**第11章.持久性层**

在某个阶段, 需要保留应用程序数据或将其保存到辅助存储。有几种方法可用, 包括将它们存储到文件、关系数据库、xml 文档等。其中, 将数据持久保存到关系数据库是最可靠和最受欢迎的。此外, 还可以从现成的功能下购买对象到关系的数据库映射工具, 以帮助 java 程序员保留 java 对象。

如果没有映射工具, 您手头还有其他选项。其中包括数据访问对象 (dao) 模式、java 数据对象 (jdo)、开源库 (如 hibernate) 等。其中, dao 模式最容易学习, 在大多数应用程序中就足够了。本章介绍如何实现数据持久性的 dao 模式。

另请注意, 由于应用程序的许多部分可能需要持久化对象, 因此一个好的设计要求您为数据持久性创建一个专用图层。此持久性层抽象工厂模式下的 dao 模式。

dao 类中的每个方法都使用 sql 语句访问数据库。遗憾的是, sql 语句可能因数据库类型而异。例如, 要将记录插入表中, oracle 数据库支持序列的概念, 以便为新记录生成序列号。因此, 在 oracle 中, 您将执行两个操作: 生成序列号并插入新记录。相比之下, mysql 支持插入新记录时生成的自动数字。在这种情况下, 插入方法将取决于它将数据保留到的数据库。若要允许应用程序支持多个数据库, 可以修改 dao 模式实现以使用抽象工厂模式。图11.4 显示了习惯 dao 接口, 该接口定义了需要存在于某个客户 dao 对象中的方法。实现将绑定到数据库类型。图11.4 中的两个实现类是 ava

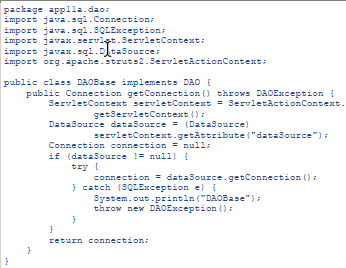
dao 接口和 dao 库类

dao 是所有 dao 类必须直接或间接实现的接口。dao 接口中只定义了一个方法 getConnection。清单11.1 给出了 dao 接口。

清单11.1。dao 接口

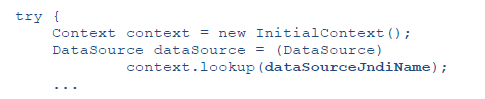
清单11.2 所示的 daobase 类提供了 dao 接口的 getConnection 方法的实现。

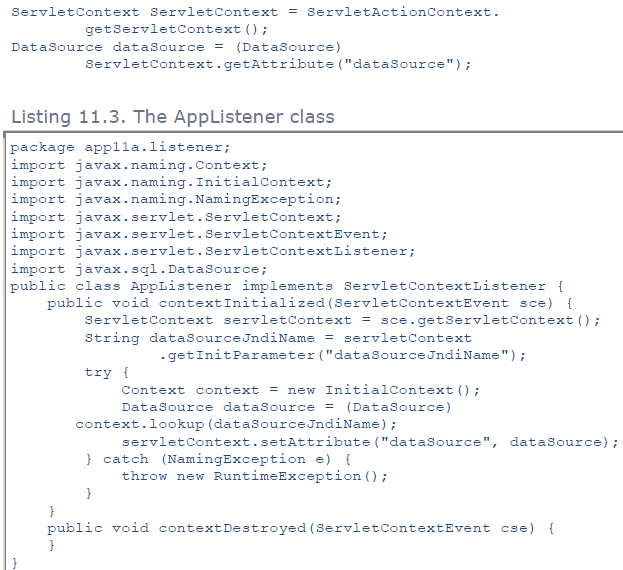
清单11.2。daobase 类



方法返回可由 dao 对象用于访问数据库的 java.sql.Connection。在 java se 中, 可以通过 java.sql.DriverManager 获取连接对象。但是, 在 java ee 中, 可伸缩性非常重要, 您肯定希望使用连接池快速获取连接对象。javax.sql.DataSource 支持连接池, 所有 java ee 容器都必须提供一个 datasource 对象, 从中可以获得连接对象。连接池非常重要, 您甚至可以在 tomcat 中找到此功能, 即使 tomcat 不是 java ee 容器。

