JSP程序设计教程

第6章 Servlet技术

第 6章 Servlet技术

- 6.1 Servlet基础√
- 6.2 Servlet API编程常用接口和类√
- 6.3 Servlet开发 √
- 6.4 Servlet的应用实例√

课件制作人：王国辉

6.1 Servlet基础

Servlet是在JSP之前就存在的运行在服务端的一种Java技术，它是用Java语言编写的服务器端程序。在JSP技术出现之前，Servlet被广泛地应用来开发动态的Web应用程序。如今在J2EE项目的开发中，Servlet仍然被广泛的使用。

6.1 Servlet基础

6.1.1 Servlet技术简介 √

6.1.2 Servlet技术功能 √

6.1.3 Servlet技术特点 √

6.1.4 Servlet的生命周期 √

6.1.5 Servlet与JSP的区别 √

6.1.6 Servlet的代码结构 √

6.1.7 开发简单的Servlet程序 √

6.1.1 Servlet技术简介

Servlet是一种独立于平台和协议的服务器端的Java技术，可以用来生成动态的Web页面。与传统的CGI（计算机图形接口）和许多其他类似CGI技术相比，Servlet具有更好的可移植性、更强大的功能，更少的投资，更高的效率，更好的安全性等特点。

Servlet是使用Java Servlet应用程序设计接口（API）及相关类和方法的Java程序。Java语言能够实现的功能，Servlet基本上都能实现（除了图形界面外）。Servlet主要用于处理客户端传来的Http请求，并返回一个响应。通常所说的Servlet就是指HttpServlet，用于处理Http请求，其能够处理的请求有doGet()、doPost()、service()等方法。在开发Servlet时，可以直接继承javax.servlet.http.HttpServlet。

6.1.1 Servlet技术简介

Servlet需要在web.xml中进行描述，例如，映射执行Servlet的名字，配置Servlet类、初始化参数，进行安全配置、URL映射和设置启动的优先权等。Servlet不仅可以生成HTML脚本输出，也可以生成二进制表单进行输出。

6.1.2 Servlet技术功能

Servlet通过创建一个框架来扩展服务器的能力，以提供在Web上进行请求和响应的服务。当客户机发送请求至服务器时，服务器可以将请求信息发送给Servlet，并让Servlet建立起服务器返回给客户机的响应。当启动Web服务器或客户机第一次请求服务时，可以自动装入Servlet，之后，Servlet继续运行直到其他客户机发出请求。Servlet的功能涉及范围很广，主要功能如下：

（1）创建并返回一个包含基于客户请求性质的动态内容的完整的HTML页面；

（2）创建可嵌入到现有HTML页面中的一部分HTML页面（HTML片段）；

6.1.2 Servlet技术功能

（3）与其他服务器资源（包括数据库和基于Java的应用程序）进行通信；

（4）用多个客户机处理连接，接收多个客户机的输入，并将结果传递到多个客户机上，例如，Servlet可以是多参与者的游戏服务器；

（5）当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至applet的新连接，并将该连接保持在打开状态；当允许客户机和服务器简单、高效地执行会话的情况下，applet也可以启动客户浏览器和服务器之间的连接，可以通过定制协议进行通信；

（6）将订制的处理提供给所有服务器的标准程序。

6.1.3 Servlet技术特点

Servlet技术带给程序员最大的优势是它可以处理客户端传来的HTTP请求，并返回一个响应。Servlet是一个Java类，Java语言能够实现的功能，Servlet基本上都可以实现（图形界面除外）。总的来说，Servlet技术具有以下特点。

1. 高效

在服务器上仅有一个Java虚拟机在运行，它的优势在于当多个来自客户端的请求进行访问时，Servlet为每个请求分配一个线程而不是进程。

6.1.3 Servlet技术特点

2．方便

Servlet提供了大量的实用工具例程，例如处理很难完成的HTML表单数据、读取和设置HTTP头、处理Cookie和跟踪会话等。

3．跨平台

Servlet是用Java类编写的，它可以在不同的操作系统平台和不同的应用服务器平台下运行。

6.1.3 Servlet技术特点

4. 跨平台

在Servlet中，许多使用传统CGI程序很难完成的任务都可以利用Servlet技术轻松地完成。例如，Servlet能够直接和Web服务器交互，而普通的CGI程序不能。Servlet还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

