即用即查——JSP 函数与对象参考手册

丛书特点：直击核心技术 解惑网络开发

- 案头必备、内容实用
- 排版精准、图文并茂
- 光盘中提供了索引，方便快速查找

书名：即用即查——JSP 函数与对象参考手册
作者：孔鹏　编著
书号：978-7-115-16111-6
定价：62
开本：185×260　1/16
印张：35.75
总页码：570
印刷色别：单
有无光盘：1CD
选题类别：网络开发
预计出版时间：2007 年 5 月

本书讲述：

- 41 个指令属性
- 50 个动作属性
- 9 大对象 268 个方法
- 6 大标记库 274 个属性和元素
- 181 个 Servlet 方法
- 134 个 Filter 和 Listener 方法
- 229 个数据库操作方法
- 509 个实例源代码

内容涵盖：

- 指令（Directive）元素：16 个实例
- 动作（Action）元素：26 个实例
- 脚本（Scripting）元素：3 个实例
- 隐式对象（Implicit Object）：95 个实例
- 核心标记库（Core Tag Library）：38 个实例
- 国际化格式标记库：45 个实例
- SQL 标记库（SQL Tag Library）：15 个实例
- XML 标记库（XML Tag Library）：27 个实例
- 函数标记库（Functions Tag Library）：16 个实例
- 定制标记库（Tag Library）：14 个实例
- 表达式语言（Expression Language）：8 个实例
- Servlet：68 个实例
- 过滤器（Filter）：88 个实例
- 监听者（Listener）：8 个实例
- JavaBean 与开发模型：9 个实例
- Java 数据库连接（JDBC）：32 个实例

技术深度：（初、中）

目标受众：

- ☐ JSP 程序员
- ☐ Web 开发人员
- ☐ 网页设计人员
- ☐ 大中院校的学生
- ☐ Java 程序员

内容提要：

本书是一本 JSP 函数与对象参考手册，涵盖了绝大多数标签、对象，全书采用实例进行讲解，旨在能够指导开发实践。

本书以 JSP 2.0 规范为依据，全面介绍了使用 JSP 语言进行动态 Web 应用开发所涉及的函数与对象。全书首先介绍了 JSP 运行与开发环境，JSP 的基本语法；然后介绍 JSP 标准标记库、定制标记库、表达式语言等内容；接着介绍了 Servlet 技术，涵盖了 Servlet、过滤器、监听者、JavaBean 以及 JSP 开发模型等；最后介绍了数据库访问技术，包括 SQL 语言、JDBC 驱动程序、JDBC 访问 MySQL、数据访问对象等内容。本书包含大量示例代码，力求精练、实用。

本书适用于使用 JSP 技术进行 Web 应用设计的开发人员阅读，可以作为 JSP 开发人员参考用书。

目录：

样章：

目　录

第 1 篇　JSP 基础技术篇

第 1 章　建立 JSP 环境……3
1.1　JSP 环境需求……3
1.1.1　操作系统……3
1.1.2　软件需求……3
1.2　Java 软件开发工具包……3
1.2.1　安装 J2SDK……4
1.2.2　配置 J2SDK……5
1.2.3　测试 J2SDK……6
1.3　Tomcat 服务器……7
1.3.1　安装 Tomcat……7
1.3.2　测试 Tomcat……9
1.3.3　配置 Tomcat……9
1.4　Eclipse 开发环境……10
1.4.1　安装 Eclipse……10
1.4.2　安装 tomcatPlugin……11
1.4.3　启动和停止 Tomcat……12
1.4.4　第一个 JSP 页面……12

第 2 章　指令（Directive）元素……14
2.1　page 指令……14
2.1.1　language 属性：指定脚本语言……14
2.1.2　extends 属性：指定扩展类名……15
2.1.3　import 属性：声明导入库……15
2.1.4　session 属性：指定是否可用 session 对象……16
2.1.5　buffer 属性：指定缓冲区大小……16
2.1.6　autoFlush 属性：指定是否自动刷新输出……17
2.1.7　isThreadSafe 属性：指定是否线程安全……17
2.1.8　info 属性：指定信息文本……17
2.1.9　errorPage 属性：指定错误页面……17
2.1.10　isErrorPage 属性：指定是否为错误页面……18
2.1.11　contentType 属性：指定内容类型……18
2.1.12　pageEncoding 属性：指定页面编码方式……19
2.1.13　isELIgnored 属性：指定是否忽略 EL……19
2.2　include 指令……19
2.3　taglib 指令……20
2.3.1　prefix 属性：指定标记前缀……20
2.3.2　tagdir 属性：指定包含标记文件的目录……20
2.3.3　uri 属性：指定标记库相对路径……21
2.4　attribute 指令……21
2.4.1　name 属性：指定属性名……22

2.4.2 required 属性：指定属性值是否必需……22
2.4.3 fragment 属性：指定为片段……22
2.4.4 rtexprvalue 属性：指定是否可由表达式设置……23
2.4.5 type 属性：指定属性值的类型……24
2.4.6 description 属性：指定描述信息……24
2.5 tag 指令……24
2.5.1 display-name 属性：指定显示名称……24
2.5.2 body-content 属性：指定本体内容类型……24
2.5.3 dynamic-attributes 属性：保存未声明属性的变量名……25
2.5.4 small-icon 属性：指定小图标……25
2.5.5 large-icon 属性：指定大图标……26
2.5.6 description 属性：指定描述信息……26
2.5.7 example 属性：指定一个范例……26
2.5.8 language 属性：指定脚本语言……26
2.5.9 import 属性：声明导入库……26
2.5.10 pageEncoding 属性：指定文件编码方式……26
2.5.11 isELIgnored 属性：指定是否忽略 EL……27
2.6 variable 指令……27
2.6.1 name-given 属性：指定变量名……27
2.6.2 name-from-attribute 属性：指定属性名……28
2.6.3 alias 属性：指定变量别名……28
2.6.4 variable-class 属性：指定变量的类型……29
2.6.5 declare 属性：指定是否创建变量声明……29
2.6.6 description 属性：指定描述信息……29
2.6.7 scope 属性：指定变量作用域……29

第 3 章 动作（Action）元素……30
3.1 <jsp:useBean>动作……30
3.1.1 id 属性：指定 bean 的名称……30
3.1.2 class 属性：指定 bean 的类名……30
3.1.3 type 属性：指定 bean 的类型名称……31
3.1.4 beanName 属性：指定 bean 的名字……31
3.1.5 scope 属性：指定 bean 的作用域……31
3.2 <jsp:setProperty>动作……32
3.2.1 name 属性：指定 bean 的名称……32
3.2.2 property 属性：指定 bean 的属性名……32
3.2.3 param 属性：指定请求参数名……32
3.2.4 value 属性：指定属性的值……33
3.3 <jsp:getProperty>动作……33
3.3.1 name 属性：指定 bean 的名称……34
3.3.2 property 属性：指定 bean 的属性名……34
3.4 <jsp:include>动作……34
3.4.1 page 属性：指定资源相对路径……35
3.4.2 flush 属性：指定是否刷新缓冲区……35

3.4.3 <jsp:param>元素：提供参数信息……36
3.5 <jsp:forward>动作……36
3.5.1 page 属性：指定资源相对路径……37
3.5.2 <jsp:param>元素：提供参数信息……37
3.6 <jsp:param>动作……38
3.6.1 name 属性：指定参数名……38
3.6.2 value 属性：指定参数值……38
3.7 <jsp:plugin>动作……38
3.7.1 type 属性：指定对象类型……38
3.7.2 code 属性：指定 applet 类名……39
3.7.3 codebase 属性：指定 applet 类相对路径……39
3.7.4 name 属性：指定对象名……39
3.7.5 archive 属性：指定 applet 归档文件……40
3.7.6 align 属性：指定对齐方式……40
3.7.7 height 属性：指定 applet 的高度……40
3.7.8 width 属性：指定 applet 的宽度……40
3.7.9 hspace 属性：指定区域左右所留空间……41
3.7.10 vspace 属性：指定区域上下所留空间……41
3.7.11 jreversion 属性：指定 JRE 版本……42
3.7.12 nspluginurl 属性：指定 Netscape 插件 URL……42
3.7.13 iepluginurl 属性：指定 IE 插件 URL……42
3.7.14 <jsp:params>元素：提供参数信息……42
3.7.15 <jsp:fallback>元素：提供替代文本……43
3.8 <jsp:params>动作……43
3.9 <jsp:fallback>动作……43
3.10 <jsp:root>动作……44
3.10.1 version 属性：指定 JSP 版本……44
3.10.2 xmlns:jsp 属性：指定使用 XML 名字空间……44
3.10.3 xmlns:prefix 属性：指定标记库……45
3.11 <jsp:declaration>动作……45
3.12 <jsp:scriptlet>动作……46
3.13 <jsp:expression>动作……47
3.14 <jsp:text>动作……47
3.15 <jsp:output>动作……48
3.15.1 omit-xml-declaration 属性：指定是否忽略 XML 声明……48
3.15.2 doctype-root-element 属性：指定文档根元素名……49
3.15.3 doctype-system 属性：指定文档系统标识……49
3.15.4 doctype-public 属性：指定文档公共标识……49
3.16 <jsp:attribute>动作……50
3.16.1 name 属性：指定属性名……50
3.16.2 trim 属性：指定是否忽略空白……50
3.17 <jsp:body>动作……50
3.18 <jsp:element>动作……51
3.19 <jsp:invoke>动作……52

3.19.1 fragment 属性：指定片段名 ····52
3.19.2 var 属性：指定 String 类型的变量 ····53
3.19.3 varReader 属性：指定 Reader 类型的变量 ····53
3.19.4 scope 属性：指定变量的作用域 ····53
3.20 <jsp:doBody>动作 ····53
3.20.1 var 属性：指定 String 类型的变量 ····53
3.20.2 varReader 属性：指定 Reader 类型的变量 ····54
3.20.3 scope 属性：指定变量的作用域 ····54

第 4 章 脚本（Script）元素 ····55
4.1 声明（Declaration） ····55
4.2 表达式（Expression） ····56
4.3 脚本段（Scriptlet） ····57

第 5 章 隐式对象（Implicit Object） ····59
5.1 application 对象 ····59
5.1.1 getAttribute 方法：获取属性值 ····59
5.1.2 getAttributeNames 方法：获取所有的属性名 ····60
5.1.3 getContext 方法：获取 ServletContext 对象 ····60
5.1.4 getInitParameter 方法：获取初始化参数 ····60
5.1.5 getInitParameterNames 方法：获取所有的初始化参数名 ····61
5.1.6 getMajorVersion 方法：获取主版本号 ····61
5.1.7 getMinorVersion 方法：获取副版本号 ····62
5.1.8 getMimeType 方法：获取 MIME 类型 ····62
5.1.9 getNamedDispatcher 方法：获取 RequestDispatcher 对象 ····62
5.1.10 getRealPath 方法：获取物理路径 ····63
5.1.11 getRequestDispatcher 方法：获取 RequestDispatcher 对象 ····63
5.1.12 getResource 方法：获取资源路径 ····64
5.1.13 getResourceAsStream 方法：获取资源的 InputStream 对象 ····64
5.1.14 getResourcePaths 方法：获取所有的资源路径 ····65
5.1.15 getServerInfo 方法：获取服务器信息 ····65
5.1.16 getServlet 方法：获取 Servlet ····66
5.1.17 getServletContextName 方法：获取 Servlet 上下文名称 ····66
5.1.18 getServletNames 方法：获取所有的 Servlet 名称 ····66
5.1.19 getServlets 方法：获取所有 Servlet ····67
5.1.20 log 方法：信息写入日志 ····67
5.1.21 removeAttribute 方法：移除属性 ····67
5.1.22 setAttribute 方法：设置属性值 ····68
5.2 config 对象 ····68
5.2.1 getInitParameter 方法：获取初始化参数 ····68
5.2.2 getInitParameterNames 方法：获取所有的初始化参数名 ····69
5.2.3 getServletContext 方法：获取 ServletContext 对象 ····69
5.2.4 getServletName 方法：获取 Servlet 名称 ····70
5.3 exception 对象 ····70

5.3.1 fillInStackTrace 方法：填充堆栈跟踪信息 ····70
5.3.2 getCause 方法：获取异常 ····71
5.3.3 getLocalizedMessage 方法：获取本地化错误信息 ····71
5.3.4 getMessage 方法：获取错误信息 ····72
5.3.5 getStackTrace 方法：获取堆栈跟踪信息 ····72
5.3.6 initCause 方法：初始化异常对象 ····72
5.3.7 printStackTrace 方法：打印堆栈跟踪信息 ····73
5.3.8 setStackTrace 方法：设置堆栈跟踪信息 ····73
5.3.9 toString 方法：转换成字符串 ····74
5.4 out 对象 ····74
5.4.1 DEFAULT_BUFFER 字段：默认缓冲区大小 ····74
5.4.2 NO_BUFFER 字段：不缓冲输出 ····75
5.4.3 UNBOUNDED_BUFFER 字段：不限制缓冲区大小 ····75
5.4.4 clear 方法：清空缓冲区 ····75
5.4.5 clearBuffer 方法：清空缓冲区 ····75
5.4.6 close 方法：关闭输出流 ····76
5.4.7 flush 方法：刷新缓冲区输出 ····76
5.4.8 getBufferSize 方法：获取缓冲区大小 ····76
5.4.9 getRemaining 方法：获取剩余大小 ····76
5.4.10 isAutoFlush 方法：指示是否自动刷新 ····77
5.4.11 newLine 方法：输出换行符 ····77
5.4.12 print 方法：输出数据 ····77
5.4.13 println 方法：输出数据并换行 ····78
5.5 page 对象 ····79
5.6 pageContext 对象 ····80
5.6.1 APPLICATION 字段：ServletContext 的名称 ····80
5.6.2 APPLICATION_SCOPE 字段：应用作用域 ····80
5.6.3 CONFIG 字段：ServletConfig 的名称 ····80
5.6.4 EXCEPTION 字段：Exception 的名称 ····80
5.6.5 OUT 字段：JspWriter 的名称 ····80
5.6.6 PAGE 字段：Servlet 的名称 ····81
5.6.7 PAGE_SCOPE 字段：页面作用域 ····81
5.6.8 PAGECONTEXT 字段：PageContext 的名称 ····81
5.6.9 REQUEST 字段：ServletRequest 的名称 ····81
5.6.10 REQUEST_SCOPE 字段：请求作用域 ····81
5.6.11 RESPONSE 字段：ServletResponse 的名称 ····81
5.6.12 SESSION 字段：HttpSession 的名称 ····82
5.6.13 SESSION_SCOPE 字段：会话作用域 ····82
5.6.14 findAttribute 方法：查找属性 ····82
5.6.15 forward 方法：转发请求 ····82
5.6.16 getAttribute 方法：获取属性值 ····83
5.6.17 getAttributeNamesInScope 方法：在作用域中获取所有属性名 ····83
5.6.18 getAttributesScope 方法：获取属性作用域 ····84
5.6.19 getErrorData 方法：返回 ErrorData 对象 ····84

5.6.20 getException 方法：获取异常对象……85
5.6.21 getExpressionEvaluator 方法：获取 expressionEvaluator 对象……85
5.6.22 getOut 方法：获取 JspWriter 对象……85
5.6.23 getPage 方法：获取 page 对象……86
5.6.24 getRequest 方法：获取 ServletRequest 对象……86
5.6.25 getResponse 方法：获取 ServletResponse 对象……86
5.6.26 getServletConfig 方法：获取 ServletConfig 对象……86
5.6.27 getServletContext 方法：获取 ServletContext 对象……87
5.6.28 getSession 方法：获取 Session 对象……87
5.6.29 getVariableResolver 方法：获取 VariableResolver 对象……87
5.6.30 handlePageException 方法：处理页面异常……87
5.6.31 include 方法：包含其他资源……88
5.6.32 initialize 方法：初始化 pageContext 对象……89
5.6.33 popBody 方法：out 对象出栈……89
5.6.34 pushBody 方法：out 对象入栈……89
5.6.35 release 方法：释放状态……90
5.6.36 removeAttribute 方法：移除属性……90
5.6.37 setAttribute 方法：设置属性……90
5.7 request 对象……91
5.7.1 BASIC_AUTH 字段：基本认证……92
5.7.2 CLIENT_CERT_AUTH 字段：客户端证书认证……92
5.7.3 DIGEST_AUTH 字段：摘要认证……92
5.7.4 FORM_AUTH 字段：表格认证……92
5.7.5 getAttribute 方法：获取属性……92
5.7.6 getAttributeNames 方法：获取所有的属性名……93
5.7.7 getAuthType 方法：获取认证类型……93
5.7.8 getCharacterEncoding 方法：获取字符编码方式……94
5.7.9 getContentLength 方法：获取内容长度……94
5.7.10 getContentType 方法：获取内容类型……95
5.7.11 getContextPath 方法：获取上下文路径……95
5.7.12 getCookies 方法：获取 Cookie 对象数组……95
5.7.13 getDateHeader 方法：获取日期值头部……95
5.7.14 getHeader 方法：获取头部信息……96
5.7.15 getHeaderNames 方法：获取所有的头部名……96
5.7.16 getHeaders 方法：获取头部所有的值……97
5.7.17 getInputStream 方法：获取输入流……97
5.7.18 getIntHeader 方法：获取整型值头部……97
5.7.19 getLocalAddr 方法：获取服务器地址……98
5.7.20 getLocale 方法：获取区域对象……98
5.7.21 getLocales 方法：获取所有的区域对象……98
5.7.22 getLocalName 方法：获取服务器主机名……98
5.7.23 getLocalPort 方法：获取服务器端口……99
5.7.24 getMethod 方法：获取请求方式……99
5.7.25 getParameter 方法：获取请求参数……99

5.7.26 getParameterMap 方法：获取参数映射 ····· 99
5.7.27 getParameterNames 方法：获取所有的参数名 ····· 100
5.7.28 getParameterValues 方法：获取参数所有的值 ····· 101
5.7.29 getPathInfo 方法：获取路径信息 ····· 101
5.7.30 getPathTranslated 方法：获取翻译后的路径 ····· 101
5.7.31 getProtocol 方法：获取协议信息 ····· 102
5.7.32 getQueryString 方法：获取查询字符串 ····· 102
5.7.33 getReader 方法：获取 Reader 对象 ····· 102
5.7.34 getRealPath 方法：获取物理路径 ····· 103
5.7.35 getRemoteAddr 方法：获取客户端地址 ····· 103
5.7.36 getRemoteHost 方法：获取客户端主机名 ····· 103
5.7.37 getRemotePort 方法：获取客户端端口 ····· 103
5.7.38 getRemoteUser 方法：获取客户端用户名 ····· 103
5.7.39 getRequestDispatcher 方法：获取 RequestDispathcer 对象 ····· 104
5.7.40 getRequestedSessionId 方法：获取请求的会话标识 ····· 104
5.7.41 getRequestURI 方法：获取发送请求的 URI ····· 105
5.7.42 getRequestURL 方法：获取响应请求的 URL ····· 105
5.7.43 getScheme 方法：获取协议名 ····· 105
5.7.44 getServerName 方法：获取服务器名称 ····· 105
5.7.45 getServerPort 方法：获取服务器端口 ····· 106
5.7.46 getServletPath 方法：获取 Servlet 路径 ····· 106
5.7.47 getSession 方法：获取 HttpSession 对象 ····· 106
5.7.48 getUserPrincipal 方法：获取认证用户的 Principal 对象 ····· 107
5.7.49 isRequestedSessionIdFromCookie 方法：判断会话 ID 是否来自 Cookie ····· 107
5.7.50 isRequestedSessionIdFromURL 方法：判断会话 ID 是否来自 URL ····· 107
5.7.51 isRequestedSessionIdValid 方法：判断会话 ID 是否有效 ····· 107
5.7.52 isSecure 方法：判断是否使用安全链接 ····· 108
5.7.53 isUserInRole 方法：判断认证用户是否属于指定角色 ····· 108
5.7.54 removeAttribute 方法：移除属性 ····· 108
5.7.55 setAttribute 方法：设置属性 ····· 108
5.7.56 setCharacterEncoding 方法：设置字符编码方式 ····· 109
5.8 response 对象 ····· 109
5.8.1 SC_ACCEPTED 字段：接受 ····· 109
5.8.2 SC_BAD_GATEWAY 字段：错误的网关 ····· 109
5.8.3 SC_BAD_REQUEST 字段：错误请求 ····· 109
5.8.4 SC_CONFLICT 字段：冲突 ····· 110
5.8.5 SC_CONTINUE 字段：继续 ····· 110
5.8.6 SC_CREATED 字段：已创建 ····· 110
5.8.7 SC_EXPECTATION_FAILED 字段：期望失败 ····· 110
5.8.8 SC_FORBIDDEN 字段：禁止 ····· 110
5.8.9 SC_FOUND 字段：找到 ····· 111
5.8.10 SC_GATEWAY_TIMEOUT 字段：网关超时 ····· 111
5.8.11 SC_GONE 字段：已不存在 ····· 111
5.8.12 SC_HTTP_VERSION_NOT_SUPPORTED 字段：不支持的 HTTP 版本 ····· 111

