Web语言程序设计

第七章： JavaBean与Servlet编程技术

第七章：JavaBean与Servlet编程技术

ICEC

1. JavaBean概述
2. 什么是JavaBean
3. JavaBean的使用
4. 常见JavaBean使用方法
5. Servlet程序基本概述
6. Servlet过滤器的使用

第七章：JavaBean与Servlet编程技术

1 JavaBean概述

Bean的英文含义是豆子，JavaBean就是“一颗Java豆子”，即一段Java小程序。JavaBean是一个Java类，一个可重复使用的软件组件。

实际的应用系统中，大量的嵌入Java代码和HTML语句交织在一起，嵌入Java代码、HTML语句，还有JavaScript语句，使编写和维护JSP网页变得很困难。

1 JavaBean概述

如何解决这个问题呢？使用JavaBean就是一个好办法。将JSP和JavaBean结合起来，在JavaBean中处理逻辑，然后在JSP网页中调用，而JSP文本着重是网页界面设计，这会使得JSP网页变得清晰，可以节省软件开发时间和降低以后维护的难度。目前，这种将显示和逻辑分开的方法深受开发者的喜爱。

2 什么是JavaBean

JavaBean就是一个可重复使用的，基于Java的软件组件，可以在软件开发工具中被直观地操作，应用程序开发者可以通过支持JavaBean的开发工具，直接使用现成的JavaBean，也可以在开发工具容器中，对JavaBean进行必要的修改、测试而不必编写和编译程序。

JavaBean是一种Java类，通过封装属性和方法成为具有某种功能或者处理某个业务的对象，简称Bean。

2.1 JavaBean的特点

JavaBean是基于Java语言的，具有以下特点：

（1）可以实现代码的重复利用，因此可以缩短开发时间。

（2）易编写，易维护、易使用。

（3）可以在任何安装了Java运行环境的平台上使用，而不需要重新编译，为JSP的应用带来了更多的可扩展性。

2.2 JavaBean的特征

JavaBean就是一个Java类，一个JavaBean在类的命名上需要遵循以下特征：

（1）如果成员变量的名字是xxx，为了修改或获取成员变量的值，类中应用以下两个方法：

getXxx()：获取属性XXX；setXxx()：修改属性XXX

（2）对于boolean类型的成员变量可以利用is代替set。

（3）类中方法的访问属性必须是public的。

（4）如果有构造方法，必须是public的，而且无参数（用于初始化工作）。

3 JavaBean的使用

3.1 创建JavaBean

创建一个标准JavaBean的基本语法如下。

（1）定义包名称

（2）定义JavaBean类

JavaBean类定义的语法结构如下：

```
public class 类名称{
//类成员的定义
//类方法的定义。
//类属性的定义。      }
```

其中需要强调的是，类名称与保存的文件名称必须一致，否则就会出现编译错误。同时，类必须声明为公有类，即public。

3.1 创建JavaBean

(3) 定义JavaBean类的构造函数

JavaBean是Java类，JavaBean就有自己的构造函数，并且构造函数的名称必须与JavaBean类的名称一致。构造函数的主要作用是用来初始化，而且构造函数无参数输入，其语法结构如下：

```
public JavaBean类名称
{
//初始化
…….}
```

3.1 创建JavaBean

(4) 定义JavaBean属性

JavaBean是Java类，那么它不但有自己的构造函数，还有自己的属性。JavaBean属性的定义的语法结构如下：

private 数据类型 属性名称；

例如：private String maker;

3.1 创建JavaBean

(5) 设定JavaBean属性值的方法

定义了JavaBean属性之后，还要对其属性值进行设定，其属性值设定的语法结构如下：

```
public void set设定方法名称(数据类型  参数)
{this.变量=参数;}
```

例如：

```
public void setMaker(String maker)
{ this.maker="Benz";   }
```

值得注意的是：JavaBean属性值的设定方法名称一般以set三个字母开头，后面跟上属性设定方法名称。

3.1 创建JavaBean

（6）读取JavaBean属性值的方法

上面讲述了如何设定JavaBean属性值，那么属性值设定好之后，如何读取呢？JavaBean属性值读取的语法结构如下：

```
public void get读取方法名称()
{   return this.变量 = 参数; }
```

例如：

```
public void getMaker()
{   return this.maker;}
```

3.1 创建JavaBean

案例名称：创建**CarBean**

程序名称：**CarBean.java**

```
package nefu;
public class CarBean {
        public CarBean(){ //构造函数 }
        private String Car;//定义JavaBean的属性
        public String getCar() {
                return Car;}
        public void setCar(String car) {
                Car = car;}
        }
```

3.2 使用JavaBean

JSP访问JavaBean，可以使用page指令的import标记引入页面中需要用到的java类。import="{package.class | package.* }, ..."

<%@ page import="java.io.*,java.util.Hashtable" %>

注意：仅仅是引入了一个类，并没有对这个类进行初始化，所以需要通过构造函数进行类对象的初始化。创建类的对象：

类名 对象名= new 类名（）用于初始化

注意：在java声明类时，可以不定义构造函数，系统将为其生成默认的构造函数，与类名字相同，没有参数，也不完成任何操作。

3.2 使用JavaBean

程序名称：CarBean.jsp

```
<%@ page language="java" import="java.util.*"
pageEncoding="UTF-8"%>

<%@ page import="nefu.CarBean"%>
<body>
<%CarBean mycar=new CarBean();%>
<%
mycar.setCar("法拉利");
%>
我有一辆：<%=mycar.getCar()%>
```

案例名称：创建**radiusBean**
程序名称：**radius.java**

```
package nefu;
public class radius {
        private int radius;
        public radius(){     }
        public int getRadius() {
                return radius; }
        public void setRadius(int radius) {
                this.radius = radius;}
        public double circlearea(){
                return Math.PI*radius*radius;}
        public double cliclelength(){
                return Math.PI*2*radius;}
}
```

程序名称：7-1.jsp

```
<%@ page import="nefu.radius" %>
 <%
   radius radius=new radius();
  radius.setRadius(100);
  radius.circlearea();
  radius.cliclelength();
   %>
    半径是：<%=radius.getRadius()%><br>
    面积是：  <%= radius.circlearea()%><BR>
    周长是：  <%=radius.cliclelength()%><BR>
```

3.2 使用JavaBean

JSP访问JavaBean，也可以使用<jsp:useBean>标记。<jsp:useBean>标记是用于JavaBean对象的动作标记，当在JSP网页中使用它时，表示会产生一个JavaBean的实例。

<jsp:useBean>标记有5个属性：id、scope、class、beanName和type，如：

<jsp:useBean id="name" scope="page | request | session | application“ class="classname"/>

<jsp:useBean id="pi" class=“nefu.radius" scope="page"/>

程序名称：7-2.jsp

```
<jsp:useBean id="pi" class="com.radius" scope="page"/>
  <%
     pi.setRadius(100);
     %>
     半径：<%=pi.getRadius()%><BR>
     面积：<%=pi.circleArea()%><BR>
     周长：<%=pi.circleLength()%>
```

