文章编号:1001 - 9081(2005)12 - 2817 - 03

基于 Hibernate 与 Struts 框架的数据持久化应用研究

高 昂,卫文学
(山东科技大学 信息科学与工程学院,山东 青岛 266510)
(tomga@163.com)

摘 要:基于使用单一框架构建企业级应用时存在拓展性差,结构复杂的问题,给出使用 Hibernate 和 Struts 两个开源框架进行整合开发的实例,同时探讨如何配置和灵活使用两种框架,简化对象持久化映射工作和开发中的 MVC 分工,以及如何充分发挥两者优势,构建结构清晰、具备强大扩展性和维护性的 J2EE 应用。

关键词:J2EE;关系映射框架;Struts;MVC;对象/关系映射;对象持久化
中图分类号: TP311 **文献标识码**:A

Application of Java data persistence with Hibernate and Struts framework

GAO Ang, WEI Wen-xue
(*College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao Shandong* 266510, *China*)

Abstract: To solve the problems such as poor expansibility and difficult work division in building Web application, a best practice of how to combine and deploy Hibernate and Struts framework to build high performance object/ relational persistence and MVC structure was provided. The combination of using them could construct a standard J2EE Web application and also improve the development efficiency greatly.

Key words: J2EE; Hibernate; Struts; MVC(Model, View, Control) ; O/R mapping; object persistence

1 Hibernate 的数据持久化技术

Hibernate 是一个开放源代码的 O/R Mapping(对象关系映射框架),它对 JDBC 进行了轻量级的对象封装,使 Java 程序员可以方便地使用对象编程思维来操纵数据库。Hibernate 的目标是简化开发者通常的数据持久化编程任务,它可以把对象模型表示的对象映射到关系型数据库中,同时提供了数据查询和获取的方法,以大幅度减少开发时人工使用 SQL 和 JDBC 处理数据的时间。

通过使用 Hibernate 可以轻松解决开发者在使用传统的 CMP 以及 JDBC 和 DAO(Data Access Object)技术开发持久层时遇到的困难,即很难把关系表记录完整地映射到持久对象的关系上来,主要体现在多表的关系无法直接映射到对持久对象的映射上来。

1.1 Hibernate 原理

Hibernate 帮助基于普通的 Java 对象模型的持久对象的创建,允许持久对象拥有复杂的结构如混合类型、集合和属性,还可以拥有用户自定义的类型。现在这些持久对象可以有效地反映出底层数据库模式的复杂结构。为了提高效率,Hibernate 包括了一些策略,如与数据库交互时的多重最优化,包括对象的缓存、有效外部连接的获取、必要时 SQL 语句的执行。

Hibernate 在构建 Java 应用中的作用是:应用程序通过 Hibernate 对数据库进行访问,对数据持久层操作,而 Hibernate 自身通过 properties 配置文件和 XML Mapping 映射文件将类映射到数据库的记录。从而 Application 应用可以通过模型中起着应用与数据库之间的桥梁作用的 PO 这个特殊的 Java Class 直接访问数据库,而不是必须使用 JDBC 和 SQL 进行数据的操作。经上述过程,通过 Hibernate 实现关系数据库的持久化操作。

1.2 选择 Hibernate 的原因及其应用优势

Hibernate 是 Java 开源项目,用户可以在需要的时候对源代码进行改写,对其部分功能进行定制和拓展。同时,Hibernate 具有一支积极活跃的开发队伍,这使其产品有稳定的发展保障。

同时由于 Hibernate 对 JDBC 进行了轻量级的对象封装,Hibernate 的 Transaction 实际上是底层的 JDBC Transaction 的封装,这样简化了数据持久层的开发与调试,大大减轻了程序员的负担。

2 基于 MVC 模式的 Web 框架 Struts 的应用

2.1 Struts 的框架结构

Struts 是基于模型(Model)、视图(View)、控制器(Controller)的 MVC 模式应用架构,MVC 减弱了业务逻辑接口和数据接口之间的耦合,常被用来帮助开发者控制设计变化。在 Struts 框架中,Model 代表的是应用的业务逻辑,通过 JavaBean、EJB 组件实现;View 是应用的表示层,由 JSP 页面产生;Controller 是提供应用的处理过程控制,一般是 Servlet。通过这种设计模型把应用逻辑,处理过程和显示逻辑分成不同的组件实现,组件之间可以进行交互和重用。这种组件化的优点更易于实现对大规模系统的开发和管理。

2.2 应用 Struts 的作用及优势

Struts 清晰地划分了控制部分,事务逻辑和外观视图,让开发者遵循一个统一的模式进行设计编码,简化了系统后期

收稿日期:2005 - 06 - 06;**修订日期**:2005 - 08 - 30

作者简介:高昂(1982 -),男,山东泰安人,硕士研究生,主要研究方向:计算机网络系统、数据库系统、软件工程; 卫文学(1967 -),男,山西运城人,副教授,主要研究方向:计算机网络系统、网络工程、数据库系统.

