生命周期

1、初始化init()

2、service(HttpServletRequest request, HttpServletResponse response)

3、doXXXX()

4、销毁destroy()

**发送响应消息**

**1、发送状态码**

setStatus(int status)方法设置Http响应消息的状态码

sendError(int sc)发送错误信息的状态码

sendError(int code,String message)发送状态码和错误信息

**2、发送响应头**

addHeader(String name,String value) 可以增加同名的响应头字段

setHeader(String name,String value) 覆盖同名的响应头字段

name:指定响应头字段的名称,value:响应头字段对应的消息

addIntHeader(String name,int value)

setIntHeade(String name,int value)

设置包含整数值的响应头，避免将int类型的设置值转换为String类型的麻烦

setContentLength(int len)

设置响应信息的实体内容的大小（单位：字节）设置Content-Length响应头字段的值

setContentType(String type)

设置输出内容的类型,也可以设置字符编码

setLocale(Locale loc)

设置编码（不常用）

setCharacterEncoding(String charset)

设置输出内容所使用的编码

SetCharacterEncoding优先级比setLocale和setCharacterEncoding高

**3、发送实体内容**

getOutputStream()获取**字节输出流**对象为ServletOutputStream类型

输出二进制格式的响应正文，需要调用getOutputStream()方法

**OutputStream out = response.getOutputStream();**

**out.方法();**

**OutputStream对象用于向客户端输出字节流，5个常用方法：**

write(int b)：将指定字节写入输出流

write(byte[] b)：将字节数组中的字节写入输出流

write(byte[] b,int off,int len)：从指定字节数组写入len字节，从偏移量off开始

flush()：刷新输出字节

close()：关闭系统资源

**getWrite()获取字符输出流对象为PrintWriter类型**

**PrintWriter printWriter = response.getWrite();**

**printWriter.方法();**

**PrintWriter对象常用方法:**

print()：输出文本信息不换行

println()：输出文本信息换行

append()：与print类似方法可以追加 out.append(“AB”).append(“A”).append(“B”);

flush()：输出缓冲区数据，向客户端输出后清除缓冲区数据

close()：关闭输出流

**获取请求信息**

**1、获取请求行**

getMethod() 获取HTTP请求消息中的请求方法

getRequestURL() 获取请求行中的资源名称部分

getQuertString() 获取请求行中的参数部分（资源路径？以后的所有内容）

getProtocol() 获取请求行中的协议名和版本

getContextPath() 获取资源名称中Web应用程序的路径

getServletPath() 获取Servlet的名称或Servlet所映射的路径

getRemoteAddr() 获取请求客户端的IP地址

getRemoteHost() 获取请求客户端的完整主机名，无法解析完整主机名会返回IP地址

getRemotePort() 获取客户端端口号

getLocalAddr() 获取服务器端IP地址

getLocalName() 获取Web服务器所在主机名

getLocalPort() 获取Web服务器的端口号

getServerName() 获取Host头字段所对应的主机名

getServerPort() 获取Host头字段所对应的端口号

getScheme() 获取请求的协议名

getRequestURL() 获取客户端发出请求时的完整URL

**2、获取请求头**

getHeader(String name)获取指定头字段的值

getHeaders(String name)获取所有头字段值

getHeaderNames() 获取所有请求头字段的Enumeration对象

getIntHeader(String name) 获取指定头字段，并转为int类型

getDateHeader(String name)获取指定头字段，转换为日期

getContentType() 获取Content-Type头字段内容

getContentLength() 获取ContentLength内容

getCharacterEncoding() 获取CharacterEncoding字符编码集

1. **获取请求参数**

getParameter(String name) 获取指定名称的参数值

getParameterValues(String name) 获取String类型的数组，请求消息可以有多个同名称的参数

getParameterNames() 获取所有参数名的Enumeration对象，可以对请求消息进行处理

getParameterMap() 获取参数名和值，放入Map对象

# Servlet

1. **请求转发**

**发生在服务器内部**

将请求的处理转发给另外一个程序（Servlet/JSP，或其他应用接口）去处理。

需要使用request的getRequestDispatcher(String url)方法

执行该方法可以获得一个用于转发的RequestDispatcher对象。

方法：

request.setAttribute(String name,Object obj)请求中设置属性

request.getAttribute(String name,Object obj)获取request范围内对应名称的属性

request.removeAttribute(String name)删除属性

RequestDispatcher rd = request.getRequestDispatcher(String url)获取转发的地址。

1. forward(ServletRequest request,ServletResponse response)

