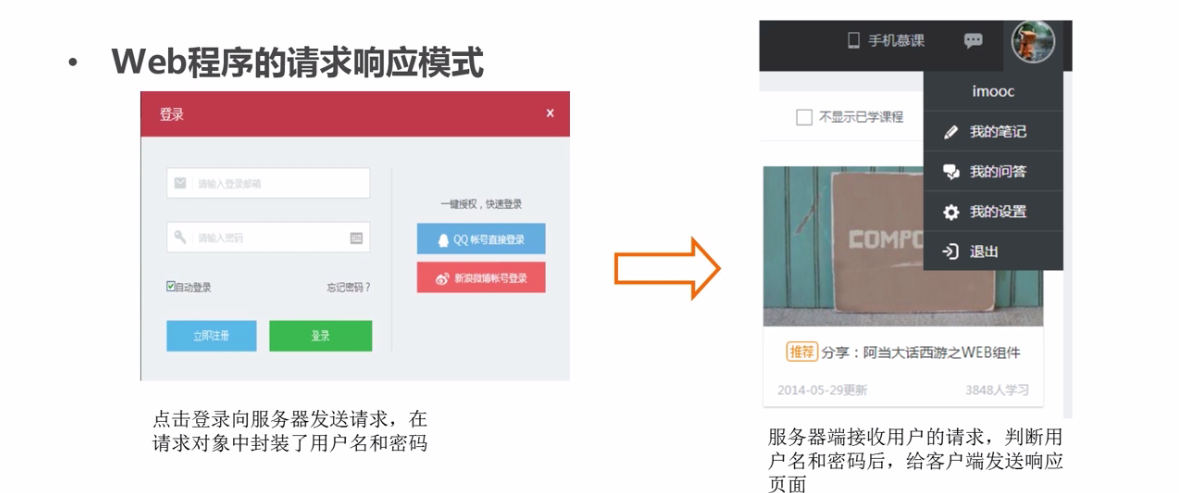
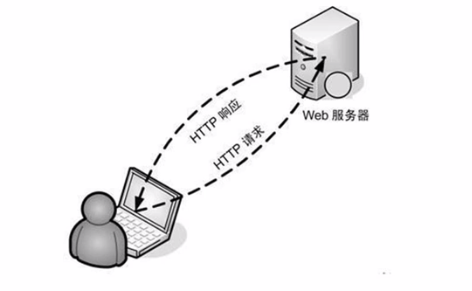
·JSP内置对象是Web容器创建的一组对象，不适用new关键字就可以使用的内置对象



·Web程序的请求响应模式

用户发送请求（request）

服务器给用户响应（response）



·什么是缓冲区

缓冲区：Buffer，所谓缓冲区就是内存的一块区域，用来保存临时数据

**·out对象**

out对象是JspWriter的实例，是向客户端输出内容常用的对象

常用方法如下：

·void println() 向客户端打印字符串

·void clear() 清除缓冲区的内容，如果在flush之后调用会抛出异常

·void clearBuffer() 清除缓冲区的内容，如果在flush之后调用不会抛出异常

·void flush() 将缓冲区内容输出到客户端

·int getBufferSize() 返回缓冲区以字节数的大小，如不设缓冲区则为0

·int getRemaining() 返回缓冲区还剩余多少可用

·boolean isAutoFlush() 返回缓冲区满时，时自动清空还是抛出异常

·void close() 关闭输入流

·get与post区别

<form name="regForm" action="动作" method="提交方式">

</from>

表单有两种提交方式：get与post

·get：以明文的方式通过URL提交数据，数据在URL中可以看到。提交的数据最多不 超过2KB。安全性较低但效率比post方式高。适合提交数据量不大，安全性不高的数 据。比如：搜索、查询等功能。

·post：将用户提交的信息封装在HTML HEADER内。适合提交数据量大，安全性高的 用户信息。比如：注册、修改、上传等功能。

**·request对象**

客户端的请求信息被封装在request对象中，通过它才能了解到客户的需求，然后做出响应。它是HttpServletRequest类的实例。request对象具有请求域，即完成客户端的请求之前，该对象一直有效。常用方法如下：

