



# Lập trình Backend Spring Hibernate

Ths. Vũ Duy Khương

1

**Giới thiệu về Hibernate**

2

**Kiến trúc của Hibernate**

3

**Maven Project , setting Project**

4

**Entity, JavaBeans và POJOs**

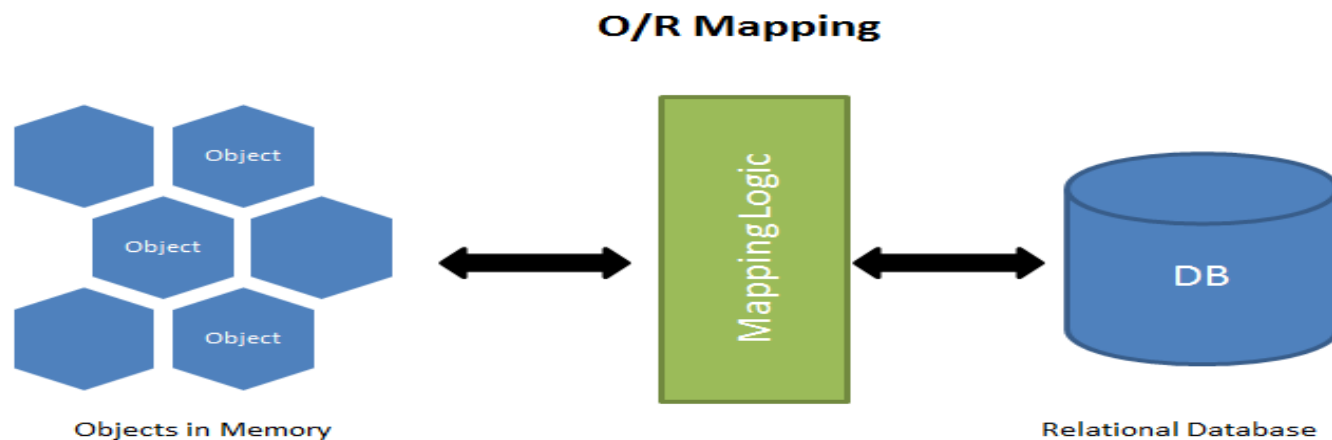
5

**Các hàm CRUD trong Hibernate**

# Giới thiệu về Hibernate

## - Khái niệm ORM (Object-Relational Mapping):

là một kỹ thuật lập trình để chuyển đổi dữ liệu giữa các cơ sở dữ liệu quan hệ và các ngôn ngữ lập trình hướng đối tượng như Java, C# ...



# Giới thiệu về Hibernate

## - Ưu điểm của ORM :

| No. | Ưu điểm   |
|-----|---|
| 1   | Cho phép đối tượng truy cập mã nghiệp vụ thay vì bảng DB.                     |
| 2   | Ẩn các chi tiết của các truy vấn SQL từ logic OO.                             |
| 3   | Dựa trên JDBC 'under the hood'  |
| 4   | Không cần phải đối phó với việc thực hiện cơ sở dữ liệu.                      |
| 5   | Các thực thể dựa trên các khái niệm nghiệp vụ thay vì cấu trúc cơ sở dữ liệu. |
| 6   | Quản lý Transaction và tạo ra key tự động.                                    |
| 7   | Phát triển ứng dụng nhanh chóng.  |

# Giới thiệu về Hibernate

## - Các thực thể của ORM :

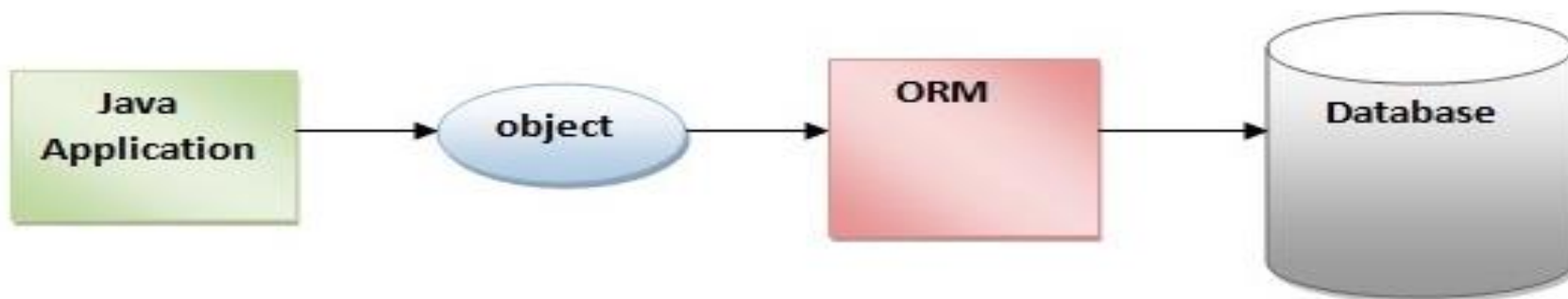
| No. | Giải pháp |
|-----|-----------|
|-----|-----------|

- |   |   |
|---|---|
| 1 | Một API để thực hiện các thao tác CRUD cơ bản trên các đối tượng của các lớp persistent.  |
| 2 | Một ngôn ngữ hoặc API để chỉ định các truy vấn đề cập đến các lớp và thuộc tính của các lớp.  |
| 3 | Một phương tiện có thể cấu hình để chỉ định siêu dữ liệu ánh xạ.  |
| 4 | Một kỹ thuật tương tác với các đối tượng giao dịch để thực hiện dirty checking, lazy loading, join fetching và các chức năng tối ưu hóa khác. |

# Giới thiệu về Hibernate

## - Giới thiệu Hibernate:

Hibernate framework là một giải pháp ORM mã nguồn mở, gọn nhẹ. Hibernate giúp đơn giản hoá sự phát triển của ứng dụng java để tương tác với cơ sở dữ liệu.



