崔群法 王咏梅 李有军 编著

Struts 2.0 从入门到精通

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Struts 2.0 从入门到精通

崔群法　王咏梅　李有军　编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书内容全面，涵盖了从事 Struts 2.0 开发所应掌握的所有基础知识。在知识的讲解上，采用理论与实践相结合的方式，从程序运行的内部机制进行分析讲解。介绍了 Struts 2.0 框架的核心组件和核心处理机制，并介绍了拦截器、国际化、输入校验、类型转换等 Struts 2.0 的关键技术，同时在书的末尾以实例方式演示了 Struts 2.0 的综合应用。

本书是介绍 Struts 2.0 框架的专业书籍，适合大专院校在校生、网站开发人员、职业技术培训人员及编程爱好者学习和参考。

图书在版编目（CIP）数据

Struts 2.0 从入门到精通 / 崔群法，王咏梅，李有军编著.—北京：电子工业出版社，2009.1
ISBN 978-7-121-07550-6

I. S… II.①崔…②王…③李… III.软件工具－程序设计 IV.TP311.56

中国版本图书馆 CIP 数据核字（2008）第 159139 号

责任编辑：李红玉
文字编辑：何　况
印　　刷：北京天竺颖华印刷厂
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
　　北京市海淀区翠微东里甲 2 号　邮编：100036
开　　本：787×1092　1/16　印张：28.5　字数：720 千字
印　　次：2009 年 1 月第 1 次印刷
定　　价：56.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Struts 是世界上第一个 MVC 框架，经过长达 6 年的发展，它已经成为了一个高度成熟的框架，不管是稳定性还是可靠性都得到了广泛的证明。它有着丰富的开发人群，几乎成为了事实上的工业标准。但是随着时间的流逝和 Web 技术的进步，Struts 1.0 的局限性也越来越多地暴露出来，制约了其继续发展。

Struts 2.0 虽然是在 Struts 1.0 的基础上发展起来的，但实质上是以 WebWork 为核心，它并没有继承 Struts 1.0 的设计理念，而是使用了 WebWork 的设计理念。与 Struts 1.0 相比，Struts 2.0 有很多革命性的改进，差别也很大。Struts 2.0 兼容了 Struts 1.0 和 WebWork 两个框架，使原来使用 Struts 1.0 和 WebWork 的开发人员都能平稳过渡到使用 Struts 2.0 框架。

Struts 2.0 的目标很简单——使 Web 开发变得更加容易。为了达成这一目标，Struts 2.0 中提供了很多新特性，比如智能的默认设置、annotation 的使用及“惯例重于配置”原则的应用，这些都大大减少了 XML 配置。

本书面向 Struts 2.0 的实际应用开发，通过大量的实例，循序渐进地为读者介绍了有关 Struts 2.0 开发所涉及的各类知识。本书内容由浅入深，涵盖了 Struts 的主要知识点，在介绍过程中，针对每个知识点都有相应的实例。本书叙述通俗易懂，结构安排合理。具体内容如下。

第 1 章：Struts 2.0 学习必备。它是学习 Struts 的预备篇，将重点介绍学习 Struts 必须掌握的一些基本技术，如 JSP 和 Servlet 等。

第 2 章：Struts 2.0 简介。主要介绍了 Struts 2.0 的发展历程、Struts 的优势和配置 Struts 运行环境。并且以一个简单的实例，介绍了 Struts 的运行流程和各个部分的作用。

第 3 章：Eclipse 开发 Struts。介绍了在广受欢迎的 Eclipse 开发工具中，进行基于 Struts 2.0 框架的 Web 开发，从而为读者开发大的应用程序并提高工作效率做好准备。

第 4 章：Struts 2.0 拦截器。介绍了拦截器的基本概念、拦截器工作原理、配置拦截器和自定义拦截器等，为后面文件的上传与下载、国际化、类型转换、数据校验等操作中实现拦截器打下了基础。