在 java ee 中, 通过使用使用以下样板代码进行 jndi 查找来获取 datasource 对象:

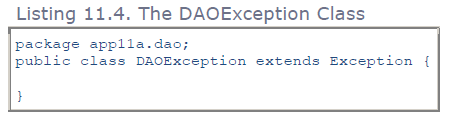


jndi 查找是昂贵的操作, 因此, 获取数据源占用大量资源。因此, 您可能希望缓存此对象, 服务器将对象将是缓存它的理想位置。在 app11a 中, 我们使用清单11.3 中的应用程序侦听器来获取一位 datasource 对象并将其存储在 servletcontext 对象中。之后, 在清单11.2 中的 daobase 类中, 您可以使用以下代码获取数据源: 在 tomcat 中的连接池要在 tomcat 中配置连接池, 请在 tomcat 的 server. xml 文件中的 <Host> 下添加此上下文元素。上面的上下文元素有助于创建一个 datasource 对象, 从中可以从池中获取 java.sql.Connection 对象。对象的详细信息在

元素。用户名和密码参数指定用于访问数据库的用户名和密码, driver类名称参数指定 jdbc 驱动程序, url 参数指定用于访问 mysql 数据库的数据库 url。url 参数指示数据库服务器与 tomcat (在 url 中使用本地主机) 驻留在同一台计算机中, 而数据源对象引用的数据库是测试数据库。

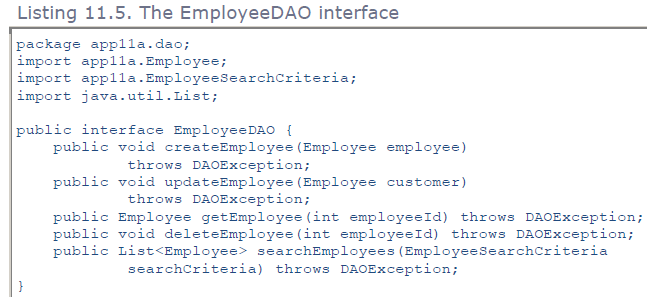
此外, 对于 dao 实现, 您可能希望扩展 java.lang.Exception 类, 使其具有自己的 dao 特定的异常。dao 对象中的方法可以引发此特定异常, 以便您可以提供处理数据访问和数据操作失败的代码。

一个简单的 dao 特定的异常类, named DAOException, is given in Listing 11.4.



