信息科学与技术丛书 程序设计系列

专家手记

AJAX开发实战

曲金龙　杨中科　主编

附光盘

- Eclipse 下 Web 开发环境的搭建
- 在 Struts 和 Struts 2 中使用 AJAX
- AJAX 框架 —— Echo2
- 功能全面的 WebOS 案例

机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书·程序设计系列

专家手记——AJAX开发实战

曲金龙　杨中科　主编
周　君　等编著

机械工业出版社

本书全面地介绍了AJAX技术。

全书分为两部分：第1部分着重介绍了Web开发的基础知识，包括Java下Web开发新概念、Eclipse下Web开发环境的搭建、基于Java EE技术的Web应用体系结构、基于Struts 2的Web开发、AJAX技术、XML技术、在struts和struts 2中使用AJAX及AJAX框架——Echo 2等内容；第2部分以一个实用的WebOS为例，讲解了基于AJAX技术的Web系统开发及应用。

随书光盘包含书中涉及的部分案例源代码。

本书适合有一定Java基础的开发人员阅读。

图书在版编目（CIP）数据

专家手记——AJAX开发实战/曲金龙，杨中科主编．—北京：机械工业出版社，2008.4
（信息科学与技术丛书·程序设计系列）
ISBN 978-7-111-23915-4

Ⅰ.专… Ⅱ.①曲… ②杨… Ⅲ.①主页制作－程序设计 ②计算机网络－程序设计 Ⅳ.TP393.09

中国版本图书馆CIP数据核字（2008）第051753号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
责任编辑：车 忱 加工编辑：唐洪昌
责任印制：李 妍

保定市中画美凯印刷有限公司印刷
2008年5月第1版·第1次印刷
184mm×260mm·28印张·690千字
0001—5000册
标准书号：ISBN 978-7-111-23915-4
ISBN 978-7-89482-641-1（光盘）
定价：52.00元（含1CD）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话（010）68326294
购书热线电话：（010）88379639 88379641 88379643
编辑热线电话：（010）88379753 88379739

出 版 说 明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术、新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书，来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术受到不断挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机在社会生活中日益普及，随着因特网延伸到人类世界的层层面面，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们对了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络、工程应用等内容，注重理论与实践相结合，内容实用，层次分明，语言流畅，是信息科学与技术领域专业人员不可或缺的图书。

现今，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设作出贡献。

机械工业出版社

前　言

随着 Web 开发技术的不断进步，AJAX 以一种新的面貌出现在我们的面前，并且迅速席卷全球，成为一种炙手可热的技术。越来越多的开发人员想快速地掌握这门技术，并把它快速应用到自己的项目中。

虽然 AJAX 技术已经非常成熟，但大多是以 CSS、JavaScript、XML 等单一形式出现的。本书将尝试着打破这种局面，在书中我们对 AJAX 的基础知识作有重点的介绍，并穿插了各种小的案例，让读者能够在实践中快速学习到 AJAX 技术；最后还安排了一个实用性非常强的 WebOS 应用，让读者能够学了就能懂，懂了立即用。

本书分两部分，内容包括：

第 1 部分着重介绍 Web 开发的基础知识。第 1 章介绍了 Java 平台下 AJAX 开发的相关技术；第 2 章介绍了搭建基于 Eclipse 的 Web 开发环境；第 3 章、第 4 章介绍了基于 Java EE 技术的 Web 开发的重难点知识；第 5 章对 AJAX 的核心技术 XMLHttpRequest、数据交换、跨域访问等难点问题进行了深入的剖析；第 6 章则对 AJAX 中数据交换的核心技术 XML 进行了通透的讲解；第 7 章讲解了如何在 Struts 中使用 AJAX 技术；第 8 章则用了大量篇幅对优秀的 AJAX 框架 Echo2 进行了详细的讲述。通过学习第 1 部分内容，读者就可以立即开始开发一个强大的 AJAX 应用。

第 2 部分以一个实用的 WebOS 为例讲解了基于 AJAX 技术的 Web 系统的开发。第 9 章介绍如何搭建系统的基础框架，包括异常处理、资源管理、上下文信息、基础组件等，并基于 Hibernate 以及 HSQLDB 技术搭建一个高度灵活的持久化层；第 10 章则基于第 9 章的基础框架开发了模拟 Windows 桌面、网络相册、爱墙系统、在线 Office、网络硬盘等经典的 Web 2.0 应用，充分展示了 AJAX 技术的强大。

