

《Java EE核心框架实战》

图书基本信息

书名：《Java EE核心框架实战》

13位ISBN编号：9787115365717

出版时间：2014-9

作者：高洪岩

页数：614

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu111.com

书籍目录

第1章 MyBatis3操作数据库 1

1.1 MyBatis介绍 1

1.2 MyBatis操作数据库的步骤 2

1.2.1 使用XML配置文件创建SqlSessionFactory对象 3

1.2.2 SqlSessionFactoryBuilder和SqlSessionFactory类的结构 4

1.2.3 使用MyBatisGenerator工具逆向 5

1.2.4 使用SqlSession对象在MsSql数据库中新建记录 7

1.2.5 使用SqlSession对象在Oracle数据库中新建记录 10

1.3 使用MyBatis针对3种数据库（Oracle、MSSQL和MySQL）实现CURD 10

1.3.1 针对Oracle的CURD 10

1.3.2 针对MSSQL的CURD 17

1.3.3 针对MySQL的CURD 20

1.4 MyBatis核心对象的生命周期与封装 22

1.4.1 创建GetSqlSessionFactory.java类 23

1.4.2 创建GetSqlSession.java类 24

1.4.3 创建DBOperate.java类 25

1.4.4 创建userinfoMapping.xml映射文件 25

1.4.5 创建连接数据库的mybatis—config.xml配置文件 26

1.4.6 创建名为test的Servlet对象 26

1.4.7 添加记录及异常回滚的测试 27

1.4.8 删除记录 29

1.4.9 更改记录 30

1.4.10 查询单条记录 31

1.4.11 查询多条记录 32

第2章 MyBatis3常用技能 33

2.1 MyBatis3的SQL映射文件 33

2.2 连接DB数据库的参数来自于Properties对象 33

2.3 < resultMap > 标签 34

2.4 < sql > 标签 35

2.5 将SQL语句作为字符串变量传入 37

2.6 动态SQL的使用 38

2.6.1 插入null值时的处理第1种方法——jdbcType 38

2.6.2 插入null值时的处理第2种方法——< if > 39

2.6.3 < choose > 标签的使用 40

2.6.4 < set > 标签的使用 42

2.6.5 < foreach > 标签的使用 43

2.7 插入超大的字符串文本内容 45

2.8 分页 46

第3章 Struts2必备开发技能 48

3.1 使用Struts2进行登录功能的开发 48

3.1.1 为什么要使用MVC 48

3.1.2 准备JAR文件 54

3.1.3 创建Web项目、添加jar文件及配置web.xml文件 55

3.1.4 创建控制层Controller文件——Login.java 56

3.1.5 创建业务逻辑层Model文件——UserinfoService.java 57

3.1.6 创建视图层View文件——login.jsp 57

3.1.7 添加核心配置文件struts.xml及解释 58

- 3.1.8 添加ok.jsp和no.jsp登录结果文件 59
- 3.1.9 运行项目 59
- 3.1.10 Struts2的拦截器 60
- 3.1.11 Struts2的数据类型自动转换 64
- 3.2 MVC框架的开发模型 71
 - 3.2.1 基础知识准备1——解析并创建xml文件 71
 - 3.2.2 基础知识准备2——Java的反射 74
 - 3.2.3 实现MVC模型——自定义配置文件 77
 - 3.2.4 实现MVC模型——ActionMapping.java封装 < action > 信息 78
 - 3.2.5 实现MVC模型——ResultMapping.java以封装 < result > 信息 78
 - 3.2.6 实现MVC模型——管理映射信息的ActionMappingManager.java对象 79
 - 3.2.7 实现MVC模型——创建反射Action的ActionManager.java对象 81
 - 3.2.8 实现MVC模型——创建核心控制器ActionServlet.java 81
 - 3.2.9 实现MVC模型——创建Action接口及控制层Controller实现类 83
 - 3.2.10 实现MVC模型——创建视图层V对应的JSP文件 84
 - 3.2.11 实现MVC模型——在web.xml中配置核心控制器 86
 - 3.2.12 实现MVC模型——运行结果 86
- 3.3 Struts2的刷新验证功能 86
 - 3.3.1 Action接口 87
 - 3.3.2 Validateable和ValidationAware接口 88
 - 3.3.3 TextProvider和LocaleProvider接口 88
 - 3.3.4 使用ActionSupport实现有刷新的验证 89
- 3.4 对Struts2有刷新验证的示例进行升级 91
 - 3.4.1 加入xml配置来屏蔽自动生成的table/tr/td代码 92
 - 3.4.2 解决“ 出错信息不能自动显示 ” 的问题 93
- 3.5 用 < s : actionerror > 标签显示全部出错信息 96
- 3.6 出错信息进行传参及国际化 98
 - 3.6.1 创建info_en_US.properties和info_zh_CN.properties属性文件 98
 - 3.6.2 在JSP文件中显示国际化的静态文本 101
 - 3.6.3 在JSP文件中显示国际化的静态文本时传递参数 102
 - 3.6.4 在Action中使用国际化功能 103
- 3.7 用实体类封装URL中的参数——登录功能的URL封装 105
- 3.8 Struts2中的转发操作 107
 - 3.8.1 Servlet中的转发操作 107
 - 3.8.2 Struts2中的转发操作 107
- 3.9 由Action重定向到Action——无参数 109
 - 3.9.1 何种情况下使用重定向 109
 - 3.9.2 新建起始控制层Login.java 109
 - 3.9.3 新建目的控制层List.java 110
 - 3.9.4 在struts.xml文件中配置重定向的重点 110
 - 3.9.5 新建显示列表的JSP文件 111
- 3.10 由Action重定向到Action——有参数 112
 - 3.10.1 何种情况下需要重定向传递参数 112
 - 3.10.2 新建起始控制层Login.java文件 112
 - 3.10.3 更改struts.xml配置文件 113
 - 3.10.4 新建目的控制层List.java文件 113
 - 3.10.5 用JSTL和EL在JSP文件中输出数据 114
- 3.11 让Struts2支持多模块多配置文件开发 115
 - 3.11.1 新建4个模块的控制层 115