5.8.13 SC_INTERNAL_SERVER_ERROR 字段：内部服务器错误 111
5.8.14 SC_LENGTH_REQUIRED 字段：需要数据长度 112
5.8.15 SC_METHOD_NOT_ALLOWED 字段：方法未允许 112
5.8.16 SC_MOVED_PERMANENTLY 字段：永久性移动 112
5.8.17 SC_MOVED_TEMPORARILY 字段：临时性移动 112
5.8.18 SC_MULTIPLE_CHOICES 字段：多重选择 112
5.8.19 SC_NO_CONTENT 字段：无内容 113
5.8.20 SC_NON_AUTHORITATIVE_INFORMATION 字段：非官方信息 113
5.8.21 SC_NOT_ACCEPTABLE 字段：无法访问 113
5.8.22 SC_NOT_FOUND 字段：未找到 113
5.8.23 SC_NOT_IMPLEMENTED 字段：未实现 114
5.8.24 SC_NOT_MODIFIED 字段：未修正 114
5.8.25 SC_OK 字段：正常 114
5.8.26 SC_PARTIAL_CONTENT 字段：局部内容 114
5.8.27 SC_PAYMENT_REQUIRED 字段：保留 114
5.8.28 SC_PRECONDITION_FAILED 字段：先决条件失败 115
5.8.29 SC_PROXY_AUTHENTICATION_REQUIRED 字段：需代理服务器认证 115
5.8.30 SC_REQUEST_ENTITY_TOO_LARGE 字段：请求实体过大 115
5.8.31 SC_REQUEST_TIMEOUT 字段：请求超时 115
5.8.32 SC_REQUEST_URI_TOO_LONG 字段：请求 URI 过长 115
5.8.33 SC_REQUESTED_RANGE_NOT_SATISFIABLE：请求范围无法满足 116
5.8.34 SC_RESET_CONTENT 字段：重置内容 116
5.8.35 SC_SEE_OTHER 字段：参见其他信息 116
5.8.36 SC_SERVICE_UNAVAILABLE 字段：服务无法获得 116
5.8.37 SC_SWITCHING_PROTOCOLS 字段：转换协议 117
5.8.38 SC_TEMPORARY_REDIRECT 字段：临时重定向 117
5.8.39 SC_UNAUTHORIZED 字段：未授权 117
5.8.40 SC_UNSUPPORTED_MEDIA_TYPE 字段：不支持的媒体格式 117
5.8.41 SC_USE_PROXY 字段：使用代理 117
5.8.42 addCookie 方法：添加 Cookie 对象 118
5.8.43 addDateHeader 方法：添加日期值头部 118
5.8.44 addHeader 方法：添加字符串值头部 118
5.8.45 addIntHeader 方法：添加整数值头部 119
5.8.46 containsHeader 方法：判断头部是否设置 119
5.8.47 encodeRedirectURL 方法：对指定 URL 编码 119
5.8.48 encodeURL 方法：对指定 URL 编码 119
5.8.49 flushBuffer 方法：刷新缓冲区 120
5.8.50 getBufferSize 方法：获取缓冲区大小 120
5.8.51 getCharacterEncoding 方法：获取字符编码方式 121
5.8.52 getContentType 方法：获取内容类型 121
5.8.53 getLocale 方法：获取区域对象 121
5.8.54 getOutputStream 方法：获取 ServletOutputStream 对象 121
5.8.55 getWriter 方法：获取 PrintWriter 对象 122
5.8.56 isCommitted 方法：判断是否已提交 122

5.8.57 reset 方法：清空缓冲区 ····· 122
5.8.58 resetBuffer 方法：清空缓冲区 ····· 122
5.8.59 sendError 方法：发送错误响应 ····· 123
5.8.60 sendRedirect 方法：发送重定向响应 ····· 123
5.8.61 setBufferSize 方法：设置缓冲区大小 ····· 124
5.8.62 setCharacterEncoding 方法：设置字符编码方式 ····· 124
5.8.63 setContentLength 方法：设置内容长度 ····· 124
5.8.64 setContentType 方法：设置内容类型 ····· 125
5.8.65 setDateHeader 方法：设置日期值头部 ····· 125
5.8.66 setHeader 方法：设置字符串值头部 ····· 125
5.8.67 setIntHeader 方法：设置整数值头部 ····· 126
5.8.68 setLocale 方法：设置区域对象 ····· 126
5.8.69 setStatus 方法：设置状态码 ····· 126
5.9 session 对象 ····· 127
5.9.1 getAttribute 方法：获取绑定对象 ····· 127
5.9.2 getAttributeNames 方法：获取所有的对象名 ····· 127
5.9.3 getCreationTime 方法：获取会话创建时间 ····· 127
5.9.4 getId 方法：获取会话标识 ····· 128
5.9.5 getLastAccessedTime 方法：获取上一次访问时间 ····· 128
5.9.6 getMaxInactiveInterval 方法：获取会话有效时间 ····· 128
5.9.7 getServletContext 方法：获取 ServletContext 对象 ····· 129
5.9.8 getSessionContext 方法：获取会话上下文 ····· 129
5.9.9 getValue 方法：获取属性值 ····· 129
5.9.10 getValueNames 方法：获取所有的属性名 ····· 129
5.9.11 invalidate 方法：使会话无效 ····· 129
5.9.12 isNew 方法：判断会话是否新 ····· 130
5.9.13 putValue 方法：设置属性值 ····· 130
5.9.14 removeAttribute 方法：移除绑定对象 ····· 130
5.9.15 removeValue 方法：移除属性值 ····· 130
5.9.16 setAttribute 方法：设置绑定对象 ····· 130
5.9.17 setMaxInactiveInterval 方法：设置会话有效时间 ····· 131

第 2 篇 JSP 高级技术篇

第 6 章 核心标记库（Core Tag Library） ····· 135
6.1 <c:out>动作 ····· 135
6.1.1 value 属性：指定表达式 ····· 135
6.1.2 default 属性：指定默认表达式 ····· 136
6.1.3 escapeXml 属性：指定是否转换 XML 字符 ····· 136
6.2 <c:set>动作 ····· 137
6.2.1 value 属性：指定表达式 ····· 137
6.2.2 var 属性：指定变量名 ····· 137
6.2.3 scope 属性：指定变量作用域 ····· 138
6.2.4 target 属性：指定目标对象 ····· 138
6.2.5 property 属性：指定目标对象的属性 ····· 139

6.3 <c:remove>动作……139
6.3.1 var 属性：指定变量名……139
6.3.2 scope 属性：指定变量的作用域……140
6.4 <c:catch>动作……140
6.5 <c:if>动作……141
6.5.1 test 属性：指定测试表达式……141
6.5.2 var 属性：指定变量名……141
6.5.3 scope 属性：指定变量的作用域……142
6.6 <c:choose>动作……142
6.7 <c:when>动作……143
6.8 <c:otherwise>动作……143
6.9 <c:forEach>动作……144
6.9.1 var 属性：指定嵌套变量名……144
6.9.2 items 属性：指定迭代集合……145
6.9.3 varStatus 属性：指定迭代状态变量名……145
6.9.4 begin 属性：指定开始索引……146
6.9.5 end 属性：指定结束索引……146
6.9.6 step 属性：指定迭代步长……146
6.10 <c:forTokens>动作……147
6.10.1 var 属性：指定嵌套变量名称……147
6.10.2 items 属性：指定迭代 token 组……148
6.10.3 delims 属性：指定分隔符列表……148
6.10.4 varStatus 属性：指定迭代状态变量名……148
6.10.5 begin 属性：指定开始索引……148
6.10.6 end 属性：指定结束索引……149
6.10.7 step 属性：指定迭代步长……149
6.11 <c:import>动作……150
6.11.1 url 属性：指定资源 URL……150
6.11.2 context 属性：指定外部上下文……150
6.11.3 var 属性：指定变量名……151
6.11.4 scope 属性：指定变量作用域……151
6.11.5 charEncoding 属性：指定字符编码方式……152
6.11.6 varReader 属性：指定 Reader 对象变量名……152
6.12 <c:url>动作……152
6.12.1 value 属性：指定 URL……152
6.12.2 context 属性：指定外部上下文……153
6.12.3 var 属性：指定变量名……153
6.12.4 scope 属性：指定变量作用域……154
6.13 <c:redirect>动作……154
6.13.1 url 属性：指定 URL……154
6.13.2 context 属性：指定外部上下文……155
6.14 <c:param>动作……155
6.14.1 name 属性：指定参数名……155
6.14.2 value 属性：指定参数值……156

第 7 章　国际化格式标记库 ····· 157
7.1　<fmt:setLocale>动作 ····· 157
7.1.1　value 属性：指定地域代码 ····· 157
7.1.2　variant 属性：指定供应商或浏览器代码 ····· 157
7.1.3　scope 属性：指定配置变量作用域 ····· 158
7.2　<fmt:requestEncoding>动作 ····· 158
7.3　<fmt:message>动作 ····· 159
7.3.1　key 属性：指定消息键 ····· 159
7.3.2　bundle 属性：指定资源束 ····· 160
7.3.3　var 属性：指定变量名 ····· 161
7.3.4　scope 属性：指定变量作用域 ····· 161
7.4　<fmt:param>动作 ····· 161
7.5　<fmt:bundle>动作 ····· 162
7.5.1　basename 属性：指定基本文件名 ····· 162
7.5.2　prefix 属性：指定消息键前缀 ····· 163
7.6　<fmt:setBundle>动作 ····· 163
7.6.1　basename 属性：指定基本文件名 ····· 163
7.6.2　var 属性：指定变量名 ····· 163
7.6.3　scope 属性：指定变量作用域 ····· 164
7.7　<fmt:formatNumber>动作 ····· 164
7.7.1　value 属性：指定数值 ····· 164
7.7.2　type 属性：指定数值类型 ····· 165
7.7.3　pattern 属性：指定格式化模式 ····· 165
7.7.4　currencyCode 属性：指定货币码 ····· 166
7.7.5　currencySymbol 属性：指定货币符号 ····· 166
7.7.6　groupingUsed 属性：是否使用分组 ····· 166
7.7.7　maxIntegerDigits 属性：指定最大整数位数 ····· 167
7.7.8　minIntegerDigits 属性：指定最小整数位数 ····· 167
7.7.9　maxFractionDigits 属性：指定最大小数位数 ····· 167
7.7.10　minFractionDigits 属性：指定最小小数位数 ····· 167
7.7.11　var 属性：指定变量名 ····· 167
7.7.12　scope 属性：指定变量作用域 ····· 168
7.8　<fmt:parseNumber>动作 ····· 168
7.8.1　value 属性：指定解析字符串 ····· 168
7.8.2　type 属性：指定数值类型 ····· 169
7.8.3　pattern 属性：指定格式化模式 ····· 169
7.8.4　parseLocale 属性：指定地域代码 ····· 170
7.8.5　integerOnly 属性：是否仅解析整数部分 ····· 170
7.8.6　var 属性：指定变量名 ····· 170
7.8.7　scope 属性：指定变量作用域 ····· 170
7.9　<fmt:formatDate>动作 ····· 171
7.9.1　value 属性：指定日期和时间 ····· 171
7.9.2　type 属性：指定成分类型 ····· 171
7.9.3　dateStyle 属性：指定日期样式 ····· 172

7.9.4 timeStyle 属性：指定时间样式……172
7.9.5 pattern 属性：指定格式化模式……173
7.9.6 timeZone 属性：指定时区……173
7.9.7 var 属性：指定变量名……174
7.9.8 scope 属性：指定变量作用域……174
7.10 <fmt:parseDate>动作……174
7.10.1 value 属性：指定日期和时间字符串……175
7.10.2 type 属性：指定成分类型……175
7.10.3 dateStyle 属性：指定日期样式……176
7.10.4 timeStyle 属性：指定时间样式……176
7.10.5 pattern 属性：指定格式化模式……176
7.10.6 timeZone 属性：指定时区……177
7.10.7 parseLocale 属性：指定地域代码……177
7.10.8 var 属性：指定变量名……177
7.10.9 scope 属性：指定变量作用域……178
7.11 <fmt:setTimeZone>动作……178
7.11.1 value 属性：指定时区标识……178
7.11.2 var 属性：指定变量名……179
7.11.3 scope 属性：指定变量作用域……179
7.12 <fmt:timeZone>动作……179

第 8 章 SQL 标记库（SQL Tag Library）……181
8.1 <sql:setDataSource>动作……181
8.1.1 dataSource 属性：指定数据源……181
8.1.2 driver 属性：指定驱动程序类名……182
8.1.3 url 属性：指定数据库 URL……182
8.1.4 user 属性：指定用户名称……182
8.1.5 password 属性：指定用户密码……182
8.1.6 var 属性：指定变量名……182
8.1.7 scope 属性：指定变量作用域……183
8.2 <sql:query>动作……184
8.2.1 sql 属性：指定 SQL 查询语句……184
8.2.2 dataSource 属性：指定数据源……185
8.2.3 maxRows 属性：指定最大行数……185
8.2.4 startRow 属性：指定开始索引……186
8.2.5 var 属性：指定变量名……187
8.2.6 scope 属性：指定变量作用域……187
8.3 <sql:update>动作……188
8.3.1 sql 属性：指定 SQL 更新语句……188
8.3.2 dataSource 属性：指定数据源……189
8.3.3 var 属性：指定变量名……190
8.3.4 scope 属性：指定变量作用域……190
8.4 <sql:param>动作……190
8.5 <sql:dateParam>动作……191

8.5.1 value 属性：指定参数值 191
8.5.2 type 属性：指定字段类型 192
8.6 <sql:transaction>动作 192
8.6.1 dataSource 属性：指定数据源 193
8.6.2 isolation 属性：指定事务隔离级别 194

第 9 章 XML 标记库（XML Tag Library） 196
9.1 <x:parse>动作 196
9.1.1 doc 属性：指定 XML 文档 196
9.1.2 xml 属性：指定 XML 文档 198
9.1.3 systemId 属性：指定系统标识 198
9.1.4 filter 属性：指定过滤器 198
9.1.5 varDom 属性：指定 DOM 变量名 199
9.1.6 scopeDom 属性：指定 DOM 变量作用域 199
9.1.7 var 属性：指定变量名 200
9.1.8 scope 属性：指定变量作用域 200
9.2 <x:out>动作 201
9.2.1 select 属性：指定 XPath 表达式 201
9.2.2 escapeXml 属性：是否转换 XML 字符 202
9.3 <x:set>动作 202
9.3.1 select 属性：指定 XPath 表达式 202
9.3.2 var 属性：指定变量名 203
9.3.3 scope 属性：指定变量作用域 203
9.4 <x:if>动作 204
9.4.1 select 属性：指定测试条件 204
9.4.2 var 属性：指定变量名 205
9.4.3 scope 属性：指定变量作用域 206
9.5 <x:when>动作 206
9.6 <x:choose>动作 207
9.7 <x:otherwise>动作 209
9.8 <x:forEach>动作 209
9.8.1 var 属性：指定变量名 210
9.8.2 select 属性：指定 XPath 表达式 210
9.8.3 varStatus 属性：指定状态变量名 211
9.8.4 begin 属性：指定开始索引 212
9.8.5 end 属性：指定结束索引 212
9.8.6 step 属性：指定迭代步长 213
9.9 <x:transform>动作 214
9.9.1 doc 属性：指定 XML 文档 214
9.9.2 xml 属性：指定 XML 文档 215
9.9.3 xslt 属性：指定 XSLT 样式表 215
9.9.4 docSystemId 属性：指定 XML 系统标识 216
9.9.5 xsltSystemId 属性：指定 XSLT 系统标识 216
9.9.6 var 属性：指定变量名 217

9.9.7 scope 属性：指定变量作用域……217
9.9.8 result 属性：指定结果对象……218
9.10 <x:param>动作……218
9.10.1 name 属性：指定参数名……219
9.10.2 value 属性：指定参数值……220

第 10 章 函数标记库（Functions Tag Library）……221
10.1 fn:contains 函数……221
10.1.1 inputString 参数：指定输入字符串……221
10.1.2 subString 参数：指定子字符串……221
10.1.3 函数返回值……221
10.2 fn:containsIgnoreCase 函数……223
10.2.1 inputString 参数：指定输入字符串……223
10.2.2 subString 参数：指定子字符串……223
10.2.3 函数返回值……223
10.3 fn:startsWith 函数……224
10.3.1 inputString 参数：指定输入字符串……224
10.3.2 prefix 参数：指定前缀字符串……224
10.3.3 函数返回值……224
10.4 fn:endsWith 函数……225
10.4.1 inputString 参数：指定输入字符串……226
10.4.2 suffix 参数：指定后缀字符串……226
10.4.3 函数返回值……226
10.5 fn:escapeXml 函数……227
10.5.1 inputString 参数：指定输入字符串……227
10.5.2 函数返回值……227
10.6 fn:indexOf 函数……228
10.6.1 string 参数：指定字符串……228
10.6.2 subString 参数：指定子字符串……228
10.6.3 函数返回值……229
10.7 fn:split 函数……230
10.7.1 string 参数：指定要分离的字符串……230
10.7.2 delimiters 参数：指定分隔符列表……230
10.7.3 函数返回值……230
10.8 fn:join 函数……231
10.8.1 array 参数：指定要连接的数组……232
10.8.2 separator 参数：指定连接字符串……232
10.8.3 函数返回值……232
10.9 fn:replace 函数……233
10.9.1 inputString 参数：指定输入字符串……233
10.9.2 beforeString 参数：指定替换前字符串……233
10.9.3 afterString 参数：指定替换后字符串……233
10.9.4 函数返回值……233
10.10 fn:trim 函数……235

10.10.1 string 参数：指定原始字符串……235
10.10.2 函数返回值……235
10.11 fn:substring 函数……236
10.11.1 inputString 参数：指定输入字符串……236
10.11.2 beginIndex 参数：指定开始索引……236
10.11.3 endIndex 参数：指定结束索引……236
10.11.4 函数返回值……236
10.12 fn:substringAfter 函数……237
10.12.1 inputString 参数：指定输入字符串……238
10.12.2 subString 参数：指定子字符串……238
10.12.3 函数返回值……238
10.13 fn:substringBefore 函数……239
10.13.1 inputString 参数：指定输入字符串……239
10.13.2 subString 参数：指定子字符串……239
10.13.3 函数返回值……239
10.14 fn:toLowerCase 函数……240
10.14.1 inputString 参数：指定输入字符串……241
10.14.2 函数返回值……241
10.15 fn:toUpperCase 函数……241
10.15.1 inputString 参数：指定输入字符串……242
10.15.2 函数返回值……242
10.16 fn:length 函数……242
10.16.1 input 参数：指定输入对象……243
10.16.2 函数返回值……243

第 11 章 定制标记库（Tag Library）……244
11.1 TagSupport 类……244
11.1.1 EVAL_BODY_AGAIN 字段：再次计算本体……244
11.1.2 EVAL_BODY_INCLUDE 字段：包括本体计算结果……244
11.1.3 EVAL_PAGE 字段：计算页面……244
11.1.4 SKIP_BODY 字段：跳过本体……246
11.1.5 SKIP_PAGE 字段：跳过页面……246
11.1.6 id 字段：id 属性值……246
11.1.7 pageContext 字段：pageContext 对象……247
11.1.8 doAfterBody 方法：计算本体后的处理……248
11.1.9 doEndTag 方法：遇到结束标记时的处理……249
11.1.10 doStartTag 方法：遇到开始标记的处理……249
11.1.11 findAncestorWithClass 方法：查找类实例……250
11.1.12 getId 方法：获取 id 属性值……251
11.1.13 getParent 方法：获取父处理器……251
11.1.14 getValue 方法：获取键值……252
11.1.15 getValues 方法：获取所有的键……252
11.1.16 release 方法：释放状态……253
11.1.17 removeValue 方法：移除键值……253