第 5 章：Struts 2.0 的 Action 和类型转换。主要介绍了 Action 在 Struts 2.0 应用中的作用，以及独立实现和配置 Action，并在本章的最后一节介绍了 Struts 2.0 的类型转换。

第 6 章：Struts 2.0 标签库。主要介绍了一个功能强大、支持广泛、高扩展性的 Struts 2.0 标签库。Struts 2.0 标签库把所有的标签都统一到了一个标签库中，简化了标签的使用。

第 7 章：Struts 2.0 文件配置。主要介绍了 Struts 2.0 在开发过程中，需要完成的配置及相关配置选项介绍，如 web.xml、struts.xml 和 struts.properties。

第 8 章：文件上传与下载。主要介绍了 Struts 2.0 使用拦截器，实现不用配置，而自动上传和下载文件。

第 9 章：Struts 2.0 的数据库应用。主要介绍了使用 Struts 2.0 进行 Web 开发时的数据库应用技巧和方法，并通过具体的实例进行了演示，这里我们使用的是 MySQL 5 数据库。

第 10 章：Struts 2.0 输入校验。输入校验是 Struts 2.0 针对用户的输入，而提供的一种校验方式，本章主要介绍了手动完成输入校验、内置校验和自定义校验等内容。

第 11 章：Struts 2.0 高级应用。主要介绍了 Struts 2.0 的国际化和异常处理，通过 Struts 2.0 的国际化处理，实现在不同地区和语言环境中以当地的语言显示 Web 应用系统。并且 Struts 2.0 采用了一种声明式的异常处理机制，从而避免了在程序中编写大量的 try()catch{}块，使异常处理和代码不再耦合。

第 12 章：SiteMesh 框架简介。详细介绍了 SiteMesh 及 Struts 2.0 整合 SiteMesh 的具体应用。通过本章的学习，希望读者能熟练地在 Struts 2.0 中应用 SiteMesh 框架进行 Web 页面开发。

第 13 章：用户在线注册系统。讲述了如何使用 Struts 2.0、JSP、JavaBean 技术编写用户注册系统，在讲解过程中把系统的需求分析、系统设计和数据库设计和模块实现等实现过程一一介绍给读者，使读者在充分学习 Struts 2.0 技术的基础之上，把握先进的设计理念和实现过程。

第 14 章：BBS 论坛开发。主要介绍使用 Struts 2.0 框架来开发一个 BBS 论坛。通过本章的学习，读者不但可以巩固前面学习的知识，更重要的是让读者了解如何设计与实现一个应用项目。

第 15 章：图书进销存管理系统。主要介绍如何开发简单的图书进销存管理系统，具体应用到了 Struts 2.0 的拦截器、输入校验、标签库、异常处理、国际化等，包括了 Struts 2.0 的方方面面。本章首先从需求分析讲起，然后介绍系统设计、数据库设计等，最后是代码实现，把本书前面所讲的 Struts 2.0 的知识很好地结合了起来。

本书是介绍 Struts 2.0 框架的专业书籍，适合大专院校在校生、网站开发人员、职业技术培训人员以及编程爱好者学习和参考。

参加本书编写与制作的人员除封面署名者以外，还有祝红涛、郝春雨、王伟平、张水波、陈军红、王俊伟、李振、唐有明、赵俊昌、刘海松、朱璟煜、朱俊成、方宁、郑千忠、郭二鹏等。在此，编者对他们表示衷心的感谢。由于编写时间仓促，作者水平有限，书中难免会有错误和疏漏之处，恳请广大读者给予批评和指正。

为了方便读者阅读，本书配套资料请登录“华信教育资源网”（http://www.hxedu.com.cn），在“资源下载”频道的“图书资源”栏目下载。

目　录

第 1 章　Struts 2.0 学习必备

内容摘要

进入 Struts 世界固然令人兴奋，但是 Struts 并不是一种完全独立的技术，而是建立在其他 Web 技术之上的一个 MVC 框架，如果脱离了这些技术，Struts 框架也就无从谈起。本章是学习 Struts 的准备篇，将重点介绍如何学习 Struts，必须要掌握一些基本技术，如 JSP 和 Servlet 等。如果读者对这些技术已经能够熟练掌握，也可以跳过此章而直接进入第 2 章开始进一步的学习。