·String getParameter(String name) 返回name指定参数的参数值

·String[] getParameterValues(String name) 返回包含参数name的所有值的数组

·void setAttribute(String,Object) 存储此请求中的属性。

·object getAttribute(String name) 返回指定属性的属性值

·String getContentType() 得到请求体的MIME类型

·String getProtocol() 返回请求用的协议类型及版本号

·String getServerName() 返回接收请求的服务器主机名

·int getServerPort() 返回服务器接收此请求所用的端口号

·String getCharacterEncoding() 返回字符编码方式

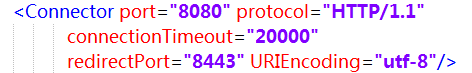
·void setCharacterEncoding() 设置请求的字符编码方式

·int getContentLength() 返回请求体的长度（以字节数）

·String getRemoteAddr() 返回发送此请求的客户端IP地址

·String getRealPath(String path) 返回一虚拟路径的真实路径

·Stirng request.getContextPath() 返回上下文路径



**·response对象**

response对象包含了响应客户请求的有关信息，但在JSP中很少直接用到它。它是HttpServletResponse类的实例。response对象具有页面作用域，即访问一个页面时，该页面内的response对象只能对这次访问有效，其他页面的response对象对当前页面无效。常用方法如下：

·String getCharacterEncoding() 返回响应用的是何种字符编码

·void setContentType(String type)设置相应的MIME类型

·PrintWriter getWriter() 返回可以向客户端输出字符的一个对象（注意比较：PrintWriter与内置out对象的区别：在没有提前调用out.flush的情况下 打印顺序先于out）

·sendRedirect(java.lang.String location) 重新定向客户端的请求

·请求转发与请求重定向

请求重定向：客户端行为，response.sendRedirect()，从本质上讲等同于两次请求。前一次的请求对象不会保存，地址栏的URL地址会改变。

请求转发：服务器行为，request.getRequestDispatcher().forward(req,resp);是一次请求，转发后请求对象会保存，地址栏的URL地址不会变

**·什么是session**

·session表示客户端与服务器的一次对话

·Web中的session指的是用户在浏览某个网站时，从进入网站到浏览器关闭所经过的 这段时间，也就是用户浏览这个网站所花费的时间

·从上述定义可以看到，session实际上是一个特定的时间概念

·在服务器的内存中保存着不同用户的session

·session对象

·session对象是一个JSP内置对象

·session对象在第一个JSP页面被装载时自动创建，完成会话期管理。

·从一个客户打开浏览器并连接到服务器开始，到客户关闭浏览器离开这个服务器结束， 被称为一个对话

·当一个客户访问一个服务器时，可能会在服务器的几个页面之间切换，服务器应当通 过某种办法知道这是一个客户，就需要session对象。

·session对象是HttpSession类的实例

·session对象常用方法如下：

·long getCreationTime()：返回SESSION创建时间

·public String getId()：返回SESSION创建时JSP引擎为它设的唯一ID号

·public Object setAttribute(String nam, Object value)：使用指定名称将对象绑定到此会话

·public Object getAttribute(String name)：返回此会话中的指定名称绑定在一起的对象， 如果没有对象绑定在该名称下，则返回null

·String[] getValueNames()；返回一个包含此SESSION中所有可用属性的数组

·int getMaxInactiveInterval()；返回两次请求间隔多长时间此SESSION被取消（单位秒）

·session的生命周期

·创建：当客户端第一次访问某个jsp或者Servlet的时候，服务器会为当前会话创建一 个SessionId，每次客户端向服务端发送请求时，都会将此SessionId携带过去，服务端 会对此SessionId进行校验。