11.1.18 setId 方法：设置 id 属性值 ····· 253
11.1.19 setPageContext 方法：设置 pageContext 对象 ····· 254
11.1.20 setParent 方法：设置父处理器 ····· 254
11.1.21 setValue 方法：设置键值 ····· 254
11.2 BodyTagSupport 类 ····· 254
11.2.1 bodyContent 字段：bodyContent 对象 ····· 255
11.2.2 EVAL_BODY_BUFFERED 字段：缓冲计算本体 ····· 256
11.2.3 EVAL_BODY_TAG 字段：计算标记本体 ····· 256
11.2.4 doInitBody 方法：初始化本体的处理 ····· 256
11.2.5 getBodyContent 方法：获取 bodyContent 对象 ····· 257
11.2.6 getPreviousOut 方法：获取外层 out 对象 ····· 258
11.2.7 setBodyContent 方法：设置 BodyContent 对象 ····· 258
11.3 SimpleTagSupport 类 ····· 258
11.3.1 doTag 方法：处理标记 ····· 258
11.3.2 findAncestorWithClass 方法：查找类实例 ····· 259
11.3.3 getJspBody 方法：获取本体片段 ····· 260
11.3.4 getJspContext 方法：获取页面上下文 ····· 261
11.3.5 getParent 方法：获取父处理器 ····· 262
11.3.6 setJspBody 方法：设置本体片段 ····· 262
11.3.7 setJspContext 方法：设置页面上下文 ····· 262
11.3.8 setParent 方法：设置父处理器 ····· 263
11.4 DynamicAttributes 接口 ····· 263
11.5 TryCatchFinally 接口 ····· 264
11.5.1 doCatch 方法：Catch 处理 ····· 264
11.5.2 doFinally 方法：Finally 处理 ····· 265
11.6 标记库描述符（Tag Library Descriptor） ····· 266
11.6.1 <taglib>元素：指定标记库 ····· 266
11.6.2 <tlib-version>元素：指定标记库版本 ····· 267
11.6.3 <jsp-version>元素：指定 JSP 版本 ····· 267
11.6.4 <short-name>元素：指定标记库简称 ····· 267
11.6.5 <uri>元素：指定标记库 URI ····· 267
11.6.6 <description>元素：指定描述信息 ····· 267
11.6.7 <display-name>元素：指定显示名称 ····· 268
11.6.8 <small-icon>元素：指定小图标 ····· 268
11.6.9 <large-icon>元素：指定大图标 ····· 268
11.6.10 <tag>元素：指定标记 ····· 268
11.6.11 <validator>元素：指定验证器 ····· 269
11.6.12 <listener>元素：指定监听器 ····· 269
11.6.13 <tag-file>元素：指定标记文件 ····· 269
11.6.14 <function>元素：指定函数 ····· 270
11.6.15 <name>元素：指定元素名称 ····· 270
11.6.16 <tag-class>元素：指定标记处理器类 ····· 270
11.6.17 <tei-class>元素：指定标记额外信息类 ····· 270
11.6.18 <body-content>元素：指定本体内容 ····· 271

11.6.19 <dynamic-attributes>元素：指定动态属性……271
11.6.20 <example>元素：指定范例……271
11.6.21 <variable>元素：指定变量……271
11.6.22 <attribute>元素：指定属性……272
11.6.23 <validator-class>元素：指定验证器类……272
11.6.24 <init-param>元素：指定初始化参数……272
11.6.25 <listener-class>元素：指定监听器类……273
11.6.26 <path>元素：指定标记文件路径……273
11.6.27 <function-class>元素：指定函数类……273
11.6.28 <function-signature>元素：指定函数声明……273
11.6.29 <name-given>元素：指定给定的变量名……273
11.6.30 <name-from-attribute>元素：指定来自属性的变量名……274
11.6.31 <variable-class>元素：指定变量类型……274
11.6.32 <declare>元素：指定是否声明……274
11.6.33 <scope>元素：指定作用域……274
11.6.34 <required>元素：指定是否必需……274
11.6.35 <rtexprvalue>元素：指定是否接受动态值……275
11.6.36 <type>元素：指定属性类型……275
11.6.37 <fragment>元素：指定是否为 JspFragment……275
11.6.38 <param-name>元素：指定参数名……275
11.6.39 <param-value>元素：指定参数值……276

第 12 章 表达式语言（Expression Language）……277
12.1 EL 标识符……277
12.1.1 保留字……277
12.1.2 保留标识符……277
12.1.3 作用域变量……277
12.2 EL 存取器……278
12.2.1 点（.）运算符……278
12.2.2 方括号（[]）运算符……278
12.3 EL 隐式对象……279
12.3.1 pageContext 对象：页面上下文……279
12.3.2 pageScope 对象：页面作用域……280
12.3.3 requestScope 对象：请求作用域……280
12.3.4 sessionScope 对象：会话作用域……280
12.3.5 applicationScope 对象：应用作用域……280
12.3.6 param 对象：请求参数……280
12.3.7 paramValues 对象：请求参数所有的值……281
12.3.8 header 对象：请求头部……282
12.3.9 headerValues 对象：请求头部所有的值……283
12.3.10 cookie 对象：Cookie……283
12.3.11 initParam 对象：初始化参数……283
12.4 EL 文字……284
12.5 EL 运算符……284

12.5.1 算术运算符……284
12.5.2 关系运算符……284
12.5.3 逻辑运算符……285
12.5.4 验证运算符……285
12.5.5 运算符优先级……285
12.6 EL 函数……286
12.6.1 函数实现类 VolFunc.java……286
12.6.2 标记库描述符文件 ccc.tld……286
12.6.3 应用部署描述文件 web.xml……287
12.6.4 EL 函数范例 elfunc.jsp……287

第 3 篇 Servlet 技术篇

第 13 章 Servlet……291
13.1 Servlet 接口……291
13.1.1 init 方法：初始化 Servlet……291
13.1.2 service 方法：处理客户端请求……291
13.1.3 destroy 方法：销毁 Servlet……293
13.1.4 getServletConfig 方法：获取 ServletConfig 对象……293
13.1.5 getServletInfo 方法：获取 Servlet 信息……293
13.2 GenericServlet 类……294
13.2.1 GenericServlet：构造函数……294
13.2.2 init 方法：初始化 Servlet……294
13.2.3 service 方法：处理客户端请求……294
13.2.4 destroy 方法：销毁 Servlet……296
13.2.5 getInitParameter 方法：获取初始化参数……296
13.2.6 getInitParameterNames 方法：获取所有初始化参数名……296
13.2.7 getServletConfig 方法：获取 ServletConfig 对象……297
13.2.8 getServletContext 方法：获取 Servlet 上下文……297
13.2.9 getServletInfo 方法：获取 Servlet 信息……298
13.2.10 getServletName 方法：获取 Servlet 名称……298
13.2.11 log 方法：信息写入日志……298
13.3 HttpServlet 类……298
13.3.1 HttpServlet：构造函数……299
13.3.2 init 方法：初始化 Servlet……299
13.3.3 service 方法：处理客户端请求……299
13.3.4 destroy 方法：销毁 Servlet……299
13.3.5 doGet 方法：处理 GET 请求……299
13.3.6 doPost 方法：处理 POST 请求……301
13.3.7 doHead 方法：处理 HEAD 请求……302
13.3.8 doPut 方法：处理 PUT 请求……302
13.3.9 doDelete 方法：处理 DELETE 请求……302
13.3.10 doTrace 方法：处理 TRACE 请求……302
13.3.11 doOptions 方法：处理 OPTIONS 请求……303
13.3.12 getInitParameter 方法：获取初始化参数……303

13.3.13 getInitParameterNames 方法：获取所有初始化参数名 ····303
13.3.14 getLastModified 方法：获取最后修改时间 ····303
13.4 ServletRequest 接口 ····304
13.4.1 getAttribute 方法：获取属性 ····304
13.4.2 getAttributeNames 方法：获取所有属性名 ····304
13.4.3 getCharacterEncoding 方法：获取字符编码方式 ····305
13.4.4 getContentLength 方法：获取内容长度 ····306
13.4.5 getContentType 方法：获取内容类型 ····306
13.4.6 getInputStream 方法：获取输入流 ····306
13.4.7 getLocalAddr 方法：获取服务器地址 ····307
13.4.8 getLocale 方法：获取区域对象 ····307
13.4.9 getLocales 方法：获取所有的区域对象 ····308
13.4.10 getLocalName 方法：获取服务器名称 ····308
13.4.11 getLocalPort 方法：获取服务器端口 ····308
13.4.12 getParameter 方法：获取参数 ····308
13.4.13 getParameterMap 方法：获取参数映射 ····308
13.4.14 getParameterNames 方法：获取所有参数名 ····309
13.4.15 getParameterValues 方法：获取所有参数值 ····309
13.4.16 getProtocol 方法：获取协议 ····311
13.4.17 getReader 方法：获取 Reader 对象 ····311
13.4.18 getRealPath 方法：获取物理路径 ····311
13.4.19 getRemoteAddr 方法：获取客户端地址 ····312
13.4.20 getRemoteHost 方法：获取客户端主机 ····312
13.4.21 getRemotePort 方法：获取客户端端口 ····312
13.4.22 getRequestDispatcher 方法：获取 RequestDispatcher 对象 ····312
13.4.23 getScheme 方法：获取协议机制 ····313
13.4.24 getServerName 方法：获取服务器名称 ····313
13.4.25 getServerPort 方法：获取服务器端口 ····313
13.4.26 isSecure 方法：判断是否使用安全链接 ····314
13.4.27 removeAttribute 方法：移除属性 ····314
13.4.28 setAttribute 方法：设置属性 ····315
13.4.29 setCharacterEncoding 方法：设置字符编码方式 ····315
13.5 ServletResponse 接口 ····315
13.5.1 flushBuffer 方法：刷新缓冲区 ····315
13.5.2 getBufferSize 方法：获取缓冲区大小 ····316
13.5.3 getCharacterEncoding 方法：获取字符编码方式 ····316
13.5.4 getContentType 方法：获取内容类型 ····316
13.5.5 getLocale 方法：获取区域对象 ····316
13.5.6 getOutputStream 方法：获取 ServletOutputStream 对象 ····316
13.5.7 getWriter 方法：获取 PrintWriter 对象 ····317
13.5.8 isCommitted 方法：判断是否已提交 ····318
13.5.9 reset 方法：重置 ····318
13.5.10 resetBuffer 方法：重置缓冲区 ····319
13.5.11 setBufferSize 方法：设置缓冲区大小 ····319

13.5.12 setCharacterEncoding 方法：设置字符编码方式……319
13.5.13 setContentLength 方法：设置内容长度……319
13.5.14 setContentType 方法：设置内容类型……320
13.5.15 setLocale 方法：设置地域代码……320
13.6 HttpServletRequest 接口……320
13.6.1 BASIC_AUTH 字段：基本认证……320
13.6.2 CLIENT_CERT_AUTH 字段：客户端证书认证……321
13.6.3 DIGEST_AUTH 字段：摘要认证……321
13.6.4 FORM_AUTH 字段：表单认证……321
13.6.5 getAttribute 方法：获取属性……321
13.6.6 getAttributeNames 方法：获取属性名集合……321
13.6.7 getAuthType 方法：获取认证类型……321
13.6.8 getContextPath 方法：获取上下文路径……322
13.6.9 getCookies 方法：获取 Cookie 对象数组……323
13.6.10 getDateHeader 方法：获取 Date 型头部……323
13.6.11 getHeader 方法：获取头部……323
13.6.12 getHeaderNames 方法：获取所有头部名……324
13.6.13 getHeaders 方法：获取头部所有的值……324
13.6.14 getInputStream 方法：获取输入流……325
13.6.15 getIntHeader 方法：获取整型头部……325
13.6.16 getMethod 方法：获取请求方式……325
13.6.17 getPathInfo 方法：获取路径信息……325
13.6.18 getPathTranslated 方法：获取翻译后的路径……326
13.6.19 getQueryString 方法：获取查询字符串……326
13.6.20 getRemoteUser 方法：获取客户端用户名……326
13.6.21 getRequestedSessionId 方法：获取请求的会话标识……327
13.6.22 getRequestURI 方法：获取发送请求的 URI……327
13.6.23 getRequestURL 方法：获取响应请求的 URL……327
13.6.24 getServletPath 方法：获取 Servlet 路径……328
13.6.25 getSession 方法：获取 HttpSession 对象……328
13.6.26 getUserPrincipal 方法：获取认证用户的 Principal 对象……329
13.6.27 isRequestedSessionIdFromCookie 方法：判断会话 ID 是否来自 Cookie……329
13.6.28 isRequestedSessionIdFromURL 方法：判断会话 ID 是否来自 URL……329
13.6.29 isRequestedSessionIdValid 方法：判断会话 ID 是否有效……329
13.6.30 isSecure 方法：判断是否使用安全链接……330
13.6.31 isUserInRole 方法：判断认证用户是否属于指定角色……330
13.7 HttpServletResponse 接口……330
13.7.1 SC_ACCEPTED 字段：已接受……331
13.7.2 SC_BAD_GATEWAY 字段：错误的网关……331
13.7.3 SC_BAD_REQUEST 字段：错误请求……331
13.7.4 SC_CONFLICT 字段：冲突……331
13.7.5 SC_CONTINUE 字段：继续……331
13.7.6 SC_CREATED 字段：已创建……332
13.7.7 SC_EXPECTATION_FAILED 字段：期望失败……332

13.7.8 SC_FORBIDDEN 字段：禁止……332
13.7.9 SC_FOUND 字段：找到……332
13.7.10 SC_GATEWAY_TIMEOUT 字段：网关超时……332
13.7.11 SC_GONE 字段：已不存在……333
13.7.12 SC_HTTP_VERSION_NOT_SUPPORTED 字段：不支持的 HTTP 版本……333
13.7.13 SC_INTERNAL_SERVER_ERROR 字段：内部服务器错误……333
13.7.14 SC_LENGTH_REQUIRED 字段：需要数据长度……333
13.7.15 SC_METHOD_NOT_ALLOWED 字段：方法未允许……333
13.7.16 SC_MOVED_PERMANENTLY 字段：永久性移动……334
13.7.17 SC_MOVED_TEMPORARILY 字段：临时性移动……334
13.7.18 SC_MULTIPLE_CHOICES 字段：多重选择……334
13.7.19 SC_NO_CONTENT 字段：无内容……334
13.7.20 SC_NON_AUTHORITATIVE_INFORMATION 字段：非官方信息……335
13.7.21 SC_NOT_ACCEPTABLE 字段：无法访问……335
13.7.22 SC_NOT_FOUND 字段：未找到……335
13.7.23 SC_NOT_IMPLEMENTED 字段：未实现……335
13.7.24 SC_NOT_MODIFIED 字段：未修正……335
13.7.25 SC_OK 字段：正常……336
13.7.26 SC_PARTIAL_CONTENT 字段：局部内容……336
13.7.27 SC_PAYMENT_REQUIRED 字段：保留……336
13.7.28 SC_PRECONDITION_FAILED 字段：先决条件失败……336
13.7.29 SC_PROXY_AUTHENTICATION_REQUIRED 字段：需代理服务器认证……336
13.7.30 SC_REQUEST_ENTITY_TOO_LARGE 字段：请求实体过大……337
13.7.31 SC_REQUEST_TIMEOUT 字段：请求超时……337
13.7.32 SC_REQUEST_URI_TOO_LONG 字段：请求 URI 过长……337
13.7.33 SC_REQUESTED_RANGE_NOT_SATISFIABLE：请求范围无法满足……337
13.7.34 SC_RESET_CONTENT 字段：重置内容……338
13.7.35 SC_SEE_OTHER 字段：参见其他信息……338
13.7.36 SC_SERVICE_UNAVAILABLE 字段：服务无法获得……338
13.7.37 SC_SWITCHING_PROTOCOLS 字段：转换协议……338
13.7.38 SC_TEMPORARY_REDIRECT 字段：临时重定向……338
13.7.39 SC_UNAUTHORIZED 字段：未授权……339
13.7.40 SC_UNSUPPORTED_MEDIA_TYPE 字段：不支持的媒体格式……339
13.7.41 SC_USE_PROXY 字段：使用代理……339
13.7.42 addCookie 方法：添加 Cookie 对象……339
13.7.43 addDateHeader 方法：添加 Date 型头部……340
13.7.44 addHeader 方法：添加字符串值头部……340
13.7.45 addIntHeader 方法：添加整数值头部……341
13.7.46 containsHeader 方法：判断头部是否设置……341
13.7.47 encodeRedirectURL 方法：对指定 URL 编码……341
13.7.48 encodeURL 方法：对指定 URL 编码……342
13.7.49 sendError 方法：发送错误响应……342
13.7.50 sendRedirect 方法：发送重定向响应……343
13.7.51 setDateHeader 方法：设置日期值头部……343

13.7.52 setHeader 方法：设置字符串值头部……344
13.7.53 setIntHeader 方法：设置整数值头部……344
13.7.54 setStatus 方法：设置状态码……345
13.8 ServletContext 接口……345
13.8.1 getAttribute 方法：获取属性值……345
13.8.2 getAttributeNames 方法：获取所有属性名……346
13.8.3 getContext 方法：获取 ServletContext 对象……347
13.8.4 getInitParameter 方法：获取初始化参数……347
13.8.5 getInitParameterNames 方法：获取所有初始化参数名……347
13.8.6 getMajorVersion 方法：获取主版本号……348
13.8.7 getMinorVersion 方法：获取副版本号……348
13.8.8 getMimeType 方法：获取 MIME 类型……348
13.8.9 getNamedDispatcher 方法：获取 RequestDispatcher 对象……349
13.8.10 getRealPath 方法：获取物理路径……349
13.8.11 getRequestDispatcher 方法：获取 RequestDispatcher 对象……350
13.8.12 getResource 方法：获取资源路径……350
13.8.13 getResourceAsStream 方法：获取资源的 InputStream 对象……351
13.8.14 getResourcePaths 方法：获取所有资源路径……352
13.8.15 getServerInfo 方法：获取服务器信息……352
13.8.16 getServlet 方法：获取 Servlet……352
13.8.17 getServletContextName 方法：获取 Servlet 上下文名称……353
13.8.18 getServletNames 方法：获取所有 Servlet 名……353
13.8.19 getServlets 方法：获取所有 Servlet……353
13.8.20 log 方法：信息写入日志……353
13.8.21 removeAttribute 方法：移除属性……354
13.8.22 setAttribute 方法：设置属性……354

第 14 章 过滤器（Filter）……355
14.1 Filter 接口……355
14.1.1 destroy 方法：销毁过滤器……355
14.1.2 doFilter 方法：过滤处理……356
14.1.3 init 方法：初始化过滤器……357
14.2 FilterConfig 接口……357
14.2.1 getFilterName 方法：获取过滤器名……357
14.2.2 getInitParameter 方法：获取初始化参数……358
14.2.3 getInitParameterNames 方法：获取所有初始化参数名……358
14.2.4 getServletContext 方法：获取 ServletContext 上下文……358
14.3 FilterChain 接口……359
14.4 ServletRequestWrapper 类……360
14.4.1 ServletRequestWrapper：构造函数……360
14.4.2 getAttribute 方法：获取属性……362
14.4.3 getAttributeNames 方法：获取所有属性名……362
14.4.4 getCharacterEncoding 方法：获取字符编码方式……362
14.4.5 getContentLength 方法：获取内容长度……363