3.2 使用JavaBean

编译好的JavaBean是一个Class文件，可以通过JAR命令将多个class文件打包成JAR。

打包过程中，需要将class所在的目录一起打包，保存到根目录下，命令：

```
jar  cvf   testbean.jar com/
```

删除class下的类文件，将JAR添加到lib下程序依然可以执行。

3.3 <jsp:getProperty>和<jsp:setProperty>

使用<jsp:getProperty>和<jsp:setProperty>标记对属性进行设置和获取。

<jsp:getProperty>标记用来取得JavaBean的属性值。

```
<jsp:getProperty name="beanname"
    property="propertyname">
```

其中，beanname是JavaBean实例名，它是在JSP文本中前面使用<jsp:useBean>标记引入的。propertyname为JavaBean的属性名。

3.3 <jsp:getProperty>和<jsp:setProperty>

<jsp:setProperty>标记

用来设置JavaBean的属性值，它一共有如下3种形式：

① <jsp:setProperty name="beanname" property="*">

② <jsp:setProperty name="beanname"

property="propertyname" param="paramname">

③ <jsp:setProperty name="beanname"

property="propertyname" value="beanvalue">

3.3 <jsp:getProperty>和<jsp:setProperty>

第一种方法：

```
<jsp:setProperty  name="beanname"  property="propertyname"
value="beanvalue">
```

其中，beanname为程序中使用的JavaBean实例名，它是在JSP文本中前面使用<jsp:useBean>标记引入的。beanvalue表示用来设定JavaBean的属性值。

3.3 <jsp:getProperty>和<jsp:setProperty>

程序名称：modify.java

```
package com;
public class modify {
        private String aa="aa";
        private String bb="bb";
        public modify(){}
        public String getAa() {
                return aa;}
        public void setAa(String aa) {
                this.aa = aa;}
        public String getBb() {
                return bb;}
        public void setBb(String bb) {
                this.bb = bb;}
}
```

3.3 <jsp:getProperty>和<jsp:setProperty>

程序名称：7-3.jsp

```
<jsp:useBean id="mybean" class="com.modify" scope="page"/>
aa调用jsp:setproperty之前的值：
<jsp:getProperty name="mybean" property="aa"/><br>
<jsp:setProperty name="mybean" property="aa" value="cc"/>
aa调用jsp:setProperty之后的值：
 <jsp:getProperty name="mybean" property="aa"/><br>
<HR>
bb调用jsp:setproperty之前的值：
<jsp:getProperty name="mybean" property="bb"/><br>
<jsp:setProperty name="mybean" property="bb" value="dd"/>
bb调用jsp:setProperty之后的值：
 <jsp:getProperty name="mybean" property="bb"/><br>
```

3.3 <jsp:getProperty>和<jsp:setProperty>

第二种方法：

```
<jsp:setProperty name="beanname" property="*">
```

利用Java Web 自省机制将From表单的元素逐一对Javabean进行赋值，表单元素需要和JavaBean中的属性名一致。

3.3 <jsp:getProperty>和<jsp:setProperty>

程序名称：7-4.jsp

```
<FORM method="post" action="">
  请输入参数1：<INPUT name="aa" type="text"><BR>
  请输入参数2：<INPUT name="bb" type="text"><BR>
  <INPUT type="submit" value="提交">
  <INPUT type="reset" value="取消"><br>


  </FORM>
  <hr>
  <jsp:useBean id="aa" class="com.aaa"></jsp:useBean>
 <jsp:setProperty name="aa" property="*"/>
  您输入的参数1是： <jsp:getProperty name="aa" property="aa"/><BR>
   您输入的参数2是： <jsp:getProperty name="aa" property="bb"/>
```

3.3 <jsp:getProperty>和<jsp:setProperty>

第三种方法：

```
<jsp:setProperty name="beanname" property="propertyname"
param="paramname">
```

利用From表单的元素对Javabean中同名属性进行赋值，param=“paramname”中的paramname是表单中的元素，property=“propertyname”中的propertyname是Javabean中的属性。

3.3 <jsp:getProperty>和<jsp:setProperty>

程序名称：7-5.jsp

```
<FORM method="post" action="">
  请输入参数1：<INPUT name="a1" type="text"><br>
   请输入参数2：<INPUT name="a2" type="text"><br>
   <INPUT type="submit">
  </FORM>
  <jsp:useBean id="aa" class="com.aaa"></jsp:useBean>
  <jsp:setProperty name="aa" property="aa" param="a1"/>
  <jsp:setProperty name="aa" property="bb" param="a2"/>
  <hr>
  您输入的参数1：
  <jsp:getProperty name="aa" property="aa"/><BR>
     您输入的参数2：
  <jsp:getProperty name="aa" property="bb"/><BR>
```

3.3 <jsp:getProperty>和<jsp:setProperty>

程序名称：修改中文字符集

```
public String getBb() {
        try{
                byte b[]=bb.getBytes("ISO-8859-1");
                bb=new String(b);
                return bb;
            }
          catch(Exception e){
              return bb;
        }
}
```

3.4 JavaBean存在的范围

在<jsp:useBean>标记中，有一个scope属性，它是用来设定JavaBean存在的范围。scope属性一共有四种属性值，分别为page、request、session和apllication:

(1) Page ：表示JavaBean实例的生命周期只在一个页面里，只能在一个页面中存取它。

(2) Request：JavaBean实例与Request对象有着很大的关系，它的存在范围除了整个网页（page）外，还包括使用动作元素<jsp:include>和<jsp:forward>包含的网页，也就是说，这些包含的网页可以访问原来网页产生的JavaBean实例。是相邻的2个页面有效。

3.4 JavaBean存在的范围

(3) Session：Session对象是JSP网页创建的内建对象。当用户使用浏览器访问某个网页时，就进行了一个连接，与此同时创建了一个代表该连接的session对象，当浏览器停止浏览一定时间（一般30 min）后，便自动结束代表该连接的session对象。JavaBean实例存在范围与Session类似。

3.4 JavaBean存在的范围

(4) Application：Application范围的JavaBean的生命周期最长，只要Tomcat服务器不重新启动，它就永远存在于服务器的内存中，所以任何页面都可以使用这个JavaBean实例。

4 常见JavaBean使用方法

4.1 表单bean

表单bean就是用来处理表单的。

程序formbean.java和7-6.jsp

如果表单的域名称和Bean中的属性不一致，使用param然后把表单和javabean对应起来。

程序formbean.java和7-7.jsp

如果没有请求参数和属性对应，不采取任何动作，系统也不会提供null作为属性值，意味着bean不需要以此填充完毕，可以只填充一部分属性。

4.2 页面bean

页面Bean为JSP保留数据，不实现应用程序流中的任何功能。

Page—程序7-8.jsp和7-9.jsp scope=“page"

程序名称：7-8.jsp

```
<jsp:useBean id="id" class="com.formbean" scope="page"></jsp:useBean>
<jsp:setProperty name="id" property="id" value="888"/>
<jsp:forward page="7-9.jsp"></jsp:forward>
```