- 3.11.2 新建3个模块的配置文件 116
- 3.11.3 使用include标记导入多个配置文件 118
- 3.11.4 创建各模块使用的JSP文件 118
- 3.11.5 运行各模块的结果 119
- 3.12 在Action中有多个业务方法时的处理 120
 - 3.12.1 第一种实现方式——通过url叹号“！”参数 120
 - 3.12.2 第二种实现方式——在action标记中加入method属性 122
- 3.13 自定义全局result 124
 - 3.13.1 新建全局result实例和控制层代码 124
 - 3.13.2 声明全局result对象 125
 - 3.13.3 部署项目并运行 126
- 3.14 在Action中使用Servlet的API（紧耦版） 126
 - 3.14.1 将数据放到不同的作用域中 126
 - 3.14.2 从不同作用域中取值 128
- 3.15 在Action中使用Servlet的API（松耦版） 128
 - 3.15.1 新建控制层 128
 - 3.15.2 新建JSP视图 129
- 3.16 Session与Cookie在request与response对象中的运行机制 130
- 3.17 在MyEclipse中使用WebService 135
- 第4章 Struts2文件的上传与下载 141
 - 4.1 使用Struts2进行单文件上传 141
 - 4.1.1 Struts2上传功能的底层依赖 141
 - 4.1.2 新建上传文件的JSP文件 141
 - 4.1.3 新建上传文件的控制层Register.java文件 142
 - 4.1.4 Action中File实例的命名规则 143
 - 4.1.5 设置上传文件的大小 143
 - 4.1.6 设计struts.xml配置文件 143
 - 4.1.7 成功上传单个文件 144
 - 4.2 使用Struts2进行多文件上传 145
 - 4.2.1 新建上传多个文件的JSP 145
 - 4.2.2 设计上传的控制层代码 145
 - 4.2.3 成功上传多个文件 147
 - 4.3 使用属性驱动形式的文件上传 148
 - 4.3.1 创建上传多个文件的JSP 148
 - 4.3.2 设计上传文件的控制层 149
 - 4.3.3 新建上传文件的封装类 150
 - 4.3.4 将JSP文件中s：file标签的name属性进行更改 151
 - 4.3.5 以属性驱动方式成功上传多个文件 152
 - 4.4 用Struts2实现下载文件的功能（支持中文文件名） 153
 - 4.4.1 新建下载文件的JSP文件 153
 - 4.4.2 新建下载文件的控制层文件 154
 - 4.4.3 更改struts.xml配置文件 155
 - 4.4.4 成功下载中文文件名的文件 155
- 第5章 JSON、Ajax、jQuery与Struts2联合使用 156
 - 5.1 JSON介绍 156
 - 5.2 用JSON创建对象 157
 - 5.2.1 用JSON创建对象的语法格式 157
 - 5.2.2 在JSP中用JSON创建对象 157
 - 5.2.3 运行结果 157