学习目标

- JSP 与 Servlet
- 应用 XML 技术
- 使用自定义标签
- MVC 框架

1.1　JSP/Servlet

JSP 技术可以让 Web 后台程序开发人员和前台设计人员快速地开发出容易维护的动态 Web 网站。使用 JSP 开发的 Web 应用程序是跨平台的，既可以在 Windows 操作系统上运行，也可以在其他操作系统上运行。JSP 技术是在 Servlet 技术的基础上形成的，并继承了 Java 语言的多种优势，如安全性、多线程和平台无关性等。

1.1.1　JSP/Servlet 技术介绍

JSP 技术是一种建立在 Servlet 规范提供的功能之上的动态网页技术。和 ASP、PHP 类似，用于产生动态内容。JSP 网页（*.jsp）就是在传统的网页 HTML 文件（*.htm 或*.html）中加入 Java 程序片段（Scriptlet）和 JSP 标记（Tag）而构成的。

在 Sun 公司正式发布 JSP（Java Server Pages）之后，这种新的 Web 应用开发技术很快引起了人们的关注。JSP 为创建高度动态的 Web 应用提供了一个独特的开发环境，使得开发者能够把页面的静态 HTML 和动态部分相分离。JSP 页面动态部分代码放入标记之内，即以“<%”开始，以“%>”结束。JSP 技术可以让 Web 开发人员和设计人员非常容易地创建和维护动态网页，特别是目前的商业系统。JSP 文件可以使用任何通常使用的编辑工具来编写，如记事本、Eclipse 和 NetBeans。

Web 服务器在遇到 JSP 网页的请求时，首先执行其中的程序片段，然后将执行结果以 HTML 网页格式返回给客户端。程序片段的功能可以是操作数据库或者重新定向网页、发送 E_mail 等，这些就是建立动态网站所需要的功能。所有程序操作都在服务器端执行，网络上传送给客

户端的仅是执行的结果，因此这样对浏览器的要求较低，可以实现无 Plugin，无 ActiveX，甚至无 Frame 的 Web 编程。而 JSP 在动态网页的建设方面有其强大而独特的功能。

Servlet 是 Java 技术 CGI（CGI 是 Common Gateway Interface 的缩写，是用于连接主页和应用程序的接口）编程其中的一种技术。与传统的 CGI 或者其他类似于 CGI 的技术相比，Servlet 具有更高的效率，更容易使用，功能更强大，而且具有更好的可移植性。Servlet 运行在 Servlet 容器中（如 Tomcat），能够自动产生 HTML 页面，是支持 Java 服务器的一般扩充。它最常见的用途是扩展 Web 服务器，提供安全的、可移植的、易于使用的 CGI 替代品。Servlet 可以动态加载程序模块，为来自 Web 服务器的请求提供服务，比如本书将要重点介绍的 Struts 2.0 框架正是通过 Servlet 来加载的。由于它在服务器端运行，因此它不依赖于浏览器的兼容性。

其实，可以将 Servlet 认为是一个基于 Java 技术的 Web 组件，它由 Servlet 容器（Tomcat 服务器等）来管理，用于生成动态的页面内容。Servlet 是平台独立的 Java 类，编写一个 Servlet，实际上就是按照 Servlet 规范编写了一个 Java 类。Servlet 执行时需要被编译为与平台无关的字节码，然后被动态地加载到 Java 技术的 Web 服务器中运行。

JSP 是一种简化 Servlet 开发的技术，因此，人们往往会形成一种误解——JSP 是一种比 Servlet 更加优秀，并可以完全替代 Servlet 的技术。首先必须承认，在实现页面逻辑与表示的分离方面，JSP 确实比 Servlet 优秀，另外，理论上 JSP 和 Servlet 可以相互替代；但是随着学习的深入及项目经验的增加，会发现它们各有优势和适应性。

