WEB应用开发

课程内容

- Java Web技术概述
- Servlet技术模型
- Servlet容器模型
- JSP技术模型
- 表达式语言
- JSP标签技术
- JDBC数据库访问
- Servlet高级应用

第一章 Java Web技术概述

- 重点：

1. web常用技术
2. MVC设计模式

1.1 web常用技术

- HTTP请求-响应
- URL和URI
- 常见HTTP标签
- form表单

例

1、（ ）要将一个名称为test的表单提交到add.jsp，提交方法为post，正确写法是：

A. <Form name=“test” action=“add.jsp” post =“method”>

B. <Form name=“test” action=“add.jsp” method=“post”>

C. <Form name=“add.jsp” action=“test” method=“post”>

D. <Form name=“test” action=“post” method=“add.jsp”>

答案：1、B

2、关于get和post两种请求，下列说法正确的是？（ ）

A. get请求是默认的。

B. get请求处理的数据量大小不受到限制。//get处理小数据量 255字节

C. post请求地址栏里是能看到数据的。

D. post请求可以由doGet方法处理。 不能处理，只能调用

答案：2、A

1.1 web常用技术

- **MVC设计模式**

JavaBean、类　　JSP　　Servlet、Filter

✓ 将Web组件分为模型（Model）、视图（View）和控制器（Controller）三种类型

① 控制器是实现模型与视图之间的数据流向与数据转换功能的应用程序，即负责View和Model之间的控制关系，即负责实现业务逻辑，使用Servlet或Filter实现

② 模型是实现控制数据访问与数据处理功能的应用程序，使用JavaBeans或POJO实现

JavaBean序列化

③ 视图是实现采集用户输入的数据并传递给控制器，或输出控制器中的处理数据给用户的应用程序，即显示用户界面，使用JSP页面实现

✓ 使用MVC设计模式可以实现表示逻辑与业务逻辑分离，从而使Web应用程序的开发和维护变得容易。

1.1 web常用技术

- **采用基于MVC的设计模式开发应用程序**

① 把Servlet用作应用程序的控**制器**，把JSP文档作为**视图**，JavaBeans被用来表示**模型**。

② 所有的请求都被发送给作为控制器的Servlet，它接受请求，并根据请求信息将它们分发给适当的JSP来响应。

③ Servlet还根据JSP的需求生成JavaBeans的实例并输出给JSP环境。JSP可以通过直接调用方法或使用UseBean的自定义标签得到JavaBeans中的数据。

第二章 Servlet技术模型

- **重点：**

1. Servlet API
2. servlet 的执行过程和生命周期
3. 分析请求与发送响应
4. Servlet处理表单数据
5. Servlet处理业务逻辑，实现请求转发等
6. 部署描述文件web.xml中的元素
7. 使用@WebServlet注解

2.1Servlet的API

- Servlet接口及方法
- ServleConfig接口作用及方法
- HttpServlet类
- HttpServletRequest接口及常用方法
- HttpServletResponse接口及常用方法