维护的工作量,尤其是当其他开发者接手项目时,这种优势体现得更加明显。

Struts 给开发者提供了良好的页面导航功能,开发者可以通过配置文件 struts-config. xml 把握整个系统各部分之间的相互关联,清晰地掌握整个系统的体系结构。同时,Struts 对 Taglib 标签库进行了扩展,使之功能更为强大。使用 Taglib 可以简化 JSP 页面的开发,使得 JSP 包含最少的代码,同时也使开发者可以更方便灵活地在 Struts 中控制程序的流程。

3 Hibernate 与 Struts 结合构建 Web 应用

在具体应用中,目标系统将实现高校新生登记信息的数据库插入、更新及删除功能。在系统开发过程中,使用功能强大、拓展性强的 Eclipse 3.0 作为开发环境,同时加载了 Eclipse 的第三方插件 MyEclipse 3.8.2 以对 Struts 及 Hibernate 提供更好地控制和支持;在数据库方面,选择了稳定性及易用性较好的 SQL Server 2000;Web 容器则使用轻便、通用的 Jakarta Tomcat 5.0。在以上开发环境中,目标系统将清晰地展现 Hibernate 与 Struts 的作用范围及其相互协作关系。

3.1 Hibernate 配置

Hibernate 配置文件可以有两种格式,即 hibernate. properties 和 hibernate. cfg. xml。在本系统中,使用后者对 Hibernate 进行配置,因为使用 XML 文件更加方便直观,当增加 Hibernate 映射文件的时候,可以直接在 hibernate. cfg. xml 里面增加,不必像 hibernate. properties 必须在初始化代码中加入。使用 XML 时部分配置代码可以通过第三方的开发插件协作生成,减少了开发者的工作量。

在对 Hibernate 的配置中,主要涉及了 XML 文档的类型定义、数据库连接的各种参数以及供 Hibernate 管理事务、产生 SQL 和管理 JDBC 连接时所涉及的数据映射文件等。

配置文件 hibernate. cfg. xml 代码如下所示:

```
<?xml version = '1.0' encoding = 'UTF-8'? >
<! DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/
  Hibernate Configuration DTD 2. 0//EN" " http: //hibernate.
  sourceforge. net/hibernate-configuration-2. 0. dtd" >
<hibernate-configuration >
  <session-factory >
    <! -- properties -- >
    <property name = "connection. username" > sa </property >
    <property name = " connection. url" > jdbc: jtds: sqlserver: //
      192. 168. 100. 1: 1433; DatabaseName = Student </property >
    <property name = " dialect" > net. sf. hibernate. dialect.
      SQLServerDialect </property >
    <property name = " connection. password" > password </
    property >
    <property name = "connection. driver_class" > net. sourceforge.
      jtds. jdbc. Driver </property >
    <! -- mapping files -- >
    <mapping resource = " edu/sdust/hibernate/Student. hbm.
      xml"/ >
  </session-factory >
</hibernate-configuration >
```

在 Hibernate 的 properties 属性配置文件中,定义了链接池访问数据库的 URL 资源定位地址及数据库名称 Student,以及符合 SQL Server 规范的 dialect 方言,数据库链接驱动则使用了开源组织 SourceForge 提供的 jTDS 驱动对 SQL Server 2000 数据库访问。在 mapping 映射文件配置部分,定义了添加新生登记信息的 Student 表对应的映射文件 Student. hbm. xml,其他用到的映射资源可以随着系统开发进行灵活的加载与更新。

3.2 定义持久化类

Hibernate 作用与普通的 Java 对象使之成为持久化类。Persistent object 持久对象是一个完全符合 Java Bean 规范的纯 Java 对象,它包含有符合统一标准的属性和方法。POJO 是一个功能单一的 Java 对象,它不同于 EJB 这样的带有繁重的容器控制功能的对象,其属性只可以通过自身的 get 和 set 方法访问,这样对外隐藏了内部实现的细节,规范了事务处理部分中每个属性所对应的数据库字段的数据操作。