程序名称：7-9.jsp

```
<jsp:useBean id="id"class="com.formbean"scope=“page"></jsp:useBean>
<jsp:getProperty name="id" property="id"/>
```

4. 3 共享bean

理解方法：在一个jsp页面中修改了Bean的属性，然后再另一个页面当中读取整个bean的属性。

Request —程序7-8.jsp和7-9.jsp scope="request"

4.3 共享bean

理解方法：在一个jsp页面中修改了Bean的属性，然后再另一个页面当中读取整个bean的属性。

Session—程序7-10.jsp和7-11.jsp scope="session"

程序名称：7-10.jsp

```
<jsp:useBean id="id" class="com.formbean"
scope="session"></jsp:useBean>
<jsp:setProperty name="id" property="id" value="888"/>
<a href="7-11.jsp">to 11.jsp</a>
```

程序名称：7-11.jsp

```
<jsp:useBean id="id"class="com.formbean"scope="session"></jsp:useBean>
<jsp:getProperty name="id" property="id"/>
```

4.3 共享bean

理解方法：在一个jsp页面中修改了Bean的属性，然后再另一个页面当中读取整个bean的属性。

application—程序7-10.jsp和7-11.jsp scope="application"

程序名称：7-12.jsp

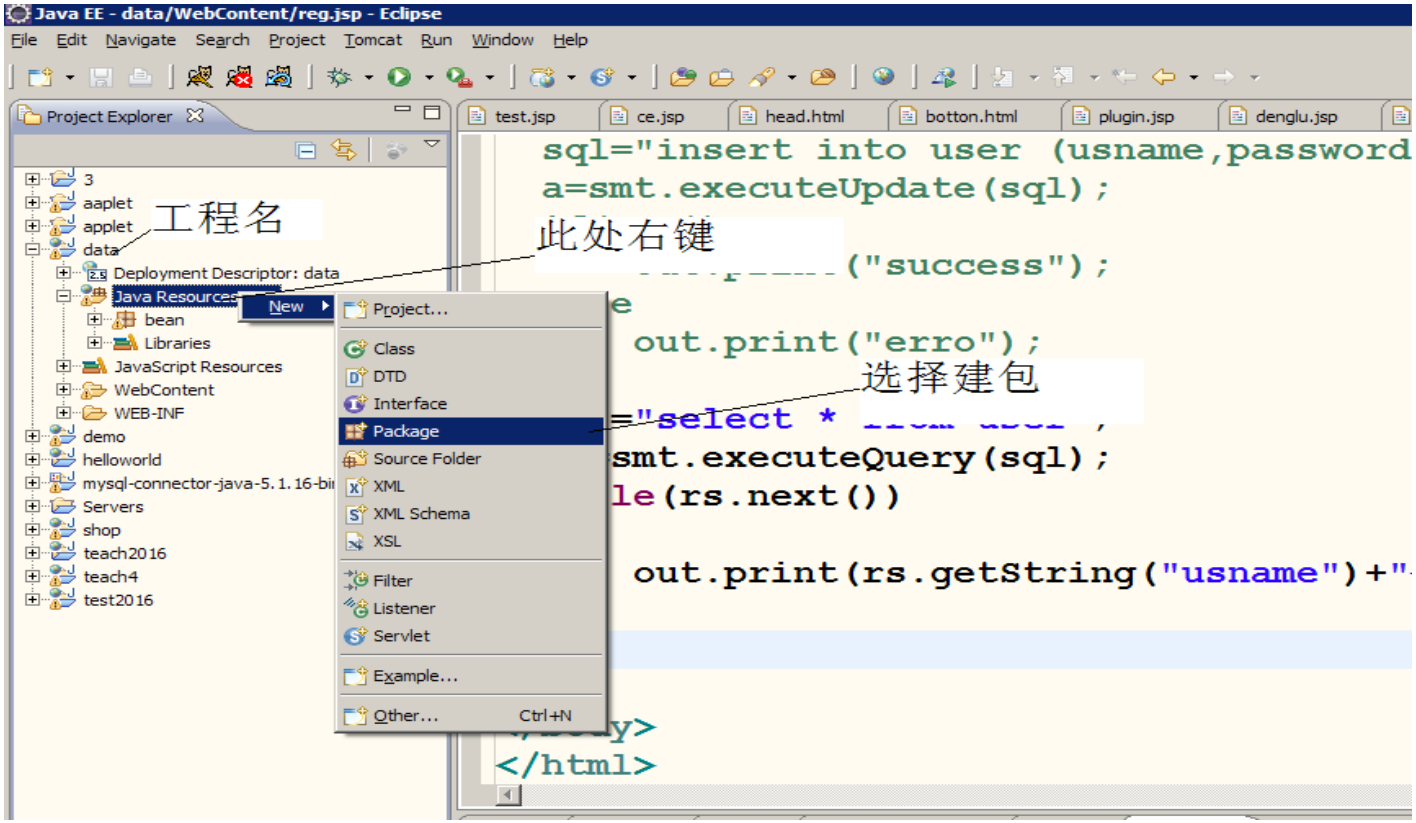
```
<jsp:useBean id="id" class="com.formbean" scope="application"></jsp:useBean>
<jsp:setProperty name="id" property="id" value="888"/>
```

程序名称：7-13.jsp

```
  <jsp:useBean
id="id"class="com.formbean"scope="application"></jsp:useBean>
  <jsp:getProperty name="id" property="id"/>
```

