

第 5 章 Hibernate5 框架技术入门

Hibernate 创造性地使用对象-关系映射(Object-Relation Mapping, ORM)模型,将关系数据转换为对象,简化数据的操作方式。在软件项目开发中,要能够创造性地提出问题的解决方案。

在 Java Web 项目开发中,有许多功能模块需要连接数据库系统,实现对数据库表的操作。在 Java 程序设计和 JSP 程序设计技术中,使用 JDBC(Java DataBase Connectivity)技术连接数据库。为了实现与数据库的高效操作,提高 Java Web 的项目性能,可以使用 Hibernate5 框架技术。本章主要介绍 Hibernate5 的基本内容。

本章主要内容如下。

- (1) Hibernate5 的发展与特点。
- (2) Hibernate5 软件包的下载与配置。
- (3) Hibernate5 的工作原理。
- (4) Hibernate5 的核心组件。
- (5) 基于 Struts2+Hibernate5 的登录和注册系统。

5.1 Hibernate5 的基础知识

Hibernate 是封装了 JDBC 的一种开放源代码的对象-关系映射框架,使程序员可以使用面向对象的思想操作数据库。Hibernate 是一种对象-关系映射的解决方案,即将 Java 对象与对象之间的关系映射到数据库表与表之间的关系。



Hibernate5
的基础知识

5.1.1 Hibernate5 的发展与特点

目前,Hibernate 是 Java 工程师招聘中要求必备的一门技能,也是 Java Web 三大经典框架之一。

2001 年,Hibernate1 发布,即 Hibernate 的第一个版本;2003 年,Hibernate2 发布,并在当年获得 Jolt 2003 大奖(Jolt 大奖素有“软件业界的奥斯卡”的美誉,共设通用类图书、技术类图书、语言和开发环境、框架库和组件、开发者网站等十余个分类大奖),同年,Hibernate 被 JBoss 公司收购,成为该公司的子项目之一;2005 年,JBoss 公司发布 Hibernate3;2006 年,JBoss 公司被 Redhat 公司收购;2012 年 11 月,Hibernate 4.1.8 发布;2015 年 3 月,Hibernate 5.0 发布。

Hibernate 是封装了 JDBC 与 ORM 技术的数据持久性解决方案。在 Java 世界中,Hibernate 是众多 ORM 软件中获得关注最多、使用最广泛的框架。它成功地实现了透明持久化,以面向对象的 HQL 语句封装 SQL 语句,为开发人员提供了一个简单灵活且面向对象的数据访问接口。Hibernate 是一个开源软件,开发人员可以很方便地获得软件源代码。

当遇到问题时,程序员可以深入源代码中查看究竟,甚至修改 Hibernate 内部错误并将修改方案提供给 JBoss 公司,从而帮助 Hibernate 框架技术改进。

Hibernate 自发布以来受到业界的欢迎,目前许多 Java 程序员学习和使用它来开发商业应用软件。另外,网络上有大量介绍和讨论 Hibernate 应用的文章,JBoss 网站也提供了一个完善的社区,所以一旦在使用中遇到问题,开发者可以轻松地在网络上搜索到相应的解决方法,这又进一步吸引了更多的程序员来学习 Hibernate,吸引更多的公司采用 Hibernate 开发软件。

Hibernate 为使用者考虑得十分周全,对于一个普通的程序员来说,只需学习不到 10 个类的用法就可以开发基于 Hibernate 框架的应用系统,实际使用起来十分方便。

Hibernate 提供了透明持久化功能,支持第三方框架,即能与其他框架进行整合,如 Struts、Spring 等,不但提供面向对象的 HQL,而且支持传统的 SQL 语句。

在基于 MVC 设计模式的 Java Web 应用中,Hibernate 可以作为应用的数据访问层或持久层。它具有以下特点。

(1) Hibernate 是一个开放源代码的对象-关系映射框架,它对 JDBC 进行了非常轻量级的对象封装,使得 Java 工程师可以随心所欲地使用面向对象编程(Object Oriented Programming, OOP)思维来操作数据库。Hibernate 可以应用在任何使用 JDBC 的场合,既可以在 Java 的客户端程序使用,也可以在 Servlet、JSP 的 Java Web 应用中使用,最具革命意义的是,Hibernate 可以在 Java EE 框架中取代 CMP,完成数据持久化的重任。