2.2 Servlet的执行过程和生命周期

■ Servlet的执行过程

① 用户通过单击超链接或提交表单想容器请求访问Servlet，容器分析这个请求，创建request和response两个对象

② 容器根据请求的URL找到正确的Servlet，为这个请求创建一个线程对象（每次请求都创建一个线程）

③ 容器调用Servlet的service方法，把请求和响应对象作为参数传递给该方法

入口

④ 调用Servlet的doGet()或doPost()方法

⑤ 向客户发送响应：Servlet使用相应对象获得输出流对象，调用有关方法将响应写给客户，响应通过容器发送给浏览器。

2.2 Servlet的执行过程和生命周期

■ Servlet的生命周期

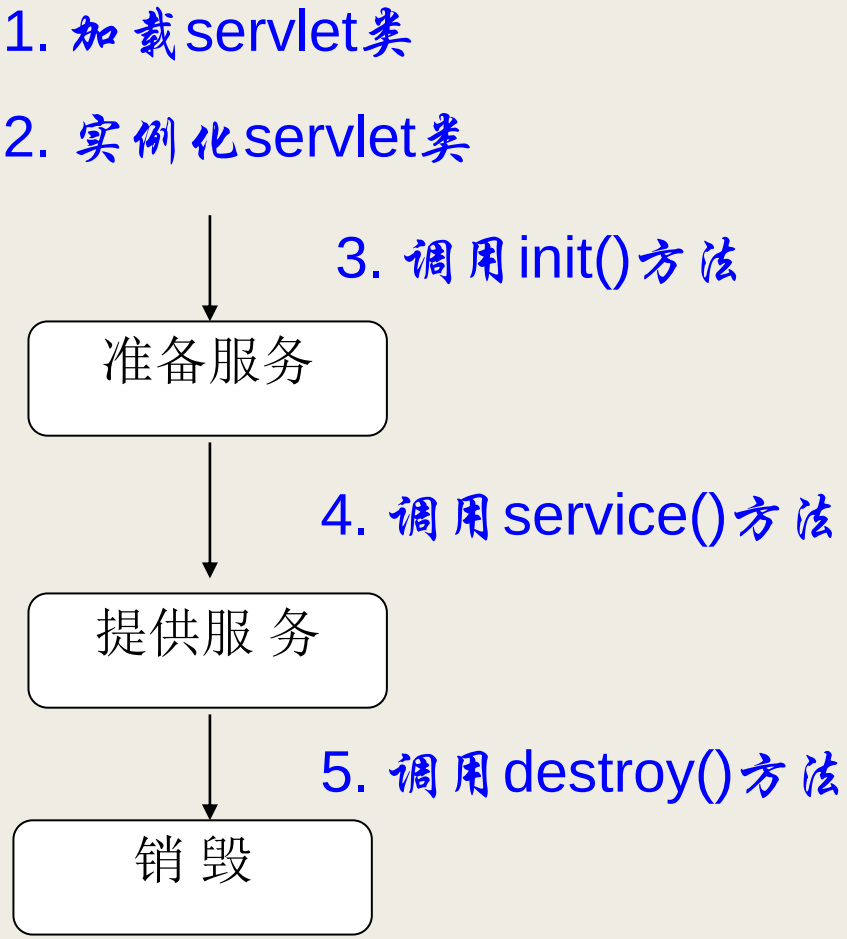
① Servlet容器创建servlet的一个实例。

② 容器调用该实例的init方法进行初始化。

③ 当客户端向该Servlet发送请求时，容器调用此实例的service方法。

④ 在service方法中，根据当前用户请求的方式进一步调用doGet或者doPost方法进行处理。

⑤ 当Servlet容器终止运行或Servlet容器重新装载Servlet的新实例时，Servlet容器调用Servlet的destroy方法释放Servlet所占用的资源。



2.3理解Web应用程序的部署描述文件web.xml

- 下面的代码展示了在部署描述文件中<servlet>元素的一个典型的使用：

```
<servlet>
    <servlet-name>helloServlet</servlet-name>
    <servlet-class>
        com.myserver.HelloServlet
    </servlet-class>
   <init-param>
        <param-name>email</param-name>
        <param-value>hacker@163.com</param-value>
   </init-param>
  <servlet-mapping>
      <servlet-name>HelloServlet</servlet-name>
   <url-pattern>/Hello</url-pattern>
</servlet-mapping>
<load-on-startup>2</load-on-startup>
</servlet>
```