本书特色：

- 突出重点和难点。本书并没有将所有的 AJAX 相关知识点进行罗列，而是有重点地对重点、难点进行解析。
- 注重实战。书中穿插了大量的实例，使读者能够学了就能用，还安排了一个实用性非常强的 WebOS 应用的开发案例，使读者能够与真实的案例项目直接面对面。
- 代码质量高。本书中所有的源代码都经过专业人员逐一验证通过，而且代码符合业界标准与规范，并且在源代码中体现了设计模式等高级开发技术。

本书由曲金龙、杨中科主编，参与编写的人员还有侯志松、周君、潘洪波、刘立业、王乐、初洪利。

本书在出版过程中得到了 CowNew 开源团队成员的大力支持，屈辰晨也为本书的出版做了大量工作，在此向他们表示深深的感谢！

如果您对本书有任何意见和建议，可以给我们发送邮件：about521@163.com，本书相关的后续资料将会发布到 CowNew 开源团队网站（http://www.cownew.com）中。

编　者

目　录

第2部分 开发案例

第1部分

AJAX基础

* Java 下 Web 开发新概念
* Eclipse 下 Web 开发环境的搭建
* 基于 Java EE 技术的 Web 应用体系结构
* 基于 Struts 2 的 Web 开发
* AJAX 技术
* XML 技术
* 在 Struts 和 Struts 2 中使用 AJAX
* AJAX 框架——Echo2

第1章 Java下Web开发新概念

当前，主流的计算机软件基本上已经转移到了B/S体系结构上。B/S体系结构的应用程序以其简单性、可移植性等诸多优点受到人们的青睐，并已被广泛接受。支持Web应用开发的程序语言也是多种多样，如CGI、ASP、PHP、Ruby、Java、.NET以及Python等。而Java是其中的热点，它无论是在大型的还是在中小型的Web应用程序领域都有足够好的表现，并以其优越的架构、优良的性能成为Web应用程序开发的首选语言之一。

1.1 Java下Web开发简介

随着Internet技术的发展，传统的基于客户端/服务器结构的胖客户端应用程序逐渐地转向了B/S体系结构的Web应用程序。Java也在这方面不断地努力，它在当时各种技术都不成熟，也没有直接可参考的应用方式的情况下推出了Java Servlet以及后来的JSP，这是Java进军Web应用领域所迈出的最重要的一步，也是Java史上的一个里程碑。

Java的Servlet/JSP技术是一项令人兴奋的技术，它使得用户编写服务器端的应用变得非常简单，并且还拥有Java天生的“一处编写、随处运行”的优势，可以将Web应用程序安装部署到任何一台支持Java的机器上，无须改动任何代码，从而实现真正的跨平台。正是由于Java将它的通用性、Servlet的速度和JSP的易用性巧妙地结合在一起，才使得它一步一步地走向Web技术开发的舞台，并迅速地在Web开发领域推广开来。随着Java Bean以及后来的Java EE、各种Servlet容器、EJB容器还有支持Web开发的各种各样的框架等技术的推出，Java在Web开发领域中越来越强大，也真正走向了成熟。

1. Servlet/JSP

Servlet是Java在Web领域中最重要的一项技术，它是客户端与服务器之间进行交互通信的接口。当客户端请求服务器时，服务器端的Servlet容器将客户端发送过来的请求包装成HttpServletRequest对象，然后将这个请求对象传递给指定的Servlet进行处理，最后将处理的结果包装成HttpServletResponse对象，由Servlet容器返回给客户端进行显示。这是Servlet处理一次请求/响应的过程。