14.4.6 getContentType 方法：获取内容类型 ····363
14.4.7 getInputStream 方法：获取输入流 ····363
14.4.8 getLocalAddr 方法：获取服务器地址 ····363
14.4.9 getLocale 方法：获取区域对象 ····364
14.4.10 getLocales 方法：获取所有的区域对象 ····364
14.4.11 getLocalName 方法：获取服务器名称 ····364
14.4.12 getLocalPort 方法：获取服务器端口 ····364
14.4.13 getParameter 方法：获取参数 ····365
14.4.14 getParameterMap 方法：获取参数映射 ····365
14.4.15 getParameterNames 方法：获取所有参数名 ····365
14.4.16 getParameterValues 方法：获取所有参数值 ····365
14.4.17 getProtocol 方法：获取协议 ····366
14.4.18 getReader 方法：获取 Reader 对象 ····366
14.4.19 getRealPath 方法：获取实际路径 ····366
14.4.20 getRemoteAddr 方法：获取客户端地址 ····366
14.4.21 getRemoteHost 方法：获取客户端主机 ····367
14.4.22 getRemotePort 方法：获取客户端端口 ····367
14.4.23 getRequest 方法：获取 ServletRequest 对象 ····367
14.4.24 getRequestDispatcher 方法：获取 RequestDispatcher 对象 ····367
14.4.25 getScheme 方法：获取协议机制 ····368
14.4.26 getServerName 方法：获取服务器名称 ····368
14.4.27 getServerPort 方法：获取服务器端口 ····368
14.4.28 isSecure 方法：判断是否使用安全链接 ····368
14.4.29 removeAttribute 方法：移除属性 ····368
14.4.30 setAttribute 方法：设置属性 ····369
14.4.31 setCharacterEncoding 方法：设置字符编码方式 ····369
14.4.32 setRequest 方法：设置 ServletRequest 对象 ····369
14.5 ServletResponseWrapper 类 ····370
14.5.1 ServletResponseWrapper：构造函数 ····370
14.5.2 flushBuffer 方法：刷新缓冲区 ····371
14.5.3 getBufferSize 方法：获取缓冲区大小 ····371
14.5.4 getCharacterEncoding 方法：获取字符编码方式 ····372
14.5.5 getContentType 方法：获取内容类型 ····372
14.5.6 getLocale 方法：获取区域对象 ····372
14.5.7 getOutputStream 方法：获取 ServletOutputStream 对象 ····372
14.5.8 getResponse 方法：获取 ServletResponse 对象 ····373
14.5.9 getWriter 方法：获取 PrintWriter 对象 ····373
14.5.10 isCommitted 方法：判断是否已提交 ····373
14.5.11 reset 方法：重置 ····373
14.5.12 resetBuffer 方法：重置缓冲区 ····374
14.5.13 setBufferSize 方法：设置缓冲区大小 ····374
14.5.14 setCharacterEncoding 方法：设置字符编码方式 ····374
14.5.15 setContentLength 方法：设置内容长度 ····374
14.5.16 setContentType 方法：设置内容类型 ····375

14.5.17 setLocale 方法：设置地域代码 ····375
14.5.18 setResponse 方法：设置 ServletResponse 对象 ····375
14.6 HttpServletRequestWrapper 类 ····375
14.6.1 HttpServletRequestWrapper：构造函数 ····376
14.6.2 getAuthType 方法：获取认证类型 ····376
14.6.3 getContextPath 方法：获取上下文路径 ····376
14.6.4 getCookies 方法：获取 Cookie 对象数组 ····377
14.6.5 getDateHeader 方法：获取日期值头部 ····377
14.6.6 getHeader 方法：获取头部 ····377
14.6.7 getHeaderNames 方法：获取所有头部名 ····377
14.6.8 getHeaders 方法：获取头部所有的值 ····378
14.6.9 getIntHeader 方法：获取整数值头部 ····378
14.6.10 getMethod 方法：获取请求方式 ····378
14.6.11 getPathInfo 方法：获取路径信息 ····378
14.6.12 getPathTranslated 方法：获取翻译后的路径 ····379
14.6.13 getQueryString 方法：获取查询字符串 ····379
14.6.14 getRemoteUser 方法：获取客户端用户名 ····379
14.6.15 getRequestedSessionId 方法：获取请求的会话标识 ····379
14.6.16 getRequestURI 方法：获取发送请求的 URI ····380
14.6.17 getRequestURL 方法：获取响应请求的 URL ····380
14.6.18 getServletPath 方法：获取 Servlet 路径 ····380
14.6.19 getSession 方法：获取 HttpSession 对象 ····380
14.6.20 getUserPrincipal 方法：获取认证用户的 Principal 对象 ····380
14.6.21 isRequestedSessionIdFromCookie 方法：判断会话 ID 是否来自 Cookie ····381
14.6.22 isRequestedSessionIdFromURL 方法：判断会话 ID 是否来自 URL ····381
14.6.23 isRequestedSessionIdValid 方法：判断会话 ID 是否有效 ····381
14.6.24 isUserInRole 方法：判断认证用户是否属于指定角色 ····381
14.7 HttpServletResponseWrapper 类 ····382
14.7.1 HttpServletResponseWrapper：构造函数 ····382
14.7.2 addCookie 方法：添加 Cookie 对象 ····382
14.7.3 addDateHeader 方法：添加日期值头部 ····383
14.7.4 addHeader 方法：添加字符串值头部 ····383
14.7.5 addIntHeader 方法：添加整数值头部 ····383
14.7.6 containsHeader 方法：判断头部是否设置 ····384
14.7.7 encodeRedirectURL 方法：对指定 URL 编码 ····384
14.7.8 encodeURL 方法：对指定 URL 编码 ····384
14.7.9 sendError 方法：发送错误响应 ····384
14.7.10 sendRedirect 方法：发送重定向响应 ····385
14.7.11 setDateHeader 方法：设置日期值头部 ····385
14.7.12 setHeader 方法：设置字符串值头部 ····385
14.7.13 setIntHeader 方法：设置整数值头部 ····385
14.7.14 setStatus 方法：设置状态码 ····386

第 15 章 监听者（Listener） ····387

15.1 ServletContextListener 接口……387
15.1.1 contextInitialized 方法：Web 应用初始化过程开始的通知……387
15.1.2 contextDestroyed 方法：Servlet 上下文将被销毁的通知……388
15.2 ServletContextEvent 类……389
15.2.1 ServletContextEvent：构造函数……389
15.2.2 getServletContext 方法：获取 Servlet 上下文……389
15.3 ServletContextAttributeListener 接口……390
15.3.1 attributeAdded 方法：新属性被添加的通知……390
15.3.2 attributeRemoved 方法：属性已被移除的通知……391
15.3.3 attributeReplaced 方法：属性已被替换的通知……391
15.4 ServletContextAttributeEvent 类……392
15.4.1 ServletContextAttributeEvent：构造函数……392
15.4.2 getServletContext 方法：获取 Servlet 上下文……392
15.4.3 getName 方法：获取属性名……393
15.4.4 getValue 方法：获取属性值……393
15.5 HttpSessionListener 接口……393
15.5.1 sessionCreated 方法：会话被创建的通知……393
15.5.2 sessionDestroyed 方法：会话将无效的通知……394
15.6 HttpSessionActivationListener 接口……394
15.6.1 sessionDidActivate 方法：会话变为有效状态的通知……395
15.6.2 sessionWillPassivate 方法：会话变为无效状态的通知……395
15.7 HttpSessionEvent 类……395
15.7.1 HttpSessionEvent：构造函数……396
15.7.2 getSession 方法：获取 HttpSession 对象……396
15.8 HttpSessionAttributeListener 接口……397
15.8.1 attributeAdded 方法：属性已添加的通知……397
15.8.2 attributeRemoved 方法：属性已移除的通知……398
15.8.3 attributeReplaced 方法：属性已替换的通知……399
15.9 HttpSessionBindingListener 接口……399
15.9.1 valueBound 方法：对象绑定到会话的通知……400
15.9.2 valueUnbound 方法：对象解除绑定的通知……400
15.10 HttpSessionBindingEvent 类……401
15.10.1 HttpSessionBindingEvent：构造函数……401
15.10.2 getName 方法：获取属性名……401
15.10.3 getSession 方法：获取 HttpSession 对象……402
15.10.4 getValue 方法：获取属性值……402
15.11 ServletRequestListener 接口……402
15.11.1 requestInitialized 方法：请求将要进入 Web 应用范围的通知……402
15.11.2 requestDestroyed 方法：请求将要离开 Web 应用范围的通知……403
15.12 ServletRequestEvent 类……404
15.12.1 ServletRequestEvent：构造函数……404
15.12.2 getServletContext 方法：获取 Servlet 上下文……404
15.12.3 getServletRequest 方法：获取 ServletRequest 对象……404
15.13 ServletRequestAttributeListener 接口……404

15.13.1 attributeAdded 方法：新属性被添加的通知 405
15.13.2 attributeRemoved 方法：属性已被移除的通知 405
15.13.3 attributeReplaced 方法：属性已被替换的通知 405
15.14 ServletRequestAttributeEvent 类 407
15.14.1 ServletRequestAttributeEvent：构造函数 407
15.14.2 getName 方法：获取属性名 407
15.14.3 getValue 方法：获取属性值 407

第 16 章 JavaBean 与开发模型 408
16.1 JavaBean 组件 408
16.2 JSP 与 JavaBean 409
16.2.1 <jsp:useBean>动作：声明 JavaBean 对象 409
16.2.2 <jsp:setProperty>动作：设置 JavaBean 的属性 410
16.2.3 <jsp:getProperty>动作：获取 JavaBean 的属性 410
16.3 自省（Introspection）机制 411
16.4 JavaBean 的范围 412
16.4.1 page：页面作用域范围 413
16.4.2 request：请求作用域范围 413
16.4.3 session：会话作用域范围 414
16.4.4 application：应用作用域范围 415
16.5 JSP 开发模型 416
16.5.1 Model 1：JSP+JavaBeans 架构 416
16.5.2 Model 2：MVC 架构 421

第 4 篇 JSP 数据库技术篇

第 17 章 结构化查询语言（SQL） 427
17.1 SQL 概述 427
17.2 数据定义语言（DDL） 427
17.2.1 CREATE DATABASE 命令：创建数据库 427
17.2.2 DROP DATABASE 命令：删除数据库 428
17.2.3 CREATE TABLE 命令：创建基本表 428
17.2.4 ALTER TABLE 命令：修改基本表 428
17.2.5 DROP TABLE 命令：删除基本表 429
17.2.6 CREATE INDEX 命令：创建索引 429
17.2.7 DROP INDEX 命令：删除索引 430
17.2.8 CREATE VIEW 命令：创建视图 430
17.2.9 DROP VIEW 命令：删除视图 430
17.3 数据查询语言（DQL） 430
17.3.1 SELECT 命令：检索数据 431
17.3.2 INTO 子句：输出查询结果到指定位置 432
17.3.3 WHERE 子句：指定条件语句 432
17.3.4 GROUP BY 子句：指定分组语句 433
17.3.5 HAVING 子句：指定过滤条件 433
17.3.6 ORDER BY 子句：指定排序语句 434

17.4 数据操纵语言（DML）……434
17.4.1 INSERT 命令：添加记录……434
17.4.2 UPDATE 命令：更新记录……435
17.4.3 DELETE 命令：删除记录……435
17.5 数据控制语言（DCL）……435
17.5.1 GRANT 命令：授予用户权限……435
17.5.2 REVOKE 命令：撤销用户权限……436
17.5.3 COMMIT 命令：提交事务……436
17.5.4 ROLLBACK 命令：回滚事务……436

第 18 章 Java 数据库连接（JDBC）……437
18.1 JDBC 驱动程序类型……437
18.1.1 Type1：JDBC-ODBC 桥接驱动程序……437
18.1.2 Type2：部分原生 API 驱动程序……438
18.1.3 Type3：JDBC 网络纯 Java 驱动程序……438
18.1.4 Type4：原生协议纯 Java 驱动程序……438
18.2 MySQL 数据库……439
18.2.1 安装 MySQL……439
18.2.2 配置 MySQL……440
18.3 JDBC 访问 MySQL……442
18.4 Connection 对象……445
18.4.1 TRANSACTION_NONE 字段：不支持事务……445
18.4.2 TRANSACTION_READ_COMMITTED 字段：读已提交事务……445
18.4.3 TRANSACTION_READ_UNCOMMITTED 字段：读未提交事务……445
18.4.4 TRANSACTION_REPEATABLE_READ 字段：可重复读事务……445
18.4.5 TRANSACTION_SERIALIZABLE：可串行化事务……446
18.4.6 clearWarnings 方法：清除所有警告……446
18.4.7 close 方法：关闭连接……446
18.4.8 commit 方法：提交事务……446
18.4.9 createStatement 方法：创建一个 Statement 对象……447
18.4.10 getAutoCommit 方法：获取自动提交模式……447
18.4.11 getCatalog 方法：获取当前目录……448
18.4.12 getHoldability 方法：获取可保存性……448
18.4.13 getMetaData 方法：获取 DatabaseMetaData 对象……448
18.4.14 getTransactionIsolation 方法：获取事务隔离级别……449
18.4.15 getTypeMap 方法：获取类型映射……449
18.4.16 getWarnings 方法：获取警告……449
18.4.17 isClosed 方法：判断是否已关闭……450
18.4.18 isReadOnly 方法：判断是否只读……450
18.4.19 nativeSQL 方法：转换 SQL 语句为原生语法……450
18.4.20 prepareCall 方法：创建一个 CallableStatement 对象……450
18.4.21 prepareStatement 方法：创建一个 PreparedStatement 对象……451
18.4.22 releaseSavepoint 方法：移除 Savepoint 对象……452
18.4.23 rollback 方法：回滚事务……452

18.4.24 setAutoCommit 方法：设置自动提交模式 ····453
18.4.25 setCatalog 方法：设置目录名 ····453
18.4.26 setHoldability 方法：设置可保存性 ····453
18.4.27 setReadOnly 方法：设置只读模式 ····454
18.4.28 setSavepoint 方法：设置 Savepoint 对象 ····454
18.4.29 setTransactionIsolation 方法：设置事务隔离级别 ····454
18.4.30 setTypeMap 方法：设置类型映射 ····455
18.5 Statement 对象 ····455
18.5.1 CLOSE_ALL_RESULTS 字段：关闭所有结果 ····455
18.5.2 CLOSE_CURRENT_RESULT 字段：关闭当前结果 ····455
18.5.3 EXECUTE_FAILED 字段：执行失败 ····455
18.5.4 KEEP_CURRENT_RESULT 字段：保持当前结果 ····456
18.5.5 NO_GENERATED_KEYS 字段：无生成的键 ····456
18.5.6 RETURN_GENERATED_KEYS 字段：返回生成的键 ····456
18.5.7 SUCCESS_NO_INFO 字段：成功但无信息 ····456
18.5.8 addBatch 方法：添加批处理 ····457
18.5.9 cancel 方法：取消 Statement 对象 ····457
18.5.10 clearBatch 方法：清除批处理 ····457
18.5.11 clearWarnings 方法：清除所有警告 ····458
18.5.12 close 方法：关闭语句 ····458
18.5.13 execute 方法：执行 SQL 语句 ····458
18.5.14 executeBatch 方法：执行批处理 ····459
18.5.15 executeQuery 方法：执行查询语句 ····459
18.5.16 executeUpdate 方法：执行更新语句 ····460
18.5.17 getConnection 方法：获取 Connection 对象 ····460
18.5.18 getFetchDirection 方法：返回获取方向 ····461
18.5.19 getFetchSize 方法：返回获取大小 ····461
18.5.20 getGeneratedKeys 方法：获取自动生成的键 ····461
18.5.21 getMaxFieldSize 方法：获取最大字段大小 ····461
18.5.22 getMaxRows 方法：获取最大行数 ····462
18.5.23 getMoreResults 方法：获取更多结果 ····462
18.5.24 getQueryTimeout 方法：获取查询超时 ····462
18.5.25 getResultSet 方法：获取 ResultSet 对象 ····463
18.5.26 getResultSetConcurrency 方法：获取结果集并发性 ····463
18.5.27 getResultSetHoldability 方法：获取结果集的可保存性 ····463
18.5.28 getResultSetType 方法：获取结果集类型 ····463
18.5.29 getUpdateCount 方法：获取更新计数 ····464
18.5.30 getWarnings 方法：获取警告 ····464
18.5.31 setCursorName 方法：设置指针名 ····464
18.5.32 setEscapeProcessing 方法：设置转义处理 ····464
18.5.33 setFetchDirection 方法：设置获取方向 ····465
18.5.34 setFetchSize 方法：设置获取大小 ····465
18.5.35 setMaxFieldSize 方法：设置最大字段大小 ····465
18.5.36 setMaxRows 方法：设置最大行数 ····466

18.5.37 setQueryTimeout 方法：设置查询超时 ····466
18.6 PreparedStatement 对象 ····466
18.6.1 addBatch 方法：添加批处理 ····467
18.6.2 clearParameters 方法：清除参数 ····467
18.6.3 execute 方法：执行 SQL 语句 ····467
18.6.4 executeQuery 方法：执行查询语句 ····468
18.6.5 executeUpdate 方法：执行更新语句 ····468
18.6.6 getMetaData 方法：获取 ResultSetMetaData 对象 ····468
18.6.7 getParameterMetaData 方法：获取 ParameterMetaData 对象 ····469
18.6.8 setArray 方法：设置参数为给定的 Array 对象 ····470
18.6.9 setAsciiStream 方法：设置参数为给定的 InputStream 对象 ····470
18.6.10 setBigDecimal 方法：设置参数为给定的 BigDecimal 值 ····471
18.6.11 setBinaryStream 方法：设置参数为给定的 InputStream 对象 ····471
18.6.12 setBlob 方法：设置参数为给定的 Blob 对象 ····471
18.6.13 setBoolean 方法：设置参数为给定的 boolean 值 ····472
18.6.14 setByte 方法：设置参数为给定的 byte 值 ····472
18.6.15 setBytes 方法：设置参数为给定的字节数组 ····472
18.6.16 setCharacterStream 方法：设置参数为给定的 Reader 对象 ····473
18.6.17 setClob 方法：设置参数为给定的 Clob 对象 ····473
18.6.18 setDate 方法：设置参数为给定的 Date 值 ····473
18.6.19 setDouble 方法：设置参数为给定的 double 值 ····474
18.6.20 setFloat 方法：设置参数为给定的 float 值 ····474
18.6.21 setInt 方法：设置参数为给定的 int 值 ····474
18.6.22 setLong 方法：设置参数为给定的 long 值 ····475
18.6.23 setNull 方法：设置参数为 NULL ····475
18.6.24 setObject 方法：设置参数为给定的对象 ····475
18.6.25 setRef 方法：设置参数为给定的 REF 值 ····476
18.6.26 setShort 方法：设置参数为给定的 short 值 ····476
18.6.27 setString 方法：设置参数为给定的 String 值 ····477
18.6.28 setTime 方法：设置参数为给定的 Time 值 ····477
18.6.29 setTimestamp 方法：设置参数为给定的 Timestamp 值 ····477
18.6.30 setUnicodeStream 方法：设置参数为给定的 InputStream 对象 ····478
18.6.31 setURL 方法：设置参数为给定的 URL 对象 ····478
18.7 CallableStatement 对象 ····478
18.7.1 getArray 方法：获取 Array 对象 ····479
18.7.2 getBigDecimal 方法：获取 BigDecimal 值 ····479
18.7.3 getBlob 方法：获取 Blob 对象 ····479
18.7.4 getBoolean 方法：获取 boolean 值 ····480
18.7.5 getByte 方法：获取 byte 值 ····480
18.7.6 getBytes 方法：获取字节数组 ····480
18.7.7 getClob 方法：获取 Clob 对象 ····481
18.7.8 getDate 方法：获取 Date 值 ····481
18.7.9 getDouble 方法：获取 double 值 ····482
18.7.10 getFloat 方法：获取 float 值 ····482

18.7.11 getInt 方法：获取 int 值 ····482
18.7.12 getLong 方法：获取 long 值 ····483
18.7.13 getObject 方法：获取对象 ····483
18.7.14 getRef 方法：获取 REF 值 ····484
18.7.15 getShort 方法：获取 short 值 ····484
18.7.16 getString 方法：获取 String 值 ····484
18.7.17 getTime 方法：获取 Time 值 ····485
18.7.18 getTimestamp 方法：获取 Timestamp 值 ····485
18.7.19 getURL 方法：获取 URL 对象 ····486
18.7.20 registerOutParameter 方法：注册输出参数 ····486
18.7.21 setAsciiStream 方法：设置参数为给定的 InputStream 对象 ····487
18.7.22 setBigDecimal 方法：设置参数为给定的 BigDecimal 值 ····488
18.7.23 setBinaryStream 方法：设置参数为给定的 InputStream 对象 ····488
18.7.24 setBoolean 方法：设置参数为给定的 boolean 值 ····488
18.7.25 setByte 方法：设置参数为给定的 byte 值 ····489
18.7.26 setBytes 方法：设置参数为给定的字节数组 ····489
18.7.27 setCharacterStream 方法：设置参数为给定的 Reader 对象 ····489
18.7.28 setDate 方法：设置参数为给定的 Date 值 ····490
18.7.29 setDouble 方法：设置参数为给定的 double 值 ····490
18.7.30 setFloat 方法：设置参数为给定的 float 值 ····490
18.7.31 setInt 方法：设置参数为给定的 int 值 ····491
18.7.32 setLong 方法：设置参数为给定的 long 值 ····491
18.7.33 setNull 方法：设置参数为 NULL ····491
18.7.34 setObject 方法：设置参数为给定的对象 ····492
18.7.35 setShort 方法：设置参数为给定的 short 值 ····492
18.7.36 setString 方法：设置参数为给定的 String 值 ····493
18.7.37 setTime 方法：设置参数为给定的 Time 值 ····493
18.7.38 setTimestamp 方法：设置参数为给定的 Timestamp 值 ····493
18.7.39 setURL 方法：设置参数为给定的 URL 对象 ····494
18.7.40 wasNull 方法：判断是否为 null ····494
18.8 ResultSet 对象 ····495
18.8.1 CLOSE_CURSORS_AT_COMMIT 字段：提交事务时关闭游标 ····495
18.8.2 CONCUR_READ_ONLY 字段：只读并发模式 ····495
18.8.3 CONCUR_UPDATABLE 字段：可更新并发模式 ····495
18.8.4 FETCH_FORWARD 字段：向前获取 ····495
18.8.5 FETCH_REVERSE 字段：向后获取 ····496
18.8.6 FETCH_UNKNOWN 字段：未知方向获取 ····496
18.8.7 HOLD_CURSORS_OVER_COMMIT 字段：提交事务时保持游标 ····496
18.8.8 TYPE_FORWARD_ONLY 字段：仅向前的游标类型 ····496
18.8.9 TYPE_SCROLL_INSENSITIVE 字段：可滚动且不受影响的对象类型 ····496
18.8.10 TYPE_SCROLL_SENSITIVE 字段：可滚动且受影响的对象类型 ····497
18.8.11 absolute 方法：移动到指定行 ····497
18.8.12 afterLast 方法：移动到尾行之后 ····497
18.8.13 beforeFirst 方法：移动到首行之前 ····498

