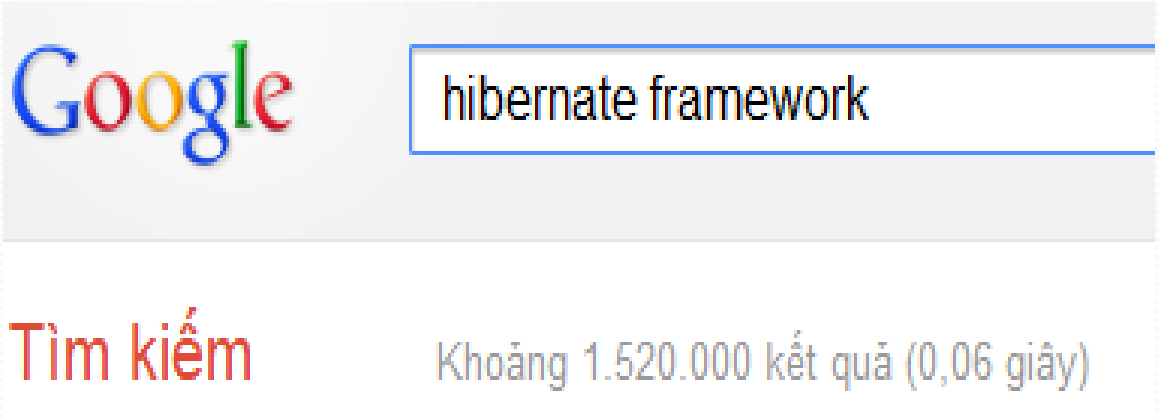
Hibernate


Google
hibernate framework
Tìm kiếm
Khoảng 1.520.000 kết quả (0,06 giây)

Hibernate

- 0.Lịch sử của Hibernate
- 1.Giới thiệu về Hibernate.
- 2.Kiến trúc Hibernate.
- 3.Một số Interface chính
- 4.Cài đặt và cấu hình
- 5.HQL

0.Lịch sử của Hibernate

- http://www.hibernate.org/about/history.html
- Hibernate được bắt đầu từ năm 2001 bởi Gavin King,như là 1 sự thay thế cho việc sử dụng EJB2.

