Hibernate入门指南

Hibernate团队

JBoss视觉设计团队

4 1 11最后

2013-03-18

表的内容

前言

表的内容

使用面向对象的软件和关系数据库可以繁琐和费时。 开发成本明显上涨,因一个范式之间不匹配的数据如何在对象表示与关系数据库。 Hibernate是一个对象/关系映射为Java环境的解决方案。 这个 术语对象/关系映射指的技巧之间映射数据对象模型表示 关系数据模型表示。 看到 http://en.wikipedia.org/wiki/Object-relational_mapping 对于一个好的高层讨论。

> **注意**
>
> 你不需要一个强大的背景在SQL中使用Hibernate,但有一个基本的了解 概念可以帮助您了解更全面和快速。冬眠 理解数据建模 原则是特别重要的。 你可能想考虑这些资源作为一个良好的起点:
>
> **数据建模资源**
> http://www.agiledata.org/essays/dataModeling101.html
> http://en.wikipedia.org/wiki/Data_modeling

Hibernate负责从Java类映射到数据库表,并从Java数据类型到SQL数据 类型。 此外,它提供了数据查询和检索设施。它可以显著减少开发 时间否则花手工数据处理在SQL和JDBC。 Hibernate的设计目标是缓解 开发人员从95%的通用数据持续相关编程任务通过消除需要手动, 手工数据处理使用SQL和JDBC。 然而,与许多其他持久性解决方案,Hibernate 没有隐藏的力量,从你和确保SQL投资关系技术和 知识是作为有效的一如既往。

Hibernate可能不是最佳的解决方案为以数据为中心的应用程序只使用存储过程来　实现业务逻辑在数据库中,它是最有用的和面向对象的领域模型和业务 逻辑在基于java的中间层。 然而,Hibernate当然可以帮助你去除或封装 特定于供应商的SQL代码和简化了常见的翻译任务,结果集从一个表格 表示对图形对象。

1。 参与

- 使用Hibernate和报告任何错误或问题你发现。 看到 http://hibernate.org/issuetracker.html 详情。
- 试着动手修复一些错误或实施改进。 再次,看到 http://hibernate.org/issuetracker.html 。
- 参与社区使用邮件列表、论坛、IRC或其他方式标价 http://hibernate.org/community.html 。
- 帮助改善或翻译这个文件。 联系我们在 开发人员邮件列表,如果你有兴趣。
- 传播这个消息。 让你其他的组织知道的好处 Hibernate。

2。 教程代码

引用的项目和代码教程在本指南可在 文件/ hibernate教程邮政编码 。

第1章。 获得Hibernate

表的内容

1.1。 发布包下载

Hibernate团队提供发布包在SourceForge托管文件发布系统,在 ZIP 和 TGZ 格式。 每次发布包包含 jar，文档、源代码和其他美德。

你可以下载版本的冬眠,在你选择的格式,从名单上 http://sourceforge.net/projects/hibernate/files/hibernate4/ 。

- 这个 **lib /要求/** 目录包含所有的罐子Hibernate要求。 所有的 ,该目录中的jar也必须被包括在您的项目的类路径中。
- 这个 **/ lib / jpa /** 目录包含 hibernate entitymanager jar和它所依赖的超越 在 **lib /要求/** 。 这定义了Hibernate支持 JPA 。
- 这个 **lib / envers** 目录包含 hibernate-envers jar和它所依赖的超越 **lib /要求/**
- 这个 **lib /可选** 目录包含jar可选特性的需要 Hibernate。

1.2。 Maven存储库工件

> **注意**
>
> 权威库Hibernate工件是JBoss Maven存储库。 这个团队负责 对JBoss Maven存储库维护一个数量的Wiki页面,包含重要的信息。
>
> **Maven存储库Wiki页面**
>
> http://community.jboss.org/docs/doc - 14900 -一般信息的存储库。
>
> http://community.jboss.org/docs/doc - 15170 ——信息设置JBoss 存储库来做开发工作在JBoss项目本身。
>
> http://community.jboss.org/docs/doc - 15169 ——信息设置访问 存储库使用JBoss项目作为你自己的软件。

Hibernate生成一个数量的工件(所有这些都在 org hibernate groupId):