18.8.14 cancelRowUpdates 方法：取消当前行的更新 ····498
18.8.15 clearWarnings 方法：清除所有警告 ····498
18.8.16 close 方法：关闭对象 ····498
18.8.17 deleteRow 方法：删除当前行 ····499
18.8.18 findColumn 方法：查找列 ····499
18.8.19 first 方法：移动到首行 ····499
18.8.20 getArray 方法：获取 Array 对象 ····499
18.8.21 getAsciiStream 方法：获取 ASCII 字符流 ····500
18.8.22 getBigDecimal 方法：获取 BigDecimal 值 ····500
18.8.23 getBinaryStream 方法：获取二进制流 ····500
18.8.24 getBlob 方法：获取 Blob 对象 ····501
18.8.25 getBoolean 方法：获取 boolean 值 ····501
18.8.26 getByte 方法：获取 byte 值 ····501
18.8.27 getBytes 方法：获取字节数组 ····502
18.8.28 getCharacterStream 方法：获取 Reader 对象 ····502
18.8.29 getClob 方法：获取 Clob 对象 ····502
18.8.30 getConcurrency 方法：获取并发模式 ····503
18.8.31 getCursorName 方法：获取游标名称 ····503
18.8.32 getDate 方法：获取 Date 值 ····503
18.8.33 getDouble 方法：获取 double 值 ····504
18.8.34 getFetchDirection 方法：返回获取方向 ····504
18.8.35 getFetchSize 方法：返回获取大小 ····504
18.8.36 getFloat 方法：获取 float 值 ····505
18.8.37 getInt 方法：获取 int 值 ····505
18.8.38 getLong 方法：获取 long 值 ····505
18.8.39 getMetaData 方法：获取 ResultSetMetaData 对象 ····506
18.8.40 getObject 方法：获取对象 ····506
18.8.41 getRef 方法：获取 REF 值 ····507
18.8.42 getRow 方法：获取当前行编号 ····507
18.8.43 getShort 方法：获取 short 值 ····507
18.8.44 getStatement 方法：获取 Statement 对象 ····507
18.8.45 getString 方法：获取 String 值 ····508
18.8.46 getTime 方法：获取 Time 值 ····508
18.8.47 getTimestamp 方法：获取 Timestamp 值 ····509
18.8.48 getType 方法：获取对象类型 ····509
18.8.49 getURL 方法：获取 URL 对象 ····509
18.8.50 getWarnings 方法：获取警告 ····510
18.8.51 insertRow 方法：插入行 ····510
18.8.52 isAfterLast 方法：是否位于尾行之后 ····510
18.8.53 isBeforeFirst 方法：是否位于首行之前 ····511
18.8.54 isFirst 方法：是否位于首行 ····511
18.8.55 isLast 方法：是否位于尾行 ····511
18.8.56 last 方法：移动到尾行 ····511
18.8.57 moveToCurrentRow 方法：移动到当前行 ····512

18.8.58 moveToInsertRow 方法：移动到插入行……512
18.8.59 next 方法：下移一行……512
18.8.60 previous 方法：上移一行……512
18.8.61 refreshRow 方法：刷新行……513
18.8.62 relative 方法：移动相对行数……513
18.8.63 rowDeleted 方法：行已被删除……513
18.8.64 rowInserted 方法：行已被插入……514
18.8.65 rowUpdated 方法：行已被更新……514
18.8.66 setFetchDirection 方法：设置获取方向……514
18.8.67 setFetchSize 方法：设置获取行数……514
18.8.68 updateArray 方法：用 Array 对象更新指定列……515
18.8.69 updateAsciiStream 方法：用 ASCII 流更新指定列……515
18.8.70 updateBigDecimal 方法：用 BigDecimal 值更新指定列……515
18.8.71 updateBinaryStream 方法：用二进制流更新指定列……516
18.8.72 updateBlob 方法：用 Blob 对象更新指定列……516
18.8.73 updateBoolean 方法：用 boolean 值更新指定列……517
18.8.74 updateByte 方法：用 byte 值更新指定列……517
18.8.75 updateBytes 方法：用字节数组更新指定列……517
18.8.76 updateCharacterStream 方法：用字符流更新指定列……518
18.8.77 updateClob 方法：用 Clob 对象更新指定列……518
18.8.78 updateDate 方法：用 Date 值更新指定列……518
18.8.79 updateDouble 方法：用 double 值更新指定列……519
18.8.80 updateFloat 方法：用 float 值更新指定列……519
18.8.81 updateInt 方法：用 int 值更新指定列……520
18.8.82 updateLong 方法：用 long 值更新指定列……520
18.8.83 updateNull 方法：用 null 更新指定列……520
18.8.84 updateObject 方法：用 Object 对象更新指定列……521
18.8.85 updateRef 方法：用 REF 值更新指定列……521
18.8.86 updateRow 方法：更新行……522
18.8.87 updateShort 方法：用 short 值更新指定列……522
18.8.88 updateString 方法：用 String 值更新指定列……522
18.8.89 updateTime 方法：用 Time 值更新指定列……523
18.8.90 updateTimestamp 方法：用 Timestamp 值更新指定列……523
18.8.91 wasNull 方法：判断是否为 null……523
附录 名词解释……524

第 9 章

XML 标记库（XML Tag Library）

XML 标记库（XML Tag Library）包含有关 XML 处理的动作。使用 XML 标记库时，必须使用 taglib 指令，并设定 prefix 和 uri 属性，通常设置如下：

```
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>
```

XML 标记库共有 10 个动作，包括 parse、out、set、if、when、choose、otherwise、forEach、transform 和 param。

9.1 <x:parse>动作

<x:parse>动作用于解析一个 XML 文档的内容，其语法格式如下：

```
<x:parse
    [doc="xmlDocument" | xml="xmlDocument"]
    [systemId="systemId"]
    [filter="filter"]
    var="varName"  [scope="page|request|session|application"]  />
或
<x:parse
    [systemId="systemId"]
    [filter="filter"]
    var="varName"  [scope="page|request|session|application"]>
    xmlDocument
</x:parse>
或
<x:parse
    [doc="xmlDocument" | xml="xmlDocument"]
    [systemId="systemId"]
    [filter="filter"]
    varDom="domName"  [scopeDom="page|request|session|application"]  />
或
<x:parse
    [systemId="systemId"]
    [filter="filter"]
    varDom="domName"  [scopeDom="page|request|session|application"]>
    xmlDocument
</x:parse>
```

<x:parse>动作共有 8 个属性，包括 doc、xml、systemId、filter、varDom、ScopeDom、var 和 scope。

9.1.1 doc 属性：指定 XML 文档

【功能说明】doc 属性用于指定要解析的源 XML 文档，可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<c:import  url="txl.xml"  charEncoding="GB2312"  var="xmldoc"  />
<x:parse  doc="${xmldoc}"  var="doc"  />
```

其中，txl.xml 文档内容如下：

```
<?xml version="1.0" encoding="GB2312"?>
<linkmans>
  <linkman id="1">
    <name>霸王丸</name>
    <sex>男</sex>
    <birthday>1965-05-04</birthday>
    <mobile>13112943800</mobile>
    <phone>022-62590099</phone>
    <email>babel@laser.com</email>
  </linkman>
  <linkman id="2">
    <name>桔右京</name>
    <sex>男</sex>
    <birthday>1966-11-12</birthday>
    <mobile>13399287400</mobile>
    <phone>010-65118888</phone>
    <email>jacinth@sanfen.com</email>
    </linkman>
```

```
    <linkman id="3">
      <name>娜可露露</name>
      <sex>女</sex>
      <birthday>1980-04-19</birthday>
      <mobile>13214095045</mobile>
      <phone>0411-2809705</phone>
      <email>naker@soldier.com</email>
    </linkman>
    <linkman id="4">
      <name>服布半藏</name>
      <sex>男</sex>
      <birthday>1977-09-09</birthday>
      <mobile>13020041126</mobile>
      <phone>0513-3235776</phone>
      <email>fubsy@wyxfive.com.cn</email>
    </linkman>
    <linkman id="5">
      <name>加尔福特</name>
      <sex>男</sex>
      <birthday>1971-01-01</birthday>
      <mobile>13906144105</mobile>
      <phone>0874-5261044</phone>
      <email>jackey@falvxyxy.net</email>
    </linkman>
    <linkman id="6">
      <name>千两狂死郎</name>
      <sex>男</sex>
      <birthday>1972-03-04</birthday>
      <mobile>13309094452</mobile>
      <phone>0577-7219756</phone>
      <email>quadron@igresve.com.jp</email>
    </linkman>
    <linkman id="7">
      <name>牙神幻十郎</name>
      <sex>男</sex>
      <birthday>1973-04-05</birthday>
      <mobile>13301140758</mobile>
      <phone>0738-5814854</phone>
      <email>ystem@beelinkss.net</email>
    </linkman>
</linkmans>
```

示例代码执行后，将 txl.xml 文档的内容解析并保存到 doc 对象中。

9.1.2 xml 属性：指定 XML 文档

【功能说明】xml 属性用于指定要解析的源 XML 文档，不再推荐使用，该属性已被 doc 属性所替代。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<c:import url="txl.xml" charEncoding="GB2312" var="xmldoc" />
<x:parse xml="${xmldoc}" var="doc" />
```

9.1.3 systemId 属性：指定系统标识

【功能说明】systemId 属性用于为要解析的 XML 文档指定系统标识 URI，可以接受动态值。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<x:parse var="doc" systemId="txl.dtd">
```

```
        <c:import    url="txl.xml"    charEncoding="GB2312"    />
</x:parse>
```

其中，txl.dtd 文档的内容如下：

```
<?xml version="1.0" encoding="GB2312"?>

<!ELEMENT linkmans (linkman)+>

<!ELEMENT linkman (name,sex,birthday,mobile,phone,email*)>
<!ATTLIST linkman id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT birthday (#PCDATA)>
<!ELEMENT mobile (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

示例代码执行后，将根据 txl.dtd 对 txl.xml 文档进行解析，并将结果保存到 doc 对象中。

9.1.4 filter 属性：指定过滤器

【功能说明】filter 属性用于指定适用于源文档的过滤器，其类型为 org.xml.sax.XMLFilter。filter 属性可以接受动态值。

9.1.5 varDom 属性：指定 DOM 变量名

【功能说明】varDom 属性用于指定保存解析结果的 DOM 变量名，其类型为 org.w3c.dom.Document。varDom 属性不可以接受动态值，示例代码参见 9.1.6 小节。

9.1.6 scopeDom 属性：指定 DOM 变量作用域

【功能说明】scopeDom 属性用于指定 DOM 变量的作用域范围，可选的作用域如下。

□ page，范围为页面作用域。

□ request，范围为请求作用域。

□ session，范围为会话作用域。

□ application，范围为应用作用域。

scopeDom 属性不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  varDom="doc"  scopeDom="application">
    <c:import  url="txl.xml"  charEncoding="GB2312"  />
</x:parse>

<c:import  var="stylesheet"  url="txl.xsl"  charEncoding="GB2312"  />
<x:transform  doc="${doc}"  xslt="${stylesheet}"  />
```

其中，XSTL 样式文件 txl.xsl 的内容如下：

```
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="linkmans">
        <table border="1" width="100%">
            <tr>
                <th>编号</th>
                <th>姓名</th>
                <th>移动电话</th>
                <th>Email</th>
            </tr>
            <xsl:for-each select="linkman">
                <tr>
                    <td align="center">
                        <xsl:value-of select="@id"/>
                    </td>
                    <td align="center">
                        <xsl:value-of select="name"/>
                    </td>
                    <td align="center">
                        <xsl:value-of select="mobile"/>
                    </td>
                    <td align="right">
                        <xsl:value-of select="email"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </xsl:template>
</xsl:stylesheet>
```

示例代码的执行效果如图 9.1 所示。

图 9.1　使用<x:parse>动作的 varDom 和 scopeDom 属性

9.1.7　var 属性：指定变量名

【功能说明】var 属性用于指定保存解析结果的变量名称，变量的类型依赖于实现。通常，var 属性是必需的。var 属性不可以接受动态值，示例代码参见 9.1.8 小节。

9.1.8　scope 属性：指定变量作用域

【功能说明】scope 属性用于指定变量的作用域范围，如 page、request、session 和 application 等。scope 属性不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="doc"  scope="application">
  <c:import  url="txl.xml"  charEncoding="GB2312"  />
</x:parse>

<table>
  <tr align=center>
    <td><B>姓名</B></td>
    <td><B>生日</B></td>
    <td><B>移动电话</B></td>
    <td><B>Email</B></td>
  </tr>
  <x:forEach  select="$doc/linkmans/linkman">
  <tr>
    <td align=center><x:out  select="name" /></td>
    <td align=right><x:out  select="birthday" /></td>
    <td align=right><x:out  select="mobile" /></td>
    <td align=right><x:out  select="email" /></td>
  </tr>
  </x:forEach>
</table>
```

示例代码的执行效果如图 9.2 所示。

图 9.2　使用<x:parse>动作的 var 和 scope 属性

9.2　<x:out>动作

<x:out>动作类似于<%= … >指令，用于输出 XPath 表达式的计算结果。<x:out>动作的语法格式如下：

```
<x:out    select="XPathExpression"
      [escapeXml="true|false"]    />
```

<x:out>动作共有两个属性，即 select 和 escapeXml。

9.2.1　select 属性：指定 XPath 表达式

【功能说明】select 属性用于指定要计算的 XPath 表达式，不可以接受动态值。

【实例演示】

```
<%@    page    contentType="text/html;charset=GB2312"    %>
<%@    taglib    prefix="c"    uri="http://java.sun.com/jsp/jstl/core"    %>
<%@    taglib    prefix="x"    uri="http://java.sun.com/jsp/jstl/xml"    %>

<x:parse    var="txl"    scope="application">
      <c:import    url="txl.xml"    charEncoding="GB2312"    />
</x:parse>

联系人（id=1）：</br>
<x:out    select="$txl//linkman"    /></br></br>
联系人（id=3）：</br>
<x:out    select="$txl/linkmans/linkman[@id=3]"    /></br>
```

示例代码的执行效果如图 9.3 所示。

图 9.3　使用<x:out>动作的 select 属性

9.2.2 escapeXml 属性：是否转换 XML 字符

【功能说明】escapeXml 属性如果为 true，则结果字符串中的“ < ”、“ > ”、“ & ”、“ ’ ”和“ ” ”等字符将转换为相应的 XML 字符实体码，否则保持不变。escapeXml 属性可以接受动态值。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<x:parse var="txl" scope="application">
    <c:import url="txl.xml" charEncoding="GB2312" />
</x:parse>

<x:out select="$txl//url" escapeXml="true" />
```

示例代码的执行效果如图 9.4 所示。

图 9.4 使用<x:out>动作的 escapeXml 属性

查看源代码，可以看到 url 中的字符已转换成相应的 XML 字符实体码，源代码如下：

```
http://localhost:8080/txl/login.jsp?name=forrid&amp;pass=1&#039;3&#034;5
```

如果将 escapeXml 属性设置为 false，则查看源代码如下：

```
http://localhost:8080/txl/login.jsp?name=forrid&pass=1'3"5
```

9.3 <x:set>动作

<x:set>动作用于将一个 XPath 表达式的计算结果保存到一个作用域变量中，其语法格式如下：

```
<x:set select="XPathExpression"
    var="varName" [scope="page|request|session|application"] />
```

<x:set>动作共有 3 个属性，包括 select、var 和 scope。

9.3.1 select 属性：指定 XPath 表达式

【功能说明】select 属性用于指定要计算的 XPath 表达式，不可以接受动态值。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<x:parse var="txl" scope="application">
    <c:import url="txl.xml" charEncoding="GB2312" />
</x:parse>

<x:set var="name" select="$txl//name" />
<x:set var="linkman" select="$txl//linkman" />
```

示例代码执行后，变量 name 中保存的内容为“ 霸王丸 ”，而变量 linkman 中保存的内容如下：

```
<linkman id="1">
  <name>霸王丸</name>
  <sex>男</sex>
  <birthday>1965-05-04</birthday>
  <mobile>13112943800</mobile>
  <phone>022-62590099</phone>
  <email>babel@laser.com</email>
</linkman>
```

要取出 XML 中的内容参见 9.3.3 小节。

9.3.2 var 属性：指定变量名

【功能说明】var 属性用于指定变量的名称，该变量从 XML 文档中取得内容，不可以接受动态值。示例代码参见 9.3.3 小节。

9.3.3 scope 属性：指定变量作用域

【功能说明】scope 属性用于指定变量的作用域范围，如 page、request、session 和 application 等。scope 属性不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="txl"  scope="application">
    <c:import  url="txl.xml"  charEncoding="GB2312"  />
</x:parse>

<x:set  var="linkman"  select="$txl//linkman"  />

<table>
  <tr>
    <td align=right>姓  名：</td>
    <td><x:out  select="$linkman//name"  /></td>
  </tr>
  <tr>
    <td align=right>性  别：</td>
    <td><x:out  select="$linkman//sex"  /></td>
  </tr>
  <tr>
    <td align=right>生  日：</td>
    <td><x:out  select="$linkman//birthday"  /></td>
  </tr>
  <tr>
    <td align=right>移动电话：</td>
    <td><x:out  select="$linkman//mobile"  /></td>
  </tr>
  <tr>
    <td align=right>固定电话：</td>
    <td><x:out  select="$linkman//phone"  /></td>
  </tr>
  <tr>
    <td align=right>E-mail：</td>
    <td><x:out  select="$linkman//email"  /></td>
  </tr>
</table>
```

示例代码的执行效果如图 9.5 所示。

图 9.5　使用<x:set>动作的 var 和 scope 属性

9.4　<x:if>动作

<x:if>动作用于表示 XML 条件判断标记，仅当指定的 XPath 表达式计算为 true 时，才计算其动作体，计算结果可以保存为一个作用域变量。<x:if>动作的语法格式如下：

```
<x:if   select="testXPathExpression"
     var="varName"   [scope="page|request|session|application"]   />
或
<x:if   select="testXPathExpression"
     [var="varName"]   [scope="page|request|session|application"]>
     JSP elements
</x:if>
```

<x:if>动作共有 3 个属性，包括 select、var 和 scope。

9.4.1　select 属性：指定测试条件

【功能说明】select 属性用于指定测试条件表达式，以确定是否执行动作体中的内容。select 属性不可以接受动态值。

【实例演示】

```
<%@   page   contentType="text/html;charset=GB2312"   %>
<%@   taglib   prefix="c"   uri="http://java.sun.com/jsp/jstl/core"   %>
<%@   taglib   prefix="x"   uri="http://java.sun.com/jsp/jstl/xml"   %>

<x:parse   var="txl"   scope="application">
     <c:import   url="txl.xml"   charEncoding="GB2312"   />
</x:parse>

