Hibernate

作成者:周东

作成日:2019.08.30

Neusoft

目录

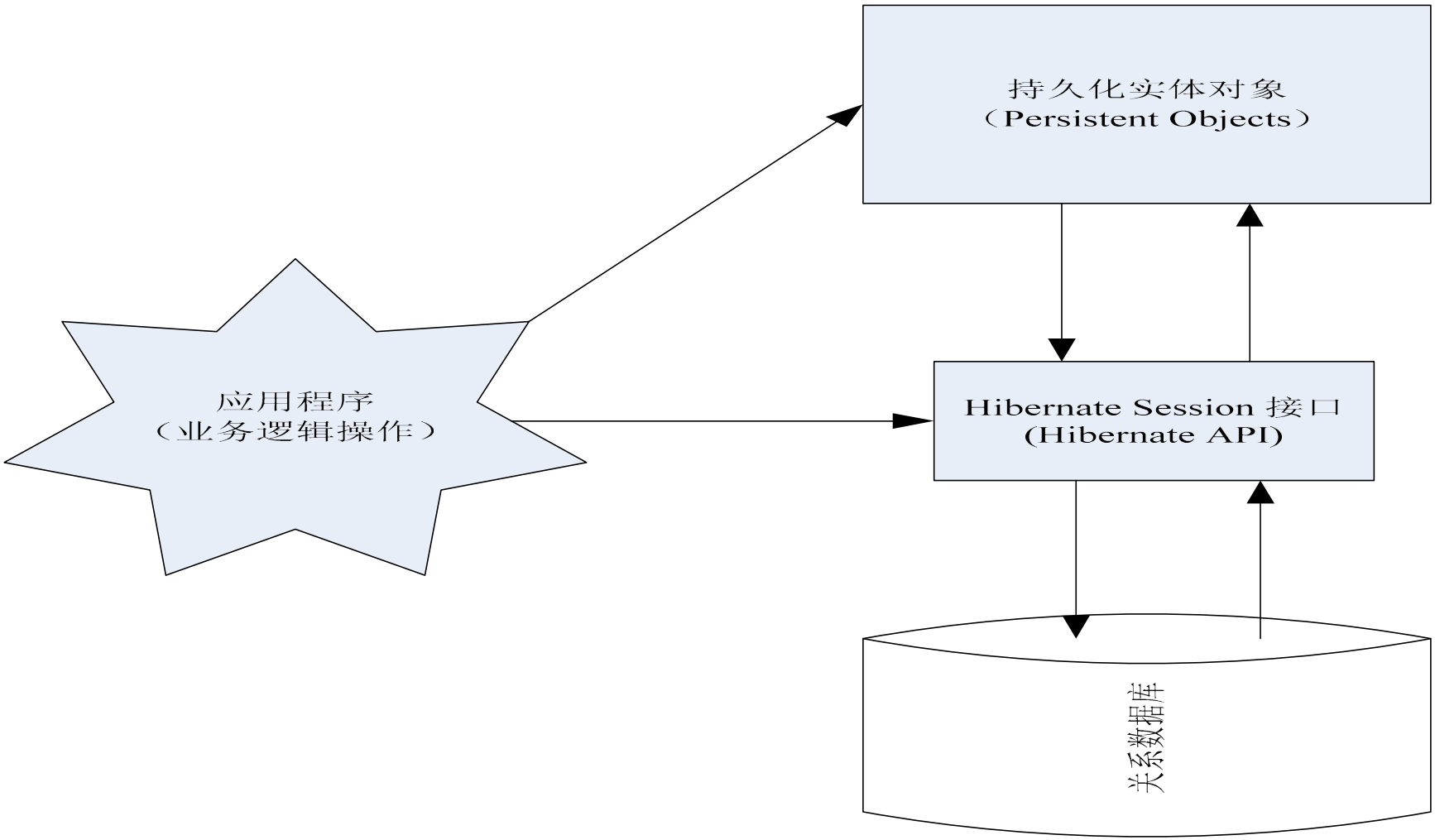
Neusoft

简介

Hibernate能帮助我们利用面向对象的思想，开发基于关系型数据库的应用程序。

第一：将对象数据保存到数据库
第二：将数据库数据读入对象中

简介


持久化实体对象
（Persistent Objects）
应用程序
（业务逻辑操作）
Hibernate Session 接口
(Hibernate API)
关系数据库

配置

创建Hibernate配置文件 - hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration
    PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
<property
name="hibernate.connection.url">jdbc:mysql://127.0.0.1/hibernate</property>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">mysql</property>
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

</session-factory>
</hibernate-configuration>
```

配置

创建持久化类 User.java

```
public class User {
        private String id;
        private String name;
        private String password;
        private Date createTime;
        private Date expireTime;
        ……getters/setters
}
```

配置

创建类的映射文件 - User.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.bjsxt.hibernate.User">
        <id name="id">
            <generator class="uuid"/>
        </id>
        <property name="name"/>
        <property name="password"/>
        <property name="createTime"/>
        <property name="expireTime"/>
    </class>
</hibernate-mapping>
```

配置

将类的映射文件加入Hibernate

为了让Hibernate能够处理User对象的持久化，需要将它的映射信息加入到Hibernate中
加入的方法很简单，在Hibernate配置文件中加入：
<mapping resource="com/bjsxt/hibernate/User.hbm.xml"/>
即可
resource属性指定了映射文件的位置和名称