5. 灵活性和可扩展性

采用Servlet开发的Web应用程序，由于Java类的继承性、构造函数等特点，使得其应用灵活，可随意扩展。

6.1.3 Servlet技术特点

6．共享数据

Servlet之间通过共享数据可以很容易地实现数据库连接池。它能方便地实现管理用户请求，简化Session和获取前一页面信息的操作。而在CGI之间通信则很差。由于每个CGI程序的调用都开始一个新的进程，调用间通信通常要通过文件进行，因而相当缓慢。同一台服务器上的不同CGI程序之间的通信也相当麻烦。

7．安全

有些CGI版本有明显的安全弱点。即使是使用最新的标准和PERL等语言，系统也没有基本安全框架。而Java定义有完整的安全机制，包括SSL\CA认证、安全政策等规范。

6.1.4 Servlet的生命周期

Servlet部署在容器里，它的生命周期由容器管理。Servlet的生命周期概括为以下几个阶段。

（1）当Web客户请求Servlet服务或当Web服务启动时，容器环境加载一个Java Servlet类。

（2）容器环境也将根据客户请求创建一个Servlet对象实例，或者创建多个Servlet对象实例，并把这些实例加入到Servlet实例池中。

（3）容器环境调用Servlet的初始化方法init()进行初始化。这需要给init()方法传入一个ServletConfig对象，ServletConfig对象包含了初始化参数和容器环境的信息，并负责向Servlet传递数据，如果传递失败，则会发生ServletException异常，Servlet将不能正常工作。

6.1.4 Servlet的生命周期

（4）容器环境利用一个HttpServletRequest和HttpServletResponse对象，封装从Web客户接收到的HTTP请求和由Servlet生成的响应。

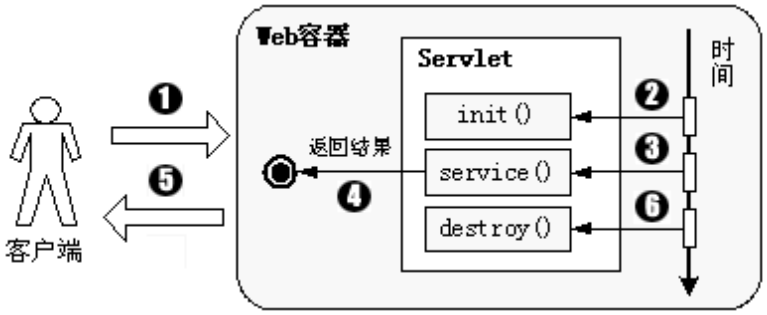
（5）容器环境把HttpServletRequest和HttpServletResponse对象传递给HttpServlet.service()方法。这样，一个定制的Java Servlet就可以访问这种HTTP请求和响应接口。service()方法可被多次调用，各调用过程运行在不同的线程中，互不干扰。

（6）定制的Java Servlet从HttpServletRequest对象读取HTTP请求数据，访问来自HttpSession或Cookie对象的状态信息，进行特定应用的处理，并且用HttpServletResponse对象生成HTTP响应数据。

（7）当Web服务器和容器关闭时，会自动调用HttpServlet.destroy()方法关闭所有打开的资源，并进行一些关闭前的处理。

6.1.4 Servlet的生命周期

在Servlet的整个生命周期中，Servlet的处理过程如下。



第一步：用户通过客户端浏览器请求服务器，服务器加载Servlet，并创建一个Servlet实例；
第二步：容器调用Servlet的init()方法；
第三步：容器调用service()方法，并将HttpServletRequest和HttpServletResponse对象传递给该方法，在service()方法中处理用户请求；
第四步：在Servlet中请求处理结束后，将结果返回给容器；
第五步：容器将结果返回给客户端进行显示；
第六步：当Web器关闭时，调用destroy()方法销毁Servlet实例。

人民邮电出版社
POSTS & TELECOM PRESS

6.1.5 Servlet与JSP的区别

Servlet是一种在服务器端运行的Java程序，从某种意义上说，它就是服务器端的Applet。所以Servlet可以像Applet一样作为一种插件（Plugin）嵌入到Web Server中去，提供诸如HTTP、FTP等协议服务甚至用户自己订制的协议服务。而JSP是继Servlet后Sun公司推出的新技术，它是以Servlet为基础开发的。Servlet与JSP相比有以下几点区别：