JSP 与 Servlet 之间的主要差异在于，JSP 提供了一套简单的标签，和 HTML 融合得比较好，即使不了解 Servlet 的用户也可以通过 JSP 做出动态网页来。因此，很多对 Java 语言不太熟悉的用户，会觉得 JSP 开发比较方便。JSP 页面修改后可以立即看到修改后的效果，不需要手工编译（JSP 引擎会来做这些工作）。而 Servlet 则需要编译、重新启动 Servlet 引擎等一系列动作。JSP 很简单，可以方便地嵌入到 HTML 中，很容易加入动态内容，方便地输出 HTML。而在 Servlet 中输出 HTML 则需要调用特定的方法，相对于 JSP 来说比较复杂。从上面的比较可以看出，在表示层的实现上 JSP 相对于 Servlet 具有很大的优势。但是，如果要开发的 Web 应用有很复杂的控制逻辑需要实现，这时使用 Servlet 就可以非常清晰地封装这些控制逻辑。事实上，很多 Web 应用框架（如 Struts）就是采用 Servlet 来实现控制逻辑。

1.1.2 JSP 页面标记和内置对象

JSP 页面中嵌入的 Java 代码，称为 JSP 脚本程序。除了脚本程序外，还存在着其他页面元素。总的来说，页面元素可以分为 3 种类型：指令元素、脚本元素和动作元素等。下面对常用的指令元素与动作元素做简单的介绍。

指令元素主要为转换阶段提供整个 JSP 页面的相关信息，语法形式如下所示。

```
<%@ directive {attr="value"}*%>
```

在起始符号<%@之后和结束符号%>之前可以加入空格，也可以不加。但要注意的是在起始符号中的<和%之间、%和@之间、以及结束符号中的%和>之间不能有任何空格。指令元素中的常用指令如表 1-1 所示。

动作元素用于为请求处理阶段提供信息，它遵循 XML 的语法，可以是一个空标签，也可以有自己的属性。JSP 2.0 规范定义了一些标准的动作，这些动作就是一些标签，它们影响 JSP

运行时的行为和对客户端请求的响应，这些动作由 JSP 容器来实现。在 JSP 2.0 动作规范中定义了标准的动作元素，部分常用的动作元素如表 1-2 所示。

表 1-1　指令元素中的常用指令

指令	说明
Page	该指令作用于整个 JSP 页面，定义了许多与页面相关的属性，这些属性将被用于和 JSP 容器通信。语法格式为<%@ page attr1="value1" attr2="value2"...%>
include	该指令用于在 JSP 页面中静态包含一个文件，该文件可以是 JSP 页面、HTML 网页、文本文件或一段 Java 代码。使用了 include 指定的 JSP 页面在转换时，JSP 容器会在其中插入所包含文件的文本或代码。语法格式为<%@ include file="relativeURL"%>
taglib	该指令允许页面使用用户制定的标签。语法格式为<%@ taglib(uri="tagLibraryURI"ltagdir="tagDir")prefix="tagPrefix"%>
uri	该属性唯一地标识和前缀（prefix）相关的标签库描述符，可以是绝对或相对的 URI。这个 URI 被用于定位标签库描述符的位置
tagdir	该属性指示前缀（prefix）将被用于标识安装在/WEB-INF/tags/目录或其子目录下的标签文件
prefix	该属性用于定义一个 prefix:tagname 形式的字符串前缀，用于区分多个自定义标签。以 jsp:，jspx:，java:，javax:，servlet:，sun:和 sunw:开始的前缀被保留。前缀的命名必须遵循 XML 名称空间的命名约定。在 JSP 2.0 规范中，空前缀是非法的