实现

将对象保存到数据库

```
public static void main(String[] args) throws Exception{
        Configuration cfg = new Configuration().configure();
        SessionFactory factory = cfg.buildSessionFactory();

        Session session = factory.openSession();
        session.beginTransaction();

        User user = new User();
        user.setName("管理员");
        user.setPassword("admin");
        user.setCreateTime(new Date());
        user.setExpireTime(new Date());

        session.save(user);
        session.getTransaction().commit();
        if(session.isOpen()){
                session.close();
        }
}
```

Configuration

概述：Configuration 类负责管理Hibernate 的配置信息。它包括如下内容：
Hibernate运行的底层信息：数据库的URL、用户名、密码、JDBC驱动类，数据库Dialect，数据库连接池等。
Hibernate映射文件（*.hbm.xml）。

Hibernate配置的两种方法：
属性文件（hibernate.properties）。
调用代码：Configuration cfg = new Configuration();
Xml文件（hibernate.cfg.xml）。
调用代码：Configuration cfg = new Configuration().configure();

SessionFactory

概述:应用程序从SessionFactory(会话工厂)里获得Session(会话)实例。它在多个应用线程间进行共享。通常情况下,整个应用只有唯一的一个会话工厂——例如在应用初始化时被创建。然而,如果你使用Hibernate访问多个数据库,你需要对每一个数据库使用一个会话工厂。

会话工厂缓存了生成的SQL语句和Hibernate在运行时使用的映射元数据。

调用代码:

```
SessionFactory sessionFactory = cfg.buildSessionFactory();
```

说明:SessionFactory由Configuration对象创建,所以每个Hibernate配置文件,实际上是对SessionFactory的配置。

Session(会话)

概述:
Session不是线程安全的, 它代表与数据库之间的一次操作, 它的概念介于Connection和Transaction之间。
Session也称为持久化管理器, 因为它是与持久化有关的操作接口。
Session通过SessionFactory打开, 在所有的工作完成后, 需要关闭。
它与Web层的HttpSession没有任何关系。

调用代码
```
Session session = sessionFactory.openSession();
```

Query

概述：
Query（查询）接口允许你在数据库上执行查询并控制查询如何执行。查询语句使用HQL或者本地数据库的SQL方言编写。

调用代码：
Query query = session.createQuery(“from User”);

Query

```
Configuration cfg = new Configuration().configure();
SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession();
session.beginTransaction();

Query query = session.createQuery("from User");
List users = query.list();
for (Iterator iter = users.iterator(); iter.hasNext();) {
        User user = (User) iter.next();
        System.out.println("user name = "+user.getName());
}

session.getTransaction().commit();
if(session.isOpen()){
        session.close();
}
```

标准化查询对象

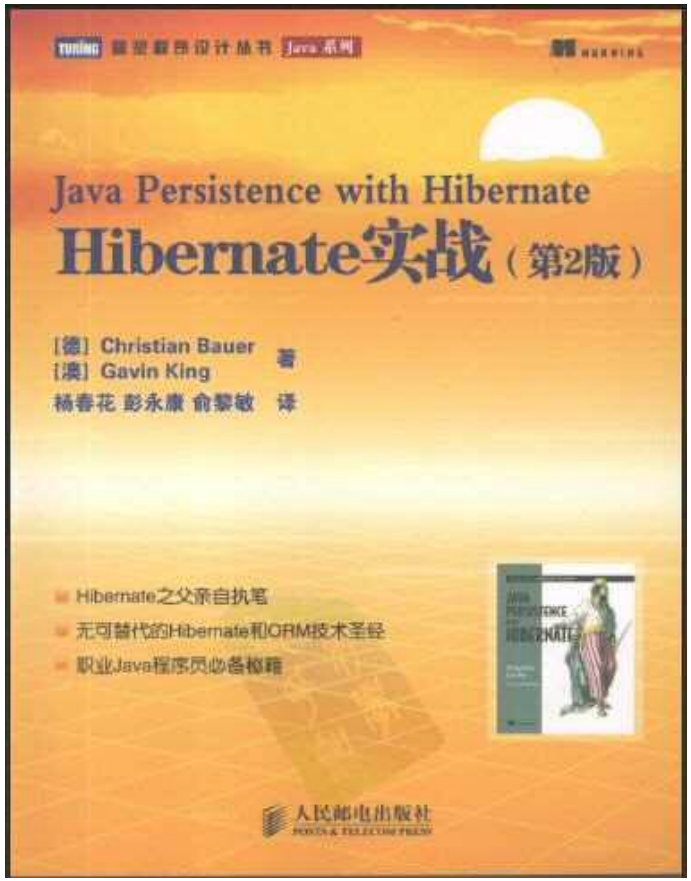
标准化对象查询(Criteria Query):以对象的方式进行查询，将查询语句封装为对象操作。优点:可读性好，符合Java 程序员的编码习惯。缺点:不够成熟，不支持投影(projection)或统计函数(aggregation)。

简单例子:查询用户名以“J”开头的所有用户。

```
Criteria criteria = session.createCriteria(User.class);
criteria.add(Expression.like("name","J%"));
List users = criteria.list();
```

参考资料

Hibernate实战(第2版)




Neusoft
Beyond Technology