- 5.3 用JSON创建字符串的限制 158
 - 5.3.1 需要转义的特殊字符 158
 - 5.3.2 在JSP中对JSON特殊字符进行转义 158
 - 5.3.3 运行结果 159
- 5.4 用JSON创建数字类型的语法格式 159
 - 5.4.1 在JSP中用JSON创建数字类型 160
 - 5.4.2 运行结果 160
- 5.5 用JSON创建数组对象的语法格式 160
 - 5.5.1 在JSP中用JSON创建数组对象 161
 - 5.5.2 运行结果 161
- 5.6 用JSON创建嵌套的对象类型 161
- 5.7 将对象转换成JSON字符串 162
 - 5.7.1 什么情况下需要将对象转换成JSON字符串 162
 - 5.7.2 在JSP中用stringify方法将对象转换成JSON字符串 163
- 5.8 将对象转换成JSON字符串提交到Action并解析（以post方式提交） 164
 - 5.8.1 在JSP中创建JSON和Ajax对象 164
 - 5.8.2 用Action控制层接收通过Ajax传递过来的JSON字符串 165
 - 5.8.3 运行结果 166
 - 5.8.4 在控制台输出的数据 166
- 5.9 将对象转换成JSON字符串提交到Action并解析（以get方式提交） 167
 - 5.9.1 新建创建JSON字符串的JSP文件 167
 - 5.9.2 新建接收JSON字符串的Action控制层 168
 - 5.9.3 运行结果 168
 - 5.9.4 在控制台输出的数据 169
- 5.10 将数组转换成JSON字符串提交到Action并解析（以get和post方式提交） 169
 - 5.10.1 在服务器端用get方法解析JSON字符串 171
 - 5.10.2 在服务器端用post方法解析JSON字符串 171
 - 5.10.3 运行结果 172
 - 5.10.4 在控制台输出的数据 172
- 5.11 使用Ajax调用Action并生成JSON再传递到客户端（以get和post方式提交） 173
 - 5.11.1 新建具有Ajax提交功能的JSP 173
 - 5.11.2 在Action控制层创建List中存放的String 176
 - 5.11.3 在Action控制层创建List中存放的Bean 177
 - 5.11.4 在Action控制层创建Map中存放的String 178
 - 5.11.5 在Action控制层创建Map中存放的Bean 178
 - 5.11.6 单击不同的button按钮调用不同的Action 179
- 5.12 jQuery、JSON和Struts2 181
 - 5.12.1 jQuery框架的Ajax功能介绍 181
 - 5.12.2 用jQuery的Ajax功能调用远程action（无返回结果） 181
 - 5.12.3 jQuery的Ajax方法的结构 183
 - 5.12.4 用jQuery的Ajax功能调用远程action（有返回结果） 184
 - 5.12.5 用jQuery的Ajax功能调用远程action并且传递JSON格式参数（有返回值） 185
 - 5.12.6 用jQuery解析从action返回List中存放String的JSON字符串 188
- 第6章 Spring4MVC实用开发 191
 - 6.1 Spring4MVC介绍 191
 - 6.1.1 Spring4MVC核心控制器 191
 - 6.1.2 基于注解的Spring4MVC开发 192
 - 6.2 Spring4MVC的第一个登录测试 193
 - 6.2.1 添加Spring4MVC的依赖jar文件 193

- 6.2.2 在web.xml中配置核心控制器 193
- 6.2.3 新建springMVC—servlet.xml配置文件 193
- 6.2.4 新建相关的JSP文件 194
- 6.2.5 新建控制层Java类文件 195
- 6.2.6 部署项目并运行 195
- 6.2.7 第一个示例的总结 196
- 6.2.8 Spring更加方便的参数获取方法 196
- 6.3 执行Controller控制层与限制提交的method方式 197
 - 6.3.1 新建控制层ListUsername.java文件 197
 - 6.3.2 新建登录及显示数据的JSP文件 198
 - 6.3.3 部署项目并测试 199
- 6.4 解决多人开发路径可能重复的问题 200
 - 6.4.1 错误的情况 200
 - 6.4.2 解决办法 201
- 6.5 在控制层中使用指定方式处理get或post提交方式 203
 - 6.5.1 控制层代码 203
 - 6.5.2 新建JSP文件并运行 204
- 6.6 控制层重定向到控制层——无参数传递 205
 - 6.6.1 新建控制层Java文件 205
 - 6.6.2 创建JSP文件并运行项目 206
- 6.7 控制层重定向到控制层——有参数传递 206
 - 6.7.1 创建两个控制层Java文件 207
 - 6.7.2 部署项目并运行 207
- 6.8 匹配URL路径执行指定Controller 208
 - 6.8.1 新建控制层文件 208
 - 6.8.2 部署项目并运行 209
- 6.9 在服务器端获取JSON字符串并解析——方式1 210
 - 6.9.1 在web.xml中配置字符编码过滤器 210
 - 6.9.2 新建JSP文件 211
 - 6.9.3 新建控制层Java文件 212
 - 6.9.4 添加依赖的jar包文件 212
 - 6.9.5 运行项目 213
- 6.10 在服务器端获取JSON字符串并解析——方式2 213
 - 6.10.1 新建封装JSON对象属性的实体类 213
 - 6.10.2 新建控制层 214
 - 6.10.3 在配置文件中添加 < mvc : annotation—driven/ > 注解 214
 - 6.10.4 新建JSP文件 215
 - 6.10.5 添加jacksonJSON解析处理类库并运行 215
 - 6.10.6 解析不同格式的JSON字符串示例 216
- 6.11 将URL中的参数转成实体的示例 218
 - 6.11.1 新建控制层文件 218
 - 6.11.2 新建登录用途的JSP文件 219
 - 6.11.3 在web.xml中注册编码过滤器 219
 - 6.11.4 运行结果 219
- 6.12 在控制层传回JSON对象示例 220
 - 6.12.1 新建控制层文件 220
 - 6.12.2 新建JSP文件 220
 - 6.12.3 部署项目并运行 222
- 6.13 在控制层传回JSON字符串示例 222