这个方法将请求从一个servlet或jsp目标资源转发到另一个资源，由该资源来产生响应数据。

例如：FirstServlet请求转发给SecondServlet,SecondServlet来响应数据。

rd.forward(request,response);

1. include(ServletRequest request,ServletResponse response)

此方法用于包含响应中的某一个资源的内容,也就是由目标资源进行响应而不是转发资源。

例如：FirstServlet请求转发给SecondServlet,FirstServlet来响应数据。

1. **重定向**

sendRedirect(URL)方法

1. 外部地址需要完整的URL，例如response.sendRedirect(“https://www.baidu.com”);
2. 内部地址可以是相对地址或绝对地址，加反斜杠\表示采用绝对路径，

可以用request.getContextPath() 获取。获取资源名称中Web应用程序的路径

## 处理session

HttpSession session = request.getSession();

创建HttpSession对象，getSession(true)返回HttpSession对象，不存在则生成新的对象。

getSession(false)返回HttpSession对象，不存在则返回Null。

可以使用HttpSession的isNew()方法判断对象是否为新建立的。

Invalidate()会销毁session中包含的属性，使之前的HttpSession直接失效，此时服务器会立即生成新的session并分配给客户端。

**Servlet处理Cookie**

1. Servlet向客户端写入Cookie

创建Cookie对象

Cookie cookie = new Cookie(String keyname,String value);

Cookie.setMax.Age(int second) 设置Cookie存活时间。

Cookie.setPath(String path) 设置Cookie的有效路径，未设置则在当前项目的根目录下的所有路径有效。

Cookie.setDomain(String pattern) 设置能够读取Cookie的域名，默认本机。

response.addCookie(Cookie c) 将Cookie写入客户端。

1. Servlet从客户端读取Cookie

request.getCookie() 返回Cookie类型的数组。

使用循环读取数据:

Cookie.getName()获取Cookie的键名。

Cookie.getValue()获取Cookie的键值。

1. Servlet从客户端删除Cookie

生成与原来名称一样的Cookie对象，覆盖原Cookie。

**ServletContext对象**

上下文环境对象，对应整个Web应用，一个Web应用只能有一个ServletContext对象。

1. ServletContext对象获取(获取当前对象)
	1. this.getServletConfig.getServletContext()
	2. this.getServletContext()
	3. Request.getServletContext()

**JSP**

**特征：跨平台、业务代码相分离、组件重用、预编译**

服务器端程序、动态生成页面、处理复杂业务。

1. 变量声明与表达式

<%! 变量声明;%>

感叹号!表示该变量的声明范围为整个网页，去掉感叹号，只能在变量声明后使用。

<%= 变量/表达式%>

将声明的变量以字符串的形式输出到客户端

表达式不能以分号结尾。

1. 程序段

<% 代码 %>

1. JSP注释

<!-- HTML风格的注释会发送给客户端，可插入JSP表达式，<%= a%> -->

<%-- JSP风格的注释，不会发送到客户端 --%>

JSP中代码的注释不会发送到客户端

1. JSP指令
2. page指令
	1. 导入包

<%@ page import = “包名.类名”,”包名.类名”%>

* 1. 设定字符集

<%@ page pageEncoding=”UTF-8”%>

* 1. 指定MIME类型和字符编码

<%@ page ContentType=”text/html;charset= UTF-8” %>

1. 打印错误信息

<%@ page language="java" contentType="text/html; charset=UTF-8"

 pageEncoding="UTF-8" isErrorPage="true"%>

1. 是否内置session对象

<%@ page language="java" contentType="text/html; charset=UTF-8"

 pageEncoding="UTF-8" isErrorPage="true" session=”true”%>

1. include指令

引入外部文件

<%@ include file=”文件名” %>

外部引入的文件定义变量名不能与主页面变量名一致。

1. JSP动作

语法：

<jsp: 动作名 属性名1 = “属性值1” 属性名2 = “属性值2” />

或

<jsp: 动作名 属性名1 = “属性值1” 属性名2 = “属性值2” >

...