(2) Hibernate 的目标是成为 Java 中处理数据持久性问题的一种完整解决方案。它协调应用程序与关系数据库的交互,把开发者解放出来,专注于项目的业务逻辑问题。

(3) Hibernate 是一种非强迫性的解决方案。开发者在写业务逻辑和持久化类时,不会被要求遵循许多 Hibernate 特定的规则和设计模式。这样,Hibernate 就可以与大多数新的和现有的应用程序进行集成,而不需要对应用程序的其余部分做破坏性改动。

5.1.2 Hibernate5 软件包的下载和配置

本书的实例和项目使用的是 Hibernate 5.2.12 版本,2017 年 10 月发布,如需使用 Hibernate 其他版本进行 Web 项目开发,可以在其官方网站下载。

1. 软件包下载

由于 Hibernate 先被 JBoss 公司收购,后来 JBoss 公司被 Redhat 公司收购,所以 Hibernate5 可以在以下 3 个网站下载: www.redhat.com、www.jboss.org 和 www.hibernate.org。根据需要在上述 3 个网站下载自己要使用的 Hibernate 版本。官方页面之一如图 5-1 所示。

单击图 5-1 所示页面中的 Hibernate ORM,出现图 5-2。

单击图 5-2 所示页面中的选择点 5.2 后,出现如图 5-3 所示的版本选择页面,在该页面的下方单击 5.2.12.Final 区域中的 Download 按钮进行下载。

单击图 5-3 所示页面中 Download 按钮后,出现图 5-4,选择下载路径后进行下载。

2. Hibernate5 软件包中主要文件

解压缩 `hibernate-release-5.2.12.Final.zip` 文件后得到一个名为 `hibernate-release-5.2.12.Final` 的文件夹,该文件夹结构如图 5-5 所示。