- 6.13.1 新建控制层文件 222
- 6.13.2 新建JSP文件及在配置文件中注册utf——8编码处理 223
- 6.13.3 运行项目 224
- 6.14 在控制层获取HttpServletRequest和HttpServletResponse对象 224
 - 6.14.1 新建控制层 224
 - 6.14.2 JSP文件中的EL代码及运行结果 225
 - 6.14.3 直接使用HttpServletResponse对象输出响应字符 225
- 6.15 通过URL参数访问指定的业务方法 227
 - 6.15.1 新建控制层文件List.java 227
 - 6.15.2 运行结果 227
- 6.16 Spring4MVC单文件上传——写法1 228
 - 6.16.1 新建控制层 228
 - 6.16.2 在配置文件springMVC—servlet.xml中声明上传请求 229
 - 6.16.3 创建前台JPS文件 229
 - 6.16.4 运行结果 230
- 6.17 Spring4MVC单文件上传——写法2 230
- 6.18 Spring4MVC多文件上传 231
 - 6.18.1 新建控制层及JSP文件 231
 - 6.18.2 运行结果 232
- 6.19 Spring4MVC支持下载文件名为中文的文件 232
- 6.20 控制层返回List对象及实体的结果 233
 - 6.20.1 新建控制层文件 233
 - 6.20.2 新建JSP文件 234
 - 6.20.3 更改springMVC—servlet.xml配置文件 234
 - 6.20.4 运行结果 235
- 6.21 控制层ModelMap对象 236
 - 6.21.1 新建控制层 236
 - 6.21.2 JSP文件代码 236
 - 6.21.3 运行结果 237
- 6.22 对Spring4MVC提交的表单进行手动数据验证 237
 - 6.22.1 创建控制层文件 237
 - 6.22.2 创建JSP文件 238
 - 6.22.3 运行结果 238
- 第7章 Spring4MVC必备知识 239
 - 7.1 web.xml中的不同配置方法 239
 - 7.1.1 将配置文件存放于src路径中 239
 - 7.1.2 指定存放路径 240
 - 7.1.3 指定多个配置文件 240
 - 7.2 路径中添加通配符的功能 241
 - 7.3 Service业务逻辑层在Controller中进行注入 241
 - 7.3.1 新建业务逻辑层 241
 - 7.3.2 创建控制层文件 242
 - 7.3.3 设计springMVC—servlet.xml配置文件 242
 - 7.3.4 运行结果 242
 - 7.3.5 多个实现类的情况 243
 - 7.4 对象ModelAndView的使用 244
 - 7.4.1 创建控制层及JSP文件 244
 - 7.4.2 程序运行结果 244
 - 7.5 控制层返回void数据的情况 245

- 7.5.1 创建控制层及index.jsp文件 245
- 7.5.2 更改配置文件 246
- 7.5.3 部署项目并运行程序 246
- 7.6 使用Spring4MVC中的注解来操作HttpSession中的对象 247
 - 7.6.1 创建控制层文件PutGetSession.java 247
 - 7.6.2 创建显示不同作用域中值的JSP文件 247
 - 7.6.3 部署项目并运行程序 248
- 第8章 Spring4MVC+MyBatis3+Spring4整合 249
 - 8.1 准备Spring4的jar包文件 249
 - 8.2 准备MyBatis的jar包文件 250
 - 8.3 准备MyBatis3与Spring4整合的jar文件 250
 - 8.4 创建Web项目 250
 - 8.5 配置web.xml文件 251
 - 8.6 配置springMVC—servlet.xml文件 252
 - 8.7 配置MyBatis配置文件 252
 - 8.8 创建MyBatis与映射有关文件 253
 - 8.9 配置applicationContext.xml文件 254
 - 8.10 创建DAO对象 255
 - 8.11 创建Service对象 256
 - 8.12 创建Controller对象 257
 - 8.13 测试整合效果 258
 - 8.14 回滚的测试 258
- 第9章 用Hibernate4操作数据库 260
 - 9.1 Hibernate概述与优势 260
 - 9.2 持久层与持久化与ORM 261
 - 9.3 用MyEclipse开发第一个Hibernate示例 262
 - 9.3.1 用MyEclipseDatabaseExplorer工具连接Oracle11g数据库 263
 - 9.3.2 创建一个支持Hibernate4环境的Web项目 265
 - 9.3.3 对数据表进行Hibernate逆向工程 267
 - 9.3.4 逆向工程后的项目orm结构 270
 - 9.3.5 使用Hibernate进行持久化 273
- 第10章 Hibernate4核心技能 274
 - 10.1 Configuration介绍 274
 - 10.2 SessionFactory介绍 275
 - 10.3 Session介绍 275
 - 10.4 使用Session实现CURD操作 275
 - 10.4.1 Session操作目标表USERINFO 276
 - 10.4.2 逆向工程后的项目结构 276
 - 10.4.3 新建添加记录的Servlet 277
 - 10.4.4 新建查询记录的Servlet 278
 - 10.4.5 新建更改记录的Servlet 279
 - 10.4.6 新建删除记录的Servlet 280
 - 10.5 在Hibernate中使用JNDI技术 281
 - 10.5.1 备份Tomcat/conf路径下的配置文件 281
 - 10.5.2 更改配置文件context.xml 281
 - 10.5.3 更改配置文件web.xml 281
 - 10.5.4 添加Hibernate框架配置的关键步骤 282
 - 10.5.5 逆向工程 282
 - 10.5.6 支持JNDI的hibernate.cfg.xml配置文件内容 282