表 1-2　常用的动作元素

元素	说明
<jsp:useBean>	该动作元素用于实例化 JavaBean，或者定位一个已经存在的 JavaBean 实例，并把实例的引用赋给一个变量。类似如下所示：<jsp:useBean id="conn" scope="session" class="com.itzcn.jack.ConnDB"/>。其中 id 为标识 JavaBean 实例的名字；scope 用于指定一个范围，可以取值：page、request、sesson 和 application，默认值是 page；class 用于指定 JavaBean 对象的完整的限定类名
<jsp:setProperty>	该动作元素与<jsp:useBean>一起使用，用来设置 JavaBean 的简单属性和索引属性，它使用 setXXX()方法，在 Bean 中设置一个或多个属性值。在 JSP 中，经常使用<jsp:setProperty>动作元素将客户端提交的数据保存到 JavaBean 的属性中
<jsp:getProperty>	该动作元素用来访问一个 Bean 的属性，并把属性的值转化成一个 Sting，然后发送到输出流中。如果属性是一个对象，将调用该对象的 toString（）方法。类似如下所示：<jsp:getProperty name="bean_name" property="propertyName"/>，其中，name 用于指定 Bean 实例的名称；property 用于指定要得到的属性的名称
<jsp:param>	该动作元素被用来以“名/值对”的形式为其他标签提供附加信息。它和<jsp:include>、<jsp:forward> 与 <jsp:plugin> 一起使用。类似如下所示：<jsp:param name="name" value="value"/>，其中，name 表示给出参数的名称；value 表示给出参数的值，可以是一个表达式
<jsp:include>	该动作元素用于在当前页面中包含静态和动态的资源，一旦被包含的页面执行完毕，请求处理将在调用页面中继续进行。被包含的页面不能改变响应的状态代码或设置报头，这防止了对类似 setCookie 这样方法的调用，任何对这些方法的调用都将被忽略。类似如下所示：<jsp:include page="urlSpec" flush="truelfalse"/>，其中，page 指定被包含资源的相对路径，该路径是相对于当前 JSP 页面的 URL；flush 属性是可选的。如果设置为 true，当页面输出使用了缓冲区，那么在进行包含工作之前，先要刷新缓冲区。如果设置为 false，则不会刷新缓冲区。该属性的默认值是 false
<jsp:forward>	这个动作允许在运行时将当前的请求转发给一个静态的资源、JSP 页面或者 Servlet，请求被转向到的资源必须位于同 JSP 发送请求相同的上下文环境中。使用格式类似如下所示：<jsp:forward page="relativeURL"/>，其中，page 用于指定请求被转向的资源的相对路径，该路径可以是相对于当前 JSP 页面的 URL，也可以是经过表达式计算得到的相对 URL
<jsp:plugin>	该动作用于产生与客户端浏览器相关的 HTML 标签（<OBJECT>或<EMBED>），从而可以在需要时下载 Java 插件（Plug-in）软件，并在插件中执行指定的 Applet 或 JavaBean。<jsp:plugin>标签将根据客户端浏览器的类型被替换为<object>或<embed>标签
<jsp:params>	它是<jsp:plugin>动作的一部分，并且只能在<jsp:plugin>动作中使用。该动作包含一个或多个<jsp:param>动作，用于向 Applet 或 JavaBean 提供参数

在 JSP 页面中存在很多重要的内置对象（也称为隐含对象），它由 JSP 容器自动为 JSP 页面提供，在编写 JSP 程序时，可以直接使用它们。这些对象使用户容易地收集通过浏览器请求发送的信息、响应浏览器及存储用户信息，从而提高了开发人员的工作效率。如表 1-3 所示为 JSP 常用的内置对象。

表 1-3　常用的内置对象

内置对象	说明

Request	request 是类 javax.servlet.HttpServletRequest 的一个对象。当客户端请求一个 JSP 页面时，JSP 容器会将客户端的请求信息包装在 request 对象中。请求信息的内容包括请求的头信息（Header）、系统信息（编码方式）、请求的方式（GET 或 POST）、请求的参数名称、参数值等。通常用得最多的是客户端请求的参数名称和参数值。可以通过 getParameter()得到参数值。使用格式类似如下所示。
	request.getParameter("parameterName")//得到参数 parameterName 的值
	Enumeration params = request.getParameterNames();//得到所有参数的名称