练习

3. 哪个不是Servlet接口的方法？（　）

A. doGet方法　　init() service() desdroy()

B. doPost方法

C. init方法

D. forward方法

答案：3、D

4.考虑下面的HTML页面代码：

请求

当用户在显示的超链接上点击时将调用HelloServlet的哪个方法？

A. doPost()　　B. doGet()

C. doForm()　　D. doHref()

答案：4、B

练习

5、Servlet程序的入口点是：（ ）

A、init（） B、main（）

C、service（） D、doGet（）

答案：5、C

6、以下HTTP响应状态码的含义描述正确的是？（ ）

A、*200 表示请求失败* 请求成功

B、*400 表示不良请求表示服务器未发现与请求URI匹配的内容。*

C、*404 表示由于语法错误而导致服务器无法理解请求信息*

D、*500 表示内部服务器错误，无法处理请求*

答案：6、D

403 服务器端有权限设置

第三章 Servlet容器模型

- **重点：**

1. ServletContext接口
2. **会话管理**
3. **Cookie及其应用**

3.1 ServletContext接口

- 得到ServletConfig接口对象的方法
  - *ServletConfig config = getServletConfig();*
- ServletConfig接口共定义了下面4个方法：
  - *public String getInitParameter(String name)*
  - *public Enumeration getInitParameterNames()*
  - *public String getServletName()*
  - *public ServletContext getServletContext()*
  - *使用RequestDispatcher实现请求转发*
  - *ServletRequest的getRequestDispatcher()方法，可以传递一个相对路径，*
  - *ServletContext的getRequestDispatcher()方法只能传递以“/”开头的路径。*
  - *通过ServletContext对象共享数据*

3.2会话管理

- 理解会话的基本思想和管理机制
- 了解会话对象HttpSession及常见方法
- 调用request.getSession获取HttpSession对象：
  HttpSession session = request.getSession(true);
- 将信息存入会话

  public void setAttribute (String name, Object value)
- 查找与会话相关联的信息

public Object getAttribute(String name)

public Enumeration getAttributeNames()

public void removeAttribute(String name)

- 使用HttpSession对象通常需要三步：

（1）为客户创建或获得与请求关联的会话对象；

（2）在会话对象中添加或删除名/值对属性；

（3）如果需要可使会话失效。

会话时间一般为半小时，若设为-1则不限定时间

不同浏览器会话不共享

会话到有效期 httpsession中有失效方法

3.3Cookie及其应用

- Cookie类的常用方法： getName、 getValue、setValue、setMaxAge和getMaxAge；
- 向客户端发送Cookie
  - *创建Cookie对象*
    - `Cookie c = new Cookie("username", "hacker");`
  - *将Cookie放入到HTTP响应中*
    - `response.addCookie(c)`
- 从客户端读取Cookie

```
Cookie[] cookies = request.getCookies();
if (cookies!=null){
  for(int i = 0;i<cookies.length;i++){
    Cookie cookie = cookies[i];
    if(cookie.getName().equals(cookieName))
      cookieValue = cookie.getValue();
  }
}
```

练习

答案：A

mySession是属于HttpSession类型的对象，则下列语句：
mySission.setMaxInactiveInterval(60);的作用为？（ ）