- 10.5.7 创建查询数据的Servlet 283
- 10.5.8 部署项目并验证结果 283
- 10.6 缓存与实体状态 283
 - 10.6.1 Hibernate的OID与缓存 283
 - 10.6.2 Hibernate中的对象状态：瞬时状态、持久化状态和游离状态 285
- 10.7 双向一对多在MyEclipse中的实现 285
 - 10.7.1 创建主表MAIN 285
 - 10.7.2 创建子表SUB 285
 - 10.7.3 添加主外键约束对象 286
 - 10.7.4 设置主外键关系 286
 - 10.7.5 逆向主从表外键关系 287
 - 10.7.6 集合与多对一 288
 - 10.7.7 新建主表main数据 289
 - 10.7.8 新建子表SUB数据 290
 - 10.7.9 删除子表SUB数据 292
 - 10.7.10 删除主表MAIN数据 293
- 10.8 Hibernate备忘知识点 294
- 10.9 对主从表结构中的HashSet进行排序 295
- 10.10 Hibernate中延迟加载的调试实验 295
 - 10.10.1 主从表表结构的设计 295
 - 10.10.2 对省表和市表内容的填充 295
 - 10.10.3 更改映射文件 295
 - 10.10.4 新建测试用的Servlet对象 296
 - 10.10.5 更改映射文件Sheng.hbm.xml 296
- 10.11 Hibernate中对Oracle中CLOB字段类型的读处理 297
- 10.12 Hibernate中的inverse与cascade的测试 297
- 第11章 在Hibernate4中使用HQL语言进行检索 302
 - 11.1 Hibernate的检索方式 302
 - 11.2 HQL表别名 305
 - 11.3 HQL对结果进行排序与list () 和iterator () 方法的区别 306
 - 11.4 HQL索引参数绑定 309
 - 11.5 HQL命名参数绑定与安全性 309
 - 11.6 HQL方法链的使用 311
 - 11.7 HQL中的uniqueResult () 方法的使用 311
 - 11.8 HQL中的Where子句与查询条件 312
 - 11.9 HQL中的聚集函数：distinct、count、min、max、sum和avg 314
 - 11.10 HQL中的分组查询 316
- 第12章 Spring4的AOP和IOC 318
 - 12.1 Spring介绍 318
 - 12.2 Spring架构 318
 - 12.3 IOC的介绍 319
 - 12.4 AOP的介绍 319
 - 12.5 IOC容器 320
 - 12.6 使用传统方式保存数据功能的测试 320
 - 12.7 使用Spring的IOC方式保存数据功能的测试 321
 - 12.8 BeanFactory与ApplicationContext 327
 - 12.9 Spring的IOC容器的注入类型 327
 - 12.9.1 通过IOC容器注入基本数据类型 327
 - 12.9.2 通过IOC容器注入引用数据类型 330

- 12.9.3 通过IOC容器注入null类型 330
- 12.9.4 通过IOC容器注入Properties类型 332
- 12.9.5 通过IOC容器对构造方法进行注入 333
- 12.10 Spring中Bean在Singleton和Prototype中的作用域 335
- 12.11 Spring中注入外部属性文件的属性值 337
- 12.12 Spring中多个applicationContext.xml配置文件的使用 339
- 12.13 AOP的概念与介绍 342
 - 12.13.1 静态代理的实现 342
 - 12.13.2 动态代理的实现 344
- 12.14 实现MethodBeforeAdvice接口——方法执行前增强 345
- 12.15 实现AfterReturningAdvice接口——方法执行后增强 348
- 12.16 实现MethodInterceptor接口——方法执行前后环绕增强 350
- 第13章 Struts2+Hibernate4+Spring4整合 353
 - 13.1 目的 353
 - 13.2 新建Oracle数据表userinfo 353
 - 13.2.1 新建数据表userinfo 353
 - 13.2.2 创建序列对象 354
 - 13.3 新建整合用的Web项目 354
 - 13.4 添加Struts2框架支持环境 354
 - 13.4.1 添加Struts2框架 354
 - 13.4.2 在web.xml文件中注册Struts2的过滤器 355
 - 13.4.3 在项目的src目录下创建struts.xml配置文件 355
 - 13.4.4 添加Struts2框架后的项目文件结构图 356
 - 13.5 添加HibernateDatabaseExplorer数据库连接 356
 - 13.6 添加Hibernate4框架支持 357
 - 13.7 添加Spring4框架支持文件 357
 - 13.8 创建的applicationContext.xml文件 358
 - 13.9 在web.xml文件中添加Spring的utf—8编码过滤器和Spring监听器 359
 - 13.10 添加Spring4框架后的Web项目结构 360
 - 13.11 对Oracle11g数据表userinfo进行Hibernate逆向工程 361
 - 13.12 创建Hibernate4的DAO类 362
 - 13.13 创建All_DAO对象 363
 - 13.14 创建UserinfoService.java服务对象 364
 - 13.15 创建AllService服务对象 364
 - 13.16 继续更改applicationContext.xml和hibernate.cfg.xml 365
 - 13.17 新建自定义action的父类BaseAction 367
 - 13.18 新建操作userinfo表中数据的Action 367
 - 13.19 在applicationContext.xml中配置 / base和/test 368
 - 13.20 部署到Tomcat容器 370
- 第14章 有状态 / 无状态会话Bean和消息驱动Bean 373
 - 14.1 EJB3概述 373
 - 14.1.1 JavaEE体系结构 374
 - 14.1.2 容器的概念 374
 - 14.2 有状态会话Bean和无状态会话Bean 375
 - 14.2.1 会话Bean的作用 375
 - 14.2.2 会话Bean的种类 375
 - 14.2.3 在MyEclipse中无状态会话Bean的创建 376
 - 14.2.4 用Web方式调用本地SayHello接口 390
 - 14.2.5 本地和远程无状态会话Bean的区别 394