雇员办公室接口

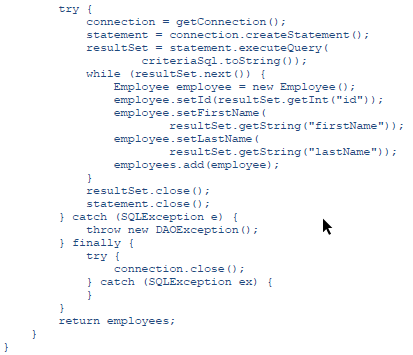
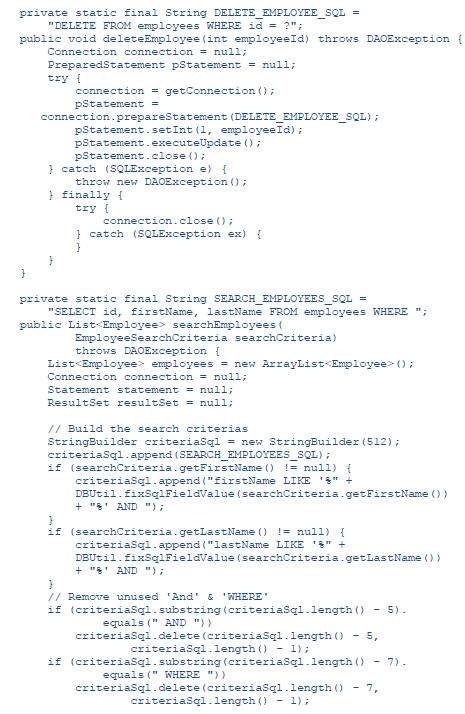
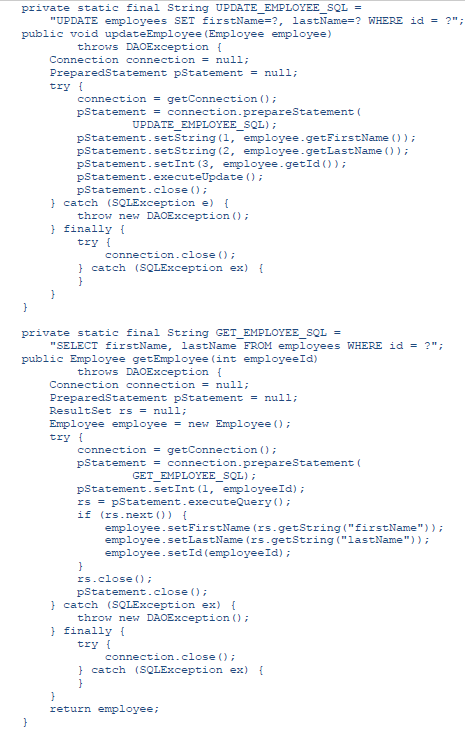
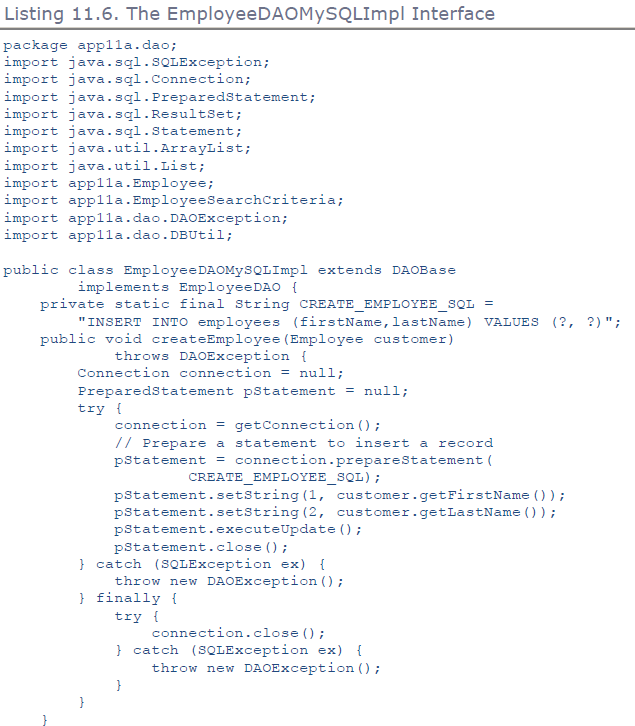
app11a 应用程序使用一个 dao 类, 雇员 dao。为了支持多个数据库, "雇员 dao" 被编写为定义占义工对象的方法的接口。清单11.5 显示了雇员 dao 接口。