<c:set   var="linkmanSex"   value="男"   />

<c:forEach   begin="1"   end="7"   var="linkmanId">
     <x:if   select="$txl/linkmans/linkman[@id=$linkmanId]/sex[. = $linkmanSex]">
          <x:out   select="$txl/linkmans/linkman[@id=$linkmanId]/name"   />的Email地址：
          <x:out   select="$txl/linkmans/linkman[@id=$linkmanId]/email"   /></br>
     </x:if>
</c:forEach>
```

其中，“ $txl/linkmans/linkman[@id=$linkmanId]/sex[. = $linkmanSex] ”的含义是，当 linkman 元素的 id 等于 linkmanId，并且 linkman 的 sex 元素等于 linkmanSex 时才会执行<x:if>动作体中的内容。

示例代码执行后，将显示所有性别为男的联系人的 Email 地址，执行效果如图 9.6 所示。

图 9.6　使用<x:if>动作的 select 属性

9.4.2　var 属性：指定变量名

【功能说明】var 属性用于指定保存测试条件结果的变量名称，其类型为 Boolean。var 属性不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="txl"  scope="application">
    <c:import  url="txl.xml"  charEncoding="GB2312"  />
</x:parse>

<c:set  var="linkmanSex"  value="女"  />

<c:forEach  begin="1"  end="7"  var="linkmanId">
    <x:if  var="isFemale"  select="$txl/linkmans/linkman[@id=$linkmanId]/sex[. = $linkmanSex]"  />
    <c:if  test="${isFemale}">
        <x:out  select="$txl/linkmans/linkman[@id=$linkmanId]/name"  />的电话号码：
        <x:out  select="$txl/linkmans/linkman[@id=$linkmanId]/mobile"  /></br>
    </c:if>
</c:forEach>
```

示例代码的执行效果如图 9.7 所示。

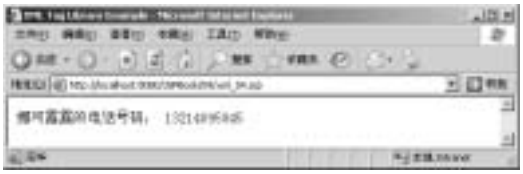

图 9.7　使用<x:if>动作的 var 属性

9.4.3 scope 属性：指定变量作用域

【功能说明】scope 属性用于指定变量的作用域范围，如 page、request、session 和 application 等。scope 属性不可以接受动态值。

【实例演示】

```
<c:forEach  begin="1"  end="7"  var="linkmanId">
    <x:if  var="isFemale"  scope="session"
        select="$txl/linkmans/linkman[@id=$linkmanId]/sex[. = $linkmanSex]"  />
    <c:if  test="${isFemale}">
        <x:out  select="$txl/linkmans/linkman[@id=$linkmanId]/name"  />的电话号码：
        <x:out  select="$txl/linkmans/linkman[@id=$linkmanId]/mobile"  /></br>
    </c:if>
</c:forEach>
```

9.5 <x:when>动作

<x:when>动作用于表示一个<x:choose>动作中的互斥选项之一，仅当该动作的测试表达式计算为 true，且是第一个测试为 true 的<x:when>动作时，才会计算其动作体。<x:when>动作的语法格式如下：

```
<x:when  select="testXPathExpression">
    JSP elements
</x:when>
```

<x:when>动作只有一个 select 属性。

【功能说明】select 属性用于指定测试条件表达式，以确定是否执行动作体中的内容。select 属性不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="txl"  scope="application">
    <c:import  url="txl.xml"  charEncoding="GB2312"  />
</x:parse>

<table border=0 width="100%">
  <tr bgcolor="lightgreen">
    <th>姓名</th>
    <th>性别</th>
    <th>生日</th>
    <th>移动电话</th>
    <th>固定电话</th>
    <th>Email</th>
  </tr>
  <x:forEach  select="$txl/linkmans/linkman">
  <x:choose>
    <x:when  select="sex[. = '女' ]">
      <tr bgcolor="lightblue">
    </x:when>
    <x:otherwise>
      <tr>
    </x:otherwise>
  </x:choose>
    <td align=center><x:out  select="name"  /></td>
    <td align=center><x:out  select="sex"  /></td>
    <td align=center><x:out  select="birthday"  /></td>
    <td align=right><x:out  select="mobile"  /></td>
    <td align=right><x:out  select="phone"  /></td>
    <td align=right><x:out  select="email"  /></td>
  </tr>
  </x:forEach>
</table>
```

示例代码的执行效果如图 9.8 所示。

图 9.8　使用<x:when>动作的 select 属性

9.6　<x:choose>动作

【功能说明】<x:choose>动作用于嵌套使用<x:when>和<x:otherwise>动作标记，其语法格式如下：

```
<x:choose>
    [<x:when   select="testXPathExpression">
        JSP elements
    </x:when>] +
    [<x:otherwise>
        JSP elements
    </x:otherwise>]
</x:choose>
```

<x:choose>动作没有属性。

【实例演示】

```
<%@   page   contentType="text/html;charset=GB2312"   %>
<%@   taglib   prefix="c"   uri="http://java.sun.com/jsp/jstl/core"   %>
<%@   taglib   prefix="fmt"   uri="http://java.sun.com/jsp/jstl/fmt"   %>
<%@   taglib   prefix="x"   uri="http://java.sun.com/jsp/jstl/xml"   %>

<fmt:requestEncoding   value="GB2312" />
<x:parse   var="book"   scope="application">
    <c:import   url="book.xml"   charEncoding="GB2312"   />
</x:parse>

<FORM method="POST" action="xml_061.jsp">
    选择感兴趣的图书：</br>
    <select name="bookname">
        <x:forEach   select="$book/books/book">
            <option><x:out   select="name"   /></option>
        </x:forEach>
    </select>
    <input type="submit" value="查看">
</FORM>
```

其中，xml_061.jsp 的源代码如下：

```
<%@   page   contentType="text/html;charset=GB2312"   %>
<%@   taglib   prefix="c"   uri="http://java.sun.com/jsp/jstl/core"   %>
<%@   taglib   prefix="fmt"   uri="http://java.sun.com/jsp/jstl/fmt"   %>
<%@   taglib   prefix="x"   uri="http://java.sun.com/jsp/jstl/xml"   %>

<fmt:requestEncoding value="GB2312" />

<x:parse   var="book"   scope="application">
    <c:import   url="book.xml"   charEncoding="GB2312"   />
</x:parse>

<x:set   var="selectBook"   select="$book/books/book[name = $param:bookname]"   />

<hr>
```

```
书名：<x:out    select="$selectBook//name"    /></br>
定价：<x:out    select="$selectBook/price/@symbol"    /><x:out    select="$selectBook//price"    /></br>
<x:choose>
    <x:when    select="$selectBook//type[. = '计算机']">
        计算机类图书    <font color=red>七五折</font>
    </x:when>
    <x:when    select="$selectBook//type[. = '文学']">
        文学类图书    <font color=red>八八折</font>
    </x:when>
    <x:otherwise>
        其他类图书    <font color=red>九五折</font>
    </x:otherwise>
</x:choose>
```

其中，book.xml 文档的内容如下：

```
<?xml version="1.0" encoding="GB2312"?>
<books>
    <book>
        <name>《JSP设计》</name>
        <type>计算机</type>
        <price symbol="￥">79.00</price>
    </book>
    <book>
        <name>《精通Struts》</name>
        <type>计算机</type>
        <price symbol="￥">49.00</price>
    </book>
    <book>
        <name>《杞人忧师》</name>
        <type>文学</type>
        <price symbol="￥">16.80</price>
    </book>
    <book>
        <name>《服从》</name>
        <price symbol="￥">15.80</price>
    </book>
    <book>
        <name>《你在忙什么》</name>
        <type>管理</type>
        <price symbol="￥">25.00</price>
    </book>
</books>
```

示例代码的执行效果如图 9.9 所示。

图 9.9　使用<x:choose>动作

9.7　<x:otherwise>动作

【功能说明】<x:otherwise>动作用于表示一个<x:choose>动作中的默认选项，仅当它前面所有的<x:when>动作的测试表达式计算都为 false 时才会计算其本体。<x:otherwise>动作

的语法格式如下：

```
<x:otherwise>
    JSP elements
</x:otherwise>
```

<x:otherwise>动作没有属性。

【实例演示】

```
<x:choose>
    <x:when    select="$selectBook//type[. = '计算机']">
        计算机类图书    <font color=red>七五折</font>
    </x:when>
    <x:when    select="$selectBook//type[. = '文学']">
        文学类图书    <font color=red>八八折</font>
    </x:when>
    <x:otherwise>
        其他类图书    <font color=red>九五折</font>
    </x:otherwise>
</x:choose>
```

示例代码的执行效果参见 9.6 节。

9.8 <x:forEach>动作

<x:forEach>动作用于表示 XML 循环标记，其语法格式如下：

```
<x:forEach    select="XPathExpression"
    [var="varName"]
    [varStatus="varStatus"]
    [begin="startIndex"]    [end="stopIndex"]    [step="increment"]>
    JSP elements
</x:forEach>
```

<x:forEach>动作共有 6 个属性，包括 var、select、varStatus、begin、end 和 step。

9.8.1 var 属性：指定变量名

【功能说明】var 属性用于指定保存当前元素的变量名称，其类型依赖于 select 属性中 XPath 表达式的计算结果。var 属性不可以接受动态值。

【实例演示】

```
<%@    page    contentType="text/html;charset=GB2312"    %>
<%@    taglib    prefix="c"    uri="http://java.sun.com/jsp/jstl/core"    %>
<%@    taglib    prefix="x"    uri="http://java.sun.com/jsp/jstl/xml"    %>

<x:parse    var="book"    scope="application">
    <c:import    url="book.xml"    charEncoding="GB2312"    />
</x:parse>

<table border=0 width="65%">
  <tr bgcolor="lightgreen">
    <th>书名</th>
    <th>类型</th>
    <th>定价</th>
  </tr>
  <x:forEach    var="item"    select="$book/books/book">
  <tr>
    <td align=center><x:out    select="$item//name"    /></td>
    <td align=center><x:out    select="$item//type"    /></td>
    <td align=center><x:out    select="$item//price/@symbol"    /><x:out    select="$item//price"    /></td>
  </tr>
  </x:forEach>
</table>
```

示例代码的执行效果如图 9.10 所示。

图 9.10　使用<x:forEach>动作的 var 属性

9.8.2　select 属性：指定 XPath 表达式

【功能说明】select 属性用于指定要计算的 XPath 表达式，不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="book"  scope="application">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:parse>

<table border=0 width="65%">
  <tr bgcolor="lightgreen">
    <th>书名</th>
    <th>类型</th>
    <th>定价</th>
  </tr>
  <x:forEach  select="$book/books/book">
  <tr>
    <td align=center><x:out  select="name"  /></td>
    <td align=center><x:out  select="type"  /></td>
    <td align=center><x:out  select="price/@symbol"  /><x:out  select="price"  /></td>
  </tr>
  </x:forEach>
</table>
```

示例代码的执行效果参见 9.8.1 小节（请注意代码的差别）。

9.8.3　varStatus 属性：指定状态变量名

【功能说明】varStatus 属性用于指定保存循环状态的变量名称，其类型为 javax.servlet.jsp.jstl.core.LoopTagStatus。varStatus 不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="book"  scope="application">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:parse>

<table border=0 width="75%">
  <tr bgcolor="lightgreen">
    <th>序号</th>
    <th>书名</th>
    <th>类型</th>
    <th>定价</th>
  </tr>
  <x:forEach  varStatus="status"  select="$book/books/book">
  <tr>
    <td align=center><c:out  value="${status.index+1}"  /></td>
    <td align=center><x:out  select="name"  /></td>
    <td align=center><x:out  select="type"  /></td>
```

```
        <td align=center><x:out    select="price/@symbol"    /><x:out    select="price"    /></td>
    </tr>
    </x:forEach>
</table>
```

示例代码的执行效果如图 9.11 所示。

图 9.11　使用<x:forEach>动作的 varStatus 属性

9.8.4 begin 属性：指定开始索引

【功能说明】begin 属性用于指定迭代开始的元素索引，集合中第一个元素的索引为 0。begin 属性可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="book"  scope="application">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:parse>

<table border=0 width="75%">
  <tr bgcolor="lightgreen">
    <th>序号</th>
    <th>书名</th>
    <th>类型</th>
    <th>定价</th>
  </tr>
  <x:forEach  varStatus="status"  select="$book/books/book"  begin="2">
  <tr>
    <td align=center><c:out  value="${status.index+1}"  /></td>
    <td align=center><x:out  select="name"  /></td>
    <td align=center><x:out  select="type"  /></td>
    <td align=center><x:out  select="price/@symbol"  /><x:out  select="price"  /></td>
  </tr>
  </x:forEach>
</table>
```

示例代码的执行效果如图 9.12 所示。

图 9.12　使用<x:forEach>动作的 begin 属性

9.8.5 end 属性：指定结束索引

【功能说明】end 属性用于指定迭代结束的元素索引，可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<x:parse  var="book"  scope="application">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:parse>

<table border=0 width="75%">
  <tr bgcolor="lightgreen">
    <th>序号</th>
    <th>书名</th>
    <th>类型</th>
    <th>定价</th>
  </tr>
  <x:forEach  varStatus="status"  select="$book/books/book"  end="3">
  <tr>
    <td align=center><c:out  value="${status.index+1}"  /></td>
```

```
        <td align=center><x:out    select="name"    /></td>
        <td align=center><x:out    select="type"    /></td>
        <td align=center><x:out    select="price/@symbol"    /><x:out    select="price"    /></td>
    </tr>
    </x:forEach>
</table>
```

示例代码的执行效果如图 9.13 所示。

图 9.13　使用<x:forEach>动作的 end 属性

9.8.6　step 属性：指定迭代步长

【功能说明】step 属性用于指定迭代时索引的递增量，可以接受动态值。

【实例演示】

```
<%@    page    contentType="text/html;charset=GB2312"    %>
<%@    taglib    prefix="c"    uri="http://java.sun.com/jsp/jstl/core"    %>
<%@    taglib    prefix="x"    uri="http://java.sun.com/jsp/jstl/xml"    %>

<x:parse    var="book"    scope="application">
      <c:import    url="book.xml"    charEncoding="GB2312"    />
</x:parse>

<table border=0 width="75%">
    <tr bgcolor="lightgreen">
        <th>序号</th>
        <th>书名</th>
        <th>类型</th>
        <th>定价</th>
    </tr>
    <x:forEach    varStatus="status"    select="$book/books/book"    step="2">
    <tr>
        <td align=center><c:out    value="${status.index+1}"    /></td>
        <td align=center><x:out    select="name"    /></td>
        <td align=center><x:out    select="type"    /></td>
        <td align=center><x:out    select="price/@symbol"    /><x:out    select="price"    /></td>
    </tr>
    </x:forEach>
</table>
```

示例代码的执行效果如图 9.14 所示。

图 9.14　使用<x:forEach>动作的 step 属性

9.9 <x:transform>动作

<x:transform>动作用于使用一个 XSLT 样式表转换一个源 XML 文档，其语法格式如下：

```
<x:transform  doc="XMLDocument"
    xslt="XSLTStylesheet"
    [docSystemId="XMLSystemId"]
    [xsltSystemId="XSLTSystemId"]
    [var="varName"  [scope="page|request|session|application"] | result="resultObject"]  />
或
<x:transform  doc="XMLDocument"
    xslt="XSLTStylesheet"
    [docSystemId="XMLSystemId"]
    [xsltSystemId="XSLTSystemId"]
    [var="varName"  [scope="page|request|session|application"] | result="resultObject"]>
    [<x:param  name="parameterName"  value="parameterValue"  />] +
</x:transform>
或
<x:transform
    xslt="XSLTStylesheet"
    [docSystemId="XMLSystemId"]
    [xsltSystemId="XSLTSystemId"]
    [var="varName"  [scope="page|request|session|application"] | result="resultObject"]>
    XMLDocument
    [<x:param  name="parameterName"  value="parameterValue"  />] +
</x:transform>
```

<x:transform>动作共有 8 个属性，包括 doc、xml、xslt、docSystemId、xsltSystemId、var、scope 和 result。

9.9.1 doc 属性：指定 XML 文档

【功能说明】doc 属性用于指定要转换的源 XML 文档。如果 XML 通过<x:set>动作输出，则必须是格式良好的 XML 文档。doc 属性可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<c:import  var="doc"  url="txl.xml"  charEncoding="GB2312"  />
<c:import  var="stylesheet"  url="txl.xsl"  charEncoding="GB2312"  />

<x:transform  doc="${doc}"  xslt="${stylesheet}"  />
```

示例代码的执行效果如图 9.15 所示。

图 9.15　使用<x:transform>动作的 doc 属性

9.9.2 xml 属性：指定 XML 文档

【功能说明】xml 属性用于指定要转换的 XML 文档，不再推荐使用，它已被 doc 属性所替代。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<c:import var="doc" url="txl.xml" charEncoding="GB2312" />
<c:import var="stylesheet" url="txl.xsl" charEncoding="GB2312" />

<x:transform xml="${doc}" xslt="${stylesheet}" />
```

9.9.3 xslt 属性：指定 XSLT 样式表

【功能说明】xslt 属性用于指定 XSLT 样式表，可以接受动态值。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<c:import var="doc" url="book.xml" charEncoding="GB2312" />
<c:import var="stylesheet" url="book.xsl" charEncoding="GB2312" />

<x:transform doc="${doc}" xslt="${stylesheet}" />
```

其中，XSLT 样式表 book.xsl 的内容如下：

```
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="books">
        <table border="0" width="70%">
            <tr bgcolor="lightblue" >
                <th>书名</th>
                <th>类型</th>
                <th>定价</th>
            </tr>
            <xsl:for-each select="book">
                <tr bgcolor="#DDDDDD">
                    <td align="center">
                        <xsl:value-of select="name"/>
                    </td>
                    <td align="center">
                        <xsl:value-of select="type"/>
                    </td>
                    <td align="right">
                        <xsl:value-of select="price/@symbol"/>
                        <xsl:value-of select="price"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </xsl:template>
</xsl:stylesheet>
```

示例代码的执行效果如图 9.16 所示。

图 9.16　使用<x:transform>动作的 xslt 属性

9.9.4　docSystemId 属性：指定 XML 系统标识

【功能说明】docSystemId 属性用于指定解析的 XML 文档的系统标识 URI，可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<c:import  var="stylesheet"  url="book.xsl"  charEncoding="GB2312"  />
<x:transform  xslt="${stylesheet}"  docSystemId="book.dtd">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:transform>
```

示例代码的执行效果参见 9.9.3 小节。

9.9.5　xsltSystemId 属性：指定 XSLT 系统标识

【功能说明】xsltSystemId 属性用于指定解析的 XSLT 样式表的系统标识 URI，可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<c:import  var="stylesheet"  url="book.xsl"  charEncoding="GB2312"  />

<x:transform  xslt="${stylesheet}"  xsltSystemId="xslt.dtd">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:transform>
```

示例代码的执行效果参见 9.9.3 小节。

9.9.6　var 属性：指定变量名

【功能说明】var 属性用于指定保存转换结果的变量名称，其类型为 org.w3c.dom.Document。var 属性不可以接受动态值。示例代码参见 9.9.7 小节。

9.9.7　scope 属性：指定变量作用域

【功能说明】scope 属性用于指定变量的作用域范围，如 page、request、session 和 application 等。scope 属性不可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<c:set  var="xsl1">
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="books">
        <myBooks>
          <xsl:for-each select="book">
              <myBook>
                  <bookName><xsl:value-of select="name" /></bookName>
                  <bookType><xsl:value-of select="type" />类</bookType>
                  <bookPrice><xsl:value-of select="price/@symbol" />
                          <xsl:value-of select="price" /></bookPrice>
              </myBook>
          </xsl:for-each>
        </myBooks>
    </xsl:template>
</xsl:stylesheet>
</c:set>

<c:set  var="xsl2">
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="myBooks">
        <center>
        <table border="0" width="70%">
            <tr bgcolor="lightblue" >
                <th>书名</th>
                <th>类型</th>
                <th>定价</th>
            </tr>
            <xsl:for-each select="myBook">
                <tr bgcolor="#DDDDDD">
                    <td align="center">
                        <xsl:value-of select="bookName"/>
                    </td>
                    <td align="center">
                        <xsl:value-of select="bookType"/>
                    </td>
                    <td align="right">
                        <xsl:value-of select="bookPrice"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
        </center>
    </xsl:template>
</xsl:stylesheet>
</c:set>

<x:transform  xslt="${xsl1}"  var="temp"  scope="application">
    <c:import  url="book.xml"  charEncoding="GB2312"  />
</x:transform>