</jsp>

|  |  |
| --- | --- |
| 语法 | 描述 |
| jsp:include | 类似于include指令，用于引入文件。 |
| jsp:forward | 请求转发到另外的页面 |
| jsp:useBean | 寻找一个JavaBean |
| jsp:setProperty  | 设置JavaBean属性 |
| jsp:getProperty | 获取JavaBean属性 |
| jsp:plugin  | 根据浏览器类型为Java插件生成OBJECT或EMBED标记 |

<jsp:include page=”文件名” flush=”true/false”/>

先编译文件再引入，不会因为变量名一致而报错。实时更新。

flush属性 是否清除缓冲区数据

<jsp:forward page=”文件名”></jsp:forward>

执行之后，当前页面不再执行

**JSP内置对象**

1、out对象:等同于response.getWriter()

2、request对象:

3、response对象:

4、session对象:等同于request.getSession()获取的HttpSession对象

5、application对象:等同于request.getServletContext()获取的ServletContext对象

6、exception对象:异常

7、page对象:JSP实例化对象，this代替

8、pageContext对象:JSP上下文,可以获取其他8个对象

9、config对象：初始化参数、配置

**JavaBean**

使用标签导入

<jsp:useBean id=”JavaBean实例名” class=”JavaBean对应类” scope=”作用范围,默认page”></jsp:useBean>

设置属性

<jsp:setProperty name=”JavaBean实例名” property=”实例的属性名” value=”设置的属性”/>

<jsp:setProperty name=”JavaBean实例名” property=”实例的属性名” param=”前端可访问的值”/>

获取属性

<jsp:getProperty name=”JavaBean实例名” property=”实例的属性名”/>

**JDBC**

1、加载注册驱动

Class.forName(“drivername”); com.mysql.jdbc.Driver

2、链接数据库

Connection conn = DriverManager.getConnection(String url,String user,String pwd);

url:链接数据库的url地址

本地Url=“jdbc:mysql://localhost:3306/web\_test?useUnicode=true&characterEncoding=UTF-8”;

3、获取Statement对象

调用createStatement()方法，获取createStatement对象。

调用prepareStatement()方法，获取prepareStatement对象。



调用prepareCall()方法，获取prepareCall对象。

Statement stmt = conn.createStatement();

1. 执行SQL语句

Statement对象的执行SQL语句的方法:

* execute():执行任何SQL语句
* executeQuery():通常执行查询语句，执行后返回代表结果集的ResultSet对象

//ResultSet rs = stmt.executeQuery(SQL);

* executeUpdate():主要执行DML(INSERT UPDATE DELETE)和DDL语句，返回受SQL影响的行数，DDL语句返回0;

5、操作ResultSet结果集

1. 纵向数据读取

getInt(索引或字段名)

getString()

getDate()

1. 横向数据读取

next()

previous()

 last()

afterlast();

absolute();

6、关闭连接，释放资源（注意顺序）

rs.close();

stmt.close();

conn.close();

7、批处理







8、事务



**EL和JSTL**

降低复杂度，提高代码重用性

一、EL(表达式语言)

语法格式：

${表达式} 在页面输出时使用转义字符\ 或 ${${} 输出 “${“

### **1、EL中的标识符**

为了避免出现非法的标识符，在定义标识符时还需要遵循以下规范。
• 不能以数字开头。
• 不能是EL中的保留字，如and、or、gt。
• 不能是EL隐式对象，如pageContext。
• 不能包含单引号（’）、双引号（"）、减号（-）和正斜杠（/）等特殊字符。

**2、方括号运算符**

(1)属性名称中包含特殊字符(数字，横线，下划线等) ${requestScope.goods[“goods\_count”]