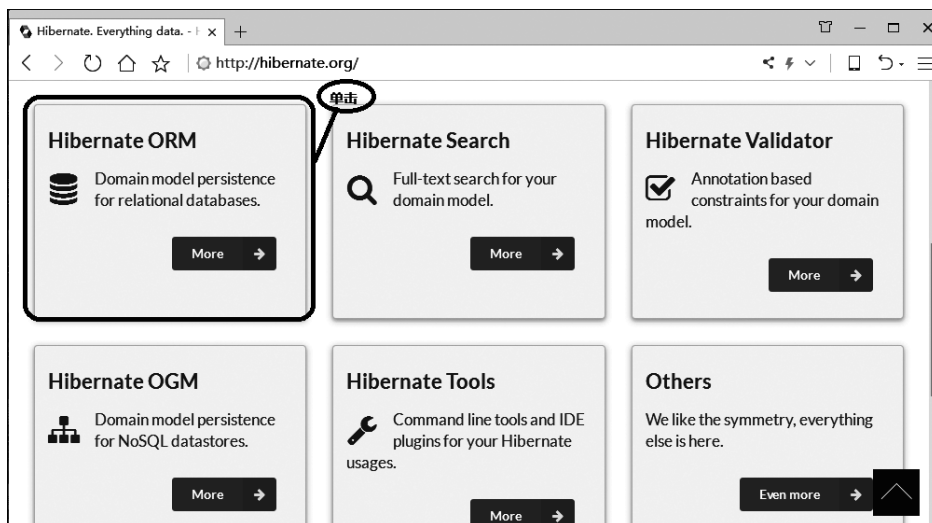


图 5-1 Hibernate 官方页面

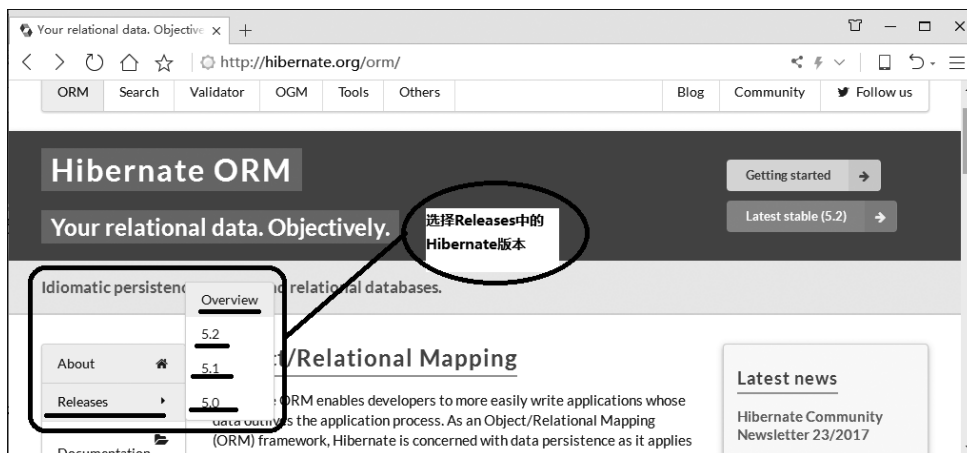


图 5-2 选择下载页面

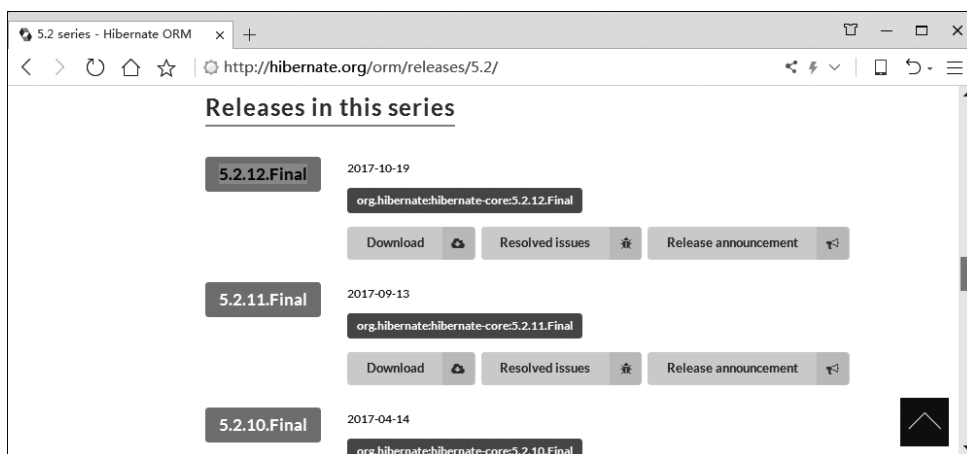


图 5-3 版本选择页面

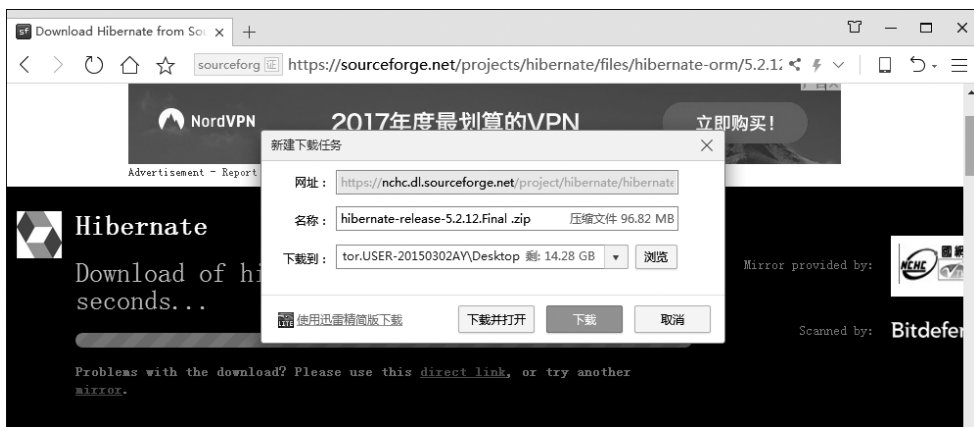


图 5-4 选择下载路径

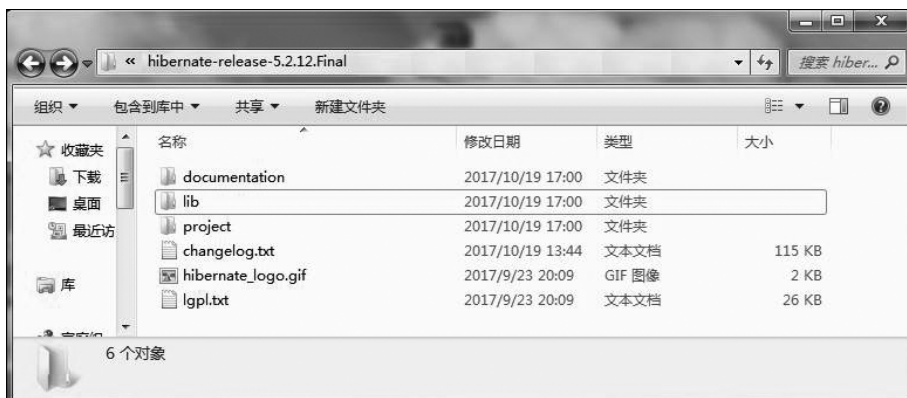


图 5-5 Hibernate5 文件夹结构

(1) documentation 文件夹：存放 Hibernate5 的相关文档,包括 Hibernate5 的参考文档和 API 文档等。

(2) lib 文件夹：存放 Hibernate5 框架的核心类库以及 Hibernate5 的第三方类库。该文件夹下的 required 子目录存放运行 Hibernate5 项目时必需的核心类库。

(3) project 文件夹：存放 Hibernate5 项目的源代码。

3. Hibernate5 的配置

Hibernate5 的 lib 文件夹有 6 个子目录,需要在项目的类库中添加 required 和 jpa-metamodel-generate 子目录下面的所有 JAR 文件,其他目录中的 JAR 文件可根据项目的实际需求添加。例如,使用连接池需要添加 lib\optional\c3p0 下面的 JAR 文件和使用缓存功能需要使用 lib\optional\chcache 下面的 JAR 文件。