A．如果用户访问本web应用程序的间隔超过了1分钟，会话将被容器终止

B．如果用户访问本web应用程序的间隔超过了1小时，会话将被容器终止

C．该会话的Cookie将在客户浏览器上保存60天

D．从该语句调用算起，再过60秒钟该会话将被容器强制终止

练习

答案：D

以下说法正确的是？（ ）

A．JSP页面可以在本地打开

B．Session永久地保存在本地

C．Cookie永久地保存在本地 设置有效期

D．会话跟踪可以用URL重写的方式实现

第四章 JSP技术模型

- **重点：**

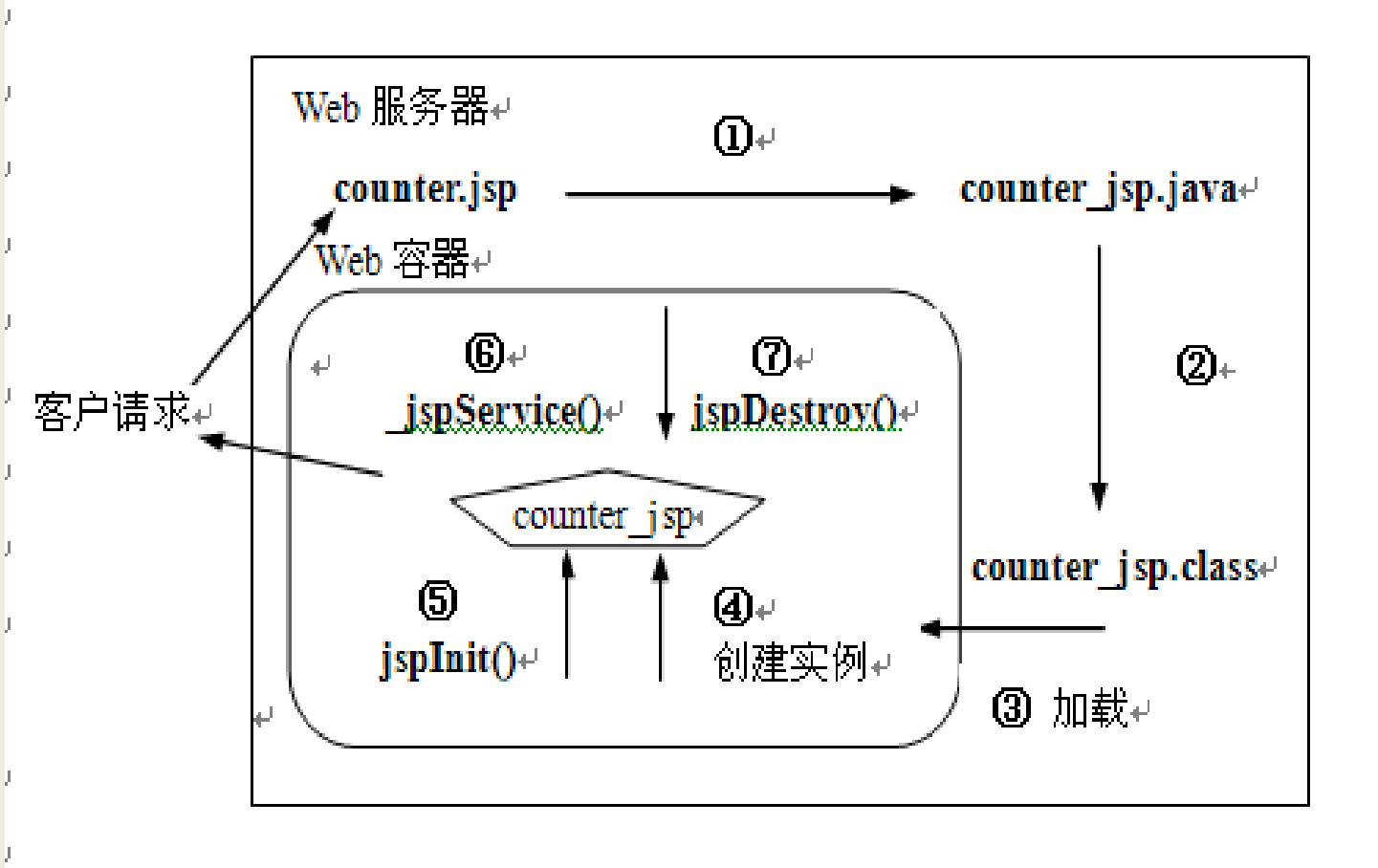
1. JSP页面元素
2. JSP生命周期
3. page指令
4. JSP脚本元素
5. JSP隐含变量
6. JSP组件包含
7. javabeans

4.1 JSP页面元素

JSP页面元素	简要说明	标签语法
声明	声明变量与定义方法	<%! Java 声明 %> 不能写语句
小脚本	执行业务逻辑的Java代码	<% Java 代码 %>
表达式	用于在JSP页面输出表达式的值	<%= 表达式 %>
指令	指定转换时向容器发出的指令	<%@ 指令 %>
动作	向容器提供请求时的指令	<jsp:动作名 />
EL表达式	JSP 2.0引进的表达式语言	${applicationScope.email}
注释	用于文档注释	<%-- 任何文本 --%>
模板文本	HTML标签和文本	同HTML规则

4.2JSP页面生命周期

- 理解JSP的执行过程
- 理解JSP页面转换
- 理解JSP页面的生命周期



4.3JSP隐含变量

request：用户端请求，此请求会包含来自GET/POST请求的参数
response：网页传回用户端的回应
pageContext：网页的属性是在这里管理
session：与请求有关的会话期
application：servlet正在执行的内容
out：用来传送回应的输出
config：servlet的构架部件
page：JSP网页本身
exception：针对错误网页，未捕捉的例外

4.4 作用域对象

- page是代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类）表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面。
- request是代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件。
- session是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求 。
- application是代表与整个 Web 应用程序相关的对象和属性。

4.5JSP组件包含

- 静态包含是在JSP页面转换阶段将另一个文件的内容包含到当前的JSP页面中，语法格式为<%@ include file=” ”%>，file属性值不能是表达式。 编译时就包含，统一转换