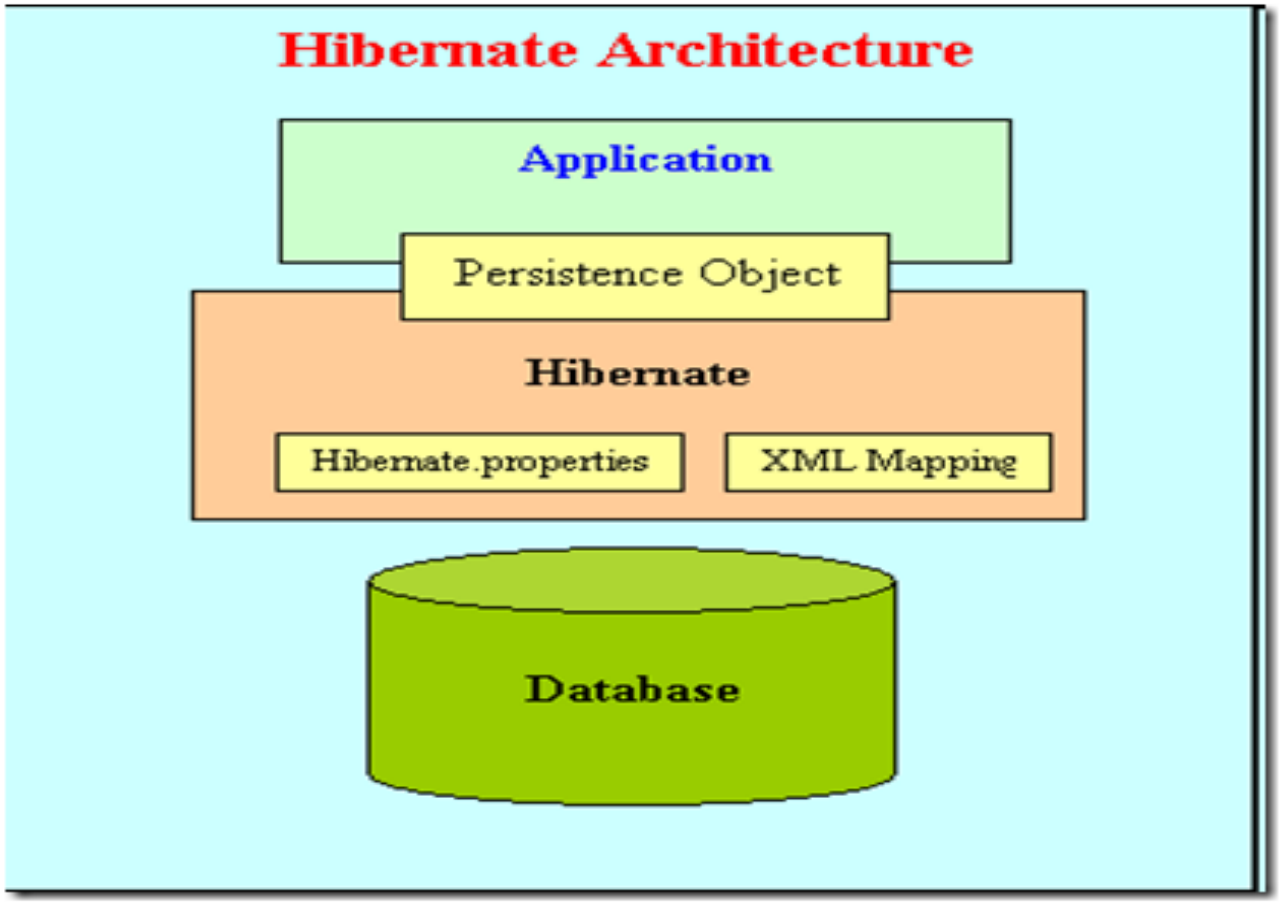
- Mục đích của Hibernate là cung cấp khả năng *persisten* đơn giản hơn EJB2,giảm bớt đi sự phức tạp và bổ sung thêm các tính năng còn thiếu.

- Đầu năm 2003,Hibernate2 bắt đầu được phát triển,cung cấp nhiều cải tiến đáng kể so với phiên bản đầu tiên và trở thành chuẩn *persistent* trong Java.