实际上，在Java Web应用程序的开发过程中离不开Servlet技术，包括本书涉及的所有Java Web开发框架。有些框架可能不需要开发人员手动写Servlet类来接收客户端的请求，那是因为框架本身就封装了一个或几个Servlet来进行请求的处理。比如，著名的Struts框架，它就提供了一个Servlet-ActionServlet类来处理所有的客户端请求。尽管在开发基于Struts框架的Web应用程序的时候没有编写任何Servlet类，但实际上也是使用了Servlet技术的。所以要学习Java Web应用程序的开发，首先要学好Servlet的相关知识。本书在后面的章节中介绍了许多Servlet的相关知识以及使用与编写方法。在掌握Servlet的这些基本知识之后，建议大家阅读一下Servlet规范，弄清Servlet如何处理请求和返回响应，以及Servlet容器

如何管理 Servlet 的生命周期。只有在全面了解了 Servlet 技术内幕之后，我们才能在进行 Java Web 应用开发的过程中游刃有余地使用 Servlet。

Servlet 的生命周期是由 Servlet 容器来管理的，不是由开发人员控制的。开发人员不能构造一个 Servlet 或者销毁一个 Servlet。即使如此，开发人员仍然可以监视 Servlet 生命周期的变化，并进行相应的操作。Servlet 的生命周期大概可以分为 5 个阶段：加载、实例化、初始化、处理请求和销毁，如图 1-1 所示。其中，初始化、请求处理和销毁这 3 个阶段程序员是可以参与的。因为 Java 中的 Servlet 都必须直接或间接地继承自 javax.servlet.Servlet 这个接口。这个接口中定义了 3 个可以让开发人员参与 Servlet 的生命周期过程的方法，它们分别是 init()、service()和 destroy()。这 3 个方法分别对应 Servlet 生命周期过程的初始化、对请求的服务以及 Servlet 的销毁 3 个阶段。Servlet 的加载和实例化是由 Servlet 容器来管理的，可以在 Servlet 容器启动的时候进行，也可以在第一个请求到达的时候进行（一般的 Servlet 容器都是通过 Web 应用程序中的/WEB _ INF/web.xml 这个文件配置来决定加载哪些 Servlet）。Servlet 容器加载并实例化 Servlet 后会调用 Servlet 中的 init()方法，如果需要在 Servlet 处理客户端的请求之前进行某些资源的初始化工作就可以重写这个方法。比如，常见的初始化工作有读取配置文件、获取数据库连接等。Servlet 容器实例化一个 Servlet 类后只调用一次 init()方法，无论该 Servlet 对客户端的请求进行多少次处理，init()方法都不会再被调用，所以 init()方法比较适合用于初始化工作环境的场所。service()方法被 Servlet 容器调用用于处理客户端的请求，每当 Servlet 容器监听到客户端的请求到来时，就会调用对应 Servlet 的 service()方法，service()方法会根据请求中设置的所要调用的方法名（如 POST 和 GET 等）找到对应的方法（如 doPost()和 doGet()等），并调用该方法对请求进行相应的处理。这些方法再通过调用 Java Bean 或者其他的组件进行具体的业务处理，最后由 Servlet 生成响应返回给客户端。Servlet 引擎可以随时随意地释放 Servlet，但是它在释放回收 Servlet 的时候需要保证 Servlet 中引用的资源都被释放掉。因此，Java 提供了一个 destory()方法供开发人员在 Servlet 被销毁之前能够有机会释放掉 Servlet 中引用的相关资源。

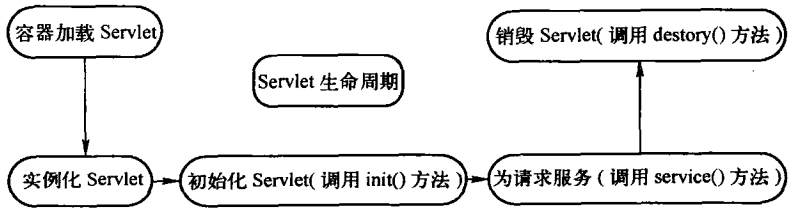

图 1-1 Servlet 的生命周期

Servlet 虽然有速度快等诸多优点，但是由于它没有为处理 HTML 代码提供很好的解决方案，所以使用纯 Servlet 编写 Web 应用程序也是一件繁琐的事情。例如，下面的一段代码就是使用 Servlet 返回响应给客户端的一个简单示例。从这段代码片段中我们可以看出，由于在 Java 代码中插入了大量的 HTML 代码字符串而使程序显得不美观，而且这样的代码也失去了 HTML 代码本身的结构性，对编写、调试以及维护 HTML 页面都造成了极大的困难。