为了简化说明,在 Student 类中只定义了三个属性,即 StudId、StudName 和 StudTitle,它的属性和数据库中 Student 表的字段是一一对应的,并且类型一致。

将数据表中 StudId 对应的持久化类结构列出如下:

```
package edu. sdust. hibernate;
public class Student
  {
    private String StudId;
    public Student ( ) { }
    public java. lang. Integer getStudId( )
      { return StudId; }
    public void setStudId( java. lang. Integer StudId)
      {
        this. hashValue = 0;
        this. StudId = StudId;
      }
  }
```

在简单的持久化类中,定义了 StudId 属性以及getStudId( )和 setStudId 方法用于操作属性所对应的数据库字段。Hibernate 对属性使用的类型不加限制,但需要注意的是,各属性中 StudId 是一个特殊的属性,代表了这个类的数据库标识符,即数据表中的主键,由于持久化类 Student 将映射到数据库中对应的表,所以主键属性的设置在类中是不可缺省的。

3.3 对持久化对象进行映射定义

数据库表对应的映射文件 Student. hbm. xml 包含了对象/关系映射所需的元数据。元数据中包含了持久化类的声明,以及类中各个属性到数据库表各个字段的映射关系。其属性可以作为一般值存在也可以是指向其他实体的关联,其在关系型数据库中体现为数据表的外键。

Student. hbm. xml

```
<?xml version = "1.0"? >
<! DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate
  Mapping DTD 2. 0//EN" " http: //hibernate. sourceforge. net/
  hibernate-mapping-2. 0. dtd" >
<hibernate-mapping package = "edu. sdust. hibernate" >
  <class name = "Student" table = "Student" >
    <id name = "StudId" column = "Stud_id" type = "java. lang.
      Integer" >
      <generator class = "native"/ >
    </id >
    <property name = "StudName" column = "Stud_name" type = "
      java. lang. String" not-null = "true" / >
    <property name = "StudTitle" column = "Stud_title" type = "
      java. lang. String" / >
    <many-to-one name = " Specialty" class = "Specialty" column
      = "SpecialtyNo"/ >
  </class >
</hibernate-mapping >
```

在持久化类中定义 Student 类的 StudId 为标识属性,其主键生成方式为 native。Hiernate 附带了多种不同的标识符生成器,分别适用于不同场合。使用 native 本地生成方式时将根据所选用数据库的规范选择 identity,sequence 或者 hilo 方式中的一个生成主键。

Student 类中其他属性也将通过 Hibernate 映射到数据表中,属性 StudName 和 StudTitle 被声明为 String 类型,它们被分别映射到数据表中对应的字段,其中设定 StudName 为非空值。接下来的 many-to-one 则展示了 Hibernate 中实体之间一对多关系的定义方式,在此定义了 Student 类与 Specialty 类的关联,即学生与其所在专业信息的一对多关联。

3.4　实现持久化操作

数据持久化操作实现时,一般不直接在 JSP 中加载 Hibernate 框架的数据操作,而是把数据库存取的业务逻辑封装在 JavaBean 中,然后在 JSP 中通过调用 JavaBean 来访问 Hibernate 封装的对象。各部分之间的逻辑关系控制及表示层、业务层的分工由 Struts 框架统一管理。

利用 Hibernate 的 Session 从数据库中存取 Stduent 记录。首先,要从 SessionFactory 中获取一个 Hibernate 的 Session 工作单元。Session 是介于数据连接与事务管理的中间接口,它的作用相当于一个持久对象的缓冲区,Hibernate 可以通过 Session 检测到持久对象的改变,并及时刷新数据库。进行数据操作前先设置对应的 XML 配置文件 Hibernate. cfg. xml,然后从 SessionFactory 类中取得 Session 的实例:

```
SessionFactory sessionFactory = new Configuration( ). configure( "/
  conf/hibernate. cfg. xml") . buildSessionFactory( ) ;
```

得到了全局的 SessionFactory,就可以从中获取 session,进行对象的操作,如查询、修改、插入、删除等。同时还可以使用 Hibernate Query Language 编写简化的 SQL 查询语句。

其中的数据库插入操作定义如下:

```
session = SessionFactory. currentSession( ) ;
session. save( data) ;
session. flush( ) ;
```

其中的数据库查询操作定义如下:

```
session = SessionFactory. currentSession( ) ;
Query query = session. createQuery ( " select StudName from edu.
  sdust. hibernate. Student Student order by Student. StudId") ;

return query. list( ) ;
```