# Giới thiệu về Hibernate

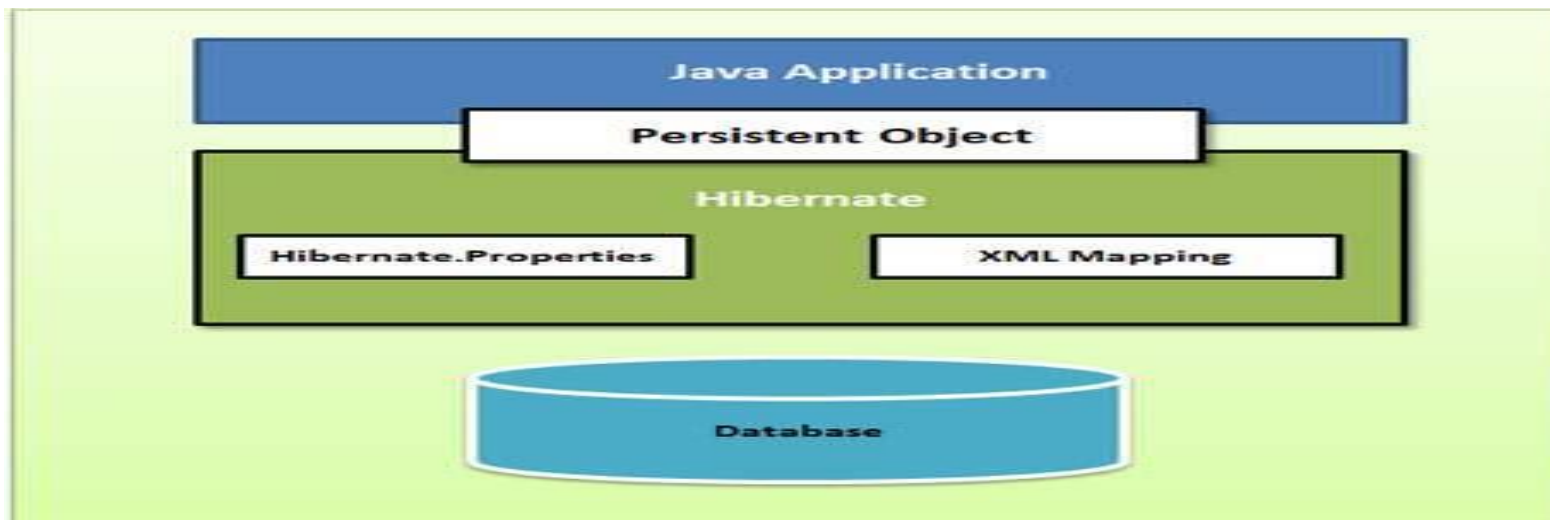
## - Ưu điểm của Hibernate:

- Mã nguồn mở và nhẹ
- Hiệu suất nhanh
- Truy vấn cơ sở dữ liệu độc lập
  - HQL là phiên bản hướng đối tượng của SQL*
- Tạo bảng tự động.
- Đơn giản lệnh join phức tạp.
- Cung cấp thống kê truy vấn và trạng thái cơ sở dữ liệu.

# Kiến trúc của Hibernate

## Kiến trúc của Hibernate

Kiến trúc Hibernate bao gồm nhiều đối tượng như đối tượng **persistent**, **session factory**, **transaction factory**, **connection factory**, **session**, **transaction**

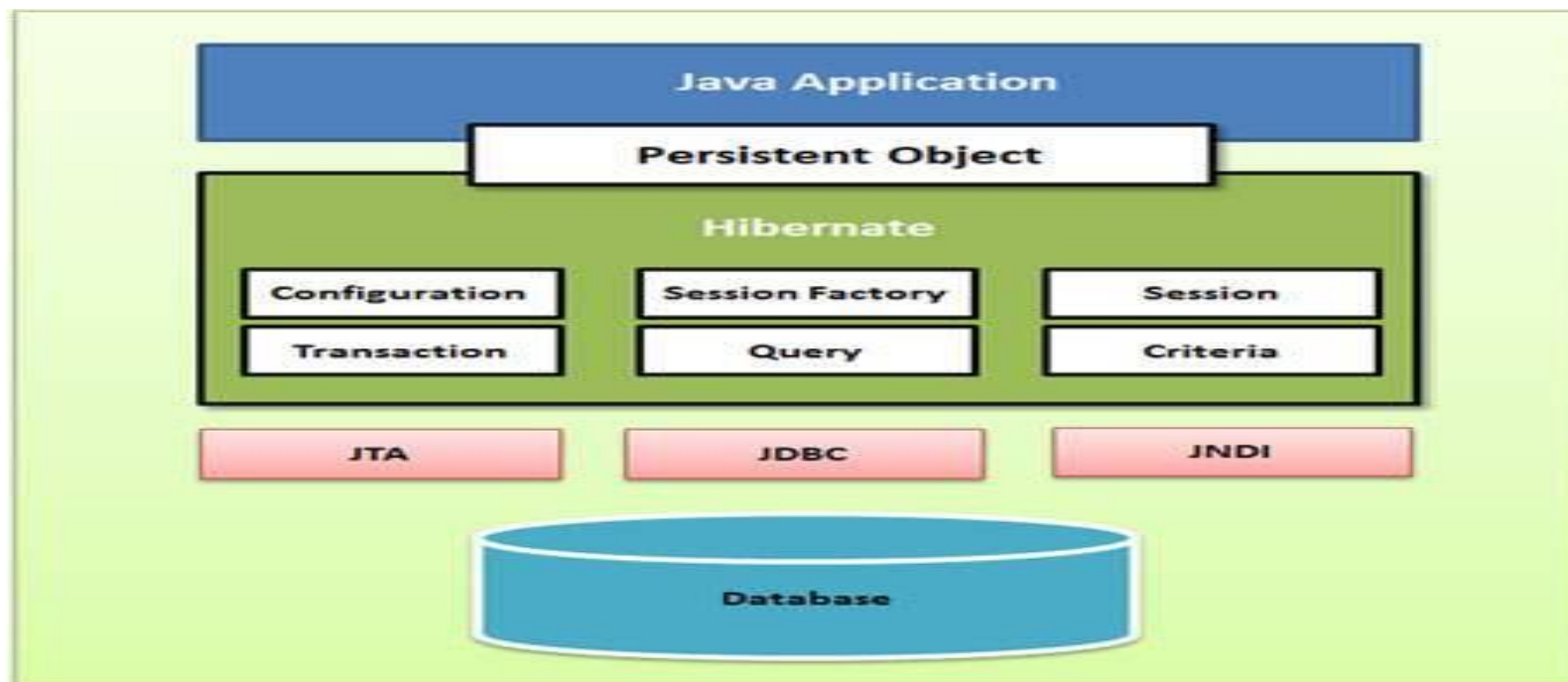




# Kiến trúc của Hibernate

## Kiến trúc của Hibernate

Kiến trúc ứng dụng Hibernate với vài lớp core quan trọng



# Kiến trúc của Hibernate

## Đối tượng Configuration

là đối tượng Hibernate đầu tiên bạn tạo trong bất kỳ ứng dụng Hibernate nào và chỉ cần tạo một lần trong quá trình khởi tạo ứng dụng. Nó đại diện cho một tập tin cấu hình hoặc thuộc tính yêu cầu của Hibernate. Đối tượng Configuration cung cấp hai thành phần chính:

**Database Connection:** Thao tác này được xử lý thông qua một hoặc nhiều tệp cấu hình được Hibernate hỗ trợ. Các tệp này là *hibernate.properties* và *hibernate.cfg.xml*.

**Class Mapping Setup:** Thành phần này tạo ra kết nối giữa các lớp Java và các bảng cơ sở dữ liệu.

# Kiến trúc của Hibernate

## Đối tượng SessionFactory

Là đối tượng được tạo ra nhờ đối tượng **Configuration**.