- 动态包含是通过JSP标准动作< jsp:include>实现的，它是在请求时将另一个页面的输出包含到主页面的输出中，语法格式为：<jsp:include page=” ” flush=”true”/> ，page属性值必须是相对URL，并指向任何静态或动态web组件，包括servlet，也可以是请求时表达式。 可传参数

4. 5JavaBeans规范

- 遵循下面3个规范的Java类作为JavaBean
  - *类必须是public的;类必须具有无参数的public构造方法,*
  - *JavaBeans类的成员变量一般称为属性（property）。对每个属性访问权限一般定义为private或protected，而不是定义为public的。注意：属性名必须以小写字母开头。*
  - *对每个属性，一般定义两个public方法，它们分别称为访问方法（getXxx）和修改方法（setXxx），允许容器访问和修改bean的属性。*
- *JavaBean的主要特性*
  - *是一个Java类*
  - *有一个无参数的构造函数*
  - *不应该有公开的实例变量*
  - *对值的获取采用getXxx和setXxx方法来访问*
  - *boolean型属性，允许用is代替get和set*

采用MVC设计模式开发程序的步骤

- 重点参考教材例子

1. 定义JavaBeans表示数据
2. 使用Servlet处理请求
3. 填写JavaBeans对象数据
4. 结果的存储
5. 转发请求到JSP页面
6. 从JavaBeans对象中提取数据

练习

答案：D

（　　）在JSP页面转换过程中，以下哪一项描述**不正确**?

A．有些指令在转换阶段产生java代码

B．所有的JSP注释都被忽略

C．所有的JSP表达式都成为_jspService()方法的一部分

D．只有部分JSP声明变成产生的Servlet类的一部分，它们也被原样拷贝

练习

答案：B

（ ）在jsp1.jsp页面中要把请求转发给jsp2.jsp，jsp1.jsp中如何实现?

A. jsp2.jsp

B. <jsp:forward page="jsp2.jsp"/> 动态

C. <jsp:include page="jsp2.jsp"/>

D. <jsp:forward file="jsp2.jsp"/> 静态

练习

（ ）某个客户端浏览器第10次访问以下JSP网页时的输出结果哪项表述正确?

```
<%! int a=0; %>
<%   int b=0;
a++;
b++; %>
a:<%= a %>   ; b:<%= b %>
```

选项:

A. a值不能确定 b=1

B. a=10 b值不能确定

C. a=1 b=10

D. a=10 b=1

答案：A

a-共享资源 线程不安全 可能有多个用户访问

b-转换后作为service()中的局部变量 安全 每次访问初始值都清0

练习

答案：C

当在JSP文件中要使用到Vector对象时，应在JSP文件中加入以下哪个语句？（　）

A.<jsp:include file=”java.util.*” />

B.<jsp:include page=”java.util.*”/>

C.<%@ page import=”java.util.*” %>

D.<%@ page include=”java.util.*” %>

练习

44. jsp:forward和sendRedirect都是用来做页面跳转的，描述错误的是？（　　）

A. forward之后可以使用原来的request对象，而且效率较高。

B. sendRedirect之后不可以使用原来的request对象，而且效率较低。

C. forward地址栏不变化，只能在Web应用程序内的页面间跳转。 只能在自己的工程里转发

D. forward地址栏变化，可以跳转到任何页面和机器。

是sendRedirect

答案：D

练习

（　　）以下关于JavaBean的描述中，哪个是**错误**的：

A. 必须要有一个无参构造方法

B. 名称为xxx的属性一般有相应的getXxx（或isXxx）和setXxx方法

C. 必须要有一个非空的无参构造方法

D. JavaBean也是一种Java类，但必须遵循一定的设计规范，其目的是为了实现组件重用。

答案：C

练习

（　）以下哪个**不属于**JavaBean的4种作用域之一：

A. response　　B. request

C. session　　D. application

答案：A

练习

- 已知某JSP网页调用了某JavaBean，程序输出为：Hello ZJUT！试使用JSP的getProperty与setProperty等动作将以下程序补充完整。

已知JavaBean的内容：

```
package simpleBean;
public class simpleBean implements java.io.Serializable{
    private String myText = new String("");
    public simpleBean() { }
    public String getMyText(){ return myText; }
    public void setMyText(String value){    this.myText=value ;  }
}
```