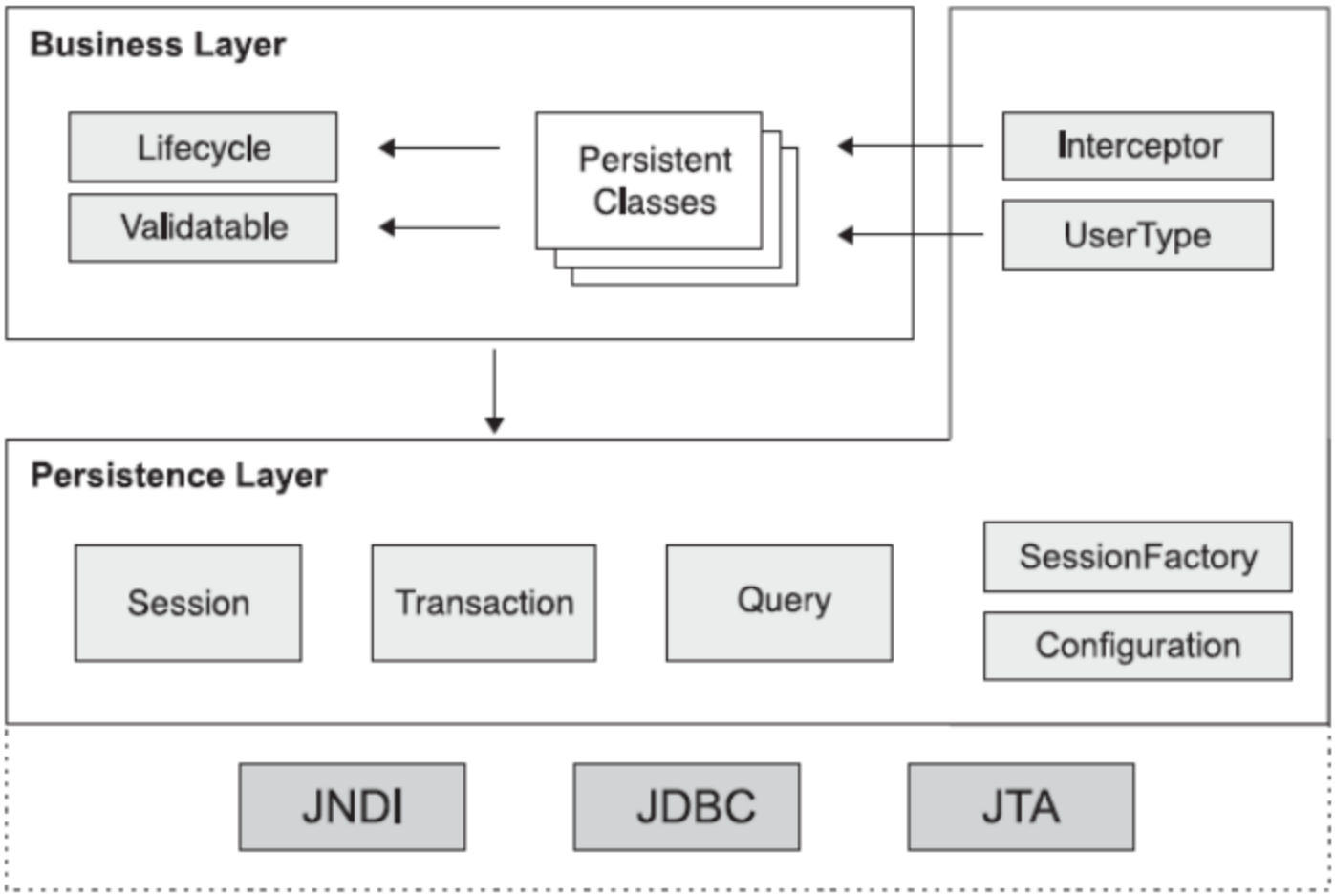
1.Giới thiệu về Hibernate

- Là một framework cho *persistence* layer.
- Là một dịch vụ lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh.
- Hỗ phát triển các class dùng để lưu trữ dữ liệu theo cách thức hướng đối tượng.
- Cho phép thực hiện các câu truy vấn dữ liệu bằng cách sử dụng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc là ngôn ngữ SQL nguyên thủy cũng như là sử dụng các API.

2.Kiến trúc Hibernate



3.Một số Interface chính



3.Một số Interface chính(cont…)

Các interface Hibernate được thể hiện trên hình trên có thể được phân thành các loại chính sau:

- Các interface được các ứng dụng gọi để thực hiện các thao tác cơ bản CRUD và các lệnh truy vấn. Chúng bao gồm Session, Transaction và Query.
- Các interface được code cơ sở ứng dụng gọi để cấu hình Hibernate, trong đó quan trọng nhất là interface Configuration.
- Các interface Callback – là các interface cho phép ứng dụng phản hồi lại các sự kiện (event) xảy ra bên trong Hibernate, gồm các interface như Interceptor, Lifecycle và Validatable.
- Các interface giúp mở rộng các chức năng ánh xạ của Hibernate, gồm các interface như UserType, CompositeUserType, và IdentifierGenerator. Các interface này được hiện thực bằng code cơ sở ứng dụng.