- SessionFactory là đối tượng nặng nên thường nó được tạo ra trong quá trình khởi động ứng dụng và lưu giữ để sử dụng sau này.
- Bạn sẽ cần một đối tượng SessionFactory cho mỗi cơ sở dữ liệu bằng cách sử dụng một tập tin cấu hình riêng biệt. Vì vậy, nếu bạn đang sử dụng nhiều cơ sở dữ liệu thì bạn sẽ phải tạo nhiều đối tượng SessionFactory

# Kiến trúc của Hibernate

## Đối tượng Session

- Một session được sử dụng để có được một kết nối vật lý với một cơ sở dữ liệu. Đối tượng Session là nhẹ và được thiết kế để được tạo ra thể hiện mỗi khi tương tác với cơ sở dữ liệu. Các đối tượng liên tục được lưu và truy xuất thông qua một đối tượng Session.
- Không nên sử dụng đối tượng Session trong 1 thời gian dài. Chúng ta cần tạo ra khi cần thiết và đóng lại khi không sử dụng

# Kiến trúc của Hibernate

## Đối tượng Transaction

- Một Transaction đại diện cho một đơn vị làm việc với cơ sở dữ liệu và hầu hết các RDBMS hỗ trợ chức năng transaction.
- Đây là một đối tượng tùy chọn và các ứng dụng Hibernate có thể chọn không sử dụng interface này, thay vào đó quản lý transaction trong code ứng dụng riêng.

# Kiến trúc của Hibernate

## Đối tượng Query

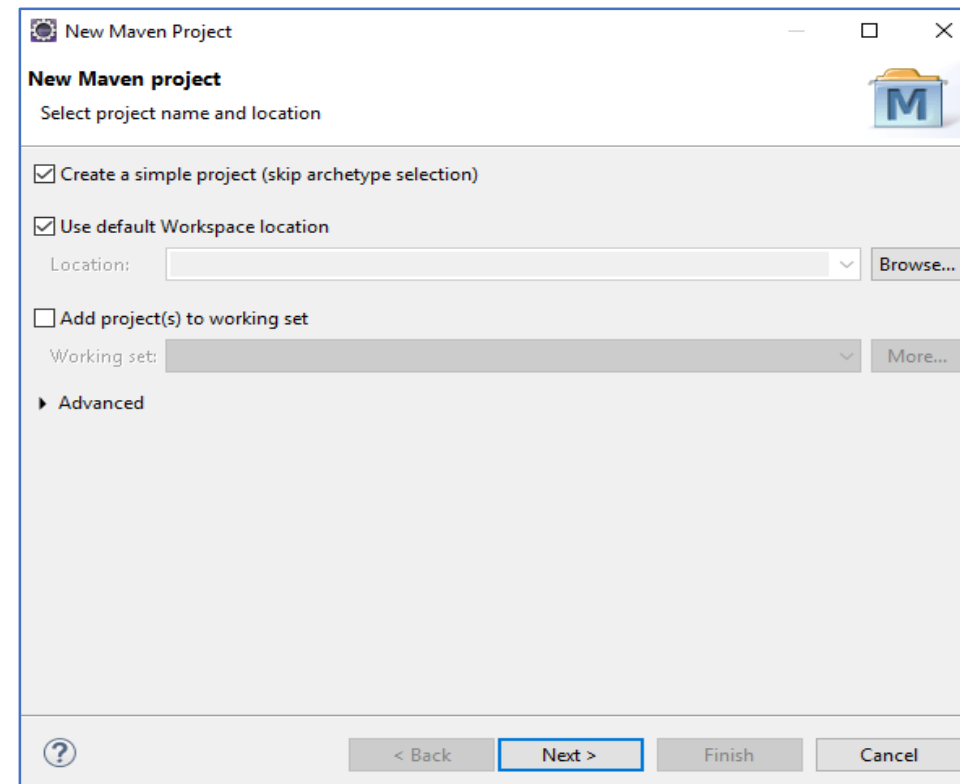
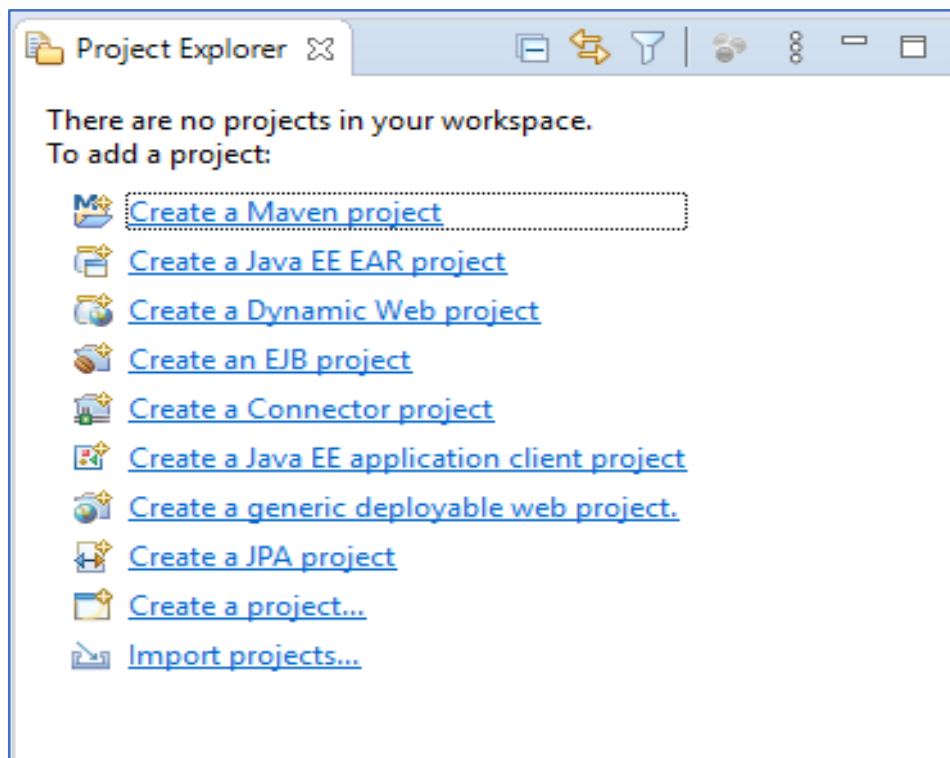
- Đối tượng Query sử dụng chuỗi truy vấn SQL hoặc Hibernate Query Language (HQL) để lấy dữ liệu từ cơ sở dữ liệu và tạo các đối tượng.