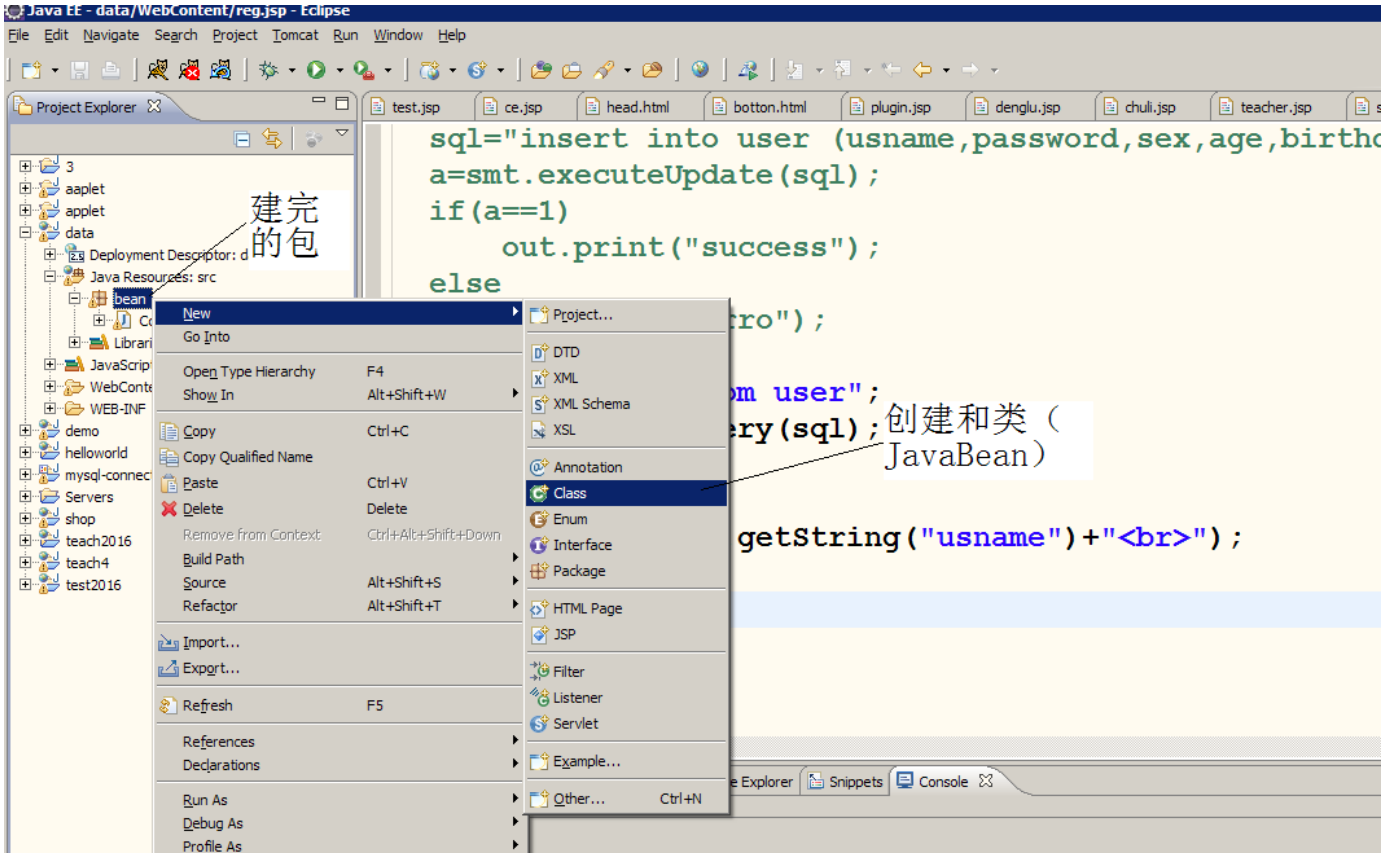
4.4 Eclipse中JavaBean的创建

先建个包



4.4 Eclipse中JavaBean的创建

再建个类



4.4 Eclipse中JavaBean的创建

建完类后，按照Bean的功能编写代码。之后可以重启下Tomcat服务，然后就可以通过运行JSP程序运行JavaBean程序了。

5.Servlet基础概述

5.1什么是Servlet

Servlet是一类Java Web程序，是JSP的前身。

它只能运行在特定的Web容器上，比如 Tomcat上。编写一个Servlet，实际上就是按照Servlet规范编写一个Java 类。

JSP与Servlet实质是相同的，Servlet是JSP的底层，JSP是Servlet的简化设计。

5. Servlet基础概述

5.1什么是Servlet

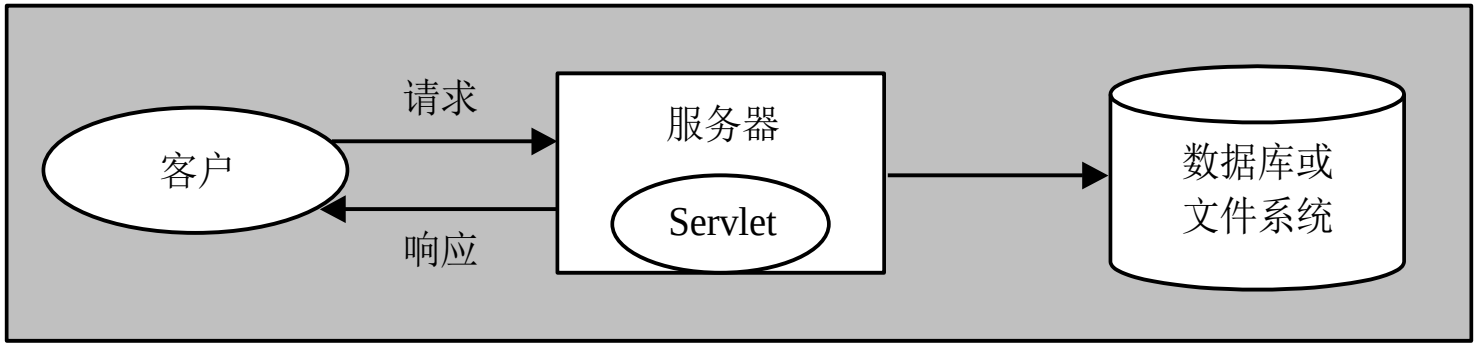


图7-1 Web服务器装载，执行并管理Servlet的过程

5. Servlet基础概述

5.1什么是Servlet

（1）客户向服务器发送对页面的请求。

（2）如果Servlet还没有装入，服务器就装入它。

（3）服务器把请求信息送给Servlet，给每一个请求创建一个执行的新线程（Java语言的线程允许同时执行多个任务）。

（4）Serlvet处理这个请求，生成一个响应并传递给服务器。

（5）服务器把响应送回给客户。

5.Servlet基础概述

5.2 Servlet的生命周期

Servlet运行在Servlet容器中，其生命周期由容器来管理。

Servlet的生命周期与javax.servlet.Servlet接口中

init(),service()和destroy()方法相对应。

Servlet的生命周期包含了4个阶段。

5. Servlet基础概述

5.2 Servlet的生命周期

(1) 初始化事件

(2) 执行 - 处理请求和响应

(3) 终止事件或卸载

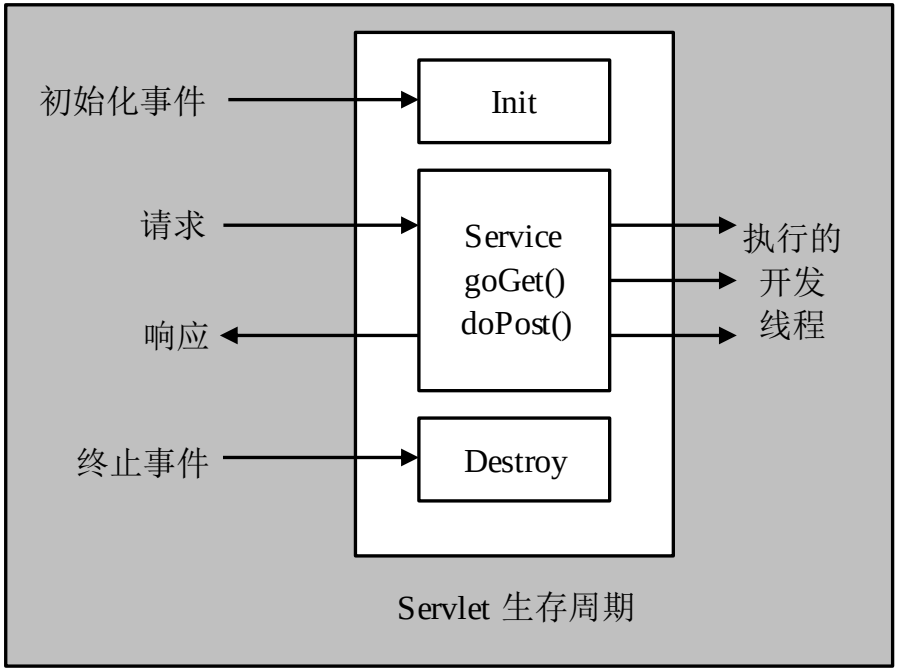


图7-2 Servlet的生命周期

5.Servlet基础概述

5.3 Servlet基本程序的结构

（1）推荐有一个包的声明；

（2）必须导入javax.servlet.*和javax.servlet.http.*包；

（3）所有的servlet类必须实现javax.servlet.Servlet接口，当编写用于处理HTTP协议执行的servlet时，一般选择继承这个接口的实现类javax.servlet.HttpServlet.

(4)在Servlet类中常使用的方法有init()(初始化)、service()(处理请求，具体由doGet()和doPost()方法实现)、destroy()(销毁).

(5)Servlet能直接使用的对象只有HttpServletRequest类的request对象和HttpServletResponse类型的response对象。其他对象由这两个对象获得

