



图书基本信息



内容概要



书籍目录

第 部分 简介

第1章 概述 2

1.1 Java EE 6 平台的亮点 3

1.2 Java EE 应用程序模型 4

1.3 分布式多层应用程序4

1.3.1 安全 5

1.3.2 Java EE 组件5

1.3.3 Java EE 客户端6

1.3.4 Web 组件8

1.3.5 业务组件8

1.3.6 企业信息系统层9

1.4 Java EE 容器9

1.4.1 容器服务9

1.4.2 容器类型10

1.5 Web Service 支持11

1.5.1 XML 12

1.5.2 SOAP 传输协议12

1.5.3 WSDL 标准格式12

1.6 Java EE 应用程序的装配和部署12

1.7 打包应用程序13

1.8.1 开发角色14

1.8.2 Java EE 产品提供方14

1.8.3 工具提供方15

1.8.4 应用程序组件提供方15

1.8.5 应用程序装配方15

1.8.6 应用程序部署方和管理方16

1.9 Java EE 6 API 16

1.9.1 Enterprise JavaBean 技术19

1.9.2 Java Servlet 技术19

1.9.3 JavaServer Faces 技术20

1.9.4 JavaServer Pages 技术20

1.9.5 JavaServer Pages 标准标签库21

1.9.6 Java 持久化API21

1.9.7 Java 事务API 21

1.9.8 支持RESTful Web Service 的Java API21

1.9.9 Managed Beans22

1.9.10 Java EE 平台上下文和依赖注入 (JSR 299) 22

1.9.11 Java 依赖注入 (JSR 330) 22

1.9.12 Bean Validation 22

1.9.13 Java 消息服务API 23

1.9.14 Java EE 连接器架构23

1.9.15 JavaMail API23

1.9.16 Java Authorization Contract for Containers 23

1.9.17 Java Authentication Service Provider Interface for Containers 24

1.10在Java 平台标准版6 和7 中的Java EE 6 API 24

1.10.1 Java 数据库连接API 24

1.10.2 Java 命名和目录接口API 24



- 1.10.3 JavaBeans Activation Framework 25
- 1.10.4 Java XML 处理API 25
- 1.10.5 Java XML 绑定架构 25
- 1.10.6 SOAP with Attachments API for Java 26
- 1.10.7 Java API for XML Web Services 26
- 1.10.8 Java 认证和授权服务 26
- 1.10.9 GlassFish Server 工具 26
- 第2章 使用本教程的示例程序 28
 - 2.1 所需软件 28
 - 2.1.1 Java 平台标准版本 28
 - 2.1.2 Java EE 6 软件开发工具集 29
 - 2.1.3 Java EE 6 教程组件 29
 - 2.1.4 NetBeans IDE 30
 - 2.1.5 Apache Ant 31
 - 2.2 启动及停止GlassFish Server 32
 - 2.3 启动管理控制台 33
 - 2.4 启动和停止Java DB 服务 33
 - 2.5 构建示例程序 34
 - 2.6 本教程示例程序的目录结构 34
 - 2.7 获取示例程序的最新更新 35
 - 2.8 调试Java EE 应用程序 35
 - 2.8.1 使用服务器日志 35
 - 2.8.2 使用调试器 36
- 第 部分 Web 层
- 第3章 JavaServer Faces 技术：高级概念 38
 - 3.1 JavaServer Faces 应用程序的生命周期 8
 - 3.1.1 JavaServer Faces 生命周期概述 39
 - 3.1.2 恢复视图阶段 41
 - 3.1.3 应用请求值阶段 42
 - 3.1.4 处理校验阶段 42
 - 3.1.5 更新模型值阶段 43
 - 3.1.6 调用应用程序阶段 43
 - 3.1.7 渲染响应阶段 43
 - 3.2 局部处理和局部渲染 44
 - 3.3 Facelets 应用程序的生命周期 44
 - 3.4 用户界面组件模型 45
 - 3.4.1 用户界面组件类 45
 - 3.4.2 组件渲染模型 47
 - 3.4.3 转换模型 48
 - 3.4.4 事件和监听器模型 49
 - 3.4.5 校验模型 50
 - 3.4.6 导航模型 51
- 第4章 在JavaServer Faces 技术中使用Ajax 54
 - 4.1 Ajax 概述 55
 - 4.2 在JavaServer Faces 技术中使用Ajax 功能 55
 - 4.3 在Facelets 中使用Ajax 56
 - 4.3.1 使用f:ajax 标签 56
 - 4.3.2 发送一个Ajax 请求 58
 - 4.3.3 使用event 属性 58