## Đối tượng Criteria

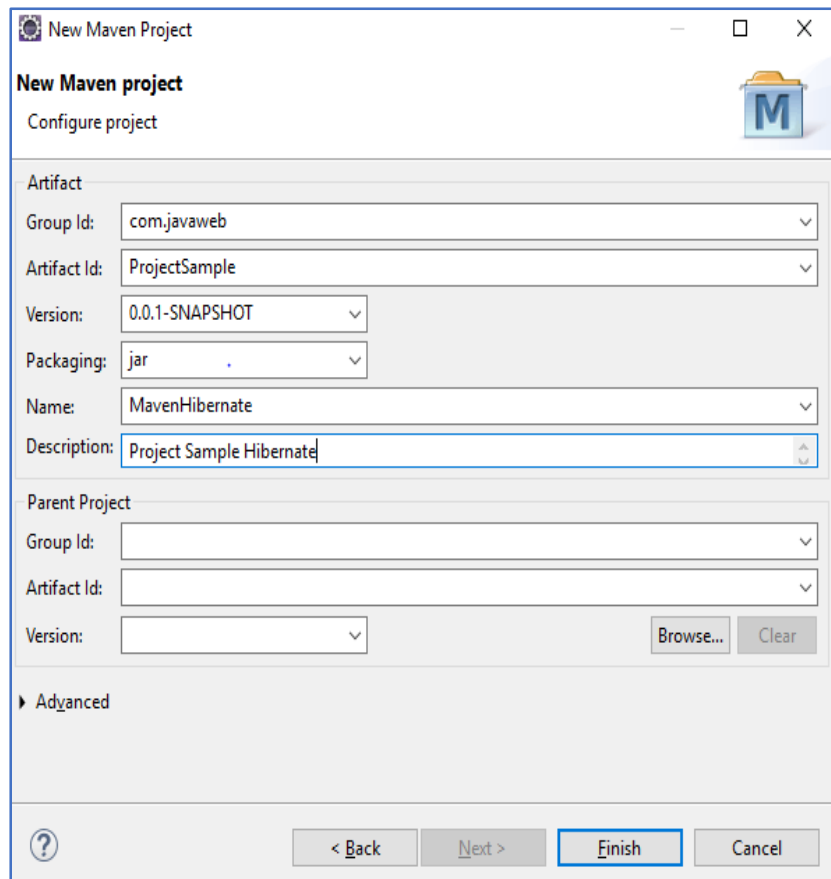
- Đối tượng Criteria được sử dụng để tạo và thực hiện truy vấn các tiêu chí định hướng đối tượng để lấy các đối tượng.

# Maven Project \_ Setting Project

## - Tạo maven Project:



# Maven Project \_ Setting Project



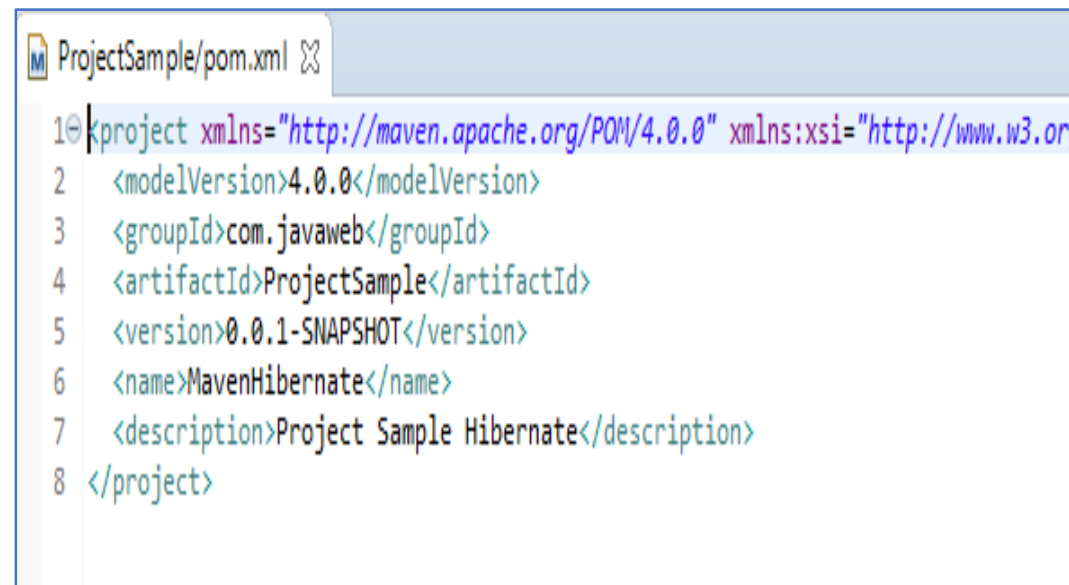
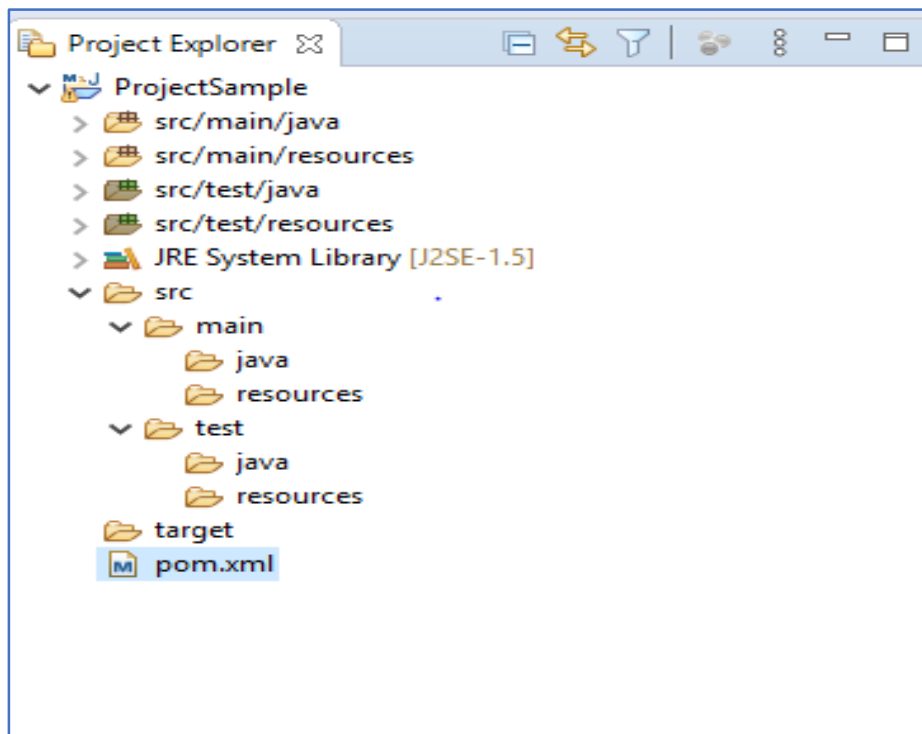
Trong đó :

- Group Id : Tên tổ chức / công ty / cá nhân của dự án. Ví dụ: com.gpcoder
- Artifact Id : Tên dự án (project). Lưu ý: tên viết liền, không có khoảng trắng ở giữa.
- Version : phiên của dự án.
- Package : để ý 2 giá trị : **jar** có nghĩa là **thư viện** or **java application**, **war** là **web application**.
- Name : Tên project (trong Eclipse)



# Maven Project \_ Setting Project

## - Cấu trúc project:



# Maven Project \_ Setting Project

- File pom.xml trong Maven Project:

File **pom.xml** là nơi khai báo tất cả những gì liên quan đến dự án được cấu hình qua maven, như khai báo các dependency, version của dự án, tên dự án, repository

# Maven Project \_ Setting Project

- Add dependencies Hibernate, MySQL:

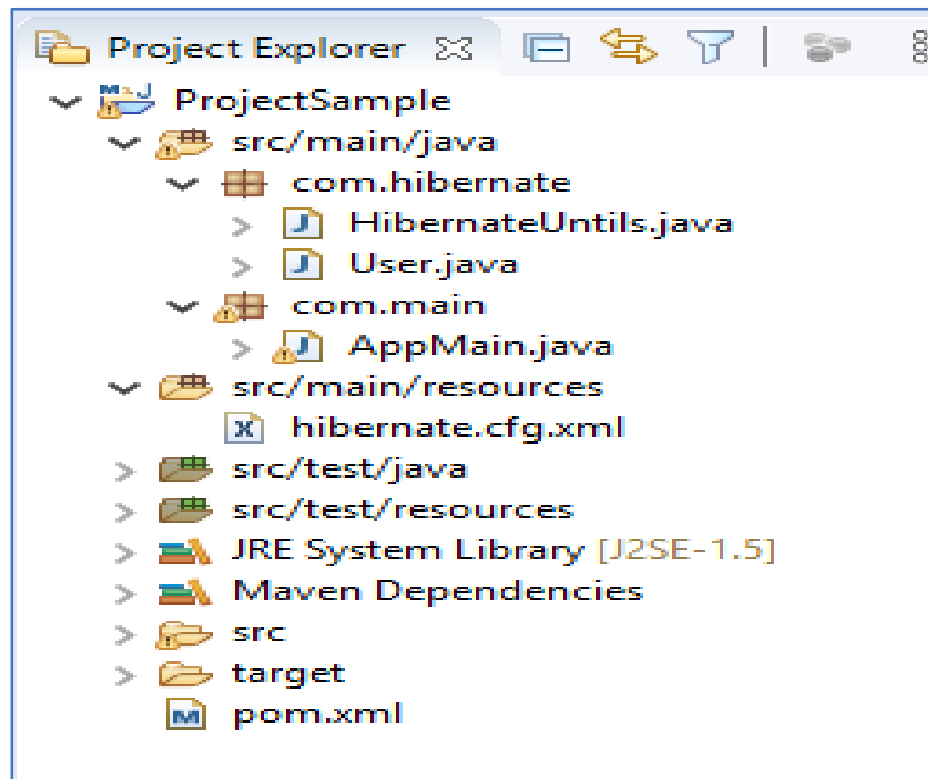
```

ProjectSample/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>com.javaweb</groupId>
6   <artifactId>ProjectSample</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <name>MavenHibernate</name>
9   <description>Project Sample Hibernate</description>
10
11   <!-- Các thư viện được khai báo bên trong thẻ dependencies -->
12   <dependencies>
13     <!-- Hibernate -->
14     <!-- http://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
15     <dependency>
16       <groupId>org.hibernate</groupId>
17       <artifactId>hibernate-core</artifactId>
18       <version>5.4.7.Final</version>
19     </dependency>
20
21     <!-- MySQL -->
22     <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
23     <dependency>
24       <groupId>mysql</groupId>
25       <artifactId>mysql-connector-java</artifactId>
26       <version>8.0.17</version>
27     </dependency>
28   </dependencies>
29   <build>
30     <finalName>${project.artifactId}</finalName>
31   </build>
32 </project>

```

# Maven Project \_ Setting Project

- Create Project Sample: Cấu trúc thư mục Project



# Maven Project \_ Setting Project

- Create Project Sample: Pojo Object

```
User.java
17
18 @Entity
19 @Table(name = "user_table")
20 public class User {
21
22     @Id
23     @Column(name = "user_id")
24     private int userid;
25     @Column(name = "user_name")
26     private String username;
27     @Column(name = "created_by")
28     private String createdBy;
29     @Column(name = "created_date")
30     private Date createdDate;
31
32     public int getUserid() {
33         return userid;
34     }
35     public void setUserid(int userid) {
36         this.userid = userid;
37     }
38     public String getUsername() {
39         return username;
40     }
41     public void setUsername(String username) {
42         this.username = username;
43     }
44     public String getCreatedBy() {
45         return createdBy;
46     }
47     public void setCreatedBy(String createdBy) {
48         this.createdBy = createdBy;
49     }
50     public Date getCreatedDate() {
51         return createdDate;
52     }
53     public void setCreatedDate(Date createdDate) {
54         this.createdDate = createdDate;
55     }
56 }
```

# Maven Project \_ Setting Project

- Create Project Sample: Class Setup Hibernate

```
1  HibernateUtils.java
6  import org.hibernate.SessionFactory;
11
12  /**
13   * @author NAM
14   *
15   */
16  // File Setup Hibernate
17  public class HibernateUtils {
18
19      private static final SessionFactory sessionFactory = buildSessionFactory();
20
21      private HibernateUtils() {
22          super();
23      }
24
25      private static SessionFactory buildSessionFactory() {
26          ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder() //
27              .configure() // Load hibernate.cfg.xml from resource folder by default
28              .build();
29          Metadata metadata = new MetadataSources(serviceRegistry).getMetadataBuilder().build();
30          return metadata.getSessionFactoryBuilder().build();
31      }
32
33      public static SessionFactory getSessionFactory() {
34          return sessionFactory;
35      }
36
37      public static void close() {
38          getSessionFactory().close();
39      }
40
41  }
```

# Maven Project \_ Setting Project

## - Create Project Sample: Class AppMain

```

AppMain.java
21 public class AppMain {
22     static User userObj;
23     static Session sessionObj;
24     static SessionFactory sessionFactoryObj;
25
26     public static void main(String[] args) {
27         System.out.println(".....Hibernate Maven Example.....\n");
28         try {
29             //
30             sessionFactoryObj = HibernateUtils.getSessionFactory();
31             //
32             sessionObj = sessionFactoryObj.openSession();
33             // Tất cả các lệnh hành động với DB thông qua Hibernate
34             // đều phải nằm trong 1 giao dịch (Transaction)
35             // Bắt đầu giao dịch
36             sessionObj.beginTransaction();
37
38             for (int i = 100; i <= 105; i++) {
39                 userObj = new User();
40                 userObj.setUserid(i);
41                 userObj.setUsername("Editor " + i);
42                 userObj.setCreatedBy("Administrator");
43                 userObj.setCreatedDate(new Date());
44                 // Save Object
45                 sessionObj.save(userObj);
46             }
47             System.out.println("\n.....Records Saved Successfully To The Database.....\n");
48
49             // Committing The Transactions To The Database
50             sessionObj.getTransaction().commit();
51         } catch (Exception sqlException) {
52             if (null != sessionObj.getTransaction()) {
53                 System.out.println("\n.....Transaction Is Being Rolled Back.....");
54                 sessionObj.getTransaction().rollback();
55             }
56             sqlException.printStackTrace();
57         } finally {
58             if (sessionObj != null) {
59                 sessionObj.close();
60             }
61         }
62     }