在对数据库执行查询操作时,使用了 Hibernate Query Language 进行查询。HQL 是 Hibernate 提供的一种简便的查询语言,它是完全面向对象的,并且具备继承、多态和关联等特性。查询执行后 Hibernate 将返回 Student 表中查询得到的各个字段。

需要注意的是,Hibernate 的每个数据库操作都是在一个事务(Transaction)中进行的,这样就可以隔离开不同的数据库操作,假如程序部署到一个由容器管理事务的环境中去,源代码不需要进行更改就可以直接应用。

3.5　用 Struts 实现逻辑控制

目标系统内部的控制逻辑利用 Struts-config. xml 文件来配置。具体配置如下:

```
< ?xml version = "1. 0" encoding = "UTF-8" ? >
< ! DOCTYPE struts-config PUBLIC "-//Apache Software
  Foundation//DTD Struts Configuration 1. 1//EN " " http: //
  jakarta. apache. org/struts/dtds/struts-config_1_1. dtd" >
< struts-config >
  < data-sources/ >
  < form-beans >
    < form-bean name = " StudentForm " type = " edu. sdust.
      formbean. StudentForm" / >
  </form-beans >
  < action-mappings >
    < action
      attribute = "StudentForm"
      input = "/AddStudent. JSP"
      name = "StudentForm"
      path = "/AddStudent"
      scope = "request"
      type = "edu. sdust. action. AddStudent" >
      < forward
        name = "success"
        path = "/AddStudent. JSP"
        contextRelative = "true" / >
      </action >
    </action-mappings >
    < controller bufferSize = "4096" debug = "0" / >
    < message-resources parameter = " edu. sdust. struts.
      ApplicationResources" / >
  </struts-config >
```

输入页面 AddStudent. JSP 将输入的学生信息提交到控制器 AddStudent. do 进行处理,同时在 AddStudent 类中新建 Student 类的实例,调用 Hibernate 的数据操作服务将学生信息插入数据库。操作成功后由 forward 控制转回到 AddStudent. JSP 继续进行数据添加并且显示添加后结果。

表示层中将展示 Hibernate 与 Struts 协同完成系统功能的效果,JSP 页面中提交学生数据时部分代码如下:

```
< html: form action = "AddStudent. do" method = "post" >
  < table border = 1 >
    < tr > < td > StudentName: </td > < td > < html: text property
      = "StudName" / >
    < td > StudentTitle: </td > < td > < html: text property = "
      StudTitle" / > </td > </tr >
  </table > < br/ >
  < html: submit/ >
</html: form >
```

JSP 页面中显示已录入的学生数据时代码如下:

```
< logic: iterate id = "element" name = "students" scope = "request"
  type = "edu. sdust. hibernate. Student" >
  < tr >
    < td > < bean: write name = "element" property = "StudName" / >
    </td >
    < td > < bean: write name = "element" property = "StudTitle" / >
    </td >
  </tr >
</logic: iterate >
```

通过上述表示层代码,实现了新生数据录入及查询的用户界面,并且通过 Struts Taglib 的协同作用将 JSP 代码在表示层中减小到最少,使得程序代码与页面显示相分离。

4　结语

Hibernate 是一个非常优秀的对象关系映射框架,利用 Hibernate 的对象持久化服务,对 JDBC 进行轻量级的对象封装,简化开发者使用 SQL 和 JDBC 处理数据的时间。

Struts 则是一个逐步走向成熟的符合 MVC 规范的 J2EE 应用框架,凭借其成功的设计模式以及众多的资源支持,Struts 现已成为构建大中型 Web 应用的首选方案之一。

文章编号:1001 - 9081(2005)12 - 2820 - 04

高可靠性分布式虚拟存储系统的研究

曹 楠,康慕宁
(西北工业大学 计算机学院,陕西 西安 710072)
(GlassesWorm@yahoo.com.cn)

摘 要:针对分布式存储安全性及可靠性方面的一系列问题,提出了全新的分布式存储解决方案 RayStorage 系统,阐述了该系统的核心分布式虚拟存储架构(Distributed Virtual Storage Architecture, DVSA)的主要思想,并重点给出了 DVSA 中有关存储模式的若干概念及其效率分析结果。

关键词:分布式系统;虚拟网络存储;存储模式
中图分类号: TP309.3 **文献标识码**:A

Study on a high reliable distributed virtual storage system

CAO Nan, KANG Mu-ning
(*College of Computer, Northwestern Polytechnical University, Xi'an Shaanxi* 710072, *China*)

Abstract: Aiming at the security and stability problems in distribute storage, a new distributed storage solution called RayStorage system was presented, and the main notion about "Distributed Visual Storage Architecture"(DVSA) which is the kernel of the ReyStorage System was discussed, and the basic concepts of storage pattern and the results of their efficiency analysis was given out.