（1）编程方式不同；

（2）Servlet必须在编译以后才能执行；

（3）运行速度不同。

6.1.6 Servlet的代码结构

下面将通过一段代码说明一个简单Servlet的基本结构。

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class MingriServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws ServletException, IOException {
        //可编写使用request读取与请求有关的信息和表单数据的代码
        //可编写使用response指定HTTP应答状态代码和应答头的代码
        PrintWriter out = response.getWriter();
        //可编写使用out对象向页面中输出信息的代码
            }
}
```

6.1.6 Servlet的代码结构

该Servlet处理的是get请求，如果读者不理解HTTP，可以把它看成是当用户在浏览器地址栏输入URL、单击Web页面中的链接、提交没有指定method的表单时浏览器所发出的请求。Servlet也可以很方便地处理post请求。post请求是提交那些指定了method="post"的表单时所发出的请求。

若要创建一个Servlet，则应使创建的类继承HttpServlet类，并覆盖doGet()、doPost()方法之一或全部。doGet()和doPost()方法都有两个参数，分别为HttpServletRequest类型和HttpServletResponse类型。HttpServletRequest提供访问有关请求的信息的方法，例如表单数据、HTTP请求头等。

6.1.6 Servlet的代码结构

HttpServletResponse除了提供用于指定HTTP应答状态（200，404等）、应答头（Content-Type，Set-Cookie等）的方法之外，最重要的是它提供了一个用于向客户端发送数据的PrintWriter。对于简单的Servlet来说，它的大部分工作是通过println()方法生成向客户端发送的页面。

注意：doGet()方法和doPost()方法抛出两个异常，因此必须在声明中包含它们。另外还必须导入java.io包（要用到PrintWriter等类）、javax.servlet包（要用到HttpServlet等类）以及javax.servlet.http包（要用到HttpServletRequest类和HttpServletResponse类）。doGet()和doPost()这两个方法是由service()方法调用的，有时可能需要直接覆盖service()方法，比如Servlet要处理Get和Post两种请求时。

6.1.7 开发简单的Servlet程序

【例6-1】 通过Servlet向浏览器中输出文本信息

6.2 Servlet API编程常用接口和类

6.2.1 Servlet接口 √

6.2.2 HttpServlet类 √

6.2.3 ServletConfig接口 √

6.2.4 HttpServletRequest接口 √

6.2.5 HttpServletResponse接口 √

6.2.6 GenericServlet类 √

6.2.1 Servlet接口

javax.servlet包中的类与接口封装了一个抽象框架，建立接收请求和产生响应的组件（即Servlet）。其中javax.servlet.Servlet是所有Java Servlet的基础接口，它的主要方法如下表所示。

方法原型	含　义
public void destroy()	当Servlet被清除时，Web容器会调用这个方法，Servlet可以使用这个方法完成如切断和数据库的连接、保存重要数据等操作
public ServletConfig getServletConfig()	该方法返回ServletConfig对象，该对象可以使Servlet和Web容器进行通信，例如传递初始变量
public String getServletInfo()	返回有关Servlet的基本信息，如编程人员姓名和时间等
public void init(ServletConfig arg0) throws ServletException	该方法在Servlet初始化时被调用，在Servlet生命周期中，这个方法仅会被调用一次，它可以用来设置一些准备工作，例如设置数据库连接、读取Servlet设置信息等，它也可以通过ServletConfig对象获得Web容器通过的初始化变量
public void service(ServletRequest arg0, ServletResponse arg1) throws ServletException, IOException	该方法用来处理Web请求、产生Web响应的主要方法，它可以对ServletRequest和ServletResponse对象进行操作

6.2.2 HttpServlet类

HttpServlet类存放在javax.servlet.http包内，是针对使用HTTP协议的Web服务器的Servlet类。HttpServlet类通过执行Servlet接口，能够提供HTTP协议的功能。HttpServlet类的主要方法如下表所示。

6.2.2 HttpServlet类

方法原型	含　义
protected void doDelete(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP DELETE请求从服务器删除文件
protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP GET请求，客户向服务器请求数据，通过URL附加发送数据
protected void doHead(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP HEAD请求从服务器要求数据，和GET不同的是并不是返回HTTP数据体
protected void doOptions(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP OPTION请求，客户查询服务器支持什么方法
protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP POST请求，客户向服务器发送数据，请求数据
protected void doPut(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP PUT请求，客户向服务器上传数据或文件
protected void doTrace(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException	对应HTTP TRACE请求，用来调试Web程序
protected long getLastModified(HttpServletRequest arg0)	返回HttpServletRequest最后被更改的时间，以ms为单位，从1970/01/01计起

6.2.3 ServletConfig接口

ServletConfig接口存放在javax.servlet包内，它是一个由Servlet容器使用的Servlet配置对象，用于在Servlet初始化时向它传递信息。ServletConfig接口的主要方法如下表所示。

方 法 原 型	含　义
public String getInitParameter(String arg0)	根据初始化变量名称返回其字符串值
public Enumeration getInitParameterNames()	返回所有初始化变量的枚举Enumeration对象，可以用来查询
public ServletContext getServletContext()	返回ServletContext对象，Java的getXxx()方法大多返回原对象，而不是对象拷贝
public String getServletName()	返回当前Servlet的名称，该名称在web.xml里指定

6.2.4 HttpServletRequest接口

HttpServletRequest接口存放在javax.servlet.http包内，该接口的主要方法如下表所示。

方 法 原 型	方 法 原 型
public String getAuthType()	返回Servlet使用的安全机制名称