Response	response 对象是类 javax.servlet.http.HttpServletResponse 的一个对象。JSP 页面会根据客户端的请求建立一个默认的 response 对象。该对象常用到的方法是 sendRedirect()，使用此方法可以将当前客户端的请求转到其页面。使用格式如下所示：response.sendRedirect("URL");
Out	out 是类 javax.servlet.jsp.JspWriter 的一个对象，它能够把信息回送给客户端的浏览器。在 out 对象中，最常用的方法是 print()和 println()，它们区别在于，后者输出完它的内容后能够自动换行，而前者不可以。使用格式如下所示：out.print("IT 在中国 http://www.itzcn.com");

Session	session 是类 javax.servlet.http.HttpSession 的一个对象。session 指的是客户端与服务器端的一次会话，会话从客户连接到服务器开始，直到与服务器断开连接为止，这之间都可以访问 session 对象的属性和方法。该对象常用的方法是 setAttribiute()、getAttribute()和 removeAttribute()，分别可以对 session 中的对象进行存取和删除。使用类似如下所示。
	session.setAttribute("SiteInfo","IT 在中国 http://www.itzcn.om");
	string siteInfo = session.getAttribute("SiteInfo");
	session.removeAttribute("SiteInfo");

Cookie	该对象可以用来将少量的信息保存到浏览器中。由于 cookie 无须通过网络的传输，因而可以提高网页处理的效率，而且也能够减少服务器端的负载。另外，由于 cookie 作用在客户端的浏览器上，在使用时有些限制，所以应该注意使用的浏览器是否支持 Cookie。使用类似如下所示。
	Cookie myCookie=new Cookie("SiteName","IT 在中国");
	response.addCookie(myCookie);//将 Cookie 保存到客户端的计算机上
	myCookie.setMaxAge(100);//设置 myCookie 的有效期为创建后的 100 秒

Application	application 对象表示的是 Servlet 上下文环境，从 Servlet 的配置对象中获取。该配置对象属于 javax.servlet.ServletContext 接口，拥有 application 范围。当 Web 应用中的任一个 JSP 页面开始执行时，将产生一个 application 对象。只到服务器关闭时，application 对象才会被撤销。当网站不止一个 Web 应用，而且客户浏览不同 Web 应用的 JSP 页面时，将产生不同的 application 对象。在一个 Web 应用中的所有 JSP 页面，都将存取同一个 application 对象，即使浏览这些 JSP 页面的不是用一个客户。因此，保存于 application 对象的数据，不仅可以跨网页分享数据，更可以联机分享数据。使用类似如下所示。
	application.setAttribute("name","JackLong");
	string name=(string)application.getAttribute("name");

本书中所有实例环境均在 Windows XP 下，Tomcat 6+JDK 6。代码均为不包含修饰部分的精简代码，如果读者有兴趣可以自行修饰页面。

1.1.3　Servlet 常用接口

创建一个 Servlet 需要继承 Servlet 包提供的类或接口。在 Servlet 体系中，包含两个包，分别为 javax.servlet 包和 javax.servlet.http 包。其中 javax.servlet 包提供了控制 Servlet 生命周期所

必需的 Servlet 接口，定义了所有的 Servlet 类都必须实现或扩展的通用接口和类。javax.servlet.http 包提供了从 Servlet 接口派生出的专门用于处理 HTTP 请求的抽象类和一般的工具类。所有的 Servlet 都要实现 Servlet 接口，大多数情况下，是通过继承已经实现了 Servlet 接口的 javax.servlet.GenericServlet 和 javax.servlet.http.HttpServlet 这两个抽象类的子类，来间接实现 Servlet 接口。也可以这样去理解这两个包，javax.servlet 包中定义了实现 Servlet 程序必须要使用到的接口，javax.servlet.http 包定义了实现 Web 应用程序（Servlet）使用到的功能。

