JSP入门基础

软件开发环境这门课程的复习资料

Web开发技术概述

URL的组成部分

协议、主机DNS名或IP地址和文件名

Tomcat服务器

Tomcat服务器的默认端口号是**8080**

概念

软件开发环境是围绕着软件开发的一定目标而组织在一起的一组**相关软件工具**的有机集合

JSP和HTML的区别

- HTML页面是静态页面，也就是事先由用户写好放在服务器上，固定内容，不会变，由web服务器向客户端发送，平时上网看的网页都是大部分都是基于html语言的。
- JSP页面是有**JSP容器**执行该页面的**Java代码**部分然后**实时**生成动态页面，可动态更新页面上的内容。

HTML部分

当前版本的HTML**不区分大小写**

标题标记

```
<title>标题</title>
```

标记之间的内容将显示到浏览器的标题上

<.meta>辅助性标记

下面的语句的功能是设置HTML文档的编码格式

```
<meta charset="utf-8">
```

超链接标记

```
<a href="URL">链接文本或图像</a>
```

在新窗口打开超链接

```
<a href="url" target="_blank">
```

<.input>输入标记

正确产生复选框：

```
<input type="checkbox">
```

把表单数据发送到服务器

```
<input type="submit">
```

常用属性

name:控制标识

<.img>图像标记

src:必选项，指定图像文件的url。

标题标记

下面语句的功能是以二级标题显示“图书信息列表”

```
<h2>图书馆信息列表</h2>
```

表格标记

下面语句的功能是创建表格，边框等于一个像素

```
<table border="1">
```

1.<.table>标记用于建立表格

2.<.table>与<./table>之间为表格标题、表头及单元格的内容

border属性:表格边框宽度，以像素为单位

3.<.caption>与<./caption>:定义表格标题

4.<.tr>与<./tr>:定义表格中的一行

5.<.td>与<./td>:定义单元格内容

表单标记

格式

```
<form method="post/get" action="URL" enctype="application/x-www-form-
urlencoded"></form>
```

其常用属性如下。

action:完成表单信息处理任务程序的完整URL

method:表单中输入数据的传输方法，默认值为get

enctype:指定表单中输入数据的编码方法

列表框

1.<.select>标记

定义下拉式列表框和滚动式列表框，其常用属性如下。

name:列表框名字

size:列表框大小

multiple:允许用户进行多项选择

2.<.option>标记

<.select>标记所定义的列表框中的各个选项，其常用属性如下

selectd:表示该项预先选定

value:指定控件的初始值

页面修饰标签

```
<p>
```

分段标记：表示新一段开始，段落间有一空行。

```
<br>
```

换行标记：另起一行，中间不插入空行。

```
<hr>
```

水平线标记：在页面上画出一条水平线

```
<pre>....</pre>
```

预格式标记：使HTML文档中的空格、Tab符、回车、换行符起作用

```
<font>...</font>
```

字体标签：标记中默认的中文字体是宋体

```
<B>字体</B>
```

粗体

```
<I>字体</I>
```

斜体

考试题程序

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <h2>
            图书信息列表
        </h2>
        <table border="1">
            <tr>
                <td>书名</td>
                <td>出版时间</td>
                <td>作者</td>
            </tr>
            <tr>
                <td>软件开发环境</td>
                <td>2022-12-08</td>
                <td>小王</td>
            </tr>
            <tr>
                <td>Java程序设计</td>
                <td>2021-08-06</td>
                <td>小李</td>
            </tr>
        </table>
    </body>
</html>
```

动态网页和静态网页的区别

静态网页服务器端返回的HTMl文件是事先存储好的，且静态网页文件里只有HTML标记，没有程序代码

动态网页服务器端返回的HTML文件是程序生成的

CSS部分

定义样式