- 14.2.6 EJB组件接口无注解时的默认情况 395
- 14.2.7 调用远程类型的无状态会话Bean 396
- 14.2.8 无状态会话Bean的回调函数和生命周期 401
- 14.2.9 无状态会话Bean实例变量值保留的问题与无状态会话Bean实例池 404
- 14.2.10 有状态会话Bean 409
- 14.2.11 有状态会话Bean的创建与状态特性 409
- 14.2.12 将远程无状态会话Bean共享的服务重命名 416
- 14.2.13 使用注解声明会话Bean的第2种写法 418
- 14.2.14 有状态会话Bean的钝化与激活 419
- 14.2.15 有状态会话Bean的回调函数和生命周期 420
- 14.2.16 有状态会话Bean的@Remove回调函数的使用 425
- 14.2.17 注入其他本地类型的EJB对象 428
- 14.3 消息驱动JavaBean (MDB) 和在WebLogic中创建消息目的 430
- 14.3.1 创建持久性存储对象 431
- 14.3.2 创建JMS服务器 433
- 14.3.3 创建JMS模块 435
- 14.3.4 在JMS模块中创建子部署 437
- 14.3.5 在JMS模块中创建资源 439
- 14.3.6 点对点式消息驱动JavaBean 441
- 14.3.7 发布—订阅式消息驱动JavaBean 445
- 14.4 WebService与在EJB3中创建基于WebService的业务服务 450
- 14.5 计时器与作业调度 463
- 第15章 实体Bean 466
- 15.1 实体Bean概述 466
- 15.2 持久层、持久化与ORM 466
- 15.2.1 在WebLogic的JNDI树中创建节点与对象 469
- 15.2.2 在WebLogic的JNDI树中创建子节点 475
- 15.2.3 在WebLogic的JNDI树中查找节点 477
- 15.2.4 在WebLogic的JNDI树中删除节点 478
- 15.3 从保存记录开始 479
- 15.3.1 安装Oracle11g数据库 479
- 15.3.2 使用Toad管理Oracle数据库 479
- 15.3.3 使用MyEclipseDatabaseExplorer工具连接Oracle11G数据库 486
- 15.3.4 创建EJB3项目 489
- 15.3.5 更改实体的主键与序列映射 492
- 15.3.6 创建调用外观的Servlet 493
- 15.3.7 更改persistence.xml配置文件 493
- 15.3.8 KODO的简要介绍 494
- 15.3.9 部署到WebLogic中并且运行 494
- 15.3.10 使用JPA技术向Oracle11g数据库成功添加记录 494
- 15.3.11 改成JNDI连接池的示例 495
- 15.4 解析实体类所使用的注解 501
- 15.5 在SQL2005数据库中插入记录 501
- 15.6 在MySQL数据库中插入记录 504
- 15.7 使用表在Oracle数据库中生成主键 505
- 15.8 使用EJB3在Oracle中插入Date时间类型 508
- 15.9 处理CLOB数据类型 510
- 15.10 在WebLogic中实现JDBC+JNDI全局性分布式事务实验 510
- 15.10.1 JTA和2PC的概述 510

- 15.10.2 进入WebLogic控制台 511
- 15.10.3 配置数据源名称 511
- 15.10.4 配置数据源属性 512
- 15.10.5 设置数据源连接数据库的详细信息 513
- 15.10.6 测试是否连接到数据库 513
- 15.10.7 将数据源归属到AdminServer服务器 514
- 15.10.8 新建名为b_jndi的数据源 514
- 15.10.9 数据源列表 514
- 15.10.10 SQL在正确的情况下实现多数据源提交事务 515
- 15.10.11 SQL在错误的情况下实现多数据源回滚事务 518
- 第16章 JPA核心技能 521
- 16.1 EntityManager类的概述 521
- 16.1.1 实体类的状态 522
- 16.1.2 EJB3中的事务 522
- 16.2 EntityManager类的方法使用 522
- 16.2.1 persist (Object) 方法 525
- 16.2.2 merge (T) 方法和find (Class < T > , Object) 方法 526
- 16.2.3 remove (Object) 方法 528
- 16.2.4 getReference (Class < T > , Object) 方法 530
- 16.2.5 createNativeQuery () 方法 533
- 16.2.6 close () 和isOpen () 方法 538
- 16.2.7 refresh (Object) 方法 538
- 16.2.8 clear () 和contains (Object) 方法 542
- 16.2.9 createQuery (String) 方法 543
- 16.2.10 createNamedQuery (String) 方法 545
- 16.3 EntityManagerFactory对象介绍 546
- 16.4 用JavaSE客户端调用远程EJB3组件 (使用逆向DAO) 547
- 16.5 在JavaSE客户端使用EntityManagerFactory实现持久化 (手动配置) 550
- 16.6 在JavaSE客户端使用EntityManagerFactory实现持久化 (自动配置) 552
- 16.7 在会话Bean中生成EntityManagerFactory 557
- 16.8 双向一对多的CURD实战 559
- 16.8.1 新建数据表Sheng 559
- 16.8.2 新建数据表Shi 559
- 16.8.3 配置主从键约束关系 559
- 16.8.4 创建企业项目 561
- 16.8.5 逆向EJB实体 561
- 16.8.6 添加主键生成策略的注解 562
- 16.8.7 配置persistence.xml文件 563
- 16.8.8 persistence.xml配置文件再次提醒 563
- 16.8.9 生成的Sheng.java和Shi.java代码引用 563
- 16.8.10 创建Sheng的Servlet 564
- 16.8.11 创建Shi的Servlet 565
- 16.8.12 更新Sheng的Servlet 566
- 16.8.13 更新Shi的Servlet 566
- 16.8.14 删除没有市的省 571
- 16.8.15 删除有市的省 571
- 第17章 JPQL必备技能 573
- 17.1 JPQL介绍 573
- 17.2 命名参数和索引式参数及实体参数式查询 573