由于 NetBeans 8.2 和 MyEclipse 2017 中都集成了 Hibernate,所以可以使用工具中自带的 Hibernate。集成的 Hibernate 版本一般不是 Hibernate 的最新版本。由于 Hibernate 各版本之间存在一些细节差异,有可能在配置文件和映射文件中存在差异导致项目无法运行。使用本书进行项目开发时建议使用 Hibernate 5.2.12。在 NetBeans 8.2、MyEclipse 2017 和 Eclipse 中配置 Hibernate 与第 1 章中介绍的配置 Struts 2.5.10 的方法相似,这里不

再介绍。

5.1.3 Hibernate5 的工作原理

Hibernate5 的工作原理如图 5-6 所示。

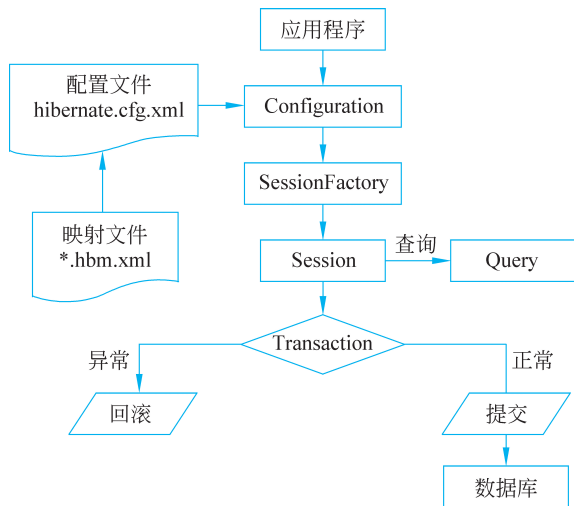


图 5-6 Hibernate5 的工作原理

Hibernate5 的工作原理如下。

首先, Configuration 读取 Hibernate5 的配置文件 hibernate.cfg.xml 和映射文件 *.hbm.xml 中的信息,即加载配置文件和映射文件,并通过 Hibernate5 配置文件生成一个多线程的 SessionFactory 对象。然后,多线程 SessionFactory 对象生成一个线程 Session 对象。Session 对象生成 Query 对象或者 Transaction 对象。可通过 Session 对象的 get()、load()、save()、update()、delete() 和 saveOrUpdate() 等方法对持久化对象(Persistent Objects, PO)进行加载、保存、更新、删除等;也可利用 Query 对象执行查询操作;如果没有异常,Transaction 对象将提交这些数据到数据库中。