```
body{color:black}
```

1）内联式样式表

2）嵌入式样式表

3）外部样式表

在HTML文档的文件头中加入语句，可建立与外部样式文件的链接。即放置在。

Internet主要技术

DNS域名解析系统

DNS（域名解析系统）在因特网上作为域名和IP地址相互映射的一个分布式数据库。

应用层协议

在字符串"http://www.sample.com:8080/login.html"中使用的访问协议是http。

Web工作原理描述

1.**HTTP协议**用于在服务器和浏览器之间传送网页数据。

2.**浏览器**负责网页数据的请求和显示。

3.**服务器负责**浏览器的请求

软件开发模式

浏览器/服务器(B/S)模式

优点:

1.用户可以在任何地点使用系统，方便、快捷、高效

2.用户可以跨平台以相同的浏览器界面访问系统

3.客户端只需要安装浏览器，升级维护简单

缺点:

1.应用服务器运行数据负荷高

2.出现服务器崩溃问题则后果严重

JSP简介

JSP运行必须需要的：操作系统，JavaJDK，支持JSP的Web服务器

JSP运行原理

当服务器上的一个JSP页面被第一次请求执行时，服务器上的JSP引擎首先将JSP页面文件转译成一个Java文件，并编译这个Java文件生成字节码文件（class文件），然后字节码文件响应客户的请求。

多个用户请求一个JSP页面

当多个用户请求一个JSP页面时，Tomcat服务器为每个客户启动一个**线程**，该线程负责执行常驻内存的字节码文件来响应相应客户的请求。

概念

安装Tomcat服务器，首先要安装JDK，并需要设置JAVA_HOME环境变量

JSP基本语法

JSP注释

1.HTML注释格式：

```
<!-- 注释内容 -->
```

2.JSP注释格式：

```
<% --注释内容-- %>
```

JSP注释写在JSP程序中，但不发送给客户，即为隐藏型注释

3.Scriptlets的注释

由于Scriptlets包含的是Java代码，所以Java中的注释规则在Scriptlets中也适用，常用Java注释使用//表示**单行注释**，使用**//表示多行注释**

字节码文件

字节码文件的任务如下：

1.JSP页面中普通的HTML标记符号，交给客户的浏览器执行显示

2.JSP标记、变量和方法声明、Java程序片段由Tomcat服务器负责执行，将需要显示的结果发送给客户的浏览器

3.Java表达式由Tomcat服务器负责计算，将结果转换为字符串，交给客户的浏览器。

执行字节码文件的结果是发生一个**HTML页面**到客户端

JSP的编译指令标记

JSP的编译指令标记通常是指page指令、include指令和taglib指令

声明

```
<%!预定义%>
```

“<%!”和“%>”之间声明的变量在整个JSP界面内都有效，与“<%!” “%>”标记在JSP页面中所在的书写位置无关。

一次可声明多个变量和方法，只要以“;”结尾就行

一个声明仅在一个页面有效

在预定义中声明的变量将在JSP页面初始化时初始化

page指令标记

<%@page%>作用于整个JSP页面

可以在一个页面中使用多个<%@page%>指令

page标记

page指令与书写的位置无关，习惯上把page指令写在JSP页面的最前面

可以用一个page指令指定多个属性值，也可以使用多个page指令分别为每一个属性指定值

Java程序片段

在“<%”和“%>”之间插入Java程序片段

Java表达式

在“<%=”和“%>”之间插入一个表达式，这个表示必须能求值。表达式的值由服务器负责计算，并将计算结果用字符串形式发送到客户端显示。

其中需要注意“<%=”是一个完整的符号，“<%”和“=”之间不要有空格

变量和方法对的声明

在“<%!”和“%>”标记符之间声明变量，变量的类型可以是Java语言允许的任何数据结构类型，这些变量称为JSP页面的成员变量。

“<%!”和“%>”之间声明的变量在整个JSP页面内部都有效，与“<%!” “%>”标记符在JSP页面所在的书写位置无关。

当多个用户同时请求一个JSP页面时，JSP引擎为每个用户启动一个**线程**，这些线程由JSP引擎来管理。这些线程共享JSP页面的成员变量，因此任何一个用户对JSP页面成员变量操作的结果，都会影响到其他用户。

JSP的基本组成

JSP以HTML为基础，将处理动态页面的部分嵌入到HTML标记中间，嵌入的部分以“<%”开始，以“%>”结束。

JSP页面元素

普通HTML标记（客户端浏览器执行）、JSP标记、成员变量和方法声明、Java脚本程序、JSP表达式、Java表达式（JSP引擎处理并将）

JSP中可以包含注释，它们是为了对程序进行必要的说明。

synchronized关键字

synchronized关键字保证一次**只有一个线程**执行

指令标记

page指令标记

include指令标记

include指令的作用是在JSP页面出现该指令的位置处静态插入一个文件，实现代码的复用，必须保证插入后形成的文件是**一个完整的JSP文件**

动作标记

1.include动作标记

2.param动作标记

param标记不能独立使用，需作为jsp:include、jsp:forward、jsp:plugin标记的子标记来使用，并为它们提供参数

3.forward动作标记

4.useBean动作标记

5.setProperty动作标记

6.getProperty动作标记

JSP内置对象

内置对象作用域从小到大是：request、session、application

JSP内置对象是指不用声明就可以在JSP页面的脚本部分使用对象。

response

在某些情况下，当响应用户时，需要将用户重新引导至另一个页面，可以使用response的sendRedirect()的方法实现重定向。

HTTP头

在HTTP响应消息中，第一行为响应状态行，紧接着的是若干响应消息头，服务器端通过响应消息头向客户端传递附件信息

状态行代码为**404**表示用户资源不可用

一个典型的HTTP请求消息包括请求行、多个请求头和信息体

方法

1）取得请求参数的方法**getParameter()**

2）取得请求HTTP头的方法**getCookies()**和**getHeader()**

3）存储和取得属性的方法**getAttribute()**和**setAttribute()**

4）用于获得主机名、主机端口号等的其他方法

request

request对象可以得到请求中的参数

通过getParameter()方法获取radio(单选按钮)

通过getParameterValues()获取checkbox(复选框)的值

处理中文乱码