- 4.3.4使用execute 属性 59
- 4.3.5使用immediate 属性 59
- 4.3.6使用listener 属性 59
- 4.4 监视客户端事件 60
- 4.5 处理错误 60
- 4.6 接收Ajax 响应 61
- 4.7 Ajax 请求生命周期 62
- 4.8 对组件进行分组 62
- 4.9 以资源形式加载JavaScript 63
 - 4.9.1在Facelets 应用程序中使用JavaScript API 63
 - 4.9.2在Bean 类中使用@ResourceDependency 注解 64
- 4.10 ajaxguessnumber 示例应用程序65
 - 4.10.1 ajaxguessnumber 源文件 65
 - 4.10.2运行ajaxguessnumber 示例程序 67
 - 4.10.3更多有关JavaServer Faces 技术中Ajax 的信息68
- 第5章 复合组件：高级主题及示例程序69
 - 复合组件的属性 69
 - 调用Managed Bean 70
 - 校验复合组件的值 70
 - compositecomponentlogin 示例程序 71
 - 复合组件文件 71
 - 调用 Managed Bean 70
 - 校验复合组件的值70
 - compositecomponentlogin 示例程序 71
 - 复合组件文件 71
 - 用到的页面 72
 - Managed Bean 72
 - 运行 compositecomponentlogin 示例程序 74
- 第6章 创建自定义UI 组件以及其他自定义对象 . 76
 - 决定你是否需要一个自定义组件或者渲染器 78
 - 何时使用自定义组件 78
 - 何时使用自定义渲染器 79
 - 组件、渲染器和标签的组合80
 - 理解图像映射示例程序 80
 - 为什么使用JavaServer Faces 技术来实现图像映射 81
 - 理解渲染的HTML 81
 - 理解Facelets 页面 82
 - 配置模型数据 83
 - Image Map 应用程序类总结85
 - 创建自定义组件的步骤 85
 - 创建自定义组件类 86
 - 指定组件类族 88
 - 执行编码 89
 - 执行解码 91
 - 允许组件属性接受表达式 91
 - 保存及恢复状态 93
 - 将渲染工作委托给渲染器 94
 - 创建渲染器类 94
 - 标识渲染器类型 96