5.2 Hibernate5 的核心组件

基于 Hibernate5 框架的项目开发的关键点就是使用 Hibernate5 的核心类和接口,即核心组件。Hibernate5 的核心组件位于业务层和持久化层之间。Hibernate5 除核心组件外还包括 Hibernate5 配置文件(hibernate.cfg.xml 或 hibernate.properties)、映射文件(xxx.hbm.xml)和持久化对象。

1. Configuration

Configuration 负责配置并启动 Hibernate5,创建 SessionFactory 对象。在 Hibernate5 的启动过程中,Configuration 类的实例首先定位映射文件位置、读取配置,然后创建 SessionFactory 对象。

2. SessionFactory

SessionFactroy 负责初始化 Hibernate5。它充当数据存储源的代理,并负责创建



Hibernate5
的核心组
件

Session 对象,这里用到了工厂模式。需要注意的是,SessionFactory 并不是轻量级的,因为一般情况下,一个项目通常只需要一个 SessionFactory 即可,当需要多次操作数据库时,可以为每个数据库指定一个 SessionFactory 线程对象。SessionFactory 是产生 Session 实例的工厂。

3. Session

Session 负责执行 PO 的操作,它用 get()、load()、save()、update()和 delete()等方法来对 PO 进行加载、保存、更新及删除等操作。但需要注意的是,Session 对象是非线程安全的。同时,Hibernate5 的 Session 不同于 JSP 应用中的 HttpSession。这里使用的 Session 术语,其实指的是 Hibernate5 中的 Session。

4. Transaction

Transaction 负责事务相关的操作,用来管理 Hibernate5 事务,它的主要方法有 commit()和 rollback(),可以使用 Session 的 beginTransaction()方法生成。它是可选的,开发人员也可以设计编写自己的底层事务处理代码。

5. Query

Query 负责执行各种数据库查询。它可以使用 HQL 对 PO 进行查询操作。Query 对象可以使用 Session 的 createQuery()方法生成。

6. Hibernate5 的配置文件

Hibernate5 的配置文件主要用来配置数据库连接参数,例如,数据库的驱动程序、URL、用户名和密码、数据库方言等。它有两种格式:hibernate.cfg.xml 和 hibernate.properties。两者的配置内容基本相同,但前者比后者使用方便一些,如 hibernate.cfg.xml 可以在其 <mapping>子元素中定义用到的映射文件(xxx.hbm.xml)列表,而使用 hibernate.properties 时则需要在程序中以编码方式指明映射文件。一般情况下,hibernate.cfg.xml 是 Hibernate5 的默认配置文件。

7. 映射文件

映射文件(xxx.hbm.xml)用来把 PO 与数据库中的表、PO 之间的关系与表之间的关系及 PO 的属性与表字段一一映射起来,它是 Hibernate5 的核心文件。

8. 持久化对象

PO 可以是普通的 JavaBean,唯一特殊的是它们与 Session 相关联。PO 在 Hibernate5 中存在 3 种状态:临时状态(Transient)、持久化状态(Persistent)和脱管状态(Detached)。当一个 JavaBean 对象在内存中孤立存在,不与数据库中的数据有任何关联关系时,那么这个 JavaBean 对象就称为临时对象(Transient Object);当它与一个 Session 相关联时,就变成持久化对象;在 Session 被关闭的同时,对象也会脱离持久化状态,变成脱管对象(Detached Object),这时可以被应用程序的任何层自由使用,例如,可用作与表示层(V)打交道的数据传输对象。

5.3 基于 Struts2 和 Hibernate5 的登录和注册系统

下面使用 Struts 2.5.10 和 Hibernate 5.2.12 开发一个实现登录和注册功能的项目,该项目的文件结构如图 5-7 所示。将项目中使用到的 Struts 2.5.10、Hibernate 5.2.12 以及