```
request.setCharacterEncodingg("UTF-8")
```

如果getParameter()的参数不存在将返回什么

用request.getParameter(“parameter_name“) 来取得参数时，如果不存在 parameter_name 这个参数，request.getParameter ()将返回 null

session

session对象驻留在服务器端，该对象调用某些方法保存用户在访问某个Web服务目录期间的有关数据

session对象可以保存用户信息

session对象提供了访问和放置页面中共享数据的方式

application

服务器启动后，新建一个对应Web服务目录的application对象，一直保持到服务器关闭。所以application的生命对象比session的生命周期长。

application对象可以被多个客户端共享(注意，不是应用，应用是服务器端)

out

JSP和JavaBean

使用getProperty或者setProperty动作标记之前，必须使用useBean动作标记获得相应的Bean

Bean是通过JSP动作标记——useBean加载成功，格式如下：

```
<jsp:useBean>
```

JavaBean的类必须是具体的、公开的，并且具有无参数构造器

JavaBean的属性要通过公共方法进行访问（即类中声明的方法的访问属性都必须是public的）

JavaBean属性和表单控件名称能很好地耦合，得到表单提交的数据

编写JavaBean

类中必须提供获取和修改方法用来获取或修改成员变量XXX的属性。

getXxx()用来获取属性xxx

setXxx()用来修改属性xxx，属性名首字母必须大写

Bean的生命周期

使用JSP动作标记useBean来加载使用Bean。useBean中的scope给出了Bean的生命周期，即scope取值决定了JSP引擎分配给用户的存活时间。

scope可以有四种取值，分别是page、request、session、application

获取和修改Bean的属性

getProperty动作标记

使用getProperty动作标记可以获得Bean的属性值，并将这个值用串的形式发送给用户的浏览器。使用getProperty动作标记之前，必须使用useBean动作标记获得相应的Bean。

getProperty动作标记的格式如下：

```
<jsp:getProperty name="Bean的名字"propertu="Bean的属性"/>
```

setProperty动作标记

通过HTTP表单的参数的值来设置Bean的响应属性的值

property=“*”

```
<jsp:setProperty="Bean的名字"property="*"/>
```

Servlet

在JSP应用开发中，Servlet程序可以在web.xml文件中配置

用于保持会话的技术：

1.Cookie

2.HTTPsession

3.HTML隐藏表单域

Servlet获得初始化参数的对象是：Servlet Config

基本功能

1.获取客户端HTML的FORM表单提交的数据和URL后面的参数信息

2.创建和客户端响应消息内容

3.访问服务器端的文件系统

4.连接数据库并开发基于数据库的应用

5.调用其他Java类

工作原理

Servlet存在于Web服务器中，并且由Servlet引擎负责分配和管理，在Servlet的声明周期中，由init、service和destroy三个方法构成。整个生命周期中Servlet只被初始化一次，destroy一次，但service()方法可能被调用多次

Servlet引擎访问Servlet

第一次访问的时候，Servlet引擎首先创建Servlet实例，然后调用Servlet的init()函数初始化Servlet,再调用Servlet的service()函数响应用户请求。

之后的访问Servlet的时候Servlet引擎会自动检测Servlet是否已经存在，如果存在就直接调用service()函数响应用户请求，不用再实例化和初始化Servlet。

翻译

Tomcat在将JSP页面翻译成Servlet程序时，会忽略JSP注释的内容，不会将JSP注释发送到客户端

主要作用

Servlet文件在Java Web开发中的主要作用是作为**控制器**

创建Servlet对象的类

当Tomcat初始化一个Servlet时，会将该Servlet的配置信息封装到一个ServletConfig对象

共享变量

当多个客户请求一个Servlet时，服务器为每个客户启动一个**线程**。

Web.xml文件规则

匹配规则：同一个Servlet指定多个不同的URL，四种不同的URL形式匹配如下：

优先级	URL形式	关联类型
1	/开始/字符串结束	精确匹配关系
2	/开始/**结束	按纯目录形式关联
3	*.jsp形式	按具体文件类型关联
4	单个/	缺省Servlet

运行Servlet

在运行Servlet文件的过程可能会遇到中文乱码的问题，需要在运行Servlet文件之前在指定的Servlet类文件中设置编码格式。

针对response请求的中文乱码或者是浏览器显示内容的乱码可以按照下面的设置方式设置编码格式和请求头文件的编码格式。