·活动： ·某次会话当中通过超链接打开的新页面属于同一次会话。

·只要当前会话页面没有全部关闭，重新打开新的浏览器窗口访问同一项目 资源时属于同一次会话。

·除非本次会话的所有页面都关闭后再重新访问某个Jsp或者Servlet将会创 建新的对话

注意事项：注意原有会话还存在，只是这个旧的SessionId仍然存在于服务端， 只不过再也美欧客户端会携带它然后交予服务端校验。

·销毁：Session的销毁只有三种方式：

·调用了session.invalidate()方法

·Session过期（超时）

·服务器重新启动

·超时

·Tomcat默认session超时时间为30分钟。

·设置session超时有两种方式：

·session.setMaxInactiveInterval(时间); //单位是秒

·在web.xml配置

<session-config>

<session-timeout>

10

</session-timeout>

</session-config>//单位是分钟

**·application对象：**

·application对象实现了用户间数据的共享，可存放全局变量

·application开始于服务器的启动，终止于服务器的关闭。

·在用户的前后连接或不同用户之间的连接中，可以对application对象的同一属性进行 操作

·在任何地方对application对象属性的操作，都将影响到其他用户对此的访问

·服务器的启动和关闭决定了application对象的生命。

·application对象是ServletContext类的实例

·常用方法如下：

·public void setAttribute(String name, Object value) 使用指定名称将对象绑定到此会话。

· public Object getAttribute(String name) 返回与此会话中的指定名称绑定在一起的对象， 如果没有对象绑定在该名称下，则返回null。

·Enumeration getAttributeNames() 返回所有可用属性名的枚举

·String getServerInfo() 返回JSP(SREVLET)引擎名及版本号

**·page对象**

page对象就是指向当前JSP页面本身，有点像类中的this指针，它是java.lang.Object 类的实例。

·常用方法如下：

·class getClass() 返回此Object的类

·int hashCode() 返回此Object的hash码

·boolean equals(Object obj) 判断此Object是否与指定的Object对象相等

·void copy(Object obj) 把此Object拷贝到指定的Object对象中

·Object clone() 克隆此Object对象

·String toString() 把此Object对象转换成String类的对象

·void notify() 唤醒一个等待的线程

·void notifyAll() 唤醒所有等待的线程

·void wait(int timeout) 使一个线程处于等待直到timeout结束或被唤醒

·void wait() 使一个线程处于等待直到被唤醒

**·pageContext对象**

·pageContext对象提供了对JSP页面内所有的对象及名字空间的访问

·pageContext对象可以访问到本页所在的session，也可以取本页面所在的application 的某一属性值

·pageContext对象相当于页面中所有功能的集大成者

·pageContext对象的本类名也叫pageContext

·常用方法如下：

·JspWriter getOut() 返回当前客户端响应被使用的JspWriter流(out)

·HttpSession getSession() 返回当前页中的HttpSession对象(session)

·Object getPage() 返回当前页的Object对象(page)

·ServletRequest getRequest() 返回当前页额ServletResponse对象(response)

·void setAttribute(String name, Object attribute) 设置属性及属性值

·Object getAttribute(String name, int scope) 在指定范围内取属性的值

·int getAttributeScope(String name) 返回某属性的作用范围

·void forward(String relativeUrlPath) 使当前页面充导到另一页面

·void include(String relativeUrlPath) 在当前位置包含另一文件

**·Config对象**

config对象是在一个Servlet初始化时，JSP引擎向它传递信息用的，此信息包括Servlet初始化时所要用到的参数（通过属性名和属性值构成）以及服务器的有关信息（通过传递一个ServletContext对象）。

·常用方法如下：

·ServletContext getServletContext() 返回含有服务器相关信息的ServletContext对象

·String getIntParameter(String name) 返回初始化参数的值

·Enumeration getInitParameterNames() 返回Servlet初始化所需所有参数的枚举

**·Exception对象**

exception对象是一个异常对象，当一个页面在运行过程中发生了异常，就产生这个对象。如果一个JSP页面要应用此对象，就必须把isErrorPage设为true，否则无法编译。他实际上是java.lang.Throwable的对象。