将以下程序的划线部分补充完整

```
<%@ page language="java" %>
  <jsp:useBean id=“simple" class=simpleBean.SimpleBean />

  <jsp:setProperty name="simple"  property="myText" value="ZJUT" />

<html>
<head>  <title>Bean Example</title>  </head>
<body bgcolor="#FFFFFF" text="#000000">

Hello:<jsp:getProperty name=" simple "  property="myText" />

<body>
<html>
```

第五章 表达式语言

- **重点：**

1. **EL表达式的作用**
2. **EL运算符**
3. **使用EL访问数据**

5.1EL表达式的作用

- 表达式语言（Expression Language，EL）是JSP 2.0的一个重要特征，使用EL可以方便地访问应用数据。
- EL表达式定义了运算符实现算术、关系等运算；
- EL表达式可以对作用域变量、JavaBeans对象、集合的元素、请求参数、Cookie等进行简单的访问；

第六章 JSP标签技术

- **重点：**

1. **自定义标签的开发**
2. **TLD文件**
3. **标准标签库**

练习

放在 文件目录下

编写一个**简单自定义标签**实现功能如下：该标签可以根据要求打印n行星号(每行为“***”)。以下是该标签的TLD配置文件和源代码，以及引用该标签的JSP页面，请补充完整。

1）标签配置文件sampleLib.tld。

```
<tag>
   <name>_____</name>
   <tag-class>sampleLib.LoopTag</tag-class>
   <body-content>empty</body-content>
   <attribute>
     <name>_____</name>
     <required>true</required>
     <rtexprvalue>true</rtexprvalue>
   </attribute>
</tag>
```

loop

count

2）LoopTag.java源代码

```
package sampleLib
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
public class LoopTag extends SimpleTagSupport{
    private int count = 0;
    public void setCount(int count){
      this.count = count
    }
    public void doTag() throws JspException,IOException{
      JspWriter out = getJspContext().getOut();
      for(int i=1; i<=count; i++){
        out.print("***<br>");
      }
    }
}
```

3）引用标签的JSP页面源代码

```
<%@ taglib prefix="demo"  uri="http://www.mydomain.com/sample" %>
<html><body>
  <demo:loop count="3" ></demo:loop>
</body></html>
```

第七章 JDBC数据库访问

- **重点：**

1. JDBC API
2. JDBC方式访问数据库
3. 数据源方式访问数据库
4. 预处理语句
5. DAO设计模式

7.1 JDBC API

- Connecttion接口及方法
- Statement接口及方法
- ResultSet接口及方法

7.2JDBC方式访问数据库的一般步骤

1、加载驱动程序：使用Class类的forName()静态方法，如

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

2、建立连接对象：使用DriverManager类的getConnection()，如

String dburl = jdbc:mysql://localhost:3306/bank

Connection conn = DriverManager.getConnection(dburl, "root", "111111");

3、创建语句对象：使用Connection接口的不同方法创建，如

Statement stmt = conn.createStatement();

4、获得SQL语句的执行结果：对于查询语句，调用executeQuery(String sql)方法,如：

```
String sql = "SELECT * FROM books" ;
ResultSet rst = stmt.executeQuery(sql) ;
while(rst.next()){
  out.print(rst.getString(1)+"\t") ;
}
```

对于语句如CREATE、ALTER、DROP、INSERT、UPDATE、DELETE等须executeUpdate(String sql)方法。

5、关闭建立的对象： close()方法释放资源

7.3数据源方式访问数据库的一般步骤

1. 建立数据源
2. 通过JNDI查找数据源对象
3. 建立连接对象
4. 通过JDBC API操作数据库

7.4DAO设计模式

- DAO（Data Access Object）称为数据访问对象。
- DAO设计模式可以在使用数据库的应用程序中实现业务逻辑和数据访问逻辑分离，从而使应用的维护变得简单。
- 它通过将数据访问实现封装在DAO类中，提高应用程序的灵活性。