<x:transform  xslt="${xsl2}"  doc="${temp}"  />
```

示例代码的执行效果如图 9.17 所示。

图 9.17　使用<x:transform>动作的 var 和 scope 属性

9.9.8 result 属性：指定结果对象

【功能说明】result 属性用于指定捕获或处理转换结果的对象，其类型为 javax.xml.transform.Result。result 属性可以接受动态值。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<c:import var="doc" url="book.xml" charEncoding="GB2312" />
<c:import var="xsl" url="book.xsl" charEncoding="GB2312" />

<x:transform doc="${doc}" xslt="${xsl}" result="${outputResult}" />
```

9.10 <x:param>动作

<x:param>动作用于为转换（Transform）添加一个参数，而且只能用于<x:transform>动作体中。<x:param>动作的语法格式如下：

```
<x:param name="parameterName"
    value="parameterValue" />
或
<x:param name="parameterName">
    parameterValue
</x:param>
```

<x:param>动作共有两个属性，即 name 和 value。

9.10.1 name 属性：指定参数名

【功能说明】name 属性用于指定转换参数的名称，可以接受动态值。

【实例演示】

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<c:import var="doc" url="book.xml" charEncoding="GB2312" />
<c:import var="xsl" url="books.xsl" charEncoding="GB2312" />

<x:transform doc="${doc}" xslt="${xsl}">
    <x:param name="color">
        green
    </x:param>
</x:transform>
```

其中，XSLT 样式表 books.xsl 的内容如下：

```
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:param name="color"/>
    <xsl:template match="books">
        <center>
        <table width="75%">
            <xsl:choose>
                <xsl:when test="$color='blue'">
                    <tr bgcolor="lightblue">
                        <th>书名</th>
                        <th>类型</th>
                        <th>定价</th>
                    </tr>
                </xsl:when>
                <xsl:when test="$color='green'">
                    <tr bgcolor="lightgreen">
                        <th>书名</th>
```

```
                        <th>类型</th>
                        <th>定价</th>
                    </tr>
                </xsl:when>
                <xsl:otherwise>
                    <tr bgcolor="#DDDDDD">
                        <th>书名</th>
                        <th>类型</th>
                        <th>定价</th>
                    </tr>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:for-each select="book">
                <tr bgcolor="#DDDDDD">
                    <td align="center">
                        <xsl:value-of select="name"/>
                    </td>
                    <td align="center">
                        <xsl:value-of select="type"/>
                    </td>
                    <td align="right">
                        <xsl:value-of select="price/@symbol"/>
                        <xsl:value-of select="price"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
        </center>
    </xsl:template>
</xsl:stylesheet>
```

示例代码的执行效果如图 9.18 所示。

图 9.18　使用<x:param>动作的 name 属性

9.10.2　value 属性：指定参数值

【功能说明】value 属性用于指定参数的值，可以接受动态值。

【实例演示】

```
<%@  page  contentType="text/html;charset=GB2312"  %>
<%@  taglib  prefix="c"  uri="http://java.sun.com/jsp/jstl/core"  %>
<%@  taglib  prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"  %>

<c:import  var="doc"  url="book.xml"  charEncoding="GB2312"  />
<c:import  var="xsl"  url="books.xsl"  charEncoding="GB2312"  />

<x:transform  doc="${doc}"  xslt="${xsl}">
    <x:param  name="color"  value="blue"  />
</x:transform>
```

示例代码的执行效果如图 9.19 所示。

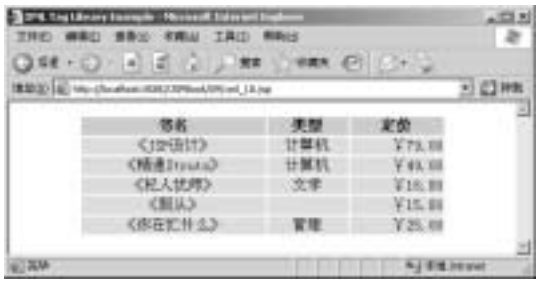

图 9.19　使用<x:param>动作的 value 属性

第 13 章

Servlet

Servlet 是一个运行于 Web 服务器内部的小型 Java 应用程序，用以处理请求和生成响应。Servlet 使用 Servlet API 所定义的接口和类实现，包含 javax.servlet 和 javax.servlet.http 两个包。普通的 Servlet 实现了 javax.servlet.Servlet 接口，可以直接实现，也可以通过扩展某个支持类实现。处理 HTTP 请求的 Servlet 通常扩展 javax.servlet.http.HttpServlet 类。

13.1　Servlet 接口

Servlet 接口是所有 Servlet 必须实现的接口。Servlet 接口定义了生命周期的方法，首先用于初始化 Servlet，然后处理请求，最后从服务器中移除 Servlet。Servlet 的生命周期如下：

□ 构造 Servlet，然后调用 init()方法初始化该对象。

□ 调用 service()方法处理所有来自客户端的请求。

□ 从服务器中移除 Servlet，调用 destroy()方法完成垃圾收集和处理。

此外，Servlet 接口还提供了 getServletConfig()和 getServletInfo()等方法。

13.1.1　init 方法：初始化 Servlet

【功能说明】init()方法用于初始化 Servlet。

【基本语法】

```
public void init (
    ServletConfig config          //  指定ServletConfig对象，该对象包含Servlet的配置和初始化参数
)
    throws ServletException
```

该方法必须在 Servlet 实例化之后才被调用，示例代码参见 13.1.2 小节。

13.1.2　service 方法：处理客户端请求

【功能说明】service()方法用于处理客户端请求。

【基本语法】

```
public void service (
    ServletRequest request,        //  指定ServletRequest对象，该对象包含客户端请求
    ServletResponse response       //  指定ServletResponse对象，该对象包含Servlet响应
)
    throws ServletException, IOException
```

该方法必须在 init()方法成功完成之后才被调用。

【实例演示】

```
package mil.zcz.jsp.servlet;

import java.io.*;
import javax.servlet.*;

public class GreetServlet implements Servlet {
    private String greeting;

    public void destroy() {
        greeting = null;
    }

    public ServletConfig getServletConfig() {
        return null;
    }

    public String getServletInfo() {
        return "Copyright 2006 Wildcat Studio. All rights reserved.";
    }

    public void init(ServletConfig config) throws ServletException {
        greeting = "你好，欢迎进入Servlet世界！";
    }

    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>GreetServlet</title>");
        out.println("</head>");
        out.println("<h2>");
        out.println(greeting);
        out.println("</h2>");
        out.close();
    }
}
```

在 web.xml 中添加以下代码：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <servlet>
        <servlet-name>GreetServlet</servlet-name>
        <servlet-class>mil.zcz.jsp.servlet.GreetServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>GreetServlet</servlet-name>
        <url-pattern>/Greet</url-pattern>
    </servlet-mapping>

</web-app>
```

示例代码的执行效果如图 13.1 所示。

图 13.1　使用 Servlet 接口的 service 方法

13.1.3　destroy 方法：销毁 Servlet

【功能说明】destroy()方法用于销毁 Servlet。

【基本语法】

```
public void destroy ( )
```

调用该方法将清除 Servlet 占用的资源，如内存、文件句柄、线程等。该方法只有当 service()方法中的所有线程均已退出或超时时才被调用，示例代码参见 13.1.2 小节。

13.1.4　getServletConfig 方法：获取 ServletConfig 对象

【功能说明】getServletConfig()方法用于返回 ServletConfig 对象，该对象包含该 Servlet 的初始化和启动参数。

【基本语法】

```
public ServletConfig getServletConfig ( )
```

【实例演示】下面的代码将返回一个 ServletConfig 对象，其值为 null。

```
public ServletConfig getServletConfig() {
    return null;
}
```

13.1.5　getServletInfo 方法：获取 Servlet 信息

【功能说明】getServletInfo()方法用于返回 Servlet 的相关信息。

【基本语法】

```
public String getServletInfo ( )
```

该方法将以字符串的形式返回 Servlet 的相关信息，如作者、版本和版权信息等，该字符串是无格式的文本，并且不支持任何标记（如 HTML、XML 等）。

【实例演示】

```
public String getServletInfo() {
    return "Copyright 2006 Wildcat Studio. All rights reserved.";
}
```

13.2 GenericServlet 类

GenericServlet 类定义了一个普通的、协议无关的 Servlet，实现了 Servlet 和 ServletConfig 等接口。GenericServlet 类简化了 Servlet 的开发，并提供了生命周期方法的简单版本。开发一个普通的 Servlet 必须重载 service()方法。

13.2.1 GenericServlet：构造函数

【基本语法】javax.servlet.GenericServlet 类的构造函数如下：

```
public GenericServlet ( )
```

该构造函数什么也不做，Servlet 所有的初始化都是由 init()方法完成的。

13.2.2 init 方法：初始化 Servlet

【功能说明】init()方法用于初始化 Servlet。

【基本语法】

```
public void init (
    ServletConfig config          //  指定ServletConfig对象，该对象包含Servlet的配置信息
)
    throws ServletException
或
public void init ( )
    throws ServletException
```

当重写 init(ServletConfig)方法时，需要调用 super.init(config)方法；而重写 init()方法时则不需要。示例代码参见 13.2.3 小节。

13.2.3 service 方法：处理客户端请求

【功能说明】service()方法用于处理客户端请求。

【基本语法】

```
public abstract void service (
    ServletRequest req,           //  指定ServletRequest对象，包含客户端请求
    ServletResponse res           //  指定ServletResponse对象，包含Servlet响应
)
    throws ServletException, IOException
```

service()方法是抽象方法，GenericServlet 的子类必须重写该方法。

【实例演示】

```
package mil.zcz.jsp.servlet;

import java.io.*;
import java.sql.*;
import javax.servlet.*;

public class DBServlet extends GenericServlet {
    private String url;

    public void init() throws ServletException {
        url = "jdbc:mysql//localhost:3306/txl?user=root&password=";
    }

    public void destroy() {
        url = null;
    }
```

```
    @Override 重载
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        String user = super.getInitParameter("user");
        String pass = super.getInitParameter("password");
        if ((user != null) && (pass != null)) {
            url = "jdbc:mysql://localhost:3306/txl?user=" + user + "&password=" + pass;
        }
        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();

        try {
            try {
                Class.forName("com.mysql.jdbc.Driver");
            } catch (ClassNotFoundException ce) {
                System.out.println(ce);
            }

            Connection conn = DriverManager.getConnection(url);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM linkman");
            out.println("<table border=1><tr>");
            out.println("<th>姓名</th>");
            out.println("<th>生日</th>");
            out.println("<th>电话</th>");
            out.println("<th>Email</th></tr>");
            while (rs.next()) {
                out.println("<tr><td>");
                out.println(rs.getString("name"));
                out.println("</td><td>");
                out.println(rs.getString("birthday"));
                out.println("</td><td>");
                out.println(rs.getString("mobile"));
                out.println("</td><td>");
                out.println(rs.getString("email"));
                out.println("</td></tr>");
            }
            out.println("</table>");
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
}
```

在 web.xml 中添加以下代码：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <servlet>
        <servlet-name>DBServlet</servlet-name>
        <servlet-class>mil.zcz.jsp.servlet.DBServlet</servlet-class>
        <init-param>
            <param-name>user</param-name>
            <param-value>root</param-value>
        </init-param>
        <init-param>
            <param-name>password</param-name>
            <param-value>kp12345</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>DBServlet</servlet-name>
        <url-pattern>/DB</url-pattern>
    </servlet-mapping>

</web-app>
```

示例代码的执行效果如图 13.2 所示。

图 13.2　使用 GenericServlet 类的 service 方法

13.2.4　destroy 方法：销毁 Servlet

【功能说明】destroy()方法用于销毁 Servlet。

【基本语法】

```
public void destroy ( )
```

示例代码参见 13.2.3 小节。

13.2.5　getInitParameter 方法：获取初始化参数

【功能说明】getInitParameter()方法用于返回初始化参数的值。

【基本语法】

```
public String getInitParameter (
    String name        //  指定参数名
)
```

如果指定的参数不存在，则返回 null。示例代码参见 13.2.3 小节。

13.2.6　getInitParameterNames 方法：获取所有初始化参数名

【功能说明】getInitParameterNames()方法用于返回所有初始化参数的名称集合。

【基本语法】

```
public Enumeration getInitParameterNames ( )
```

该方法将 Servlet 所有的初始化参数的名称作为一个字符串枚举对象返回。如果 Servlet 没有初始化参数则返回空枚举对象。

【实例演示】

```
Enumeration e = this.getInitParameterNames();
while (e.hasMoreElements()) {
    String paramName = e.nextElement().toString();
    out.println(paramName + " = " + this.getInitParameter(paramName));
}
```

13.2.7　getServletConfig 方法：获取 ServletConfig 对象

【功能说明】getServletConfig()方法用于返回 ServletConfig 对象，该对象包含该 Servlet 的初始化参数、上下文和实例名。

【基本语法】

```
public ServletConfig getServletConfig ( )
```

通过 ServletConfig 对象的 getInitParameter()方法也可以获取初始化参数。

【实例演示】

```
public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
    PrintWriter out = response.getWriter();

    String user = super.getServletConfig().getInitParameter("user");
    out.println(user);
}
```

示例代码执行后将输出字符串“ root ”。

13.2.8 getServletContext 方法：获取 Servlet 上下文

【功能说明】getServletContext()方法用于返回该 Servlet 的上下文。

【基本语法】

```
public ServletContext getServletContext ( )
```

通过 ServletContext 对象可以设置和访问属性。

【实例演示】

```
public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
    PrintWriter out = response.getWriter();

    ServletContext ctx = super.getServletContext();
    ctx.setAttribute("user", "Forrid");
    out.println(ctx.getAttribute("user"));
}
```

示例代码执行后将输出字符串“ Forrid ”。

13.2.9 getServletInfo 方法：获取 Servlet 信息

【功能说明】getServletInfo()方法用于返回该 Servlet 的信息，比如作者、版本和版权等信息。

【基本语法】

```
public String getServletInfo()
```

默认情况下，该方法返回一个空字符串。如果要返回一个有意义的值，则需要重载该方法。示例代码参见 13.1.5 小节。

13.2.10 getServletName 方法：获取 Servlet 名称

【功能说明】getServletName()方法用于返回该 Servlet 实例的名称。

【基本语法】

```
public String getServletName ( )
```

对于 DBServlet 来说，该方法将返回实例名称，即 DBServlet。

13.2.11 log 方法：信息写入日志

【功能说明】log()方法用于将指定信息写入 Servlet 日志文件中。

【基本语法】

```
public void log (
    String msg          //  指定写入日志文件的信息字符串
)
或
public void log (
    String message,     //  指定描述错误或异常的字符串
    Throwable t         //  指定java.lang.Throwable类型的错误或异常
)
```

【实例演示】

```
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        ……
        super.log("……");
    }
```

13.3 HttpServlet 类

HttpServlet 类是一个抽象类，扩展了 GenericServlet 类。HttpServlet 类用于创建一个适用于 Web 站点并支持 HTTP 协议的 Servlet。一个 HttpServlet 的子类必须至少重载以下方法中的一个。

- □ doGet()方法，适用于 HTTP GET 请求。
- □ doPost()方法，适用于 HTTP POST 请求。
- □ doPut()方法，适用于 HTTP PUT 请求。
- □ doDelete()方法，适用于 HTTP DELETE 请求。
- □ init()和 destroy()方法，管理 Servlet 生命周期中的资源。
- □ getServletInfo()方法，提供 Servlet 本身的信息。

HttpServlet 类中的 getServletConfig()、getServletContext()、getServletInfo()、getServletName()、log()等方法和 GenericServlet 类中的同名方法功能相同，本节不再介绍。

13.3.1 HttpServlet：构造函数

【基本语法】javax.servlet.http.HttpServlet 类的构造函数如下：

```
public HttpServlet ( )
```

该构造函数什么也不做，因为该类是一个抽象类。

13.3.2 init 方法：初始化 Servlet

【功能说明】init()方法用于初始化 Servlet，参见 13.2.2 小节。

13.3.3 service 方法：处理客户端请求

【功能说明】service()方法用于处理客户端请求。

【基本语法】

```
protected void service (
    HttpServletRequest request,          //  指定HttpServletRequest对象，包含客户端请求
    HttpServletResponse response         //  指定HttpServletResponse对象，包含Servlet的响应
)
    throws ServletException, IOException
或
public void service (
    ServletRequest request,              //  指定HttpServletRequest对象，包含客户端请求
    ServletResponse response             //  指定HttpServletResponse对象，包含Servlet的响应
)
    throws ServletException, IOException
```

通常情况下不需要重载该方法。

13.3.4 destroy 方法：销毁 Servlet

【功能说明】destroy()方法用于销毁 Servlet，参见 13.2.4 小节。

13.3.5 doGet 方法：处理 GET 请求

【功能说明】doGet()方法用于处理 HTTP 的 GET 请求。

【基本语法】

```
protected void doGet (
    HttpServletRequest request,              //  指定HttpServletRequest对象，包含客户端请求
    HttpServletResponse response             //  指定HttpServletResponse对象，包含Servlet的响应
)
    throws ServletException, IOException
```

重载 doGet()方法可以支持 HTTP 的 GET 请求，同时自动支持 HTTP 的 HEAD 请求。HEAD 请求是仅请求头部的 GET 请求。

在使用 PrintWriter 对象返回响应之前，最好先设置响应的内容类型和编码方式。如果请求的格式不正确，该方法将返回一个“HTTP BAD REQUEST”错误信息。

【实例演示】

```
package mil.zcz.jsp.servlet;
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class WelcomeServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();
        String user = request.getParameter("user");
        String msg = "Hi " + user + ", welcome to the servlet's world!";
        out.println(msg);
        out.close();
    }
}
```

在 web.xml 中添加以下代码：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <servlet>
        <servlet-name>WelcomeServlet</servlet-name>
        <servlet-class>mil.zcz.jsp.servlet.WelcomeServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>WelcomeServlet</servlet-name>
        <url-pattern>/Welcome</url-pattern>
    </servlet-mapping>

</web-app>
```

测试 JSP 页面 welcome.jsp 的源代码如下：

```
<html>
<head>
<title>Welcome</title>
</head>

<B>Please input your name:</B></br>
<form method="GET" action="/JSPBook/Welcome">
    <input type="TEXT" name="user" />
    <input type="SUBMIT" value="Submit" />
</form>
```

示例代码的执行效果如图 13.3 所示。

图 13.3　使用 HttpServlet 的 doGet 方法

13.3.6　doPost 方法：处理 POST 请求

【功能说明】doPost()方法用于处理 HTTP 的 POST 请求。

【基本语法】

```
protected void doPost (
    HttpServletRequest request,                    //  指定HttpServletRequest对象，包含客户端请求
```

```
        HttpServletResponse response        //  指定HttpServletResponse对象，包含Servlet的响应
    )
        throws ServletException, IOException
```

在使用 PrintWriter 对象返回响应之前，最好先设置响应的内容类型和编码方式。如果请求的格式不正确，则该方法将返回一个“ HTTP BAD REQUEST ”错误信息。

【实例演示】

```
package mil.zcz.jsp.servlet;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class WelcomeServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();
        String user = request.getParameter("user");
        String msg = "Hi " + user + ", welcome to the servlet's world!";
        out.println(msg);
        out.close();
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        this.doGet(request, response);          //  调用doGet()方法

    }
}
```

将测试 JSP 页面 welcome.jsp 的源代码修改如下：

```
<B>Please input your name:</B></br>
<form method="POST" action="/JSPBook/Welcome">
    <input type="TEXT" name="user" />
    <input type="SUBMIT" value="Submit" />
</form>
```

示例代码的执行效果参见 13.3.5 小节。

13.3.7 doHead 方法：处理 HEAD 请求

【功能说明】doHead()方法由 service()方法调用，用于处理 HTTP 的 HEAD 请求。

【基本语法】

```
protected void doHead (
    HttpServletRequest request,         //  指定传递给Servlet的请求对象
    HttpServletResponse response        //  指定Servlet返回给客户端的HTTP头部
)
    throws ServletException, IOException
```

该方法不向客户端返回任何数据，仅仅返回包含内容长度的头部信息。如果请求的格式不正确，则该方法将返回一个“ HTTP BAD REQUEST ”错误信息。通常情况下不需要重载该方法。

13.3.8 doPut 方法：处理 PUT 请求

【功能说明】doPut()方法由 service()方法调用，用于处理 HTTP 的 PUT 请求。

【基本语法】

```
protected void doPut (
    HttpServletRequest request,             //  指定HttpServletRequest对象，包含客户端请求
    HttpServletResponse response        //  指定HttpServletResponse对象，包含Servlet的响应
)
```

```
    throws ServletException, IOException
```