javax.servlet 包中常用的接口和类如表 1-4 所示。

表 1-4　javax.servlet 包中常用的接口和类

接口或类	说明
interface RequestDispatcher	定义一种对象，用于从客户接受请求，并将请求发送到服务器上任何指定的资源，如一个 Servlet、JSP 或 HTML 文件
interface Servlet	定义了所有 Servlet 必须实现的方法
interface ServletConfig	定义 Servlet config 对象，由 Servlet 引擎用在 Servlet 初始化时，向 Servlet 传递信息
interface ServletContext	定义了一系列方法，以便 Servlet 与其运行的环境通信
interface ServletRequest	定义了用于向 Servlet 传递客户请求信息的对象
interface ServletResponse	定义了一个对象，由 Servlet 用于向客户发送响应
interface SingleThreadModel	用于保证 Servlet 在任一时刻，只处理一个请求
class GenericServlet	继承 Servlet 接口，定义了一个通用的，与协议无关的 Servlet
class ServletInputStream	定义了一个输入流，用于由 Servlet 从中读取客户请求的二进制数据
class ServletOutputStream	定义了一个输出流，用于由 Servlet 向客户发送二进制数据
class ServletException	定义了一个当 Servlet 遇到问题时可以抛出的异常
class UnavailableException	定义了一种异常，用于由 Servlet 指明它永远或暂时不可用

在表 1-4 中，接口 Servlet 是 Servlet 体系的核心接口，所有 Servlet 类和程序都要继承该接口。GenericServlet 类是一个跨协议的 Servlet 类。如果要实现一个 Servlet 程序，无论采用何种协议该类必须要继承 GenericServlet 类。

javax.servlet.http 包中常用的接口和类如表 1-5 所示。

表 1-5　javax.servlet.http 包中常用的接口和类

类或接口	说明
interface HttpServletRequest	继承了 ServletRequest 接口，为 HTTPServlet 提供请求信息
interface HttpServletResponse	继承了 ServletResponse 接口，为 HTTPServlet 输出响应信息提供支持
interface HttpSession	为维护 HTTP 用户的会话状态提供支持
interface HttpSessionBindingListener	使得某对象在加入一个会话或从会话中删除时能够得到通知
interface HttpSessionContext	由 Servlet 2.1 定义，该对象在新版本已不被支持
class Cookie	用在 Servlet 中使用 Cookie 技术
class HttpServlet	定义了一个抽象类，继承 GenericServlet 抽象类，应被 HTTPServlet 继承
class HttpSessionBindingEvent	定义了一种对象，当某一个实现了 HttpSessionBindingListener 接口的对象被加入会话或从会//话中删除时，会收到该类对象的一个句柄
class HttpUtils	提供了一系列便于编写 HTTPServlet 的方法

在表 1-5 中，接口和类大部分都继承 javax.servlet 包中类和接口，都在 HTTP 协议下运行。其中接口 HttpServletRequest 和 HttpServletResponse 主要用来处理客户端请求和响应客户端请求。类 HttpServlet 一个抽象类，是所有 Servlet 程序都要继承的接口。

JSP 与 Servlet 本质上没有什么区别，一个 JSP 页面最终也要被转译为一个 Servlet。Servlet 可以实现与 JSP 同样的功能，下面创建一个 Servlet，MyServlet.java 代码如下。

```
package my.test;
import java.io.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet extends HttpServlet {
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    Date date=new Date();
    out.println("<br><br><center>"+date+"</center>");
}
public void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException {
    doGet(request,response);
}
}
```

将上述文件编译后的 class 文件放在 WEB-INF/classes 目录下，在 Web 应用的 web.xml 文件中配置 Servlet 如下。

```
    ..........
    <servlet>
        <servlet-name>testSverlet</servlet-name>
        <servlet-class>my.test.MyServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>testSverlet</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>
</web-app>
```