·常用方法如下：

·String getMessage() 返回描述异常的消息

·String toString() 返回关于异常的尖端描述信息

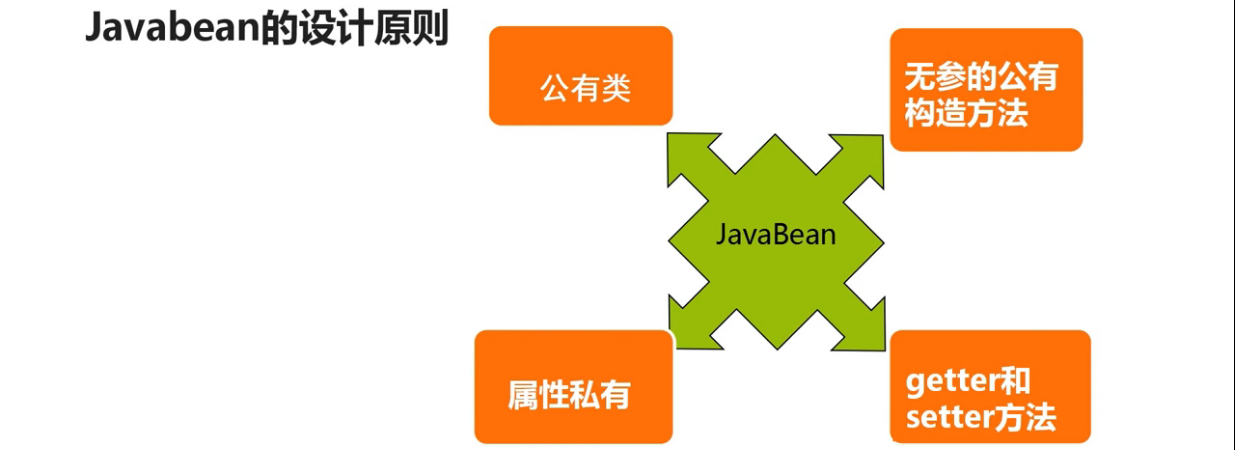
·void printStackTrace() 显示异常及其栈轨迹

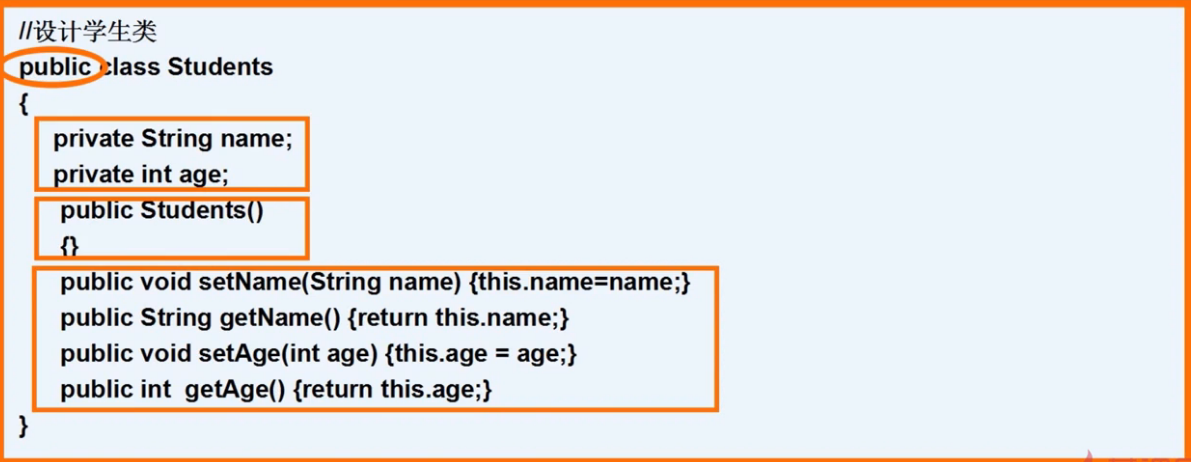
·Throwable FillInStackTrace() 重写异常的执行栈轨迹