- 17.2.1 参数索引式查询 574
 - 17.2.2 命名式参数查询 575
 - 17.2.3 为实现主从关联示例创建sheng表和shi表 576
 - 17.2.4 sheng表和shi表两种关联查询的方式 577
 - 17.3 JPQL支持的运算符 578
 - 17.3.1 +、-、*、/、=、>=、>、<、<=、<>、between、like、in运算符的使用 578
 - 17.3.2 not运算符的使用 581
 - 17.3.3 isnull运算符的使用 582
 - 17.3.4 isempty运算符的使用 582
 - 17.4 orderby的使用 583
 - 17.5 查询指定字段的示例 584
 - 17.6 聚合函数avg、count、max、min、sum的使用 586
 - 17.7 groupby和having的使用 587
 - 17.8 左外连接的使用 588
 - 17.9 通过distinct去除重复记录 590
 - 17.10 JPQL的字符串操作函数 590
 - 17.11 通过JPQL取得当前的日期和日期时间 594
 - 17.12 JPQL语言对日期的判断 595
 - 17.13 JPQL的数学函数 598
 - 17.14 JPQL中的分页功能 599
- 第18章 FreeMarker模板引擎的使用 600
- 18.1 输出8种简单数据类型 600
 - 18.2 简单数据类型的计算 603
 - 18.3 输出复杂数据类型——数组 604
 - 18.4 输出集合对象——List 605
 - 18.5 输出集合对象——Set 606
 - 18.6 输出集合对象——Map 606
 - 18.7 输出嵌套类型——List中有Map 607
 - 18.8 输出嵌套类型——Map中有List 608
 - 18.9 判断#if和#else标签的使用 609
 - 18.10 输出实体类的属性及boolean类型注意事项 609
 - 18.11 FreeMarker中的注释 611
 - 18.12 FreeMarker中的导入 611
 - 18.13 FreeMarker中对不存在的变量或null值的处理 612

精彩短评

- 1、.....spring mybatis搭框架那会看的
- 2、第一次收到同事送的书。慢慢学吧。
- 3、烂的一比。一个MyBatis，你告诉我怎么用就可以了，尼玛要说，怎么连接MySQL，怎么连接Oracle，怎么连接SqlServer，有必要吗？有必要吗？完全凑字数。坑钱货

章节试读

1、《Java EE核心框架实战》的笔记-第3页

```

<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE configuration PUBLIC
"-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"></transactionManager>
      <dataSource type="POOLED">
        <property name="driver"
          value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
        <property name="url"
          value="jdbc:sqlserver://localhost:1433;databaseName=j2eeTest" />
        <property name="username" value="sa" />
        <property name="password" value="Passw0rd" />
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="com/byhard/mybatistest/orm/userinfoMapper.xml"></mapper>
  </mappers>
</configuration>

```

2、《Java EE核心框架实战》的笔记-第58页

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
"http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
  <constant name="struts.devMode" value="true"></constant>
  <package name="com.byhard.Struts2Login" extends="struts-default">
    <action name="login" class="com.byhard.Struts2Login.controller.Login">
      <result name="toOKJSP">ok.jsp</result>
      <result name="toNOJSP">/no.jsp</result>
    </action>
  </package>
</struts>

```

3、《Java EE核心框架实战》的笔记-第210页

```

<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-class>
    org.springframework.web.filter.CharacterEncodingFilter
  </filter-class>
  <init-param>
    <param-name>encoding</param-name>

```

```

    <param-value>UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/* </url-pattern>
</filter-mapping>

```

4、《Java EE核心框架实战》的笔记-第196页

Web.xml配置：

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">

  <servlet>
    <servlet-name>springMVC</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:spring-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>springMVC</servlet-name>
    <url-pattern>*.spring</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>

```

spring-servlet.xml配置：

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"

```



```
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
http://www.springframework.org/schema/cache
http://www.springframework.org/schema/cache/spring-cache-4.0.xsd">
```

```
<context:component-scan
base-package="com.byhard.springlogin.controller"></context:component-scan>
</beans>
```

5、《Java EE核心框架实战》的笔记-第215页

415错误需要参考文章：

<http://blog.csdn.net/yixiaoping/article/details/45281721>

重要配置：

spring 4.x要配 jackson2.x