Session Interface

Session interface

Một thực thể Session ít tốn tài nguyên (có tính lightweight), dễ dàng tạo và hủy. Đặc điểm này khá quan trọng vì trong một ứng dụng cần tạo và xóa các session mọi lúc, có thể là trên mỗi request. Các Hibernate session không có tính an toàn về thread (not threadsafe) nên được thiết kế để chỉ được một thread sử dụng một thời điểm.

Định nghĩa của Hibernate về một session là một cái giữa connection và transaction. Có thể hình dung về session như một bộ đệm hoặc một tập hợp các đối tượng đã được nạp vào có liên quan đến một đơn vị công việc. Hibernate có thể dò tìm ra những thay đổi các đối tượng trong đơn vị công việc này. Nó cũng là interface cho các hàm liên quan đến persistent, ví dụ như là việc lưu và rút trích đối tượng.

SessionFactory Interface

SessionFactory interface

Ứng dụng tạo được một thực thể Session từ một SessionFactory. SessionFactory không có tính lighweight. Nó được dùng chung cho nhiều thread ứng dụng. Điển hình là có một SessionFactory cho toàn ứng dụng, được tạo trong phần khởi tạo ứng dụng. Tuy nhiên, nếu ứng dụng cần truy xuất nhiều database bằng cách sử dụng Hibernater thì cần một SessionFactory cho mỗi database.

SessionFactory lưu lại các câu lệnh SQL đã tạo và các siêu dữ liệu (metadata) ánh xạ khác mà Hibernate sử dụng ở thời điểm chạy. Nó cũng lưu các dữ liệu đã từng được đọc (được cache lại) trong một đơn vị công việc và có thể được sử dụng trong một đơn vị công việc trong tương lai (chỉ nếu các ánh xạ class và collection xác định rằng cache cấp hai này là cần thiết).

Configuration and Transaction Interface

Configuration interface

Đối tượng Configuration được sử dụng để định cấu hình và boostrap Hibernate. Ứng dụng sử dụng một thực thể Configuration để xác định vị trí của các tài liệu ánh xạ (mapping document) và các thuộc tính xác định Hibernate và sau đó tạo ra SessionFactory.

Transaction interface

Interface Transaction là một API tùy chọn. Nghĩa là các ứng dụng Hibernate có thể không sử dụng interface này, thay vào đó nó sẽ quản lý các transaction bằng code cơ sở của riêng nó. Một Transaction interface tách biệt (trừu tượng hóa) code ứng dụng khỏi sự hiện thực transaction bên dưới (có thể là transaction JDBC, UserTransaction JTA hoặc transaction CORBA) cho phép ứng dụng điều khiển các biên transaction (transaction boundary – cho phép xác định khi nào một transaction bắt đầu và kết thúc) thông qua một API nhất quán.

Điều này giúp cho các ứng dụng Hibernate có tính khả chuyển trên các môi trường thực thi và các container khác nhau.

Query and Crietia Interface

Query và Criteria interface

Interface Query cho phép thực hiện các truy vấn đến các cơ sở dữ liệu và điều khiển việc các truy vấn này được thực thi như thế nào. Các truy vấn được viết bằng HQL (Hibernate Query Language) hoặc bằng ngôn ngữ SQL nguyên thuỷ. Một thực thể Query được dùng để bind các thông số truy vấn, giới hạn của số kết quả trả về bởi truy vấn và nhiệm vụ cuối cùng là thực thi truy vấn.

Interface Criteria cũng tương tự như vậy, nó cho phép tạo và thực thi các truy vấn chuẩn hướng đối tượng.

Một thực thể Query có tính lightweight và không thể sử dụng ngoài Session đã tạo ra nó.