冬眠构件在groupId org hibernate

hibernate核心

主要的工件,需要构建应用程序使用本机Hibernate api包括 定义元数据在两个注释以及冬眠的 **hbm xml** 格式。

hibernate entitymanager

代表了Hibernate的实施 JPA 规定,在 http://jcp.org/en/jsr/detail?id=317 。

这个工件取决于 hibernate核心

hibernate-envers

一个可选的模块,提供了历史审计变更你的实体。

这个工件取决于双方 hibernate核心 和 hibernate entitymanager 。

hibernate c3p0

提供集成Hibernate和之间 C3P0 连接 池的图书馆。 看到 http://sourceforge.net/projects/c3p0/ 信息 约C3P0 。

这个工件取决于 hibernate核心 ,但通常包括 在一个项目作为一个运行时依赖。 它将在 C3P0 依赖自动。

hibernate-proxool

提供集成Hibernate和之间 proxool 连接 池的图书馆。 看到 http://proxool.sourceforge.net/ 为更多的信息关于 这个图书馆。

这个工件取决于 hibernate核心 ,但通常包括 在一个项目作为一个运行时依赖。 它将在 proxool 依赖自动. .

hibernate ehcache

Privides Hibernate和之间的集成 EHCache ,作为一个 二级缓存。 看到 http://ehcache.sourceforge.net/ 更多信息 EHCache 。

这个工件取决于 hibernate核心 ,但通常包括 在一个项目作为一个运行时依赖。 它将在 EHCache 依赖自动。

hibernate infinispan

提供集成Hibernate和之间 Infinispan ,作为一个 二级缓存。 看到 http://jboss.org/infinispan 为更多的信息 约

Infinispan 。

这个工件取决于 hibernate核心 ,但通常包括 在一个项目作为一个运行时依赖。 它将在 Infinispan 依赖自动。

第二章。 教程使用本机Hibernate api和 hbm xml 映射

表的内容

本教程是位于下下载包的 **基本/** 。

目标

- 使用Hibernate映射文件(hbm xml)提供映射信息
- 使用 原生 Hibernate api

2.1。 Hibernate配置文件

资源文件 **hibernate cfg xml** 定义了Hibernate配置 信息。

这个 连接驱动程序类，连接url，连接用户名 和 连接密码 财产 元素定义的JDBC连接信息。 这些教程利用H2 内存数据库,所以这些属性的值都是特定于运行在内存模式H2。 连接池大小 用于配置的连接数量在Hibernate的吗 内置的连接池。

> **重要**
>
> 内置的Hibernate连接池绝不是用于生产使用。 它缺少几个 特点被发现在生产就绪的连接池。 看到部分讨论 **Hibernate开发者指南** 为进一步的信息。

这个 方言 属性指定特定的SQL变体,Hibernate将 匡威。

> **提示**
>
> 在大多数情况下,Hibernate是能够正确确定哪些方言使用。 这是特别有用 如果你的应用程序目标多个数据库。 这是详细的讨论 **Hibernate开发者指南**

这个 hbm2ddl.auto 属性允许自动生成数据库模式直接进入 数据库。

最后,添加映射文件(s)为持久化类的配置。 这个 **资源** 属性的 映射 元素使Hibernate试图定位,映射 类路径资源,使用java朗类加载器 查找。

2.2。 实体的Java类

实体类本教程 org hibernate教程hbm事件 。

笔记关于实体

- 这类使用JavaBean标准命名约定财产getter和setter方法,以及 私有可见性的字段。 虽然这是推荐的设计,它不是必需的。
- 这个 无参数 构造函数,这也是一个JavaBean公约,是一个 要求所有的持久化类。 Hibernate需要创建对象,用Java反射。 构造函数可以是私有的。 然而,包或公共可见性是需要运行时 代理生成和高效的数据检索没有字节码插装。

2.3。 映射文件

这个 **hbm xml** 映射文件对于本教程是类资源 **org/hibernate/tutorial/hbm/Event.hbm.xml** 当我们看到在 2.1节, “Hibernate配置文件”

Hibernate使用映射元数据来决定如何加载和存储对象的持久化类。 这个 Hibernate映射文件是一个选择与此元数据提供冬眠。

例2.1。 类 映射元素

```
<class name="Event" table="EVENTS">
    ...
</class>
```

功能 类 映射元素

1. 这个 **名称** 属性(联合一起在这里 **包** 属性从 包含 hibernate映射 元素)的类的名字FQN要 定义为一个实体。
2. 这个 **表** 属性名称数据库表包含数据 实体。