public String getContextPath()	返回请求URI的Context部分，实际是URI中指定Web程序的部分，例如URI为“http://localhost:8080/mingrisoft/index.jsp”，这一方法返回的是“mingrisoft”
public Cookie[] getCookies()	返回客户发过来的Cookie对象
public long getDateHeader(String arg0)	返回客户请求中的时间属性
public String getHeader(String arg0)	根据名称返回客户请求中对应的头信息
public Enumeration getHeaderNames()	返回客户请求中所有的头信息名称
public Enumeration getHeaders(String arg0)	返回客户请求中特定头信息的值
public int getIntHeader(String arg0)	以int格式根据名称返回客户请求中对应的头信息（header），如果不能转换成int格式，生成一个NumberFormatException异常
public String getMethod()	返回客户请求的方法名称，例如：GET、POST或PUT
public String getPathInfo()	返回客户请求URL的路径信息

6.2.4 HttpServletRequest接口

续表

方法原型	含　义
public String getPathTranslated()	返回URL中在Servlet名称之后、检索字符串之前的路径信息
public String getQueryString()	返回URL中检索的字符串
public String getRemoteUser()	返回用户名称，主要应用在servlet安全机制中检查用户是否已经登录
public String getRequestURI()	返回客户请求使用的URI路径，是URI中的host名称和端口号之后的部分，例如URL为“http://localhost:8080/mingrisoft/index.jsp”，这一方法返回的是“/index.jsp”
public StringBuffer getRequestURL()	返回客户Web请求的URL路径
public String getServletPath()	返回URL中对应servlet名称的部分
public HttpSession getSession()	返回当前会话期间对象
public Principal getUserPrincipal()	返回java.security.Principal对象，包括当前登录用户名称
public boolean isRequestedSessionIdFromCookie()	当前session ID是否来自一个cookie
public boolean isRequestedSessionIdFromURL()	当前session ID是否来自URL的一部分
public boolean isRequestedSessionIdValid()	当前用户期间是否有效
public boolean isUserInRole(String arg0)	已经登录的用户是否属于特定角色

6.2.5 HttpServletResponse接口

HttpServletResponse接口存放在javax.servlet.http包内，它代表了对客户端的HTTP响应。HttpServletResponse接口给出了相应客户端的Servlet()方法。它允许Serlvet设置内容长度和回应的MIME类型，并且提供输出流ServletOutputStream。HttpServletResponse接口的主用方法如下表所示。

6.2.5 HttpServletResponse接口

方法原型	含　义
public void addCookie(Cookie arg0)	在响应中加入cookic对象
public void addDateHeader(String arg0, long arg1)	加入对应名称的日期头信息
public void addHeader(String arg0, String arg1)	加入对应名称的字符串头信息
public void addIntHeader(String arg0, int arg1)	加入对应名称的int属性
public boolean containsHeader(String arg0)	对应名称的头信息是否已经被设置
public String encodeRedirectURL(String arg0)	对特定的URL进行加密，在sendRedirect()方法中使用
public String encodeURL(String arg0)	对特定的URL进行加密，如果浏览器不支持cookie，同时加入session ID
public void sendError(int arg0) throws IOException	使用特定的错误代码向客户传递出错响应
public void sendError(int arg0, String arg1) throws IOException	使用特定的错误代码向客户传递出错响应，同时清空缓冲器
public void sendRedirect(String arg0) throws IOException	传递临时相应，相应的地址根据location指定
public void setHeader(String arg0, String arg1)	设置指定名称的头信息
public void setIntHeader(String arg0, int arg1)	设置指定名称头信息，其值为int数据
public void setStatus(int arg0)	设置响应的状态编码

6.2.6 GenericServlet类

GenericServlet类存放在javax.servlet.包中，它提供了对Servlet接口的基本实现。GenericServlet类是一个抽象类，它的service()方法是一个抽象方法。该类的主要方法如下表所示。

6.2.6 GenericServlet类

方法原型	含　义
public void destroy()	Servlet容器使用这个方法结束Servlet服务
public String getInitParameter(String arg0)	根据变量名称查找并返回初始变量值
public Enumeration getInitParameterNames()	返回初始变量的枚举对象
public ServletConfig getServletConfig()	返回ServletConfig对象
public ServletContext getServletContext()	返回ServletContext对象
public String getServletInfo()	返回关于Servlet的信息，如作者、版本、版权等
public String getServletName()	返回Servlet的名称
public void init() throws ServletException	代替super.init(config)的方法
public void init(ServletConfig arg0) throws Servlet-Exception	Servlet容器使用这个指示Servlet已经被初始化为服务状态
public void log(String arg0, Throwable arg1)	这个方法用来向Web容器的log目录输出运行记录，一般文件名称为Web程序的servlet名称
public void log(String arg0)	这个方法用来向Web容器的log目录输出运行纪录和弹出的运行错误信息
public void service(ServletRequest arg0, ServletRe-sponse arg1) throws ServletException, IOException	由Servlet容器调用，使Servlet对请求进行响应

6.3 Servlet开发

6.3.1 Servlet的创建 √

6.3.2 Servlet的配置 √

6.3.1 Servlet的创建

创建一个Servlet，通常涉及下列4个步骤。

（1）继承HttpServlet抽象类。

（2）重载适当的方法，如覆盖（或称为重写）doGet()方法或doPost()方法。

（3）如果有HTTP请求信息的话，获取该信息。可通过调用HttpServletRequest类对象的以下3个方法获取：