```
jackson-annotations-2.6.2.jar
jackson-core-2.6.2.jar
jackson-databind-2.6.2.jar
```

spring-servlet.xml配置：

```
<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">
<property name="messageConverters">
<list>
<ref bean="jsonHttpMessageConverter" />
</list>
</property>
</bean>
```

```
<bean id="jsonHttpMessageConverter"
class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">
<property name="supportedMediaTypes">
<list>
<value>application/json;charset=UTF-8</value>
</list>
</property>
</bean>
```

mvc配置：

```
<mvc:annotation-driven />
```

jsp中ajax请求：

```
function userinfo(username, password){
    this.username = username;
    this.password = password;
}

function sendAjax2(){
    var userinfoRef = new userinfo('高洪岩new123', '123new');
    var jsonStringRef = JSON.stringify(userinfoRef);
    $.ajax({
        type: 'POST',
        data: jsonStringRef,
        url: 'createJSONObjectURL.spring?t=' + new Date().getTime(),
        dataType: 'json',
        contentType: 'application/json;charset=UTF-8'
    });
}
```

controller测试文件：

```
@RequestMapping(value = "/createJSONObjectURL", method = RequestMethod.POST, consumes =
"application/json")
public String createJSON2(@RequestBody Userinfo userinfo) {
    java.util.Date date = new java.util.Date();
    System.out.println(date.toGMTString() + ":username value=" + userinfo.getUsername());
    System.out.println(date.toGMTString() + ":password value=" + userinfo.getPassword());
    return "index.jsp";
}
```

6、《Java EE核心框架实战》的笔记-第12页

```
<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD
Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.byhard.mybatisitest.orm">
    <resultMap type="com.byhard.mybatisitest.orm.Userinfo" id="userinfoResultMap">
        <id property="id" column="id" />
        <result property="username" column="username" />
        <result property="password" column="password" />
    </resultMap>

    <insert id="insertUserinfo" parameterType="com.byhard.mybatisitest.orm.Userinfo">
        insert into
        userinfo(username, password)
        values(#{username},#{password})
```

```

</insert>

<!-- 查询用户，根据id -->
<select id="getUser" parameterType="int" resultType="com.byhard.mybatisest.orm.UserInfo"
    resultMap="userInfoResultMap">
    SELECT * from userinfo ui WHERE ui.id = #{id}
</select>

<!-- 查询用户列表 -->
<select id="getUserAll" resultType="com.byhard.mybatisest.orm.UserInfo"
    resultMap="userInfoResultMap">
    SELECT * from userinfo
</select>

<!-- 更新用户密码 -->
<update id="updateUserPassword" parameterType="com.byhard.mybatisest.orm.UserInfo">
    update userinfo
    set password = #{password}
    where id = #{id};
</update>

<!-- 根据用户id删除用户记录 -->
<update id="deleteUserById" parameterType="int">
    delete from userinfo where id = #{id};
</update>

<update id="findPwd" statementType="CALLABLE" parameterType="hashmap">
    <![CDATA[
        { call P_J2EE_GetValue(#{username, mode=IN, jdbcType=NVARCHAR}, #{pwd, mode=OUT,
        jdbcType=NVARCHAR}) }
    ]]>
</update>

</mapper>

```

7、《Java EE核心框架实战》的笔记-第213页

方式1：

```

@RequestMapping(value="/getJSONString")
public String getJSONString(@RequestParam("jsonString") String jsonString){
    ObjectMapper om = new ObjectMapper();
    try {
        @SuppressWarnings("rawtypes")
        Map map = om.readValue(jsonString, Map.class);
        System.out.println(map.get("username"));
        System.out.println(map.get("password"));
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

```

    }
    return "index.jsp";
}

```

方式2 :

8、《Java EE核心框架实战》的笔记-第55页

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">

```

```

<filter>
    <filter-name>Struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>Struts2</filter-name>
    <url-pattern>*.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>Struts2</filter-name>
    <url-pattern>*.js</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>Struts2</filter-name>
    <url-pattern>*.action</url-pattern>
</filter-mapping>
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>

```

9、《Java EE核心框架实战》的笔记-第197页

Url中同名的参数将要自动传递给控制层方法中同名的参数，并且不再需要@RequestParam("username")注解。

指定请求路径与请求方法：

```
@RequestMapping(value = "/login", method = RequestMethod.POST)
```

控制层重定向到控制层，带参数：

```

@RequestMapping(value = "/login", method = RequestMethod.POST)
public String login(@RequestParam("username") String username,

```

```
@RequestParam("password") String password, Model model){
if(username.equals("byhard") && password.equals("123")) {
    model.addAttribute("username", username);
    //return "ok.jsp";
    return "redirect:/listUsername.spring";
} else {
    //return "no.jsp";
    return "redirect:/listUsername2.spring?username=" + username;
}
}
```

匹配url路径，执行指定的controller

```
@RequestMapping("/test1/{userId}")
public String test1(@PathVariable String userId){
    System.out.println("run test1 userId=" + userId);
    return "/index.jsp";
}
```

```
@RequestMapping("/test2/{userId}")
public String test2(@PathVariable("userId") String userIdParam){
    System.out.println("run test2 userId=" + userIdParam);
    return "/index.jsp";
}
```

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.tushu111.com