```

# Maven Project \_ Setting Project

## - Create Project Sample: Console Eclipse

```

.....Hibernate Maven Example.....

Sep 15, 2020 5:25:13 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {5.4.7.Final}
Sep 15, 2020 5:25:14 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
Sep 15, 2020 5:25:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Sep 15, 2020 5:25:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/hibernatedb?serverTimezone=UTC]
Sep 15, 2020 5:25:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
Sep 15, 2020 5:25:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Sep 15, 2020 5:25:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 4 (min=1)
Sep 15, 2020 5:25:16 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
Sep 15, 2020 5:25:17 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@21
Hibernate: create table user_table (user_id integer not null, created_by varchar(255), created_date datetime, user_name varchar(255), primary key (user_id)) engine=MyISAM

.....Records Saved Successfully To The Database.....

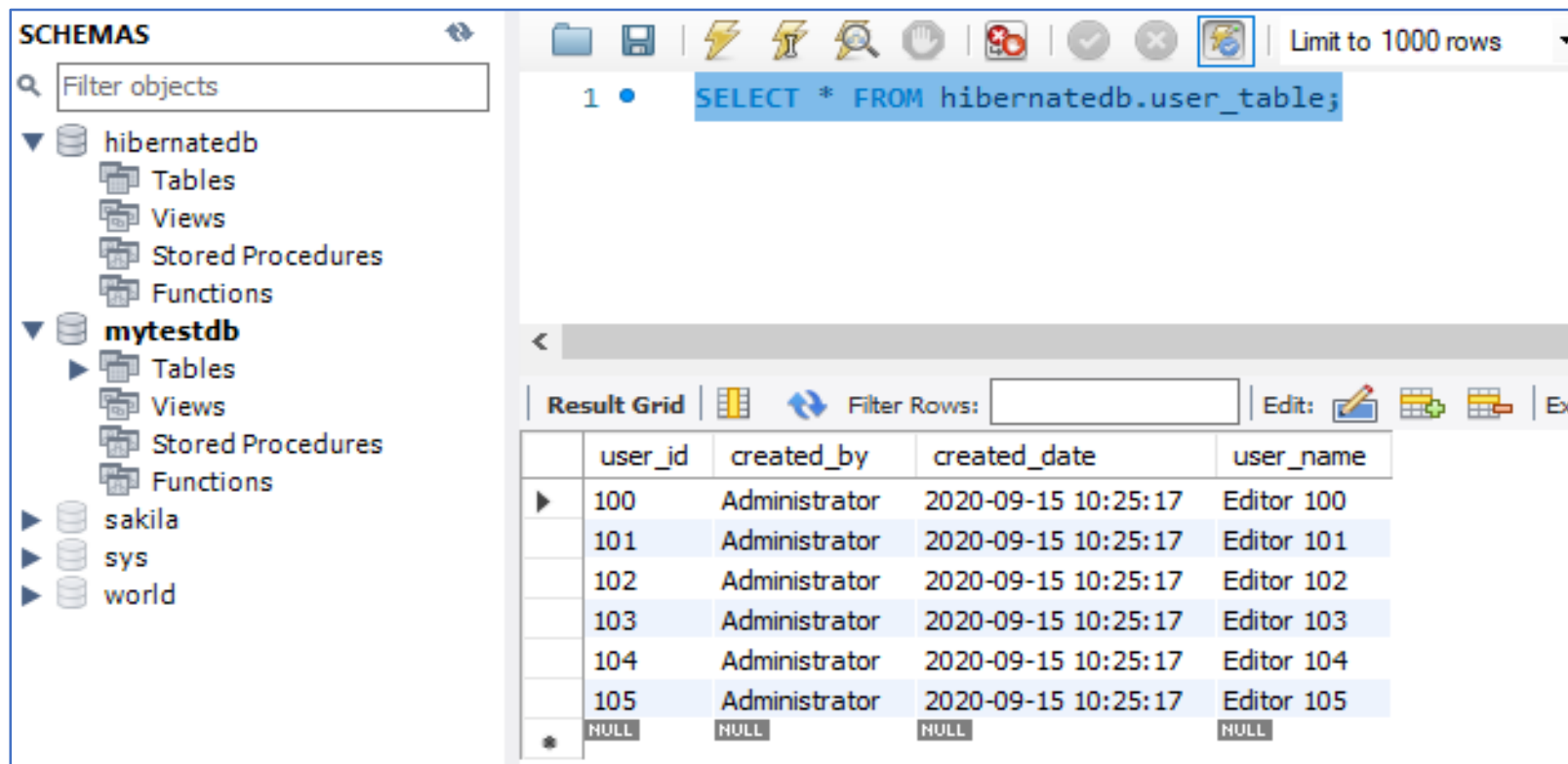
Hibernate: insert into user_table (created_by, created_date, user_name, user_id) values (?, ?, ?, ?)
Hibernate: insert into user_table (created_by, created_date, user_name, user_id) values (?, ?, ?, ?)
Hibernate: insert into user_table (created_by, created_date, user_name, user_id) values (?, ?, ?, ?)
Hibernate: insert into user_table (created_by, created_date, user_name, user_id) values (?, ?, ?, ?)
Hibernate: insert into user_table (created_by, created_date, user_name, user_id) values (?, ?, ?, ?)
Hibernate: insert into user_table (created_by, created_date, user_name, user_id) values (?, ?, ?, ?)

```



# Maven Project \_ Setting Project

## - Create Project Sample:Database



The screenshot shows a database management interface. On the left, a 'SCHEMAS' tree lists databases: hibernatedb, mytestdb, sakila, sys, and world. The 'mytestdb' database is selected, and its 'Tables' folder is expanded. In the center, a SQL query is entered: `SELECT * FROM hibernatedb.user_table;`. Below the query, a 'Result Grid' displays the data from the 'user\_table' in the 'hibernatedb' database. The grid has four columns: 'user\_id', 'created\_by', 'created\_date', and 'user\_name'. It shows six rows of data, with the last row being a NULL record.

| user_id | created_by    | created_date        | user_name  |
|---------|---------------|---------------------|------------|
| 100     | Administrator | 2020-09-15 10:25:17 | Editor 100 |
| 101     | Administrator | 2020-09-15 10:25:17 | Editor 101 |
| 102     | Administrator | 2020-09-15 10:25:17 | Editor 102 |
| 103     | Administrator | 2020-09-15 10:25:17 | Editor 103 |
| 104     | Administrator | 2020-09-15 10:25:17 | Editor 104 |
| 105     | Administrator | 2020-09-15 10:25:17 | Editor 105 |
| NULL    | NULL          | NULL                | NULL       |

# Entity, JavaBeans và POJOs

---

Entity là gì:

- Entity là các Class dùng để mô tả một bảng trong DB.

# Entity, JavaBeans và POJOs

## Entity minh họa là gì:

```
package com.gpcoder.entities;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;

@Data
@Entity
@Table(name = "user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column
    private String fullname;

    @Column
    private String username;

    @Column
    private String password;

    @Column(name = "created_at")
    private Date createdAt;

    @Column(name = "modified_at")
    private Date modifiedAt;
}
```

# Entity, JavaBeans và POJOs

## Một số Annotation được sử dụng trong Entity:

- **@Entity** : cho biết đây là một Entity.
- **@Table** : cho biết đây là một Table trong database, chúng ta có thể chỉ định tên tương ứng và các ràng buộc trong database. Mặc định, Hibernate sẽ lấy tên class tương ứng với tên table trong database nếu nó không được chỉ định name.
- **@Id** : đây là Identity của Entity. nó tương đương với khóa chính (Primary key) của table.
- **@GeneratedValue** : được sử dụng để Hibernate tự động tạo ra giá trị và gán vào cho một cột trong trường hợp insert mới một Entity vào database.
- **@Column** : được sử dụng để chú thích đây là một column trong database. Nó có thể bao gồm các thông tin ràng buộc của column như độ dài của cột, cho phép null hay không, ... Mặc định, Hibernate sẽ lấy tên property tương ứng với tên column trong database nếu nó không được chỉ định name.