- 实现事件监听器 96
- 实现值改变监听器实现动作监听器 98
- 处理自定义组件的事件98
- 在标签库描述符中定义自定义组件标签 100
- 使用自定义组件 101
- 创建和使用自定义转换器 102
- 创建自定义转换器 103
- 使用自定义转换器105
- 创建和使用自定义校验器 107
- 实现校验器接口 108
- 指定自定义标签 110
- 使用自定义校验器 111
- 将组件值和实例与Managed Bean 属性绑定112
- 将组件值与bean 属性绑定 113
- 将组件值与隐式对象绑定114
- 将组件实例与bean 属性绑定 115
- 将转换器、监听器以及校验器与Managed Bean 属性绑定 116
- 第7章 配置JavaServer Faces 应用程序 118
- 使用注解来配置Managed Bean 119
- 使用Managed Bean 作用域 119
- 应用程序配置资源文件 120
- 应用程序配置资源文件的顺序 121
- 配置Managed Bean 123
- 使用managed-bean 元素 123
- 使用managed-property 元素来初始化属性 126
- 初始化Map 和List 131
- 注册应用程序消息 132
- 使用FacesMessage 来创建消息 133
- 引用错误消息 133
- 使用默认校验器 134
- 注册自定义校验器 135
- 注册自定义转换器 135
- 配置导航规则 136
- 隐式的导航规则 139
- 使用渲染套件来注册自定义渲染器. 139
- 注册自定义组件 141
- JavaServer Faces 应用程序的基本要求 142
- 使用web 部署描述符来配置应用程序 143
- 配置项目阶段 146
- 包含类、页面和其他资源 147
- 第8章 使用Java Servlet 技术上传文件148
- @MultipartConfig 注解 148
- getParts 和getPart 方法 149
- fileupload 示例程序 150
- fileupload 示例程序的架构 150
- 运行fileupload 示例 153
- 第9章 国际化和本地化Web 应用程序. 155
- Java 平台本地化类 155
- 提供本地化的消息和标签 (label) 156



建立语言环境 157
设置资源绑定 157
获取本地化消息 158
日期和数字格式化 159
字符集和编码 159
字符集 159
字符编码 160
第 部分 Web Service
第10 章 JAX-RS : 高级主题和示例162
用于资源类字段和Bean 属性的注解 162
提取路径参数 163
提取查询参数 164
提取表单数据 164
提取请求或响应中的Java 类型 165
子资源和运行时资源解决方案165
子资源方法 165
子资源定位符 166
整合JAX-RS、EJB 技术和CDI 167
条件性HTTP 请求 168
运行时内容协商169
在JAX-RS 中使用JAXB 171
使用Java 对象为数据建模从已有的XML schema 定义开始 174
在JAX-RS 和JAXB 中使用JSON 176
customer 示例程序 177
customer 示例程序概述177
Customer 和Address 实体类 178
CustomerService 类 181
CustomerClientXML 和CustomerClientJSON 类 184
修改示例，根据已有的schema 生成实体类 186
运行customer 示例 188
第 部分 Enterprise Beans
第11 章 Message-Driven Bean 示例 196
simplemessage 示例概述 196
simplemessage 应用程序客户端 197
Message-Driven Bean 类 197
onMessage 方法 199
运行simplemessage 示例程序 200
simplemessage 示例的被管理对象 200
删除simplemessage 示例的被管理对象 202
第12 章 使用嵌入式Enterprise Bean 容器 203
嵌入式enterprise bean 容器概述 203
开发嵌入式enterprise bean 应用程序 203
运行嵌入式应用程序 204
创建enterprise bean 容器 204
查找session bean 引用 205
关闭enterprise bean 容器 206
standalone 示例程序206
第13 章 在Session Bean 中使用异步方法调用208
异步方法调用 208



创建异步的业务方法209
从enterprise bean 客户端调用异步方法 210
async 示例程序211
async 示例程序的架构 211
运行async 示例 212
第 部分 Java EE 平台上下文和依赖注入
第14 章 Java EE 平台上下文和依赖注入：高级篇 218
在CDI 应用程序中使用替代类 218
使用特例 219
在CDI 应用程序中使用生产者方法、生产者字段以及清理方法 220
使用生产者方法221
使用生产者字段来生成资源 222
使用清理方法222
在CDI 应用程序中使用预定义的Bean223
在CDI 应用程序中使用事件 224
定义事件 224
使用观察者方法来处理事件触发事件 225
在CDI 应用程序中使用拦截器 226
在CDI 应用程序中使用装饰器 228
在CDI 应用程序中使用模板229
第15 章 运行上下文和依赖注入的高级示例程序 231
encoder 示例：使用替代类231
Coder 接口和实现 232
encoder 示例中的Facelets 页面和managed bean 232
运行encoder 示例 234
producermethods 示例：使用生产者方法来选择bean 实现236
producermethods 示例的组件 237
运行producermethods 示例 238
producerfields 示例：使用生产者字段来生成资源 239
producerfields 示例的生产者字段 239
producerfields 实体和session bean 241
producerfields 示例的Facelets 页面和managed bean 242
运行producerfields 示例244
billpayment 示例：使用事件和拦截器 246
PaymentEvent 事件类246
PaymentHandler 事件监听器 247
billpayment 示例的Facelets 页面和managed bean 247
LoggedInterceptor 拦截器类 250
运行billpayment 示例 251
decorators 示例：装饰bean 252
decorators 示例的组件 253
运行decorators 示例 254
第 部分 持久化
第16 章 创建并使用基于字符串的条件 (Criteria) 查询 258
基于字符串的Criteria API 查询概述 258
创建基于字符串的查询 259
执行基于字符串的查询 260
第17 章 使用锁来控制对实体数据的并发访问261
实体锁和并发概述261