5.Servlet基础概述

5.4 Servlet的编写及部署

普通编辑器中编写部署：

1）在记事本等一般编辑工具中直接编写java类即可；

2）将tomcat安装目录下的lib文件夹中的jar文件servlet-api.jar的路径添加到CLASSPATH环境变量中；

3）将编写好的servlet类用javac编译生成.class文件；

4）将编译好的class文件拷入到Web发布目录的根目录中WEB-INF文件夹下的classes目录中设定的包中（如果没有需创建）

5）在WEB-INF文件夹中的web.xml文件中添加<servlet>及<servlet-mapping>标记分别定义servlet的类对象及发布模式；

6）重启tomcat服务

7）在浏览器中发布

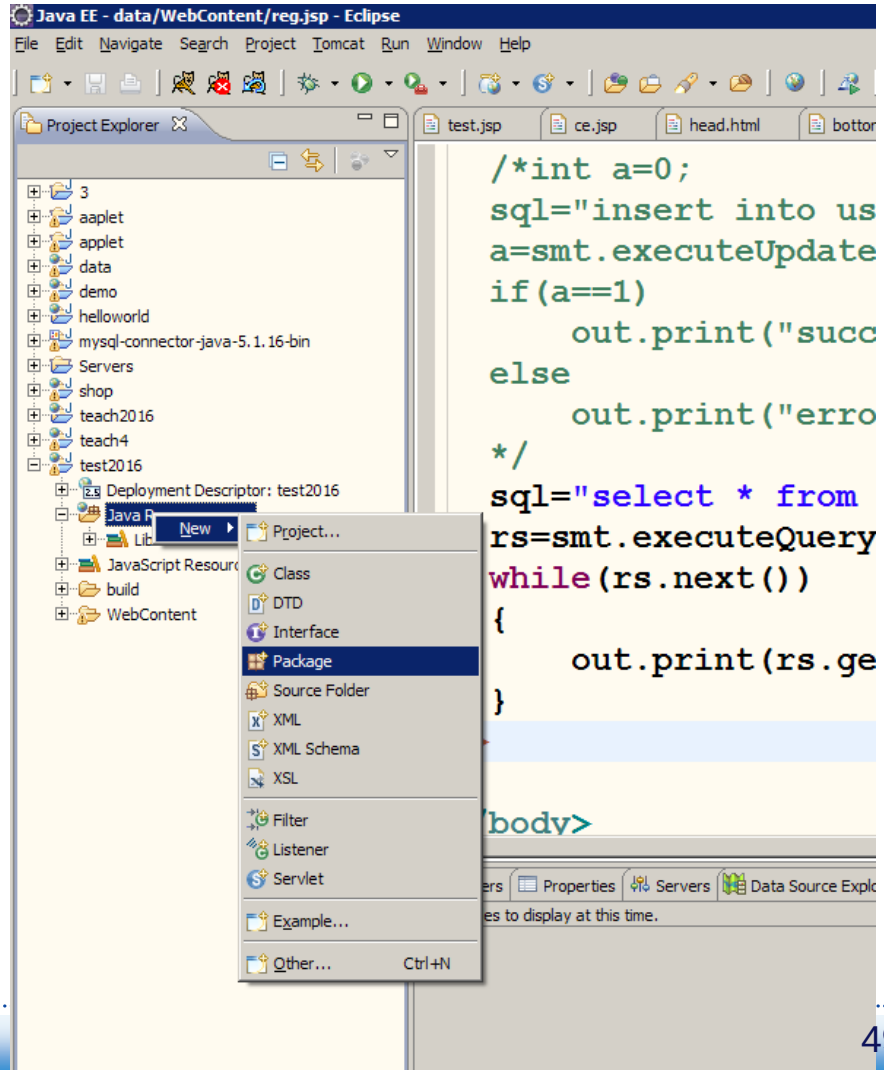
第七章：JavaBean与Servlet编程技术

5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

1）先建个包

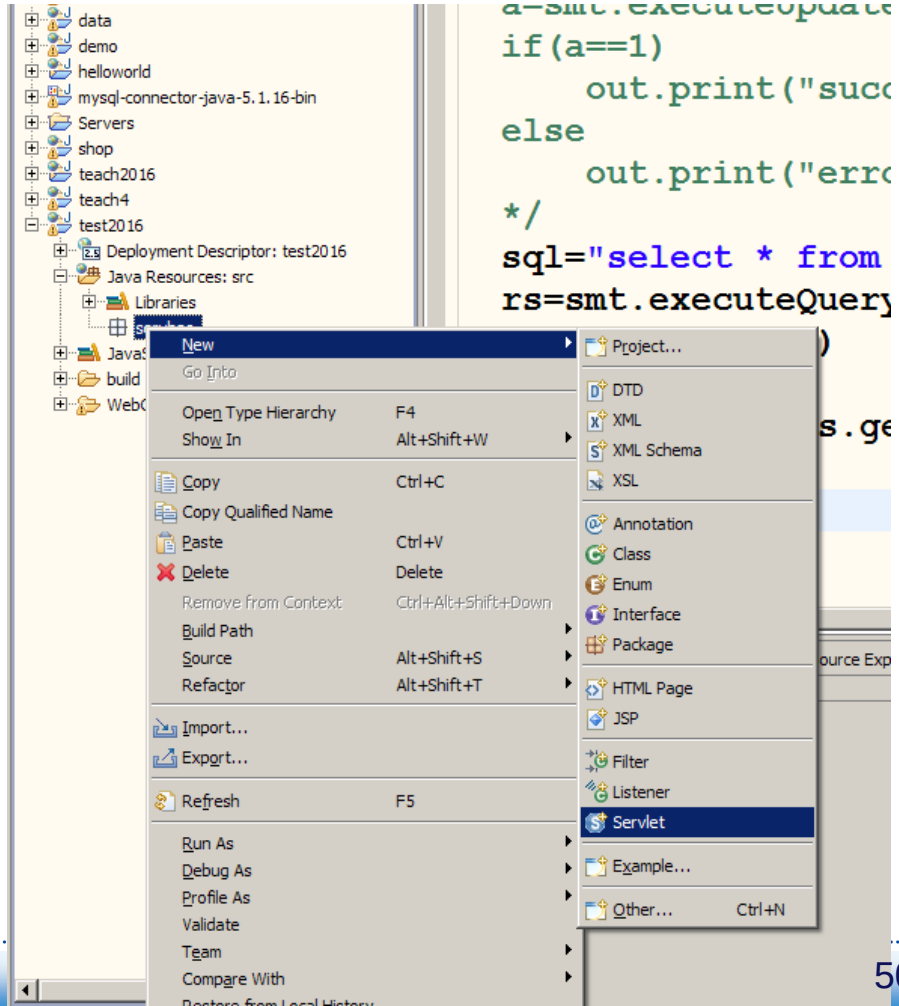