Key words: distribute system; virtual network storage; storage pattern

0 引言

随着计算机技术的发展,存储设备容量不断增长,在带来数据存储便利的同时,有研究表明[1],在 Windows2000 系统中,存储设备所拥有的资源并没有得到充分的利用,并且这种趋势随着存储设备容量的不断增长而愈加严重。同时,网络技术给存储界带来了全新的变革,由于本地数据传输速率与网络传输速率之间差异正在逐渐缩小,数据存储正在经历着由本地化到网络化的巨大转变。因此,如何利用分布于网络中各个异构主机节点之上的不可靠空闲存储资源来实现分布式的网络存储具有非常重要的研究与应用价值。

本文所提出的 RayStorage 系统是一种应用在 Internet 范围内的分布式虚拟存储系统,系统运行在一系列分布的、不可靠的,并且平台异构的主机节点之上,通过定制架构在主机级虚拟存储概念之上的虚拟存储服务,实现了对网络存储资源的管理、分配与利用。由于存储资源自身的分散性、异构性及不可靠性使得如何为用户提供安全可靠的服务成为系统研究的核心。虚拟存储[2]作为下一代网络存储的发展趋势,具有非常广阔的内涵与外延,其中主机级的虚拟存储,屏蔽了一切存储设备硬件之上的差异,具有很强的兼容性及可扩展性,相关领域的基本概念及成功应用为 RayStorage 的研究与设计提供了宝贵的经验及参考,同时也为 RayStorage 奠定了理论基础。

RayStorage 不同于现有的集中式的网络存储解决方案,例如 NAS、SAN[3]等,它提供了更为广阔、更加安全的数据复制机制,避免了系统单边崩溃的隐患;也不同于现有的大多数分布式存储系统,例如 RDSS[4,11]、OceanStore[5]等,RayStorage 提供了多层次的数据存储与访问机制,既能够进行上层存储模式级的数据操作,也能够实现安全高效的底层 Block 级的存储控制;虚拟存储相关概念的引入,也使得 RayStorage 系统具有更好的可控性及扩展性。

RayStorage 的目的在于利用分布在网络异构主机之上的不可靠存储资源提供可靠的存储服务。存储资源的不可靠性来源于系统的应用环境,任何提供资源的节点可以随时脱离系统的管理。因此,为了提供安全可靠的存储服务,RayStorage 系统必须充分考虑到以下情况:1)所有存储节点都有可能随时加入或者脱离系统的管理;2)与失效节点之间

收稿日期:2005 - 06 - 16;**修订日期**:2005 - 09 - 13

作者简介:曹楠(1981 -),男,陕西西安人,硕士研究生,主要研究方向:网格计算、计算机网络存储、分布式系统; 康慕宁(1955 -),男,陕西西安人,教授,主要研究方向:软件理论与软件工程.

在本项目开发实践中,我们将这两个作用领域不同的优秀框架进行整合来搭建 J2EE 架构。通过框架的结合,充分发挥了两者的优点,不但使资源得到最大限度的节省和利用,也使得项目开发简洁、结构清晰,并且具备了更好的可扩展性和可维护性。

参考文献:

[1] 夏昕,曹晓钢,唐勇. 深入浅出 HIBERNATE[M]. 北京:电子工业出版社,2005.
[2] 孙卫琴. 精通 HIBERNATE: Java 对象持久化技术详解[M]. 北京:电子工业出版社,2005.
[3] CRAWFORD W, KAPLAN J. J2EE 设计模式[M]. 刘绍华,毛天露,译. 北京:中国电力出版社,2005.
[4] GAMMA E, HELM R, JOHNSON R, *et al*. . 设计模式:可复用面向对象软件的基础[M]. 李英军,马晓星,蔡敏,等译. 北京:机械工业出版社,2000.
[5] TURNER J, BEDELL K. STRUTS Kick Start 中文版[M]. 孙勇,译. 北京:电子工业出版社,2004.
[6] 孙卫琴. 精通 STRUTS:基于 MVC 的 Java Web 设计与开发[M]. 北京:电子工业出版社,2004.