- 使用乐观锁262
- 锁模式262
- 设置锁模式 263
- 使用悲观锁264
- 第18章 在Java 持久化 API 应用程序中使用二级缓存 266
 - 二级缓存概述 266
 - 控制实体是否可能被缓存 267
 - 指定缓存模式设置以提高性能268
 - 设置缓存读取和存储模式用编程方式控制二级缓存270
- 第 部分 安全
- 第19章 Java EE 安全：高级篇 274
 - 使用数字签名 274
 - 创建服务器证书275
 - 将用户添加到证书域中 277
 - 在GlassFish Server 中使用不同的服务器证书 277
 - 认证机制 278
 - 客户端认证 279
 - 双向认证279
 - 在JavaServer Faces Web 应用程序中使用基于表单的登录 283
 - 在JavaServer Faces 表单中使用j_security_check 283
 - 在JavaServer Faces 应用程序中使用managed bean 进行认证284
 - 使用JDBC 域进行用户认证286
 - 保护HTTP 资源的安全290
 - 保护应用程序客户端的安全 293
 - 使用登录模块 294
 - 使用编程式登录 294
 - 保护企业信息系统应用程序的安全 295
 - 由容器管理的登录 295
 - 由组件管理的登录 295
 - 配置资源适配器安全296
 - 使用部署描述符来配置安全选项 298
 - 在部署描述符中指定基本认证在部署描述符中覆盖默认的用户-角色映射299
 - 关于安全的更多信息 299
- 第 部分 Java EE 的其他技术
- 第20章 Java 消息服务概念 302
 - JMS API 概述 302
 - 什么是消息传递 302
 - 什么是JMS API 303
 - 什么时候可以使用JMS API 303
 - JMS API 如何与Java EE 平台一起工作 304
 - JMS API 基础概念 305
 - JMS API 架构消息传递域 306
 - 消息接收 308
 - JMS API 编程模型 308
 - JMS 管理对象 309
 - JMS 连接 310
 - JMS 会话 311
 - JMS 消息生产者 311
 - JMS 消息消费者 312