(2)属性名称为动态取值。String attribute; attribute可能有多个取值; ${requestScope.goods[attribute]}

(3)获取数组中的元素。${requestScope.goodsArray[0]};

${empty A}判断是否为空 A为null；A不存在；A为空字符串；A为空数组；返回true。

**3、隐式对象**

1、pageContext读取页面上下文对象

2、pageScope 读取page范围内属性的值

3、requestScope 读取request范围内属性的值

4、sessionScope 读取session范围内属性的值

5、applicationScope读取applicationScope范围内属性的值

6、param 读取单个参数

7、paramValues 读取数组类型的参数

8、header 获取HTTP请求某个header的值

9、headerValues 获取HTTP请求多个header的值

10、cookie 读取cookie的值

11、initParam 读取web.xml中定义的参数

**获取隐式对象**

**${pageContext.response.characterEncoding}**

**pageContext对象用于获取其他的隐式对象。**

1. JSTL(标准标签库)由5个功能不同的标签库组成
2. 核心标签库
3. 国际化/格式化标签库
4. SQL标签库
5. XML标签库
6. 函数标签库
7. **JSTL的配置**

Tomcat的安装目录的\webapps\examples\WEB-INF\lib文件夹下的taglibs-standard-impl-1.2.5.jar以及taglibs-standard-spec1.2.5.jar两个文件，复制到WEB-APP\WEB-INF\lib下

需要JSP中使用<%@ taglib %>指令

<%@ taglib prefix=”自定义前缀名称” uri=”对应的标签库”%>

**JSTL标签库URL以及前缀**

|  |  |  |  |
| --- | --- | --- | --- |
| JSTL  | 推荐前缀 |  URI  | 示例 |
| 核心标签库 | c | http://java.sun.com/jsp/jstl/core | <c:out> |
| 函数标签库 | fn | http://java.sun.com/jsp/jstl/functions | <fn:length> |
| 118N标签库 | fmt | http://java.sun.com/jsp/jstl/fmt | <fmt:fomatDate> |
| SQL标签库 | sql | http://java.sun.com/jsp/jstl/sql | <sql:query> |
| XML标签库 | X/xml | http://java.sun.com/jsp/jstl/xml | <x:set> |

1. **表达式标签**
2. <c:out value=”输出的内容” default=”默认值” escapeXml=”true/false”/>

default属性：（可选属性）没有内容输出时，value值为空时，default属性值作为默认值输出。

escapeXml属性：（可选属性）决定对输出变量中的特殊字符是否转义，

为true时，表示忽略并输出，为false时，对特殊字符进行转义输出。

1. <c:set value=”变量值” var=”变量名” scope=”page/request/session/application/>

对JavaBean进行设置时

<c:set value=”属性值” target=”对象名” property=”属性名” />

1. <c:remove var=”变量名” scope=”作用范围” />

不指定范围时，相同的两个同名变量都被移除，因此在移除变量时要添加作用范围。

1. **流程控制标签**

(1)<c:if test=”${条件表达式}” scope=”作用范围”/>

Body content

</c:if>

(2)<c:choose>

<c:when test=”${条件1}”>...</c:when>

<c:when test=”${条件2}”>...</c:when>

...

<c:when test=”${条件n}”>...</c:when>

<c:otherwise>以上条件都不满足时执行</c:otherwise>

</c:choose>

1. **循环标签**

(1)<c:forEach var=”元素名” items=”集合名” begin=”起始” end=”结束” step=”步长” varStatus=”循环状态名称”>

body content

</c:forEach>

var属性用于指定将当前迭代的元素保存到page域中的名称

items属性用于指定将要迭代的集合对象

varStaus属性用于指定当前迭代状态信息对象保存到page域中的名称

bengin属性用于指定从集合中第几个元素开始进行迭代，begin的索引值从0开始。如果没有指定items属性，就从begin指定的值开始迭代，直到迭代结束为止。

step属性用于指定迭代的步长，即迭代因子的增量。



(2)<c:forTokens>标签

<c:forTokens items=”字符串” delims=”分隔符” var=”字串名” begin=”起始” end=”结束” step=”步长”>

</c:forTokens>