```
response.setHeader("content-type","text/html;charset=UTF-8");
response.setCharacterEncoding("UTF-8");
```

请求转发

通过请求转发来实现目标资源的访问是服务器内部的行为，对于客户端来说是一次请求的过程

请求转发和重定向对的方式

实现请求转发的方式如下：

1.得到RequestDispatcher对象：

```
RequestDispatcher dispatcher=request.getRequestDispatcher("a.jsp");
```

2.转发

```
dispatcher.forward(request,response);
```

实现重定向的方式如下：

```
response.sendRedirect("a.jsp");
```

RequestDispatcher.forwad()方法只能在同一个Web应用程序内的资源之间转发请求。

sendRedirect()方法还可以重定向到同一个站点上的其他应用程序中的资源，甚至是使用绝对的URL重定向到其他站点的资源。

请求转发和重定向的不同

1.资源使用范围不同

请求转发：同一服务器中的资源

重定向：任意服务器资源

2.请求响应的次数不一样

请求转发：一次请求

重定向：两次请求

3.地址栏

请求转发：URL地址栏不变

重定向：URL地址栏变化

4.请求响应对象

请求转发：两个Servlet实例共用请求响应对象

重定向：需要创建两次请求与响应对象

MVC模式

模型 Model：用于存储数据的对象

视图 View：向控制器提交所需数据以及显示模型中的数据

控制器 Controller：进行具体的业务逻辑操作

MVC模式工作原理

1.用户发出请求

2.View视图接受用户请求

3.Controller控制器处理用户请求

4.Model模型存储处理好的数据

5.View视图显示模型的数据

6.用户获得响应

基于JSP的MVC模式

1）基于JSP的Web开发中，JSP、Servlet、JavaBean就是通过MVC有机结合到了一起

2）JSP作为视图(View)，负责提供页面为用户展示数据

3）Servlet作为控制器(Controller)，用来接受用户提交的请求，进行数据处理

4）JavaBean作为模型(Model)，用来存储用户提交的数据以及数据处理的结果

在JSP中使用数据库

JSP使用JDBC提供的API和数据库进行交换信息。使用JDBC的应用程序一旦和数据库建立连接，就可以使用JDBC提供的API操作数据库

当查询ResultSet对象中的数据时，**不可以**关闭和数据库的连接。

使用PreparedStatement对象可以提高操作数据库的效率

数据库类型

1.关系型数据库

2.树型数据库

3.网状数据库

4.对象数据库

在JDBC语句访问数据库，正确导入SQL类库的语句是:

```
1 <%@page import="java.sql.*">
```

在对SQL进行预处理时可以使用通配符"？"来代替字段值

数据库管理系统

DBMS是（Data Base Management System）的缩写，是管理数据库软件的集合。

DBMS包含面向用户接口功能和面向系统维护功能。

1.面向用户接口功能是提供用户访问数据库的一些必要手段

2.面向系统维护功能是为数据库管理者提供数据库的维护工具

加载JDBC-数据库驱动程序

用Java语言编写的数据库驱动程序称为JDBC数据库驱动程序。

使用JDBC-数据库驱动方式和数据库建立连接需要经过两个步骤：

1.加载JDBC-数据库驱动程序

2.指定的数据库建立连接

驱动数据库

使用纯Java数据库驱动加载MySQL驱动程序代码如下：

```
Class.forName("com.mysql.jdbc.Driver");
```

建立起一个JDBC-ODBC桥接器“

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver")
```

结果集与查询

让连接对象con调用方法createStatement()创建执行SQL语句的Statement对象：

```
Statement sql=con.createStatement()
```

sql对象就可以调用相应的方法，实现对数据库中表的查询和修改，并将查询结果存放在一个ResultSet类声明的对象中：

```
Result rs=sql.execute("SELECT*FROM product");
```