- JMS 消息 314
- JMS 队列浏览器316
- JMS 异常处理 316
- 创建健壮JMS 应用程序 317
- 使用基础的可靠性机制 318
- 使用高级的可靠性机制321
- 在Java EE 应用程序中使用JMS API 325
- 在enterprise bean 或web 容器中使用@Resource 注解 325
- 使用session bean 来生产和同步接收消息 326
- 使用Message-Driven Bean 来异步接收消息 326
- 管理分布式事务 329
- 在应用程序客户端和web 组件中使用JMS API 330
- 关于JMS 的更多信息 331
- 第21 章 Java 消息服务示例 332
- 编写简单的JMS 应用程序 333
- 同步消息接收的简单示例 333
- 异步消息接收的简单示例 343
- 浏览队列中消息的简单示例348
- 在多个系统上运行JMS 客户端 353
- 取消部署并清理JMS 示例 359
- 编写健壮JMS 应用程序 359
- 消息应答示例 359
- 可持续订阅示例 362
- 本地事务示例 364
- 使用JMS API 和Session Bean 的应用程序 370
- 为clientsessionmdb 示例编写应用程序组件370
- 为clientsessionmdb 示例创建资源 372
- 运行 clientsessionmdb 示例 372
- 使用JMS API 和实体的应用程序 374
- clientmdbentity 示例程序概述 374
- 为clientmdbentity 示例编写应用程序组件375
- 为clientmdbentity 示例创建资源378
- 运行 clientmdbentity 示例 378
- 从远程服务器接收消息的应用程序示例 381
- consumerremote 示例模块概述 382
- 为consumerremote 示例编写模块组件 383
- 为consumerremote 示例创建资源 383
- 为consumerremote 示例使用两个应用程序服务器 383
- 运行consumerremote 示例 384
- 在两个服务器上部署Message-Driven Bean 的应用程序示例 387
- sendremote 示例模块概述 388
- 编写sendremote 示例的模块组件 389
- 为sendremote 示例创建资源 390
- 运行sendremote 示例 392
- 第22 章 Bean Validation : 高级主题 398
- 创建自定义约束 398
- 使用内置约束来创建新的约束 398
- 自定义校验器消息 399
- ValidationMessages 资源绑定 399



- 约束分组 400
- 自定义组校验顺序 400
- 第23章 使用Java EE 拦截器 402
 - 拦截器概述 402
 - 拦截器类 403
 - 拦截器的生命周期 403
 - 拦截器和CDI 403
 - 使用拦截器 403
 - 拦截方法调用 404
 - 拦截生命周期回调事件 406
 - 拦截超时事件 407
 - interceptor 示例程序 408
 - 运行interceptor 示例 409
- 第24章 资源适配器示例 410
 - 资源适配器 410
 - Message-Driven Bean 411
 - Web 应用程序 411
 - 运行mailconnector 示例 411
- 第 部分 案例研究
- 第25章 Duke ' s Bookstore 案例研究示例 416
 - Duke ' s Bookstore 的设计和架构 416
 - Duke ' s Bookstore 的接口 417
 - Java 持久化API 实体Book 417
 - Duke ' s Bookstore 中使用的Enterprise beans 418
 - Duke ' s Bookstore 中使用的Facelets 页面和Managed Beans 418
 - Duke ' s Bookstore 中使用的自定义组件和其他自定义对象 420
 - Duke ' s Bookstore 中使用的属性文件 420
 - Duke ' s Bookstore 中使用的部署描述符 421
 - 运行 Duke ' s Bookstore 案例研究应用程序 422
- 第26章 Duke ' s Tutoring 案例研究示例 424
 - Duke ' s Tutoring 的设计和架构 424
 - 主界面 426
 - 主界面中使用的Java 持久化API 实体 426
 - 主界面中使用的enterprise bean 426
 - 主界面中使用的Facelets 文件 427
 - 主界面中使用的辅助类 428
 - 属性文件 429
 - Duke ' s Tutoring 中使用的部署描述符 429
 - 管理界面 430
 - 管理界面中使用的enterprise bean 430
 - 管理界面中使用的Facelets 文件 430
 - 运行Duke ' s Tutoring 案例研究应用程序 431
 - 设置GlassFish Server 431
 - 运行Duke ' s Tutoring 432
- 第27章 Duke ' s Forest 案例研究示例 434
 - Duke ' s Forest 的设计和架构 435
 - events 项目 437
 - entities 项目 438
 - dukes-payment 项目 440



dukes-resource 项目 440
Duke ' s Store 项目 440
Duke ' s Shipment 项目 445
构建并部署Duke ' s Forest 案例研究应用程序 447
前提条件447
运行Duke ' s Forest 应用程序 450



版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu000.com