5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

2）再建个Servlet类

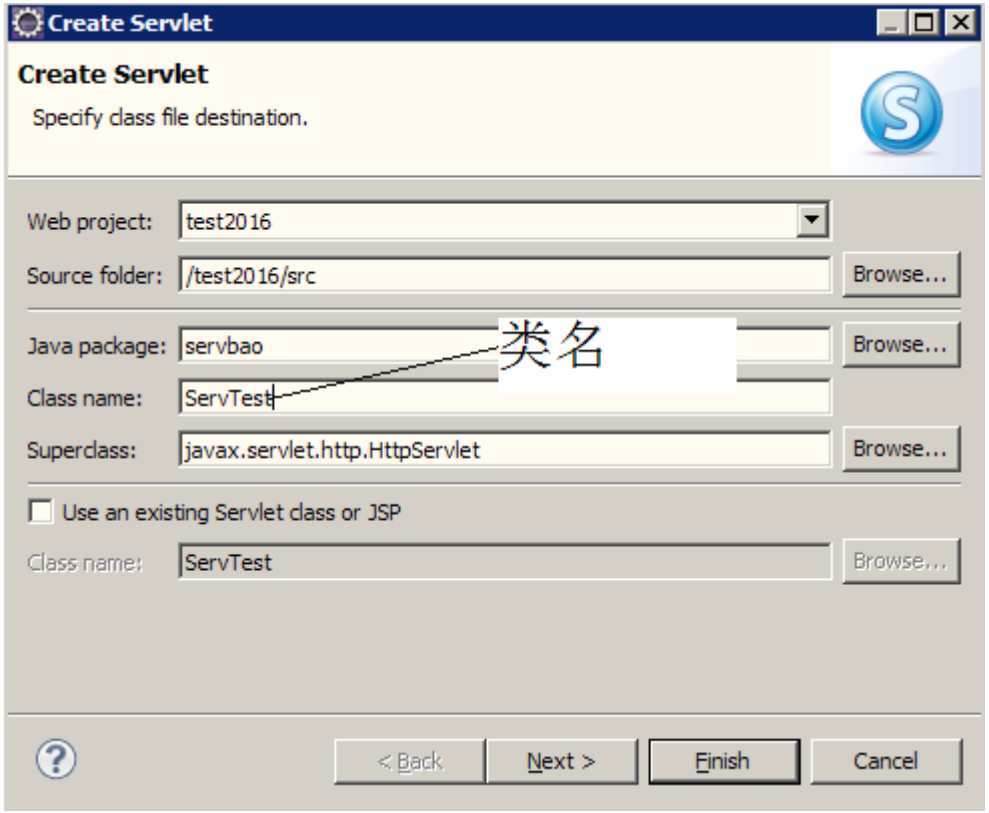


5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

3）命名Servlet类名字



5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

4）编写具体内容后保存

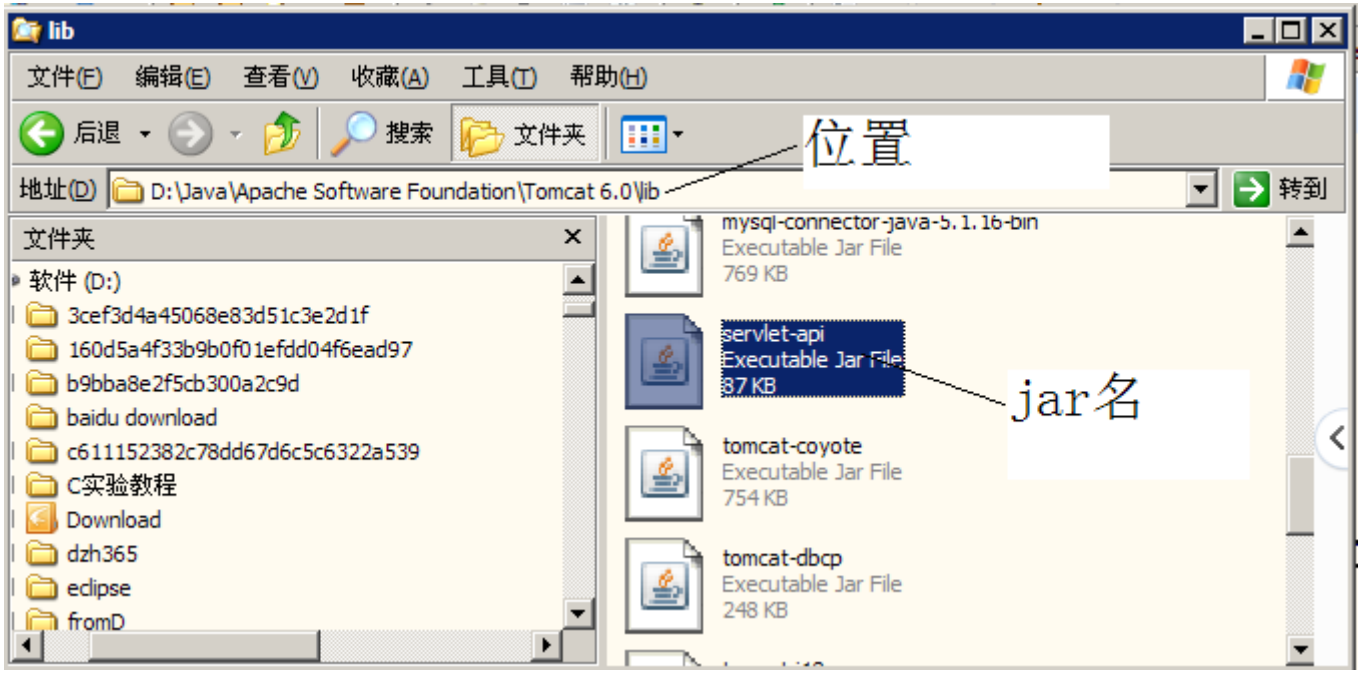
```
package servbao;

import java.io.IOException;

/**
 * Servlet implementation class ServTest
 */
public class ServTest extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServTest() {
        super();
        // TODO Auto-generated constructor stub
    }
```