# Entity, JavaBeans và POJOs

## JavaBeans trong Java:

- là các thành phần phần mềm có thể tái sử dụng cho Java có thể được thao tác trực quan trong một công cụ xây dựng.
- Một số nhà phát triển xem JavaBeans là các Đối tượng Java cũ đơn giản tuân theo các quy ước đặt tên cụ thể.

# Entity, JavaBeans và POJOs

## JavaBeans trong Java (tiếp):

- Lớp JavaBean phải thực hiện Serializable hoặc Externalizable
- Lớp JavaBean phải có hàm tạo không có đối số
- Tất cả các thuộc tính JavaBean phải có các phương thức setter và getter công khai
- Tất cả các biến đối tượng JavaBean phải ở chế độ riêng tư

# Entity, JavaBeans và POJOs

## JavaBeans trong Java (tiếp):

```
@Entity
public class Employee implements Serializable{

    @Id
    private int id;
    private String name;
    private int salary;

    public Employee() {}

    public Employee(String name, int salary) {
        this.name = name;
        this.salary = salary;
    }
    public int getId() {
        return id;
    }
    public void setId( int id ) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName( String name ) {
        this.name = name;
    }
    public int getSalary() {
        return salary;
    }
    public void setSalary( int salary ) {
        this.salary = salary;
    }
}
```

# Entity, JavaBeans và POJOs

## Pojos là gì:

- POJO là từ viết tắt của Plain Old Java Object.
- Tên được sử dụng để nhấn mạnh rằng đối tượng được đề cập là một đối tượng Java thông thường, không phải là một đối tượng đặc biệt và đặc biệt không phải là JavaBean dành cho doanh nghiệp (đặc biệt là trước EJB 3)..



# Entity, JavaBeans và POJOs

## Pojos là gì(tiếp):

- Một POJO (đối tượng Java cũ đơn giản) không được xác định chặt chẽ.
- Đó là một đối tượng Java không có yêu cầu triển khai một giao diện cụ thể hoặc xuất phát từ một lớp cơ sở cụ thể hoặc sử dụng các chú thích cụ thể

# Entity, JavaBeans và POJOs

---

## Phân biệt JavaBean và Pojo

Tất cả các JavaBeans đều là POJO nhưng không phải tất cả POJO đều là JavaBeans.

# Các hàm CRUD trong Hibernate

## Cấu trúc thư mục Project và file pom.xml



The image shows an IDE with two panels. The left panel, titled 'Project Explorer', displays the project structure for 'CRUDHibernate'. The right panel, titled 'CRUDHibernate/pom.xml', shows the content of the Maven POM file.

**Project Structure (Left Panel):**

- CRUDHibernate
  - src/main/java
    - confighibernate
      - HibernateUtils.java
    - entity
      - User.java
    - main
      - HibernateExample.java
  - src/main/resources
    - hibernate.cfg.xml
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-1.7]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- ProjectSample

**pom.xml Content (Right Panel):**

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>com.hibernate</groupId>
6   <artifactId>CRUDHibernate</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <name>CRUDinHibernate</name>
9
10  <!-- Các thư viện được khai báo bên trong thẻ dependencies -->
11  <dependencies>
12    <!-- Hibernate -->
13    <!-- http://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
14    <dependency>
15      <groupId>org.hibernate</groupId>
16      <artifactId>hibernate-core</artifactId>
17      <version>5.4.7.Final</version>
18    </dependency>
19
20    <!-- MySQL -->
21    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
22    <dependency>
23      <groupId>mysql</groupId>
24      <artifactId>mysql-connector-java</artifactId>
25      <version>8.0.17</version>
26    </dependency>
27  </dependencies>
28
29 </project>
  
```

# Các hàm CRUD trong Hibernate

## File config : hibernate.cfg.xml

```
hibernate.cfg.xml
1 <!DOCTYPE hibernate-configuration PUBLIC
2   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
3   "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4 <hibernate-configuration>
5   <session-factory>
6     <!-- Database setting -->
7     <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
8     <property name="connection.url">jdbc:mysql://localhost:3306/hibernatedb?serverTimezone=UTC</property>
9     <property name="connection.username">root</property>
10    <property name="connection.password">root@123</property>
11
12    <!-- JDBC connection pool (use the built-in) -->
13    <property name="connection.pool_size">4</property>
14
15    <!-- SQL dialect -->
16    <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
17
18    <!-- Enable Hibernate's automatic session context management -->
19    <property name="current_session_context_class">thread</property>
20
21    <!-- Disable the second-level cache -->
22    <property name="cache.provider_class">org.hibernate.cache.internal.NoCacheProvider</property>
23
24    <!-- Show all executed SQL to console -->
25    <property name="show_sql">true</property>
26
27    <!-- Setting Update or Create if table don't exists-->
28    <property name="hbm2ddl.auto">update</property>
29
30    <!-- Entity mapping -->
31    <mapping class="entity.User" />
32
33  </session-factory>
34 </hibernate-configuration>
```

# Các hàm CRUD trong Hibernate

## File User.java( Entity)

```
User.java
19 @Entity
20 @Table(name = "UserSample2")
21 public class User {
22
23     @Id
24     @GeneratedValue(strategy = GenerationType.IDENTITY)
25     private Long id;
26
27     @Column
28     private String fullname;
29
30     @Column
31     private String username;
32
33     @Column
34     private String password;
35
36     @Column(name = "created_at")
37     private Date createdAt;
38
39     @Column(name = "modified_at")
40     private Date modifiedAt;
41
42     public Long getId() {
43         return id;
44     }
45
46     public void setId(Long id) {
47         this.id = id;
48     }
49
50     public String getFullname() {
51         return fullname;
52     }
53
54     public void setFullname(String fullname) {
55         this.fullname = fullname;
56     }
57
58     public String getUsername() {
59         return username;
60     }
61 }
```

# Các hàm CRUD trong Hibernate

## Class config Hibernate