预处理语句

对于JDBC，如果使用Connection和某个数据库建立连接对象con，那么con就可以调用prepareSatement(String sql)方法，方法对参数sql指定的SQL语句进行预编译处理，生成该数据库底层的内部命令，并将该命令封装子啊PreparedStatement对象中。

```
PreparedStatement pre=prepareStatement(String sql)
```

预处理语句设置通配符“?”的值的常用方法有：

```
void setDate(int parameterIndex,Date x)
void setDouble(int parameterIndex,double x) void setFloat(int
parameterIndex,float x)
```

事务

事务是保证数据库中数据完整性与一致性的重要机制。

1.连接对象使用setAutoCommit(boolean autoCommit)方法，将参数autoCommit取值为false来关闭自动提交模式：

```
con.setAutoCommit(false)
```

2.commit()方法

con调用commit()方法就是让事务中的SQL语句全部生效

3.rollback()方法

只要事务中任何一个SQL语句没有生效，就抛出SQLException异常。在处理SQLException异常时，必须让con调用rollback()方法。

关闭数据库

关闭数据库操作的顺序与打开数据库操作的顺序相反

1.先关闭结果集(ResulrSet)

2.再关闭操作(Statement)

3.最后关闭连接(Connection)

数据接口操作实例

```
try {
    // 加载MySQL数据库驱动程序
    Class.forName("com.mysql.cj.jdbc.Driver");
} catch (Exception e) {
    // 打印加载驱动程序过程中的异常信息
    e.printStackTrace();
}

// MySQL数据库连接URL，指定本地主机地址、端口和数据库时区及字符编码
String url = "jdbc:mysql://127.0.0.1:3306/BookStore?
serverTimezone=Asia/Shanghai&characterEncoding=utf8";

// 准备SQL语句，向BookInfo表插入书籍信息
String sql = "insert into BookInfo(bookName, publisher, author) values(?, ?,
?)";

try (
    // 获取数据库连接对象，使用完毕后自动关闭连接
    Connection conn = DriverManager.getConnection(url, "root", "lj123$%^");
    // 创建PreparedStatement对象，用于执行带参数的SQL语句
    PreparedStatement ps = conn.prepareStatement(sql)
) {
    // 设置SQL语句中的参数值
    ps.setString(1, book.getBookName());
    ps.setString(2, book.getPublisher());
    ps.setString(3, book.getAuthor());

    // 执行更新操作并获取影响的行数
    int ret = ps.executeUpdate();
} catch (Exception e) {
    // 捕获执行SQL操作过程中的异常，并打印异常信息
    e.printStackTrace();
}
```

JSP文件操作

File类的创建

创建File类，指向当前路径下bookstory.xml

```
File inputFile=new File("bookstroy.xml");
```

字节流

字节输入流：InputStream

字节输出流：OutputStream

在JSP中使用XML

XML文件是由标记构成的文本文件。标记的名称可以由字母、数字、下划线“_”、点号“.”或连字符“-”组成，但必须以字母或下划线开头，而且标记名称区分大小写，例如:Graren与Graren是完全不同的标记。

XML

空元素

XML空元没有内容，可以有属性

数据结构

XML采用**树状结构**

XML实例

```
<servlet>
    <!-- 定义一个名为 AddBook 的Servlet -->
    <servlet-name>AddBook</servlet-name>
    <!-- 指定 Servlet 类的完整路径 -->
    <servlet-class>lj.test.AddBookServlet</servlet-class>
</servlet>
<servlet-mapping>
    <!-- 将 Servlet 名称 AddBook 映射到 URL 模式 /addBook -->
    <servlet-name>AddBook</servlet-name>
    <!-- 定义 Servlet 的 URL 映射模式 -->
    <url-pattern>/addBook</url-pattern>
</servlet-mapping>
```

SAX

和DOM解析器不同的是，SAX解析器不在内存中建立和XML文件相对应的树形结构数据，SAX解析器的核心是事件处理机制（即SAX是**事件驱动型XML解析器**）。和DOM解析器相比，SAX解析器占有的内存少，对于许多应用程序，用SAX解析器来获取XML“数据效率具有较高的效率。

使用SAX解析器的基本步骤

1.使用java.xml.parsers包中的SAXParerFactory类调用其方法newInstance()实例化一个SAXParserFactory对象。

```
SAXParserFactory factory=SAXParserFactory.newInstance();
```

2.SAXParserFactory对象调用newSAXParser()方法返回一个SAXParser对象，称为SAX解析器。

```
SAXParser saxParser=factory.newSAXParser();
```

3.saxParser对象调用public void parse(File f,DefaultHandler dh)方法解析参数f指定的XML文件。

```
saxParser.parse(new File("tour.xml"),handler);
```