基于 Struts2
和 Hibernate5
的登录和注
册系统

MySQL 的 JDBC 驱动 JAR 文件添加到项目 ch05 的“库”中,如图 5-8 所示。

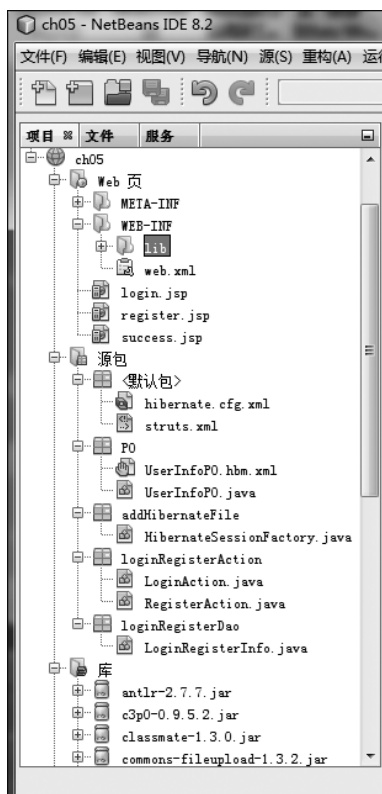


图 5-7 项目的文件结构

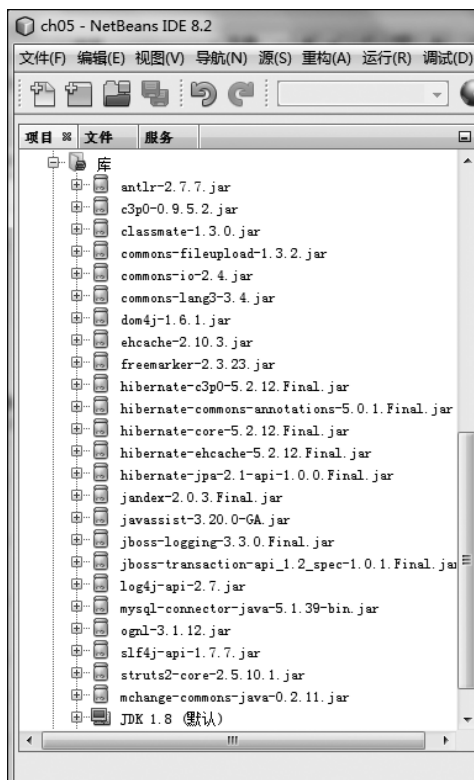


图 5-8 项目所需的 JAR 文件

1. 项目介绍

本项目实现用户登录和用户注册功能。有一个登录页面(login.jsp),代码如例 5-1 所示;登录页面对应的业务控制器为 LoginAction 类,该 Action 中覆盖了 validate()方法,使用手工验证对登录页面进行验证,该业务控制器类代码如例 5-4 所示;如果输入的用户名和密码都正确,进入登录成功页面(success.jsp),代码如例 5-2 所示;如果用户没有注册需先注册,注册页面(register.jsp)代码如例 5-3 所示;注册页面对应的业务控制器为 RegisterAction 类,代码如例 5-5 所示,注册成功后返回登录页面。此外,还需要配置 web.xml,代码如例 1-3 所示;在 struts.xml 中配置 Action,代码如例 5-6 所示。

该项目使用 MySQL 数据库。数据库名为 test,有一张名为 info 的表,表的字段名称、类型以及长度如图 5-9 所示。

在基于 Struts2+Hibernate5 的项目开发中,连接数据库时需要 Hibernate5 的配置文件 hibernate.cfg.xml 或者 hibernate.properties,本项目使用的是 hibernate.cfg.xml,代码如例 5-7 所示,配置文件主要用于加载数据库的驱动以及与数据库建立连接等,该配置文件一般与 struts.xml 文件放在同一位置,参考图 5-7。该配置文件在项目运行时需要加载,本项目编写一个加载该配置文件的类 HibernateSessionFactory,该类封装了配置文件的加载方法,即加载配置文件的 JavaBean,代码如例 5-8 所示。