```
HibernateUtils.java
1  /**
2   *
3   * package confighibernate;
4   *
5   * import org.hibernate.SessionFactory;
6   *
7   * /**
8   *  * @author NAM
9   *  *
10  */
11  public class HibernateUtils {
12
13      private static final SessionFactory sessionFactory = buildSessionFactory();
14
15      private HibernateUtils() {
16          super();
17      }
18
19      private static SessionFactory buildSessionFactory() {
20          ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder() //
21              .configure() // Load hibernate.cfg.xml from resource folder by default
22              .build();
23          Metadata metadata = new MetadataSources(serviceRegistry).getMetadataBuilder().build();
24          return metadata.getSessionFactoryBuilder().build();
25      }
26
27      public static SessionFactory getSessionFactory() {
28          return sessionFactory;
29      }
30
31      public static void close() {
32          getSessionFactory().close();
33      }
34  }
```

# Các hàm CRUD trong Hibernate

## Class main

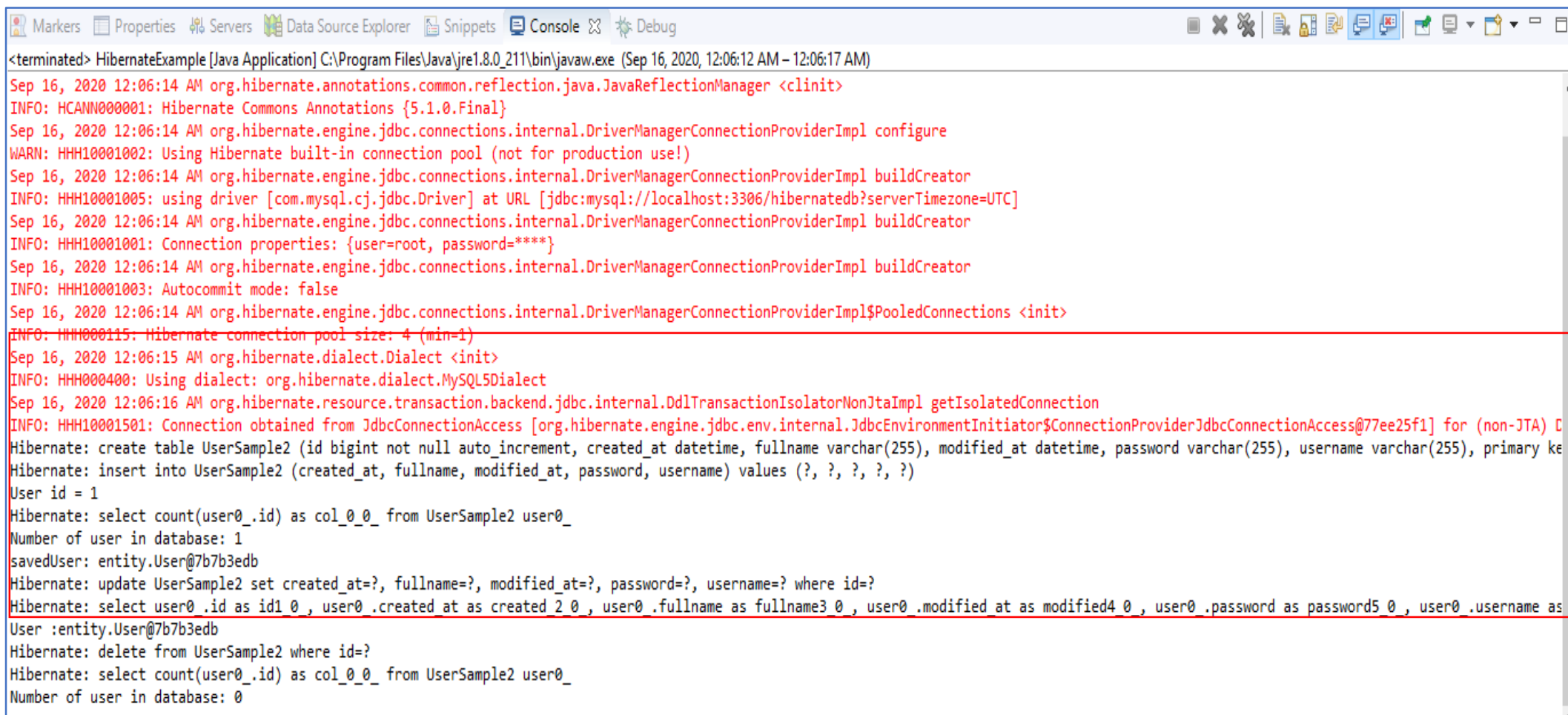
```

18 public class HibernateExample {
19
20     public static void main(String[] args) {
21
22         try (Session session = HibernateUtils.getSessionFactory().openSession();) {
23             // Begin a unit of work
24             session.beginTransaction();
25
26             // Insert user
27             Date currentDate = new Date();
28             User user1 = new User();
29             user1.setFullname("Hibernate Example");
30             user1.setUsername("gpocoder");
31             user1.setPassword("123456"); // Should encode password
32             user1.setCreatedAt(currentDate);
33             user1.setModifiedAt(currentDate);
34             Long userId = (Long) session.save(user1);
35             System.out.println("User id = " + userId);
36
37             // Count user from database
38             Long numberOfUser = session.createQuery("SELECT COUNT(id) FROM User", Long.class).uniqueResult();
39             System.out.println("Number of user in database: " + numberOfUser);
40
41             // Get user by id
42             User savedUser = session.find(User.class, userId);
43             System.out.println("savedUser: " + savedUser);
44
45             // Update user
46             savedUser.setFullname("GP Coder");
47             session.update(savedUser);
48
49             // Get users
50             List<User> users = session.createQuery("FROM User", User.class).list();
51             for (User user : users) {
52                 System.out.println("User : " + user);
53             }
54
55             // Delete user
56             session.delete(savedUser);
57
58             // Count user from database
59             numberOfUser = session.createQuery("SELECT COUNT(id) FROM User", Long.class).uniqueResult();
60             System.out.println("Number of user in database: " + numberOfUser);
61
62             // Commit the current resource transaction, writing any unflushed changes to the
63             // database.
64             session.getTransaction().commit();
65         }
    }

```

# Các hàm CRUD trong Hibernate

## Màn hình console của IDE



```
<terminated> HibernateExample [Java Application] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (Sep 16, 2020, 12:06:12 AM - 12:06:17 AM)
Sep 16, 2020 12:06:14 AM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
Sep 16, 2020 12:06:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Sep 16, 2020 12:06:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/hibernatedb?serverTimezone=UTC]
Sep 16, 2020 12:06:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
Sep 16, 2020 12:06:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Sep 16, 2020 12:06:14 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 4 (min=1)
Sep 16, 2020 12:06:15 AM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
Sep 16, 2020 12:06:16 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@77ee25f1] for (non-JTA) C
Hibernate: create table UserSample2 (id bigint not null auto_increment, created_at datetime, fullname varchar(255), modified_at datetime, password varchar(255), username varchar(255), primary ke
Hibernate: insert into UserSample2 (created_at, fullname, modified_at, password, username) values (?, ?, ?, ?, ?)
User id = 1
Hibernate: select count(user0_.id) as col_0_0_ from UserSample2 user0_
Number of user in database: 1
savedUser: entity.User@7b7b3edb
Hibernate: update UserSample2 set created_at=?, fullname=?, modified_at=?, password=?, username=? where id=?
Hibernate: select user0_.id as id1_0_, user0_.created_at as created_2_0_, user0_.fullname as fullname3_0_, user0_.modified_at as modified4_0_, user0_.password as password5_0_, user0_.username as
User :entity.User@7b7b3edb
Hibernate: delete from UserSample2 where id=?
Hibernate: select count(user0_.id) as col_0_0_ from UserSample2 user0_
Number of user in database: 0
```



# Q&A





**THANK YOU**