方法接受要插入或更新的员工对象。方法接受员工标识符, 而搜索雇员方法接受 "雇员搜索标准"。

在 app11a 中, "雇员搜索" 类类似于 "雇员" 类, 但在其他应用程序中, 它可能包括与搜索相关的属性, 如 sortorder 和

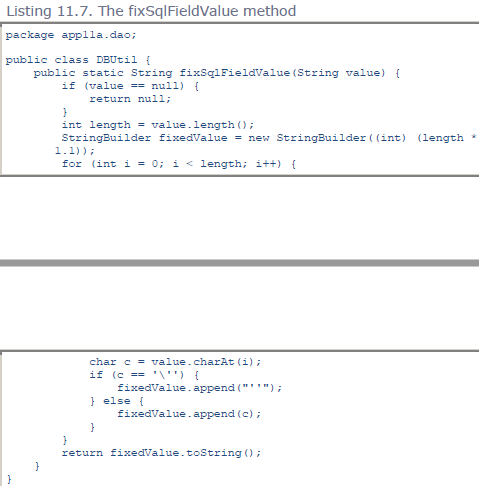
在员工中不存在的结果。因此, 需要另一个封装用户搜索条件的类。雇员人数

清单11.6 中介绍的雇员 daoysqlpll 类是 "雇员 dao" 接口的实现。为了支持另一个数据库, 您可以创建另一个雇员 dao 实现, 如雇员 daooracleimpl、雇员 daosqserverimpl 等。



除 searsel之所以使用的所有方法的 sql 语句都被定义为静态最终字符串, 因为它们永远不会更改。使它们成为静态最终值可避免一次又一次地创建相同的字符串。此外, 所有这些方法都使用 "预置语句" 而不是 java.sql.Statement 即使预置语句对象只执行一次。使用预置语句可以避免您必须检查其中一个参数是否包含单引号。使用语句时, 您必须转义参数中的任何单引号。

另一方面, searsteln 方法基于动态 sql 语句。这就要求我们使用语句对象。因此, 必须使用 dbuttil 类的 fixsqlfielvalue 方法检查参数中的单引号。清单11.7 介绍了 fixsqlfieldvalue 方法。



注意: 您可以将 fixsqlfieldvalue 方法替换为像这样的 string 类的所有方法。

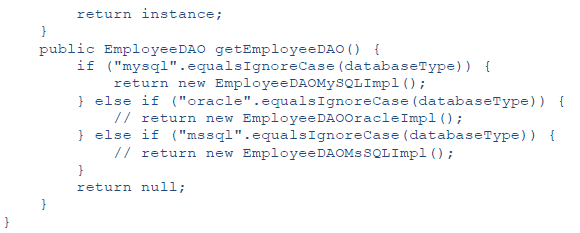
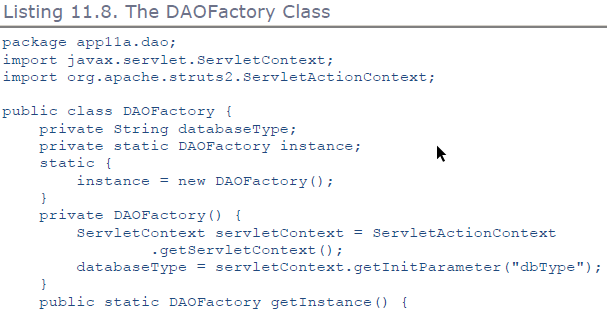


但是, 此方法计算密集型, 因为它使用正则表达式, 应避免在设计为可扩展的应用程序中使用。

dao工厂级

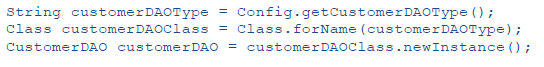
类帮助客户端实例化 dao 类。此外, 应用程序中 daod杆菌类的必要性来自于这样一个事实, 即在设计时不知道实现类名称, 例如, 它是雇员 daoysqlimpl 还是雇员 daooracleimpl。因此, daod厂类隐藏了创建 dao 对象的复杂性。