该方法允许客户端将文件放置到服务器上，类似于通过 FTP 发送文件。如果请求的格式不正确，则该方法将返回一个“ HTTP BAD REQUEST ”错误信息。通常情况下不需要重载该方法。

13.3.9 doDelete 方法：处理 DELETE 请求

【功能说明】doDelete()方法由 service()方法调用，用于处理 HTTP 的 DELETE 请求。

【基本语法】

```
protected void doDelete (
    HttpServletRequest request,          //  指定HttpServletRequest对象，包含客户端请求
    HttpServletResponse response         //  指定HttpServletResponse对象，包含Servlet的响应
)
    throws ServletException, IOException
```

该方法允许客户端请求从服务器上移除文档或页面。如果请求的格式不正确，则该方法将返回一个“ HTTP BAD REQUEST ”错误信息。通常情况下不需要重载该方法。

13.3.10 doTrace 方法：处理 TRACE 请求

【功能说明】doTrace()方法由 service()方法调用，用于处理 HTTP 的 TRACE 请求。

【基本语法】

```
protected void doTrace (
    HttpServletRequest request,          //  指定HttpServletRequest对象，包含客户端请求
    HttpServletResponse response         //  指定HttpServletResponse对象，包含Servlet的响应
)
    throws ServletException, IOException
```

该方法将产生一个响应，该响应包含所有在 TRACE 请求中发送的头部信息。通常情况下不需要重载该方法。

13.3.11 doOptions 方法：处理 OPTIONS 请求

【功能说明】doOptions()方法由 service()方法调用，用于处理 HTTP 的 OPTIONS 请求。

【基本语法】

```
protected void doOptions (
    HttpServletRequest request,          //  指定HttpServletRequest对象，包含客户端请求
    HttpServletResponse response         //  指定HttpServletResponse对象，包含Servlet的响应
)
    throws ServletException, IOException
```

该方法自动决定支持哪一种 HTTP 方法，并返回一个适当的头部。例如，如果 Servlet 重载了 doGet()方法，doOptions()方法将返回以下头部：

```
Allow: GET, HEAD, TRACE, OPTIONS
```

通常情况下不需要重载该方法。

13.3.12 getInitParameter 方法：获取初始化参数

【功能说明】getInitParameter()方法用于返回初始化参数的值，参见 13.2.5 小节。

13.3.13 getInitParameterNames 方法：获取所有初始化参数名

【功能说明】getInitParameterNames()方法用于返回所有初始化参数的名称集合，参见13.2.6 小节。

13.3.14 getLastModified 方法：获取最后修改时间

【功能说明】getLastModified()方法用于返回 HttpServletRequest 对象的最后修改时间。

【基本语法】

```
protected long getLastModified (
    HttpServletRequest request          //  发送给Servlet的HttpServletRequest对象
)
```

该方法返回的时间为自 GMT 1970 年 1 月 1 日以来的毫秒数。默认情况下，该方法返回−1，标识最后修改时间未知。

为了支持 GET 操作，必须重载该方法，使浏览器和代理服务器更加有效地工作，减少装载服务器和网络资源。

13.4 ServletRequest 接口

ServletRequest 接口定义了一个对象，该对象用于为 Servlet 提供客户端请求信息。Servlet 容器创建一个 ServletRequest 对象，并将其作为参数传递给 Servlet 的 service()方法。

一个 ServletRequest 对象提供的数据包括参数名、参数值、属性以及输入流。扩展 ServletRequest 的接口能够提供附加的协议相关的数据，例如，HttpServletRequest 对象提供 HTTP 数据。

13.4.1 getAttribute 方法：获取属性

【功能说明】getAttribute()方法用于返回指定属性的值。

【基本语法】

```
public Object getAttribute (
    String name         //  指定属性的名称
)
```

该方法返回一个对象，该对象包含属性的值。如果指定的属性不存在，则返回 null。示例代码参见 13.4.2 小节。

13.4.2 getAttributeNames 方法：获取所有属性名

【功能说明】getAttributeNames()方法用于返回请求中所有属性的名称集合。

【基本语法】

```
public Enumeration getAttributeNames ( )
```

该方法返回一个包含请求中所有有效属性的名称枚举。如果请求中没有属性，则返回一个空枚举对象。

【实例演示】

```
package mil.zcz.jsp.servlet;
```

```
import java.io.*;
import java.util.*;
import javax.servlet.*;

public class RequestServlet extends GenericServlet {
    @Override  重载
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        request.setAttribute("Author", "Forrid");      //  设置属性
        request.setAttribute("Version", "1.0");
        request.setAttribute("Copyright", "Copyright 2006 Wildcat Studio");

        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();
        Enumeration e = request.getAttributeNames();        //  获取所有属性的名称
        String name;
        out.println("<table border=1>");
        out.println("<tr><th>属性名</th><th>属性值</th></tr>");
        while (e.hasMoreElements()) {
            name = e.nextElement().toString();
            out.println("<tr><td align=center><I>");
            out.println(name);
            out.println("</I></td><td>");
            out.println(request.getAttribute(name)); //  获取指定属性的值
            out.println("</td><tr>");
        }
        out.println("</table>");
        out.close();
    }
}
```

在 web.xml 中添加以下代码：

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <servlet>
        <servlet-name>RequestServlet</servlet-name>
        <servlet-class>mil.zcz.jsp.servlet.RequestServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>RequestServlet</servlet-name>
        <url-pattern>/Request</url-pattern>
    </servlet-mapping>

</web-app>
```

示例代码的执行效果如图 13.4 所示。

图 13.4　使用 RequestServlet 接口的 getAttribute 和 getAttributeNames 方法

13.4.3　getCharacterEncoding 方法：获取字符编码方式

【功能说明】getCharacterEncoding()方法用于返回请求的字符编码方式。

【基本语法】

```
public String getCharacterEncoding ( )
```

该方法返回一个包含字符编码方式的字符串。如果请求未指定字符编码方式，则返回 null。

【实例演示】

```
public class RequestServlet extends GenericServlet {
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        request.setCharacterEncoding("GB2312");                    //  设置字符编码方式
        out.println(request.getParameter("user"));
        out.println("(" + request.getCharacterEncoding() + ")");    //  获取字符编码方式
        out.close();
    }
}
```

测试 JSP 页面 welcome.jsp 的源代码参见 13.3.5 小节。浏览 welcome.jsp 页面，在文本框中输入“ 没有角的牛 ”，提交后执行 RequestServlet，并输出“ 没有角的牛（GB2312）”。

13.4.4 getContentLength 方法：获取内容长度

【功能说明】getContentLength()方法用于返回内容的长度（字节数）。

【基本语法】

```
public int getContentLength ( )
```

如果内容长度未知，则返回−1。示例代码参见 13.4.6 小节。

13.4.5 getContentType 方法：获取内容类型

【功能说明】getContentType()方法用于返回内容的 MIME 类型。

【基本语法】

```
public String getContentType ( )
```

如果内容类型未知，则返回 null。示例代码参见 13.4.6 小节。

13.4.6 getInputStream 方法：获取输入流

【功能说明】getInputStream()方法用于返回请求输入流。

【基本语法】

```
public ServletInputStream getInputStream ( )
    throws IOException
```

该方法使用 ServletInputStream 对象将请求体内容作为二进制数据返回。

【实例演示】

```
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        ServletInputStream sis = request.getInputStream();
        byte b[] = new byte[1024];
        int len = sis.readLine(b, 0, 1024);

        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();

        out.println("内容类型：");
        out.println(request.getContentType());          //  获取内容类型
        out.println("</br>");
        out.println("内容长度：");
        out.println(request.getContentLength());        //  获取内容长度
        out.println("<hr>");

        out.println("内容如下：");
        for (int i=0; i<len; i++) {
            out.print(b[i] + " ");
        }
```

```
            out.close();
        }
```

测试 JSP 页面 welcome.jsp 的源代码如下：

```
<B>Please input your name:</B></br>
<form method="POST" action="/JSPBook/Request">
    <input type="TEXT" name="user" />
    <input type="SUBMIT" value="Submit" />
</form>
```

示例代码的执行效果如图 13.5 所示。

图 13.5　使用 ServletRequest 接口的 getInputStream 方法

13.4.7　getLocalAddr 方法：获取服务器地址

【功能说明】getLocalAddr()方法用于返回接收请求的服务器 IP 地址。

【基本语法】

```
public String getLocalAddr ( )
```

示例代码参见 13.4.25 小节。

13.4.8　getLocale 方法：获取区域对象

【功能说明】getLocale()方法用于返回首选的区域对象。

【基本语法】

```
public Locale getLocale ( )
```

【实例演示】

```
    out.println(request.getLocale());
```

示例代码执行后将输出“ zh_CN ”。

13.4.9　getLocales 方法：获取所有的区域对象

【功能说明】getLocales()方法用于返回所有区域对象的集合。

【基本语法】

```
public Enumeration getLocales ( )
```

【实例演示】

```
    Enumeration e = request.getLocales();
    while (e.hasMoreElements()) {
        out.println(e.nextElement());
```

```
    }
```

示例代码执行后将输出“ zh_CN ”。

13.4.10 getLocalName 方法：获取服务器名称

【功能说明】getLocalName()方法用于返回接收请求的服务器名称。

【基本语法】

```
public String getLocalName ( )
```

示例代码参见 13.4.25 小节。

13.4.11 getLocalPort 方法：获取服务器端口

【功能说明】getLocalPort()方法用于返回接收请求的服务器端口号。

【基本语法】

```
public int getLocalPort ( )
```

示例代码参见 13.4.25 小节。

13.4.12 getParameter 方法：获取参数

【功能说明】getParameter()方法用于返回请求参数的值。

【基本语法】

```
public String getParameter (
    String name        //  指定参数名称
)
```

如果指定的参数不存在，则返回 null。示例代码参见 13.4.15 小节。

13.4.13 getParameterMap 方法：获取参数映射

【功能说明】getParameterMap()方法用于返回请求参数的参数映射。

【基本语法】

```
public Map getParameterMap ( )
```

该方法返回一个 java.util.Map，该映射包含用作键的参数名和用作值的参数值。参数映射中，键的类型是 String，而值的类型是 String 数组。示例代码参见 13.4.15 小节。

13.4.14 getParameterNames 方法：获取所有参数名

【功能说明】getParameterNames()方法用于返回请求参数所有的名称集合。

【基本语法】

```
public Enumeration getParameterNames ( )
```

该方法返回一个 String 对象枚举，包含请求中所有的参数名称。如果请求中无参数，则该方法返回一个空的枚举。示例代码参见 13.4.15 小节。

13.4.15 getParameterValues 方法：获取所有参数值

【功能说明】getParameterValues()方法用于返回指定参数的所有值的集合。

【基本语法】

```
public String[] getParameterValues (
    String name          //   指定参数名称
)
```

该方法返回一个 String 对象数组，包含指定请求参数所有的值。如果指定的参数不存在，则该方法返回 null。如果参数只有一个值，则数组的长度为 1。

【实例演示】

```
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("GB2312");
        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();
        Enumeration e = request.getParameterNames();        //   获取所有参数的名称
        String name;
        out.println("<font size=2 color=red><I>getParameter()与getParameterNames()演示</I></font>");
        out.println("<table border=1>");
        out.println("<tr><th>参数名</th><th>参数值</th></tr>");
        while (e.hasMoreElements()) {
            name = e.nextElement().toString();
            out.println("<tr><td align=center><I>");
            out.println(name);
            out.println("</I></td><td>");
            out.println(request.getParameter(name));          //   获取指定参数的值
            out.println("</td><tr>");
        }
        out.println("</table>");

        String fond[] = request.getParameterValues("fond");          //   获取指定参数所有的值
        out.println("<font size=2 color=red><I>getParameterValues()演示</I></font></br>");
        out.println("爱好：");
        for (int i=0; i<fond.length; i++) {
            out.println(" " + fond[i]);
        }

        out.println("</br><font size=2 color=red><I>getParameterMap()演示</I></font>");
        Map m = request.getParameterMap();         //   获取参数映射
        Set s = m.keySet();
        Iterator i = s.iterator();
        Object key, value;
        out.println("<table border=1>");
        out.println("<tr><th>映射键</th><th>映射值</th></tr>");
        while (i.hasNext()) {
            key = i.next();
            value = m.get(key);
            out.println("<tr><td align=center><I>");
            out.println(key.toString());
            out.println("</I></td><td>");
            fond = (String[])value;
            int len = fond.length;
            out.print(fond[0]);
            if (len > 1) {
                for (int j=1; j<len; j++) {
                    out.println("、" + fond[j]);
                }
            }
            out.println("</td><tr>");
        }
        out.println("</table>");
        out.close();
    }
```

测试 JSP 页面 reg.jsp 的源代码如下：

```
<%@ page contentType="text/html;charset=GB2312" %>
<html>
<head>
<title>Registration</title>
</head>
```

```
<FORM METHOD=POST ACTION="/JSPBook/Request">
    注册名：<INPUT TYPE="text" NAME="id"></br>
    密&amp;nbsp;&amp;nbsp;码：<INPUT TYPE="password" NAME="pass"></br>
    爱&amp;nbsp;&amp;nbsp;好：
    <INPUT TYPE="checkbox" NAME="fond" VALUE="爬山">爬山
    <INPUT TYPE="checkbox" NAME="fond" VALUE="阅读">阅读
    <INPUT TYPE="checkbox" NAME="fond" VALUE="上网">上网</br>
    备&amp;nbsp;&amp;nbsp;注：<TEXTAREA NAME="remark" ROWS="5" COLS="25"></TEXTAREA></br>
    <INPUT TYPE="submit" VALUE="提交">
    <INPUT TYPE="reset" VALUE="重置">
</FORM>
```

示例代码的执行效果如图 13.6 所示。

图 13.6　使用 ServletRequest 接口的 getParameterValues 方法

13.4.16 getProtocol 方法：获取协议

【功能说明】getProtocol()方法用于返回请求所使用协议的名称和版本。

【基本语法】

```
public String getProtocol ( )
```

该方法返回一个包含协议名称和版本的字符串，其格式为“协议名称/主版本号.副版本号”，如“HTTP/1.1”。

13.4.17 getReader 方法：获取 Reader 对象

【功能说明】getReader()方法用于返回包含请求体的 BufferedReader 对象。

【基本语法】

```
public BufferedReader getReader ( )
    throws IOException
```

如果所使用的字符集编码不被支持，则该方法将抛出一个 UnsupportedEncodingException 异常。

【实例演示】

```
public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=GB2312");
    PrintWriter out = response.getWriter();
    BufferedReader reader = request.getReader();
    char buf[] = new char[1024];
    int len = reader.read(buf, 0, 1024);
    for (int i=0; i<len; i++) {
        out.print(buf[i]);
    }
    out.close();
}
```

测试 JSP 页面 reg.jsp 的源代码参见 13.4.15 小节。在浏览器中打开测试页面 reg.jsp，输入表格数据，单击“提交”按钮，示例代码的执行效果如图 13.7 所示。

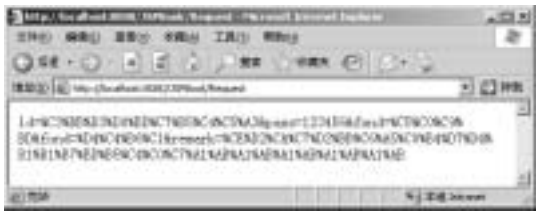

图 13.7　使用 ServletRequest 接口的 getReader 方法

13.4.18 getRealPath 方法：获取物理路径

【功能说明】自 Java Servlet 2.1 API 以来，不再推荐使用，该方法已被 ServletContext.getRealPath(String)方法所替代。

13.4.19 getRemoteAddr 方法：获取客户端地址

【功能说明】getRemoteAddr()方法用于返回发送请求的客户端 IP 地址。

【基本语法】

```
public String getRemoteAddr ( )
```

示例代码参见 13.4.25 小节。

13.4.20 getRemoteHost 方法：获取客户端主机

【功能说明】getRemoteHost()方法用于返回发送请求的客户端主机名。

【基本语法】

```
public String getRemoteHost ( )
```

示例代码参见 13.4.25 小节。

13.4.21 getRemotePort 方法：获取客户端端口

【功能说明】getRemotePort()方法用于返回发送请求的客户端端口号。

【基本语法】

```
public int getRemotePort ( )
```

示例代码参见 13.4.25 小节。

13.4.22 getRequestDispatcher 方法：获取 RequestDispatcher 对象

【功能说明】getRequestDispatcher()方法用于返回 RequestDispatcher 对象。

【基本语法】

```
public RequestDispatcher getRequestDispatcher (
    String path          //  指定资源路径
)
```

该方法返回一个 RequestDispatcher 对象，该对象用作给定路径上的资源包装器。RequestDispatcher 对象可用于转发一个请求到指定资源，或者将指定资源包含在响应中。如果 Servlet 容器无法返回一个 RequestDispatcher 对象，则该方法返回 null。

【实例演示】

```
public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
    RequestDispatcher rd = request.getRequestDispatcher("/Greet");
    rd.forward(request, response);
}
```

示例代码的执行效果如图 13.8 所示。

图 13.8　使用 ServletRequest 接口的 getRequestDispatcher 方法

13.4.23 getScheme 方法：获取协议机制

【功能说明】getScheme()方法用于返回构成请求的机制名称。

【基本语法】

```
public String getScheme ( )
```

该方法返回的机制包括 HTTP、FTP 或 HTTPS 等，通常情况下返回 HTTP。

13.4.24 getServerName 方法：获取服务器名称

【功能说明】getServerName()方法用于返回服务器的名称。

【基本语法】

```
public String getServerName ( )
```

示例代码参见 13.4.25 小节。

13.4.25 getServerPort 方法：获取服务器端口

【功能说明】getServerPort()方法用于返回服务器的端口号。

【基本语法】

```
public int getServerPort ( )
```

【实例演示】

```
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=GB2312");
        PrintWriter out = response.getWriter();
        out.println("<table border=1>");
        out.println("<tr><th>方法</th><th>结果</th></tr>");
        out.println("<tr><td>getLocalAddr</td><td>");
        out.println(request.getLocalAddr());
        out.println("</td></tr>");
        out.println("<tr><td>getLocalName</td><td>");
        out.println(request.getLocalName());
        out.println("</td></tr>");
        out.println("<tr><td>getLocalPort</td><td>");
        out.println(request.getLocalPort());
        out.println("</td></tr>");
        out.println("<tr><td>getRemoteAddr</td><td>");
        out.println(request.getRemoteAddr());
        out.println("</td></tr>");
        out.println("<tr><td>getRemoteHost</td><td>");
        out.println(request.getRemoteHost());
        out.println("</td></tr>");
        out.println("<tr><td>getRemotePort</td><td>");
        out.println(request.getRemotePort());
        out.println("</td></tr>");
        out.println("<tr><td>getServerName</td><td>");
        out.println(request.getServerName());
        out.println("</td></tr>");
        out.println("<tr><td>getServerPort</td><td>");
        out.println(request.getServerPort());
        out.println("</td></tr>");
        out.println("</table>");
        out.close();
    }
```

示例代码的执行效果如图 13.9 所示。

图 13.9　使用 ServletRequest 接口的 getServerPort 方法

13.4.26　isSecure 方法：判断是否使用安全链接

【功能说明】isSecure()方法用于判断请求是否使用安全链接，例如 HTTPS。

【基本语法】

```
public boolean isSecure ( )
```

如果请求建立在安全链接上，则返回 true，否则返回 false。

13.4.27　removeAttribute 方法：移除属性

【功能说明】removeAttribute()方法用于移除一个属性。

【基本语法】

```
public void removeAttribute (
    String name              //  指定属性名称
)
```

【实例演示】

```
    request.removeAttribute("Author");
```

示例代码执行后，将从请求中移除名为 Author 的属性。

13.4.28　setAttribute 方法：设置属性

【功能说明】setAttribute()方法用于将一个属性存储到请求中。

【基本语法】

```
public void setAttribute (
    String name,             //  指定属性名称
    Object obj               //  指定要存储的对象
)
```

该方法常与 RequestDispatcher()方法协作使用。示例代码参见 13.4.2 小节。

13.4.29　setCharacterEncoding 方法：设置字符编码方式

【功能说明】setCharacterEncoding()方法用于设置请求的字符编码方式。

【基本语法】

```
public void setCharacterEncoding (
    String env          //  指定字符编码方式
)
    throws UnsupportedEncodingException
```

该方法必须在读取请求参数或使用 getReader()读取输入之前进行调用，否则没有效果。示例代码参见 14.4.3 小节。