实例的 事件 类现在映射到行 事件 表。

例2.2。 ID 映射元素

```
<id name="id" column="EVENT_ID">
    ...
</id>
```

Hibernate使用属性指定的 ID 元素来唯一地标识中的行 表。

> **重要**
>
> 它是不需要的 ID 元素映射到表的实际主键 列(s),但这是正常的惯例。 表映射在Hibernate甚至不需要定义 主键。 然而,它是强烈建议所有模式定义适当的引用 完整性。 因此 ID 和 主键 交替使用 在Hibernate文档。

这个 ID 元素这里标识 事件id 列 的主键 事件 表。 它还指定 ID 财产的 事件 类的属性包含 标识符值。

这个 发电机 元素嵌套在 ID 元素告诉Hibernate 哪些策略是用来生成的主键值对该实体。 这个例子使用一个简单的递增计数。

例2.3。 财产 映射元素

```
<property name="date" type="timestamp" column="EVENT_DATE"/>
<property name="title"/>
```

这两个 财产 剩下的两个元素声明的属性 事件 类: 日期 和 标题 。 这个 日期 属性映射包括 **列** 属性,但 标题 不。 在缺乏 **列** 属性,冬眠 使用属性名作为列名。 这是适合 标题 ,但由于 日期 是一个保留字在大多数数据库,您需要指定一个非保留吗 字为列名。

这个 标题 映射还缺乏一个 **类型** 属性。 类型 声明和使用映射文件既不是Java数据类型也不是SQL数据库类型。 相反,他们是 Hibernate映射类型 。 Hibernate映射类型 转换器,其中翻译Java和SQL数据类型之间。 Hibernate试图确定正确的 自动转换和映射类型如果 **类型** 属性不存在的 映射,通过使用Java反射来确定Java类型的声明的属性和使用 默认的映射类型的Java类型。

在某些情况下,这种自动检测可能不选择默认你期望或需要,因为见过的 日期 财产。 Hibernate不能知道这个属性,它的类型是 java util日期 ,应该映射到SQL 日期 , 时间 ,或 时间戳 数据类型。 完整的日期和时间信息保存通过映射属性为一个 时间戳 转换器, 这标识类的一个实例吗 org.hibernate.type.TimestampType 。

> **提示**
>
> Hibernate映射类型确定使用反射当映射文件进行处理。 这 过程增加了开销方面的时间和资源。 如果启动性能是很重要的,可以考虑 明确定义类型使用。

2.4。 示例代码

这个 org.hibernate.tutorial.hbm.NativeApiIllustrationTest 类演示了使用 Hibernate 本机API 。

> **注意**
>
> 这些教程中的例子作为JUnit测试,为便于使用。 的一个好处 方法 设置 和 拆卸 大致说明 如何 org hibernate sessionfactory 是在创建的启动,一个吗 应用程序和关闭 应用程序生命周期的结束。

例2.4。 获得 org hibernate sessionfactory

```
protected void setUp() throws Exception {
    // A SessionFactory is set up once for an application
    sessionFactory = new Configuration()
            .configure() // configures settings from hibernate.cfg.xml
            .buildSessionFactory();
}
```

程序2.1。 教程工作流

1. **加载的配置。**

   这个 org hibernate cfg配置 类是第一件要注意的事。 在这个 教程中,所有的配置细节都位于 **hibernate cfg xml** 文件 讨论 2.1节,“Hibernate配置文件” 。

2. **这个 org hibernate sessionfactory 被创建。**

   这个 org hibernate cfg配置 然后创建 org hibernate sessionfactory 这是一个线程安全的对象,是吗 实例化 一次为整个应用程序。