1. **JavaBeans**

·Javabean简介

Javabeans就是符合某种特定的规范的Java类。使用Javabeans的好处是解决代码重复编写，减少代码冗余，功能区分明确，提高了代码的维护性。





·什么是Jsp动作

Jsp动作（action elements），动作元素为请求处理阶段提供信息。动作元素遵循XML元素的语法，有一个包含元素名的开始标签，可以由属性、可选的内容、与开始标签匹配的结束标签。

·第一类是与存取JavaBean有关的，包括：

<jsp:useBean><jsp:setProperty><jsp:getProperty>

·第二类是JSP1.2就开始有的基本元素，包括6个动作元素

<jsp:include><jsp:forward><jsp:param><jsp:plugin><jsp:params><jsp:fallback>

·第三类是JSP2.0 新增加的元素，主要与JSP Document有关，包括6个元素

<jsp:root><jsp:declaration><jsp:scriptlet><jsp:expression><jsp:text><jsp:output>

·第四类是JSP2.0新增的动作元素，主要是用来动态生成XML元素标签的值，包括3 个动作

<jsp:attribute><jsp:body><jsp:element>

·第五类是JSP新增的动作元素，主要是用在Tag File中，有2个元素

<jsp:invoke><jsp:dobody>

·在Jsp页面中如何使用Javabeans

·像使用普通java类一样，创建javabean实例。

·在Jsp页面中通常使用jsp动作标签使用javabean

·useBeans动作

·<jsp:useBeans>

作用：在jsp页面中实例化或者在指定范围内使用javabean：

<jsp:useBean id="标示符" class="java类名" scope="作用范围"/>

·setProperty动作

·作用：给已经实例化的Javabean对象的属性赋值，一共有四种形式。

·<jsp:setProperty name="JavaBean实例名" property="\*"/>(跟表单关联)

·<jsp:setProperty name="JavaBean实例名" property="JavaBean属性名"/> （跟表单关联）

·<jsp:setProperty name="JavaBean实例名" property="JavaBean属性名" value="BeanValue"/>（手工设置）

·<jsp:setProperty name="JavaBean实例名" property="propertyName" param="request对象中的参数名"/>（跟request参数关联）

·getProperty动作

·作用：获取指定Javabean对象的属性值

<jsp:getProperty name="JavaBean实例名" property="属性名"/>

·Javabean的四个作用域范围

说明：使用useBeans的scope属性可以用来指定javabean的作用范围。

·page //仅在当前页面有效

·request //可以通过HttpRequest.getAttribute()方法取得JavaBean对象。

·session //可以通过HttpSession.getAttribute()方法取得JavaBean对象

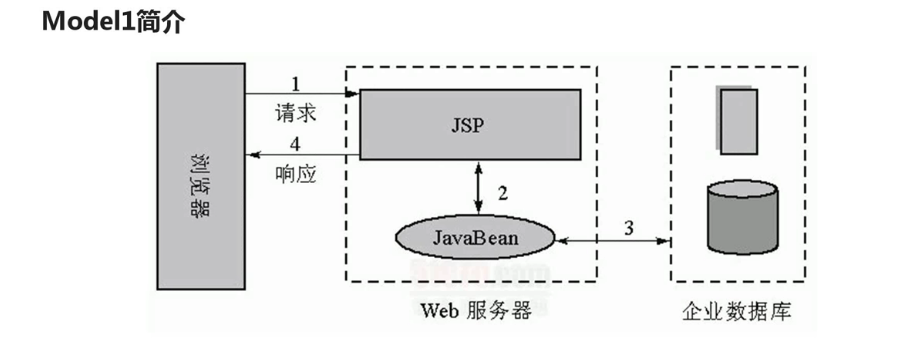
·application //可以通过application.getAttribute()方法取得Javabean对象