```
getParameterNames()          //获取请求中所有参数的名字
getParameter()               //获取请求中指定参数的值
getParameterValues()         //获取请求中所有参数的值
```

6.3.1 Servlet的创建

（4）生成HTTP响应。HttpServletResponse类对象生成响应，并将它返回到发出请求的客户机上。它的方法允许设置“请求”标题和“响应”主体。“响应”对象还含有getWriter()方法以返回一个PrintWriter类对象。使用PrintWriter的print()方法和println()方法以编写Servlet响应来返回给客户机，或者直接使用out对象输出有关HTML文档内容。

下面我们来看一下按照上述步骤创建的Servlet类。

6.3.1 Servlet的创建

```
package com;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class MyServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                // 第三步：获取HTTP 请求信息
                String myName = request.getParameter("myName");
                // 第四步：生成 HTTP 响应
                PrintWriter out = response.getWriter();
                response.setContentType("text/html;charset=gb2312");
                response.setHeader("Pragma", "No-cache");
                response.setDateHeader("Expires", 0);
                response.setHeader("Cache-Control", "no-cache");
                out.println("<html>");
                out.println("<head><title>一个简单的Servlet程序</title></head>");
                out.println("<body>");
                out.println("<h1>一个简单的Servlet程序</h1>");
                out.println("<p>"+myName+"您好，欢迎访问！");
                out.println("</body>");
                out.println("</html>");
                out.flush();
        }
        public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                this.doGet(request, response);
        }
}
```

6.3.2 Servlet的配置

要正常运行Servlet程序还需要在web.xml文件中进行配置。下面将详细介绍如在web.xml文件中对Servlet进行配置。

1. Servlet的名称、类和其他选项的配置

在web.xml文件中配置Servlet时，必须指定Servlet的名称、Servlet的类的路径，可选择性地给Servlet添加描述信息和指定在发布时显示的名称。具体代码如下：