5.Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

5）把servlet-api.jar拷入到WEB-INF的lib中，如果已经在CLASSPATH中添加其路径则不用此步

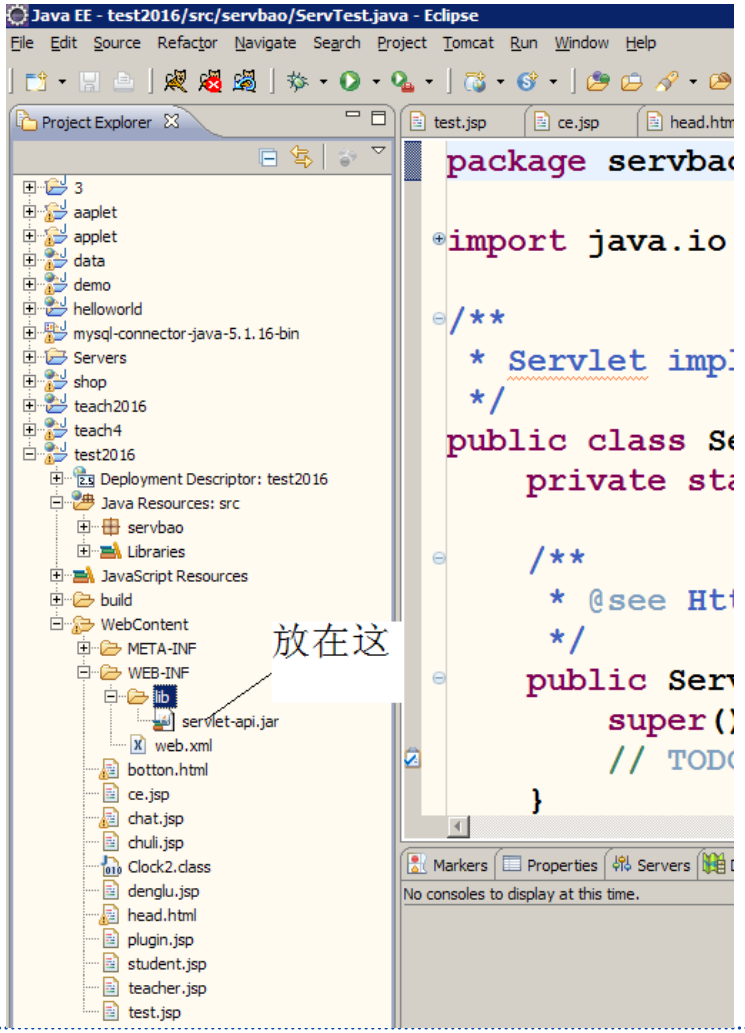


5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

5）把servlet-api.jar拷入到WEB-INF的lib中，如果已经在CLASSPATH中添加其路径则不用此步



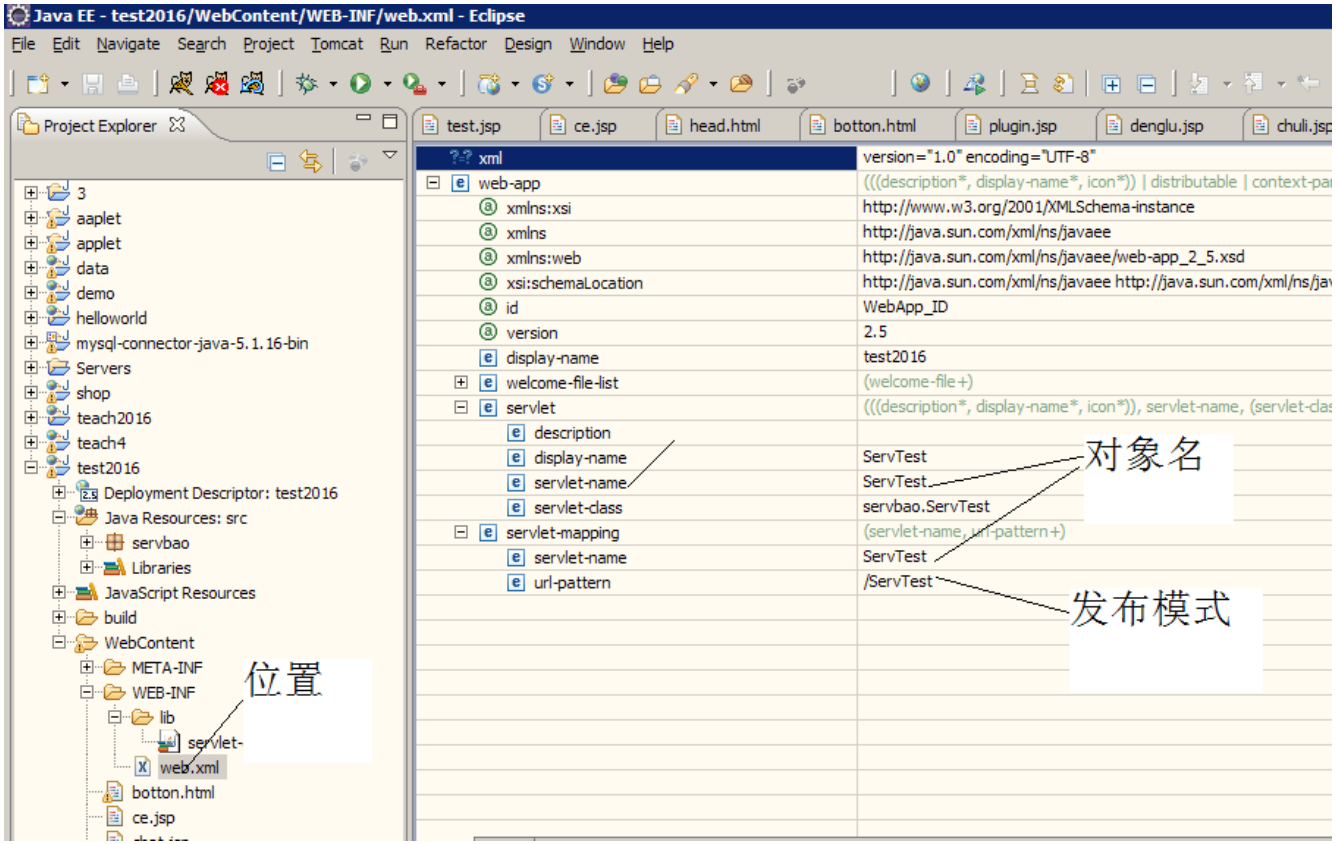
第七章：JavaBean与Servlet编程技术

5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

6）修改web.xml的内容



第七章：JavaBean与Servlet编程技术

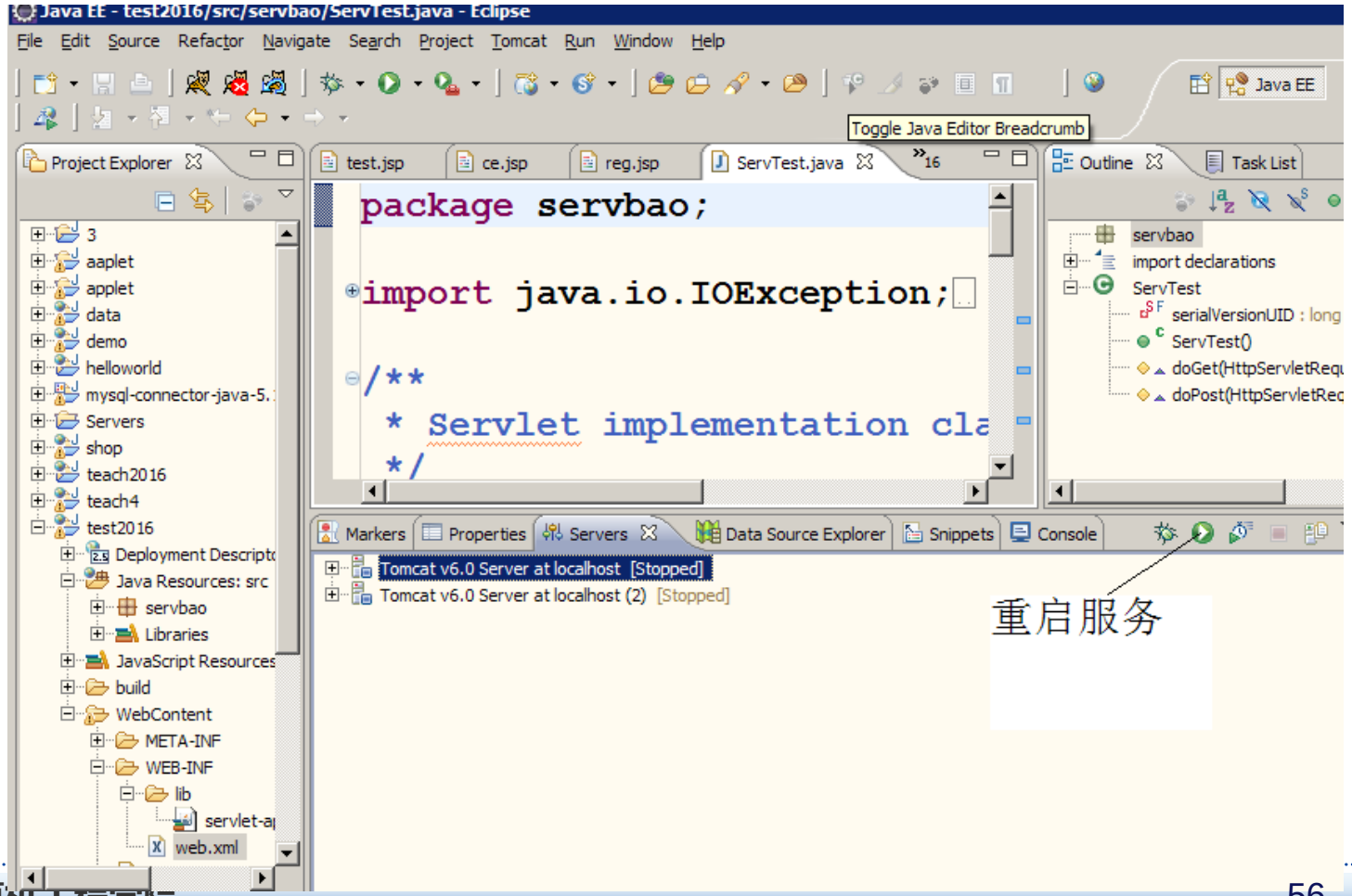
5. Servlet基础概述

5.4 Servlet的编写及部署

在eclipse中编写及部署：

7）重启服务

然后就可以发布了。



6. Servlet过滤器的使用

6.1什么是Servlet过滤器

Servlet过滤器是一中Java的接口，位于客户端和处理的程序之间，能够对请求和响应进行检查。 主要目的如下：

1）在客户端的请求访问后端资源之前，拦截这些请求。

2）在服务器的响应发送回客户端之前，处理这些响应

主要包括：身份验证过滤器、统一字符编码过滤器、字符压缩及加密等。

6. Servlet过滤器的使用

6.2 Servlet过滤器实现方法

Servlet过滤器是一个实现了 javax.servlet.Filter 接口的 Java 类。javax.servlet.Filter 接口定义了三个方法：

1） public void doFilter (ServletRequest, ServletResponse, FilterChain)

该方法完成实际的过滤操作，当客户端请求的方法与过滤器设置相匹配的URL时，Servlet容器将先调用过滤器的doFilter方法。FilterChain实现用户访问后续过滤器。

6. Servlet过滤器的使用

6.2Servlet过滤器实现方法

2） public void init(FilterConfig filterConfig)

web 应用程序启动时，web 服务器将创建Filter 的实例对象，并调用其init方法，读取web.xml配置，完成对象的初始化功能，从而为后续的用户请求作好拦截的准备工作。filter对象只会创建一次，init方法也只会执行一次。

开发人员通过init方法的参数，获得代表当前filter配置信息的FilterConfig对象。