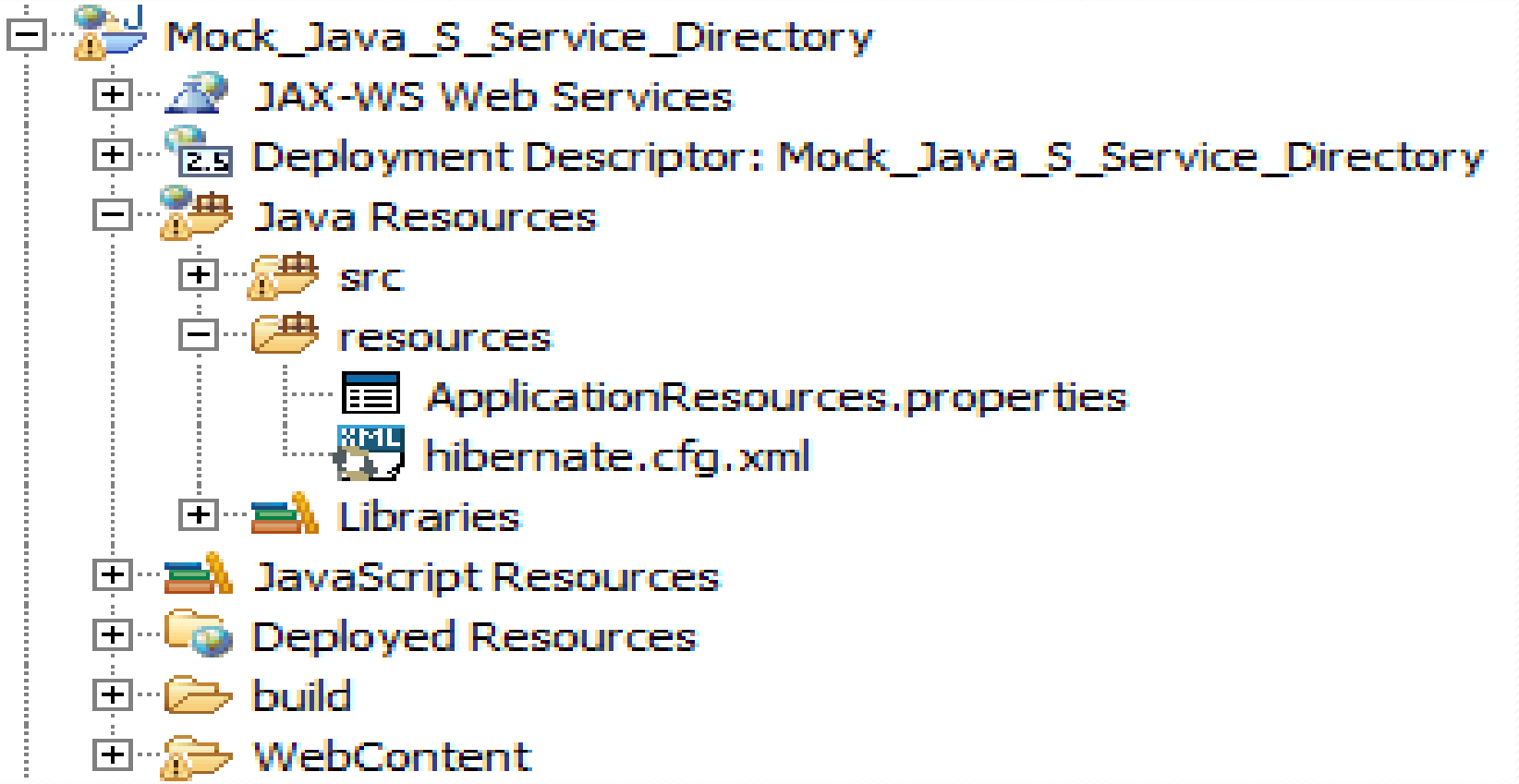
4.1Cài đặt

- 1.Add thư viện vào thư mục lib của project
- 2.Download jdbc Driver cho database