清单11.8 中介绍了 daodm表达类。

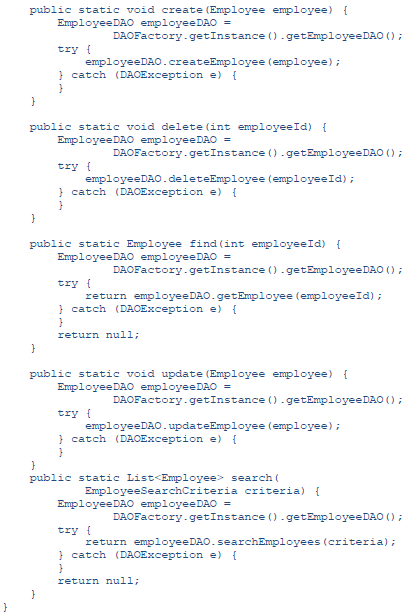
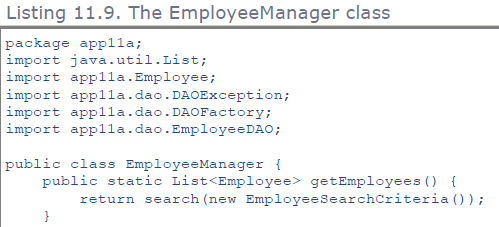


如果在编写应用程序时知道所有 dao 的实现类, 则可以使用 daodam幼特。这意味着, 如果您只考虑支持两个数据库 (mysql 和 oracle), 则事先知道雇员 dao 类的类型是雇员 daoysqlimpl 或雇员 daooracleimpl。如果将来您的应用程序需要支持 microsoft sql server, 则必须重写 dao向特杆菌类, 即在 get就此场类中添加另一个 if 语句。

如果使用反射创建 dao 对象, 则可以添加对更多数据库的支持, 而无需重新编译 dao工厂类。而不是您的 web. xml 文件中的 dbtype 参数, 您将有雇员 daotype。然后, 您的 daow动工厂类的 get也是为了使用以下代码。

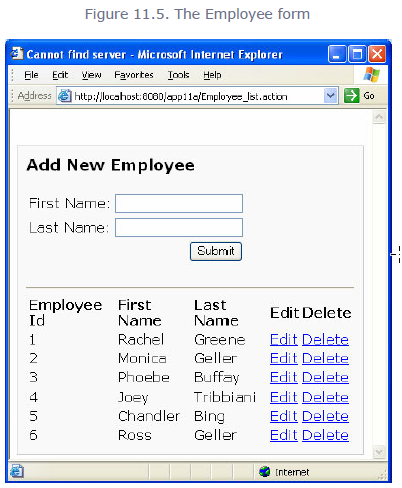
自雇人员类别

"雇员管理器" 类 (如清单11.9 所示) 是 dao 类的客户端。此类提供 struts 操作和 dao 类之间的另一层。



运行应用程序

app11a 应用程序提供用于创建新员工、更新和删除现有员工以及搜索员工的操作类。主要入口点是 "雇员" 列表操作。若要调用此操作, 请使用以下 url。

You will see something similar to Figure 11.5. 

首次运行此应用程序时, 将看不到现有员工的列表。

冬眠

在过去的几年里, hibernate 作为 java ee 和其他应用程序的加载项越来越受欢迎。其网站 (www.hibernate.org) 将这种免费产品宣传为 "一种功能强大、超高性能的对象--java 的关系持久性和查询服务"。

使用 hibernate, 您不需要实现自己的持久性层。而是使用工具创建数据库和相关表, 并确定应如何保留对象。hibernate 几乎支持当今市场上的各种数据库服务器, 其 hibernate 查询语言提供了 "对象和关系世界之间的优雅桥梁"。

在不久的将来, 更多的人将使用 hibernate。如果你有时间, 就投资吧。

总结

大多数应用程序都需要持久性层来持久化值对象。持久性层隐藏了从其客户端 (尤其是操作对象) 访问数据库的复杂性。持久性层可以通过使用 hibernate 等实现为实体 bean、dao 模式。

本章详细介绍了如何实现 dao 模式。此模式有许多变体, 您选择哪种变体取决于项目规范。最灵活的 dao 模式更可取, 因为如果将来需要更改应用程序, 您可以轻松地扩展应用程序。