另外,在使用 Hibernate5 时,项目所用数据库中的每张表都对应一个 PO。通过 PO 把

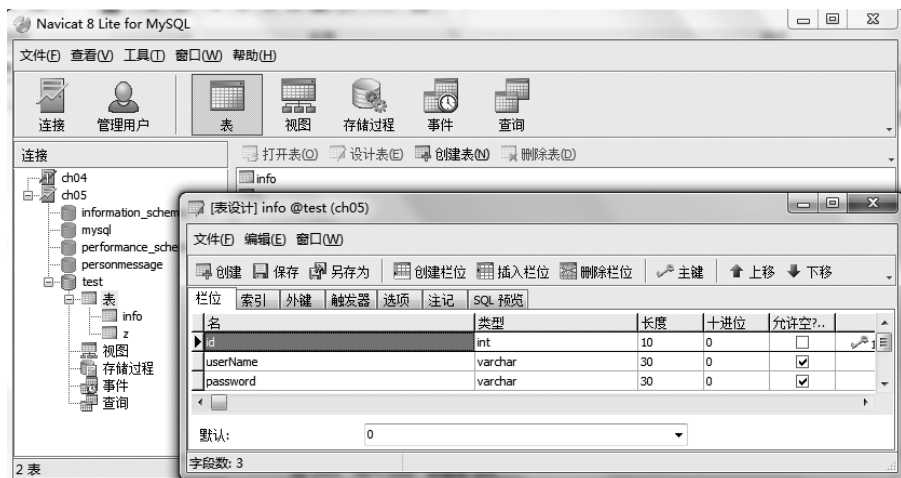


图 5-9 数据库 info 表的结构

页面中的数据保存起来并把数据存到数据库中。为了简化开发,本项目的登录页面和注册页面都使用同一张表,登录页面和注册页面中的数据都保存在 UserInfoPO 中,代码如例 5-9 所示。每个 PO 一般都会对应一个映射文件 UserInfoPO.hbm.xml,代码如例 5-10 所示,映射文件配置 PO 与数据库中表之间的映射关系;映射文件一般都放在和其对应的 PO 所在包中。为了封装登录和注册功能对数据的操作,即登录和注册页面要实现的业务逻辑,本项目编写一个名为 LoginRegisterInfo 的 JavaBean,该类提供了登录和注册业务处理,可以实现登录和注册的持久化业务操作,代码如例 5-11 所示。

以上文件所在的路径可参考项目文件结构图 5-7。

2. 在 web.xml 中配置核心控制器 StrutsPrepareAndExecuteFilter

参考 1.3.1 节中的例 1-3。

3. 编写视图组件(JSP 页面)

登录页面(login.jsp)运行效果如图 5-10 所示,代码如例 5-1 所示。



图 5-10 登录页面

【例 5-1】 登录页面(login.jsp)。


```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="s" uri="/struts-tags"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title><s:text name="基于 SH 的应用"></s:text></title>
  </head>
  <body bgcolor="#CCCCFF">
    <s:form action="login" method="post">
      <br><br><br><br><br><br>
      <table border="1" align="center" bgcolor="#AABBCCDD">
        <tr>
          <td>
            <s:textfield name="userName" label="用户名字"
              size="16"/>
          </td>
        </tr>
        <tr>
          <td>
            <s:password name="password" label="用户密码"
              size="16"/>
          </td>
        </tr>
        <tr>
          <td colspan="2" align="center">
            <s:submit value="登录"/>
          </td>
        </tr>
        <tr>
          <td colspan="2" align="center">
            <s:a href="http://localhost:8084/ch05/register.jsp">
              注册
            </s:a>
          </td>
        </tr>
      </table>
    </s:form>
  </body>
</html>

```

登录成功页面(success.jsp),代码如例 5-2 所示。

【例 5-2】 登录成功页面(success.jsp)。

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="s" uri="/struts-tags"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title><s:text name="基于 SH 的应用"></s:text></title>
  </head>
  <body bgcolor="#CCCCFF">
    <hr>
    <table border="0" align="center" bgcolor="#AABBCCDD">
      <tr>

```

```

        <td>
            欢迎${userName}, 登录成功!
        </td>
    </tr>
</table>
<hr>
</body>
</html>

```

注册页面(register.jsp)如图 5-11 所示,代码如例 5-3 所示。



图 5-11 注册页面

【例 5-3】 注册页面(register.jsp)。

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="s" uri="/struts-tags"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title><s:text name="基于 SH 的应用"></s:text></title>
    </head>
    <body bgcolor="#CCCCFF">
        <s:form action="register" method="post">
            <br><br><br><br><br><br>
            <table border="1" align="center" bgcolor="#AABBCCDD">
                <tr>
                    <td>
                        <s:textfield name="userName" label="用户名字"
                            size="16"/>
                    </td>
                </tr>
                <tr>
                    <td>
                        <s:password name="password1" label="用户密码"
                            size="18"/>
                    </td>
                </tr>
                <tr>
                    <td>

```