```
<servlet>
   <description>Simple Servlet</description>
   <display-name>Servlet</display-name>
   <servlet-name>myServlet</servlet-name>
   <servlet-class>com.MyServlet</servlet-class>
</servlet>
```

Serlvet的描述信息

发布时Serlvet的名称

Servlet的名称

Servlet的名称

6.3.2 Servlet的配置

如果要对一个JSP页面文件进行配置，则可通过下面的代码进行指定：

```
<servlet>
   <description>Simple Servlet</description>
   <display-name>Servlet</display-name>
   <servlet-name>Login</servlet-name>
   <jsp-file>login.jsp</jsp-file>
</servlet>
```

要访问的JSP文件名

2．初始化参数

Servlet可以配置一些初始化参数，例如，下面的代码：

6.3.2 Servlet的配置

```
<servlet>
    <init-param>
        <param-name>number</param-name>
        <param-value>1000</param-value>
    </init-param>
</servlet>
```

这段代码用于指定number的参数值为1000。在Servlet中可以在init()方法体中通过getInitParameter()方法访问这些初始化参数

3．启动装入优先权

启动装入优先权通过<load-on-startup>元素指定，例如，下面的代码：

6.3.2 Servlet的配置

```
<servlet>
        <servlet-name>ServletONE</servlet-name>
        <servlet-class>com.ServletONE</servlet-class>
        <load-on-startup>10</load-on-startup>
</servlet>
<servlet>
        <servlet-name>ServletTWO</servlet-name>
        <servlet-class>com.ServletTWO</servlet-class>
        <load-on-startup>20</load-on-startup>
</servlet>
<servlet>
        <servlet-name>ServletTHREE</servlet-name>
        <servlet-class>com.ServletTHREE</servlet-class>
        <load-on-startup>AnyTime</load-on-startup>
</servlet>
```

在这段代码中，ServletONE类先被载入，ServletTWO类则后被载入，而ServletTHREE类可在任何时间内被载入。

6.3.2 Servlet的配置

4. Servlet的映射

在web.xml配置文件中可以给一个Servlet做多个映射，因此，可以通过不同的方法访问这个Servlet，例如下面的代码：

```
<servlet-mapping>
        <servlet-name>OneServlet</servlet-name>
        <url-pattern>/One</url-pattern>
</servlet-mapping>
```

通过上述代码的配置，若请求的路径中包含“/One”，则会访问逻辑名为“OneServlet”的Servlet。再如下面的代码：

6.3.2 Servlet的配置

```
<servlet-mapping>
        <servlet-name>OneServlet</servlet-name>
        <url-pattern>/Two/*</url-pattern>
</servlet-mapping>
```

通过上述配置，若请求的路径中包含“/Two/a”或“/Two/b”等符合“/Two/*”的模式，则同样会访问逻辑名为“OneServlet”的Servlet。

人民邮电出版社
POSTS & TELECOM PRESS

6.4 Servlet的应用实例

6.4.1 应用Servlet实现留言板 √

6.4.2 应用Servlet实现购物车 √

6.4.1 应用Servlet实现留言板

留言板对于大家来说并不陌生，本节将介绍应用Servlet实现一个简单留言板的实例，在实例的开发过程中，应用了在第5章中介绍的工具JavaBean，该JavaBean用来转换HTML中的特殊字符、格式化时间以及解决出现的中文乱码问题。

【例6-2】 应用Servlet实现留言板

6.4.2 应用Servlet实现购物车

在第5章JavaBean的应用实例中，通过JavaBean实现了一个购物车，本节将应用Servlet来实现一个具有相同功能的购物车。读者可对这两种方法进行比较，了解两种方法各自的优势。

在程序开发过程中，应用了第5章实现的购物车实例中名为“MyTools”和“ShopCar”的工具JavaBean，以及名为“GoodSingle”的值JavaBean，这体现了JavaBean在程序开发中的可重复使用的特性。

【例6-3】 应用Servlet实现购物车