·Model1简介

·Model 1模型出现前，整个Web应用的情况：几乎全部由JSP页面组成，JSP页面结合搜处理客户端请求，对请求处理后直接做出响应

·弊端：在界面层充斥着大量的业务逻辑的代码和数据访问层的页码，Web程序的可扩展性和可维护性非常差。

·Javabean的出现可以使jsp页面中使用Javabean封装的数据或者调用Javabean的业务逻辑代码，这样大大提升了程序的可维护性。



1. **JSP状态管理**

·http协议无状态性

无状态是指，当浏览器发送请求给服务器的时候，服务器响应客户端请求。

但是当同一个浏览器再次发送请求给服务器的时候，服务器并不知道它就是刚才那个浏 览器。

简单地说，就是服务器不会去记得你，所以就是无状态协议。

·保存用户状态的两大机制

·Session

·Cookie

·Cookie简介

·什么是Cookie？

Cookie：中文名称为“小甜饼”，是Web服务器保存在客户端的一系列文本信息。

典型应用一：判定注册用户是否已经登录网站。

典型应用二：“购物车”的处理

·Cookie的作用

·对特定对象的追踪

·保存用户网页浏览记录与习惯

·简化登录

安全风险：容易泄露用户信息

·Cookie的创建与使用

·创建

·创建Cookie对象

Cookie newCookie = new Cookie(String key, Object value);

·写入Cookie对象

response.addCookie(newCookie)

·读取Cookie对象

Cookie[] cookies = request.getCookies();

·常用方法

void setMaxAge(int expiry) //设置cookie的有效期，以秒为单位

void setValue(String value) //在cookie创建后，对cookie进行赋值

String getName() //获取cookie的名称

String getValue() //获取cookie的值

int getMaxAge() //获取cookie的有效时间，以秒为单位

java.net.\* //调用前需先调用request.setCharacterEncoding()来将表单信息转码

String URLEncoder.encode(str,charset) //存入cookie时转字符编码

String URLDecoder.decode(str,charset) //从取出cookie中取出时解码

·Session与Cookie的对比



1. **JSP指令与动作元素**

·include指令

·语法：<%@ include file="URL"%>

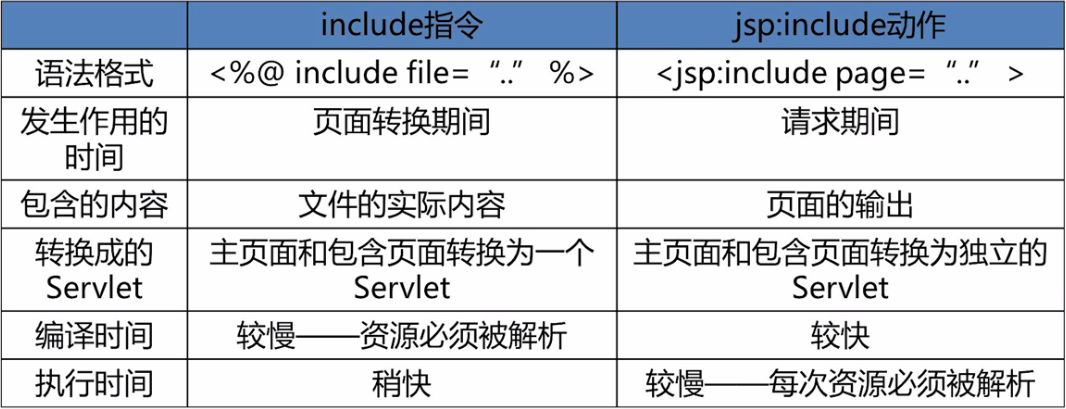
·include动作

·语法：<jsp:include page="URL" flush="true|false"/>

·page：要包含的页面

·flush：被包含的页面是否从缓冲区读取

·include指令与include动作的区别



·<jsp:forward>动作

·语法：<jsp:forward page="URL">

等同于：request.getRequestDispatcher("/url").forward(request,response);

·<jsp:param>动作

·语法：<jsp:param name="参数名" value="参数值">

常常与<jsp:forward>一起使用，作为其子标签

·<jsp:plugin>动作

1. **案例项目**