3. **SessionFactory 创建 会话 实例。**

   这个 org hibernate sessionfactory 作为一个工厂 org hibernate会话 实例中可以看出 testBasicUsage 法。

4. **会话 年代执行工作。**

   一个 org hibernate会话 应该被认为是一个必然的“单位的 工作”。

例2.5。 拯救实体

```
Session session = sessionFactory.openSession();
session.beginTransaction();
session.save( new Event( "Our very first event!", new Date() ) );
session.save( new Event( "A follow up event", new Date() ) );
session.getTransaction().commit();
session.close();
```

testBasicUsage 首先创建一些新的 事件 对象和 手他们冬眠管理、使用 保存 法。 Hibernate现在 负责执行 **插入** 在 数据库上。

例2.6。 获得一个实体列表

```
session = sessionFactory.openSession();
session.beginTransaction();
```

```
List result = session.createQuery( "from Event" ).list();
for ( Event event : (List<Event>) result ) {
    System.out.println( "Event (" + event.getDate() + ") : " + event.getTitle() );
}
session.getTransaction().commit();
session.close();
```

testBasicUsage 演示了使用 Hibernate查询语言 (HQL) 加载所有现有 事件 对象从数据库和生成 适当 选择 SQL,将其发送到数据库和填充 事件 对象与结果集数据。

2.5。 走的更远!

实践练习

- 重新配置示例连接到您自己的持久的关系数据库。
- 帮助 **开发者指南** ,添加一个协会 这个 事件 实体模型的一个消息线程。

第三章。 教程使用本机Hibernate api和注释映射

表的内容

本教程是位于下下载包的 **注释** 。

目标

- 使用注释提供映射信息
- 使用 原生 Hibernate api

3.1。 Hibernate配置文件

内容都是相同的 2.1节, “Hibernate配置文件” ,其中一个重要的 差异。 这个 映射 元素在快结束的时候,命名的实体类使用 这个 **类** 属性。

3.2。 带注释的实体的Java类

实体类在本教程 org hibernate教程注释事件 哪一个 遵循JavaBean规范。 事实上,类本身是相同的 2.2节,“实体的Java类” ,除了注释是用来提供元数据, 而不是一个单独的 **hbm xml** 文件。

例3.1。 识别类作为一个实体

```
@Entity
@Table( name = "EVENTS" )
public class Event {
    ...
}
```

这个 @javax.persistence.Entity 注释是用来标记一个类一个实体。 它的功能一样的 类 映射元素讨论 2.3节,“映射文件” 。 此外, @javax.persistence.Table 注释明确指定了表 的名字。 没有这个规格,默认的表名是 事件)。

例3.2。 识别标识符属性

```
@Id
@GeneratedValue(generator="increment")
@GenericGenerator(name="increment", strategy = "increment")
public Long getId() {
    return id;
}
```

@javax.persistence.Id 标志着财产定义 实体的标识符。 @javax.persistence.GeneratedValue 和 @org.hibernate.annotations.GenericGenerator 协同工作 表明Hibernate应该使用Hibernate的 增量 代 这个实体的标识符的战略价值。

例3.3。 识别基本属性

```
public String getTitle() {
    return title;
}

@Temporal(TemporalType.TIMESTAMP)
@Column(name = "EVENT_DATE")
public Date getDate() {
    return date;
}
```

在 2.3节,“映射文件” ,日期 房地产需求 特殊处理以适应其特殊的命名和它的SQL类型。

3.3。 示例代码

org.hibernate.tutorial.annotations.AnnotationsIllustrationTest 基本上 一样
org.hibernate.tutorial.hbm.NativeApiIllustrationTest 讨论 2.4节，“示例代码” 。

3.4。 走的更远!

实践练习

- 添加一个协会的 事件 实体模型的一个消息线程。 使用 **开发者指南** 作为一个指南。
- 添加一个回调函数,当一个接收通知 事件 创建、更新或 删除。 尝试相同的与一个事件侦听器。 使用 **开发商 指南** 作为一个指南。

第四章。 教程使用 Java持久化API(JPA)

表的内容

本教程是位于下下载包的 **EntityManager** 。

目标

- 使用注释提供映射信息。
- 使用 JPA 。

4.1。 persistence . xml

前面的教程使用hibernate具体 **hibernate cfg xml** 配置文件。 JPA ,然而,定义了不同的引导过程,利用自己的配置 文件命名 **persistence . xml** 。 这个自举过程的定义 JPA 规范。 在 JAVA ™SE环境持久性 提供者(在这种情况下 Hibernate)是需要定位所有 JPA 配置文件 通过类路径的查找 **meta - inf / persistence . xml** 资源名称。