6. Servlet过滤器的使用

6.2 Servlet过滤器实现方法

3） **public void destroy( )**

Servlet容器在销毁过滤器实例前调用该方法，在该方法中释放Servlet过滤器占用的资源。

用户编写的Servlet过滤器由上述三个方法构成并改写。

6. Servlet过滤器的使用

6.3.Servlet过滤器的配置

同样需要在 web.xml 文件中进行配置：

```
<filter>
    <filter-name>过滤实例名</filter-name>
    <filter-class>过滤器各级包名.类名</filter-class>
    <init-param>
        <param-name>初始参数</param-name>
        <param-value>参数值</param-value>
    </init-param>
</filter>
<filter-mapping>
  <filter-name>过滤实例</filter-name>
  <url-pattern>/具体url</url-pattern>
</filter-mapping>
```

6. Servlet过滤器的使用

6.4.Servlet过滤器实例

```
package  各级包名     //导入必需的 java 库
import javax.servlet.*;
import java.util.*;
//实现 Filter 类
public class LogFilter implements Filter  {
    public void  init(FilterConfig config) throws ServletException {
    String site = config.getInitParameter("参数");     // 获取初始化参数
     System.out.println("网站名称: " + site); /      / 输出初始化参数
    }
    public void  doFilter(ServletRequest request, ServletResponse response, FilterChain ch
                throws java.io.IOException, ServletException {
     System.out.println("站点网址：实际URL值");     // 输出站点名称
               chain.doFilter(request,response); /       / 把请求传回过滤链
    }
    public void destroy( ){
       /* 在 Filter 实例被 Web 容器从服务移除之前调用 */
    }
}
```

Web语言程序设计

下　课！！