- DAO设计模式的开发步骤：

（1）设计传输对象

（2）设计DAO对象

（3）使用DAO对象和传输对象

- 重点：教材中的例子

练习

（ ）在使用JDBC技术访问数据库的过程中，Statement类的作用是：

A. 加载数据库驱动程序

B. 实例化后用于执行SQL语句

C. 生成数据库连接对象

D. 在数据库中创建一张表

答案：B

练习

- 说明使用数据源对象连接数据库的优点是什么？通过数据源对象如何获得连接对象？

【答】 使用数据源是目前Web应用开发中建立数据库连接的首选方法。这种方法是事先建立若干连接对象，存放在连接池中。当应用程序需要一个连接对象时就从连接池中取出一个，使用完后再放回连接池。这样就可避免每次请求都创建连接对象，从而降低请求的响应时间，提高效率。

第八章 Servlet高级应用

- 重点：

1. 监听器
2. 过滤器
3. Servlet的多线程问题

8.1 web监听类和接口

监听对象	事件	监听器接口
ServletContext	ServletContextEvent	ServletContextListener
	ServletContextAttributeEvent	ServletContextAttributeListener
HttpSession	HttpSessionEvent	HttpSessionListener
		HttpSessionActivationListener
	HttpSessionBindingEvent	HttpSessionAttributeListener
		HttpSessionBindingListener
ServletRequest	ServletRequestEvent	ServletRequestListener
	ServletRequestAttributeEvent	ServletRequestAttributeListener
	AsyncEvent	AsyncListener

重点参考教材例子

8.2过滤器的概念、作用和工作原理

- 过滤器是Web服务器上的组件，它们对客户和资源之间的请求和响应进行过滤。
- 采用过滤器验证用户提交的数据的有效性，为请求设置编码字符集，验证用户是否已登录，记录日志审计用户的操作行为，对数据进行压缩或加密等。
- 常见的过滤器包括：

✓ 登录和审计过滤器
✓ 验证过滤器
✓ 图像转换过滤器
✓ 数据压缩过滤器
✓ 加密过滤器

过滤器是如何工作的(工作原理)

- *当容器接收到对某个资源的请求，它要检查是否有过滤器与之关联。如果有过滤器与该资源关联，容器将把该请求发送给过滤器，而不是直接发送给资源。在过滤器处理完请求后，它将做下面3件事：*

- 产生响应并将其返回给客户；
- 如果有过滤器链，它将把（修改过或没有修改过）请求传递给下一个过滤器；
- 将请求传递给不同的资源。

- *当请求返回到客户时，它是以相反的方向经过同一组过滤器返回。过滤器链中的每个过滤器够可能修改响应。*

8.3Servlet的多线程问题

- 用方法的局部变量保存请求中的专有数据
- 只用Servlet的成员变量来存放那些不会改变的数据。
- 对可能被请求修改的成员变量同步（使用synchronized关键字）
- 重点参考教材例子

异步：调用另一个线程在后台执行

练习

初始化方法一般只执行一次

答案：A

- 有关过滤器init(　　)方法错误的是：
  - A. *每次调用过滤器时都会执行init方法。*
  - B. *init方法可以访问FilterConfig对象。 FilterConfig对象提供了对servlet环境及web.xml文件中指派的过滤器名的访问。*
  - C. *利用init将FilterConfig对象存放在一个字段中，以便doFilter方法能够访问servlet环境或过滤器名*
  - D. *FilterConfig对象具有一个getInitParameter方法，它能够访问部署描述符文件（web.xml）中分配的过滤器的初始化参数。*

练习

- 请简要描述过滤器的概念和工作原理。

  过滤器是Web服务器上的组件，它们对客户和资源之间的请求和响应进行过滤。

  - *当容器接收到对某个资源的请求，它要检查是否有过滤器与之关联。如果有过滤器与该资源关联，容器将把该请求发送给过滤器，而不是直接发送给资源。在过滤器处理完请求后，它将做下面3件事：*
- 产生响应并将其返回给客户；
- 如果有过滤器链，它将把（修改过或没有修改过）请求传递给下一个过滤器；
- 将请求传递给不同的资源。
  - *当请求返回到客户时，它是以相反的方向经过同一组过滤器返回。过滤器链中的每个过滤器够可能修改响应。*

考试题型

- 选择题 20分，共10小题
- 简答题 20分，共4小题
- 程序分析题30分，共2-3小题
- 编程题2题30分，第1题10分，第2题20分