例4.1。 persistence . xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persiste
        version="2.0">
    <persistence-unit name="org.hibernate.tutorial.jpa">
        ...
    </persistence-unit>
</persistence>
```

persistence . xml 文件应该提供一个独特的名称为每个 持久性 单位 。 应用程序使用该名称来引用配置当获得一个 javax.persistence.EntityManagerFactory 参考。

中定义的设置 属性 元素进行 2.1节，“Hibernate配置文件” 。 这里的 javax.persistence 前缀 品种是用于当可能的。 注意,其余hibernate具体配置设置名称 现在加上前缀 Hibernate。 。

此外, 类 元素功能相同 3.1节，“Hibernate配置文件” 。

4.2。 带注释的实体的Java类

实体是相同 3.2节，“带注释的实体的Java类”

4.3。 示例代码

前面的教程使用Hibernate api。 本教程使用了 JPA api。

例4.2。 获得 javax.persistence.EntityManagerFactory

```
protected void setUp() throws Exception {
    entityManagerFactory = Persistence.createEntityManagerFactory( "org.hibernate.tutorial.jpa" );
}
```

请再次注意,持久性单元名称是 org hibernate教程jpa 匹配的 例4.1，“ **persistence . xml** ”

例4.3。 储蓄(坚持)实体

```
EntityManager entityManager = entityManagerFactory.createEntityManager();
entityManager.getTransaction().begin();
entityManager.persist( new Event( "Our very first event!", new Date() ) );
entityManager.persist( new Event( "A follow up event", new Date() ) );
entityManager.getTransaction().commit();
entityManager.close();
```

代码是相似的 例2.5，“拯救实体” 。 一个 javax.persistence.EntityManager 接口是用来代替一个 org hibernate会话 接口。 JPA 调用这个 操作 持续 而不是 保存 。

例4.4。 获得一个实体列表

```
entityManager = entityManagerFactory.createEntityManager();
entityManager.getTransaction().begin();
List<Event> result = entityManager.createQuery( "from Event", Event.class ).getResultList();
for ( Event event : result ) {
    System.out.println( "Event (" + event.getDate() + ") : " + event.getTitle() );
}
entityManager.getTransaction().commit();
entityManager.close();
```

再次,代码十分相似 例2.6,“获取实体列表” 。

4.4。 走的更远!

实践练习

- 开发一个EJB会话bean来调查的含义使用容器管理的 持久性上下文(javax.persistence.EntityManager)。 试试这两个 无状态和有状态的用例。

第五章。 教程使用Envers

表的内容

本教程是位于下下载包的 **envers** 。

目标

- Envers配置。
- 使用Envers api来查看和分析历史数据。

5.1。 persistence . xml

专 家 进 阶 入 门 全文翻译

翻译级别

5.2。 带注释的实体的Java类

再次,实体基本上是一样的 4.2节, “带注释的实体的Java类” 。 主要 区别是附加的 @org.hibernate.envers.Audited 注释, 告诉Envers自动跟踪更改这个实体。

5.3。 示例代码

再次,本教程利用 JPA api。 然而,代码也要改变一个 的实体,然后使用Envers API来拉回最初 修订 以及更新 修订。 修订指的是一个版本的一个实体。

例5.1。 使用 org.hibernate.envers.AuditReader

```
public void testBasicUsage() {
    ...
    AuditReader reader = AuditReaderFactory.get( entityManager );
    Event firstRevision = reader.find( Event.class, 2L, 1 );
    ...
    Event secondRevision = reader.find( Event.class, 2L, 2 );
    ...
}
```

程序5.1。 描述的例子

1. 一个 org.hibernate.envers.AuditReader 获得 org.hibernate.envers.AuditReaderFactory 这包装了 javax.persistence.EntityManager 。
2. 接下来, 找到 方法检索特定修订的实体。 第一次调用 读取 找到修订编号1的事件id为2 。 第二个调用读取 找到 修订号2的事件id为2 。

5.4。 走的更远!

实践练习

- 提供一个自定义修改实体另外捕捉谁做了修改。
- 写一个查询来检索历史数据只有满足某些标准。 使用 **Envers用户指南** 如何构建Envers查询。
- 实验与审计实体具有多对一、多对多的关系以及集合。 尝试检索历史版本(修正)这样的实体和导航对象树。