【HelloWorldServlet 代码】

```
public class HelloWorldServlet extends HttpServlet
{
  /*
   * (非 Javadoc)
   *
   * @see javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
   *      javax.servlet.http.HttpServletResponse)
   */
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
  {
    ServletOutputStream out = response.getOutputStream();

    out.println("<! DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//
EN\">");

    out.println("<html>");
    out.println("  <head>");
    out.println("    <title>My First Servlet</title>");
    out.println("  </head>");
    out.println();
    out.println("  <body>");
    out.println("    This is my first Servlet. ");
    out.println("  </body>");
    out.println("</html>");
  }
}
```

正是由于 Servlet 在应付 HTML 的时候表现得非常无助，所以就有了 JSP 的产生。JSP 的本质就是 Servlet，根据“不要重复发明轮子”的理论，直接使用已有的 Servlet 功能，并补充完善它，这才是最好的选择。那么 JSP 又在什么地方表现出它的本质是一个 Servlet 呢？关于这一点有一个最直接的方法，就是查看 Servlet 容器中由 JSP 编译后的文件。例如，创建一个非常简单的 Web 应用，命名为 helloworld，该应用程序中只有一个 index.jsp 文件，该文件的内容如下：

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

<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'index.jsp' starting page</title>
  </head>
```

```
    <body>
      This is my JSP page. <br>
    </body>
</html>
```

当我们将其部署到 Tomcat 容器中，然后在启动 Tomcat 服务器之后，输入网址 http://localhost:8080/helloworld/index.jsp，这时 Tomcat 就会解析 index.jsp 文件，将其重新编译为一个 Java 文件，然后再编译成一个 class 文件供服务器使用。在 helloworld 的例子中，index.jsp 编译后的 Java 文件存储于 Tomcat 安装目录下的一个名称为 work 的目录里，具体为：$TOMCAT_HOME/work/Catalina/localhost/helloworld/org/apache/jsp/index_jsp.java，并在同一目录下存在一个 index_jsp.class 文件。下面就是 index.jsp 被编译后的文件，先来看看它的具体内容，看看为什么说它是一个 Servlet。

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import java.util.*;

public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

  private static java.util.List _jspx_dependants;

  public Object getDependants() {
    return _jspx_dependants;
  }

  public void _jspService(HttpServletRequest request, HttpServletResponse response)
      throws java.io.IOException, ServletException {

    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;
```

```
        try {
          _jspxFactory = JspFactory.getDefaultFactory();
          response.setContentType("text/html;charset=UTF-8");
          pageContext = _jspxFactory.getPageContext(this, request, response,
                      null, true, 8192, true);
          _jspx_page_context = pageContext;
          application = pageContext.getServletContext();
          config = pageContext.getServletConfig();
          session = pageContext.getSession();
          out = pageContext.getOut();
          _jspx_out = out;

          out.write("\r\n");
          out.write("\r\n");
          out.write("<! DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//
EN\">\r\n");
          out.write("<html>\r\n");
          out.write("  <head>\r\n");
          out.write("    <title>My JSP 'index.jsp' starting page</title>\r\n");
          out.write("  </head>\r\n");
          out.write("\r\n");
          out.write("  <body>\r\n");
          out.write("    This is my JSP page. <br>\r\n");
          out.write("  </body>\r\n");
          out.write("</html>\r\n");
        } catch (Throwable t) {
          if (! (t instanceof SkipPageException)){
            out = _jspx_out;
            if (out ! = null && out.getBufferSize() ! = 0)
              out.clearBuffer();
            if (_jspx_page_context ! = null) _jspx_page_context.handlePageException(t);
          }
        } finally {
          if (_jspxFactory ! = null) _jspxFactory.releasePageContext(_jspx_page_context);
        }
      }
    }
```

根据上面提供的代码没有发现 index_jsp 类中有任何一点关于 Servlet 的信息，这是因为 index_jsp 类继承自 org.apache.jasper.runtime.HttpJspBase。因此我们可以下载 Tomcat 的源代码，然后查看解压后的源代码文件夹下 jasper/src/share/org/apache/jasper/runtime 目录中