4.2 Cấu hình

- Tạo file cấu hình hibernate.cfg.xml.
- Xây dựng các file định nghĩa ánh xạ để cung cấp cho Hibetnate các thông tin về các lớp persistent(file này có đuôi là .hbm.xml và phải hợp lệ với DTD (Document Type Definition) mà Hibernate đã đưa ra)
- Có một java bean cho mỗi table trong database. Các java bean này sẽ có các getters / setters

4.2.1 File hibernate.cfg.xml.



4.2.2 Chi tiết của file hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
        <property name="hibernate.connection.url">jdbc:sqlserver://localhost:1433;databaseName=Service_Directory</property>
        <property name="hibernate.connection.username">sa</property>
        <property name="hibernate.connection.password">123456</property>
        <property name="connection.pool_size">1</property>
        <property name="hibernate.show_sql">true</property>
        <property name="format_sql">true</property>
        <property name="hibernate.current_session_context_class">thread</property>
        <property name="hibernate.query.factory_class">org.hibernate.hql.ast.ASTQueryTranslatorFactory</property>
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>
        <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>
        <property name="hbm2ddl.auto">update</property>
        <mapping class="mock.appcode.common.utility.User" />
    </session-factory>
</hibernate-configuration>
```

4.2.3 File Bean Account.java

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import sample.util.HibernateUtil;

public class Account implements java.io.Serializable {
    private String username;
    private String password;
    private String lastname;
    private Integer roles;

    public Account() {...}

    public Account(String username) {...}

    public Account(String username, String password, String lastname, Integer roles) {...}

    public String getUsername() {...}

    public void setUsername(String username) {...}

    public String getPassword() {...}

    public void setPassword(String password) {...}

    public String getLastname() {...}

    public void setLastname(String lastname) {...}

    public Integer getRoles() {...}

    public void setRoles(Integer roles) {...}
```

4.2.4 File Account.hbm.xml

```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">
<!-- Generated Mar 13, 2012 10:15:47 AM by Hibernate Tools 3.2.1.GA -->
<hibernate-mapping>
    <class name="sample.entity.Account" table="account" schema="dbo" catalog="hibernate_demo">
        <id name="username" type="string">
            <column name="username" length="50" />
            <generator class="assigned" />
        </id>
        <property name="password" type="string">
            <column name="password" length="50" />
        </property>
        <property name="lastname" type="string">
            <column name="lastname" length="50" />
        </property>
        <property name="roles" type="java.lang.Integer">
            <column name="roles" />
        </property>
    </class>
</hibernate-mapping>
```

4.3 Cấu hình sử dụng Annotations

```
@Entity
@Table(name="User2")
public class User implements Serializable{

    private String userid;
    private String account;
    private String email;
    private String role;
    private String password;

    public User(String userid, String account, String email, String role,

    public User() {

    @Id
    @GeneratedValue
    @Column(name="UserID")
    public String getUserid() {
        return userid;
    }

    public void setUserid(String userid) {

    @Column(name="Account")
    public String getAccount() {
        return account;
    }
```

4.4 File HibernateUtil.java

```java
package mock.appcode.common.utility;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            return new AnnotationConfiguration().configure()
                    .buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Khoi tao SessionFactory bi loi." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

4.5 File UserDAO.java

```
public class UserDAO extends HibernateUtil {

    private Session session = null;
    User user = null;

    public UserDAO() {

    public boolean authenticate(String account, String password) {
        try {
            Session session = HibernateUtil.getSessionFactory()
                    .getCurrentSession();
            session.beginTransaction();

            Query query = (Query) session
                    .createQuery("from User u where u.account='" + account
                            + "' and u.password='" + password + "'");
            Object result = null;
            result = query.uniqueResult();
            session.flush();
            session.getTransaction().commit();
            if (result != null) {
                return true;
            }
        } catch (Exception e) {
            if (session.getTransaction().isActive()) {
                session.getTransaction().rollback();
            }
            e.printStackTrace();
            System.out.println("++++++++++co loi+++++++++++++++");
        }
        return false;
    }
```

4.6 File UserAction.java