此 Servlet 的运行结果如图 1-1 所示。

配置文件中的<url-pattern>/servlet</url-pattern>决定了请求这个 Servlet 的 URL。配置中其

他元素的作用都很好理解，正如它们的名字，servlet-name 表示 Servlet 的名字，servlet-class 表示 Servlet 对应的类。

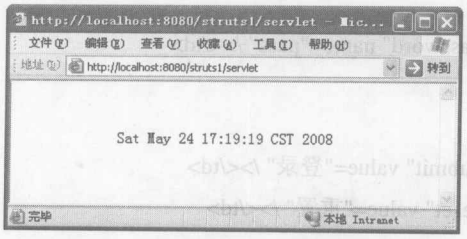

图 1-1　Servlet 的运行结果

如果把 Servlet、JSP、JavaBean（它用于处理业务逻辑，可以是一个普通的 Java 类）三者相结合，可以实现一个原始的 MVC 架构：JavaBean 用于业务处理模型（M），JSP 用于显示视图（V），Servlet 用于控制逻辑（C），这样作的好处是可以将视图与业务处理简单的分离（MVC 的优势将在本章的 1.4 节详细介绍）。

1.1.4　MVC 登录实例

使用 MVC 模式开发网站，可以成功的将数据表示、数据控制、数据模型三者分离开来。其中 JSP 页面负责显示数据，Servlet 负责控制数据流向，JavaBean 负责具体功能实现。现在以用户登录为例，演示 MVC 模式的构建和使用。如果登录成功则返回欢迎页面，否则返回错误页面。其中页面转向控制使用 Servlet 来实现，验证用户是否登录成功使用 JavaBean 实现，登录、欢迎、错误三个页面使用 JSP 页面来实现。

本实例需要实现三个视图（在此，视图要直接放置在 Web 站点目录下，比如笔者使用的是 struts1 站点），登录页面 login.jsp，效果如图 1-2 所示，代码如下。

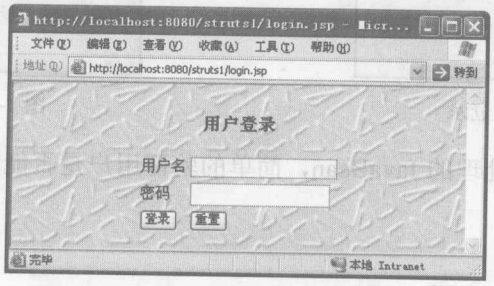

图 1-2　用户登录

```
<center>
  <h3>用户登录</h3>
  <form action="Login" method="post">
  <table>
    <tr>
      <td>用户名</td>
      <td><input type="textfield" name="name" /></td>
```

```
    </tr>
    <tr>
      <td>密码</td>
      <td><input type="password" name="pass" /></td>
    </tr>
    <tr>
      <td><input type="submit" value="登录" /></td>
      <td><input type="reset" value="重置" /></td>
    </tr>
  </table>
  </form>
</center>
```

登录成功页面 wellcome.jsp，运行效果如图 1-3 所示，代码如下。

```
<center>
  <b>恭喜您登录成功！</b>
</center>
```

登录失败页面 error.jsp，运行效果如图 1-4 所示，核心代码如下。

```
<center>
  <b>对不起登录失败！</b>
</center>
```

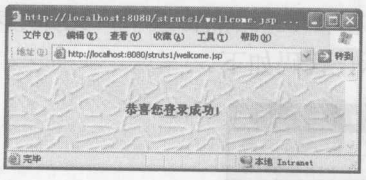

图 1-3　成功登录

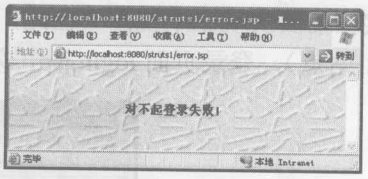

图 1-4　登录失败

下面是实现登录业务处理的 JavaBean，简单的验证用户是否能够成功登录。Check.java 代码如下。

```
package my.test;

public class Check{
  //如果验证成功则返回 true，不成功则返回 false
  public boolean check(String name,String pass){
    if("name".equals(name) && "pass".equals(pass))
    return true;
    return false;
  }
}
```