```java
public class UserAction extends ActionSupport implements ModelDriven{

    private String account = null;
    private String password = null;

    public String doLogin() throws Exception {
        UserDAO userDAO = new UserDAO();
        boolean result = userDAO.authenticate(getAccount(), getPassword());
        if (result) {
            return SUCCESS;
        }

        return "errorlogin";

    }
```

4.7 Kết quả trong màn hình Console của Eclipse

```
Hibernate:
    select
        user0_.UserID as UserID0_,
        user0_.Account as Account0_,
        user0_.Email as Email0_,
        user0_.Password as Password0_,
        user0_.Role as Role0_
    from
        User2 user0_
    where
        user0_.Account='luannt004'
        and user0_.Password='123'
```

4.8 Insert

- Để insert một đối tượng vào CSDL, ta sử dụng phương thức save().

```
User user = new User();
user.setId(120);
user.setName(“Name”);
user.setPassword(“Password”);
Session session = sessionFactory.openSession();
Transaction tx = Session.beginTransaction();
session.save(user);
user.setName(“NewName”);
tx.commit();
session.close();
```

4.9 Delete

```
User user = new User();
User.setId(100);
Session session =
sessionFactory.openSession();
Transaction tx = session.beginTransaction();
session.delete(user);
tx.commit();
session.close();
```

5.HQL(Hibernate Query Language)

trong Hibernate ta có thể dùng đến 3 cách để biểu diễn câu truy vấn.

- Cách thứ nhất là sử dụng HQL:

Ví dụ:

```
session.
createQuery("from Category c where c.name like 'Laptop%'");
```

- Cách thứ hai là sử dụng hàm API Criteria cho truy vấn, gồm truy vấn theo chuẩn (query by criteria, QBC) và truy vấn theo ví dụ (query by example, QBE).

Ví dụ:

```
session.
createCriteria((Category.class).add(Expression.like("name",
                                                    "Laptop%") );
```

- Cách thứ ba là sử dụng câu SQL trực tiếp với sự ánh xạ tự động tập kết quả thành các đối tượng

Ví dụ:

```
session.createSQLQuery("select {c.*} from CATEGORY {c} where NAME
like 'Laptop%'","c",Category.class);
```

5.1 Interface Query

Giao tiếp Query cho phép điều khiển các đối tượng được trả về như giới hạn số đối tượng được trả về, thiết lập timeout.

```
Query q = session.createQuery("from Event");
List results = q.list();
```

Thiết lập giới hạn số đối tượng Event được trả về

```
Query q = session.createQuery("from Event");
q.setMaxResults(15);
List results = q.list();
```

5.2 Sử dụng hàm

Hibernate hỗ trợ 5 hàm sau: avg, min, max và sum.

avg(expression) - tính giá trị trung bình của biểu thức.

count(expression) - đếm số dòng được trả về bởi biểu thức.

max(expression) - trả về giá trị lớn nhất trong biểu thức.

min(expression) - trả về giá trị nhỏ nhất trong biểu thức.

sum(expression) - trả về tổng các giá trị trong biểu thức.

5.3 Criteria

- Criteria API cung cấp phương pháp khác để truy vấn các đối tượng persistent. Nó cho phép xâu dựng các câu truy vấn động. Criteria được sử dụng khi có nhiều tham số cho chức năng search.
- Ví dụ:
- Bạn có màn hình search nâng cao, cho phép người dùng chọn nhiều trường để tìm kiếm.
-
- Criteria criteria = session.createCriteria(Event.class);
- criteria.add(Restrictions.between("duration", new Integer(60), new Integer(90));
- criteria.add(Restrictions.like("name", "Presen%"));
- criteria.addOrder(Order.asc("name"));
- List results = criteria.list();


I love you!
coolmyspacecomments.com