**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Hibernate Tutorial**

Hibernate là một dịch vụ truy vấn và lưu trữ Đối tượng – Quan hệ có hiệu suất cao, được cấp phép theo nguồn mở GNU Lesser General Public License (LGPL)và miễn phí để tải xuống. Hibernate không chỉ quản lí việc ánh xạ từ các Class của Java đến các Table của Database( và từ kiểu dữ liệu Java sang kiểu dữ liệu SQL) mà còn cung cấp các truy vấn và phương tiện truy suất dữ liệu. Hướng dẫn này sẽ chỉ cho bạn các sử dụng Hibernate để phát triển các ứng dụng web dựa trên cơ sở dữ liệu của bạn một cách đơn giản và dễ dàng.

Bài hướng dẫn này được thiết kế cho tất cả các lập trình viên Java mà muốn tìm hiểu Hibernate framework và API của nó.

Yêu cầu: có sự hiểu biết cơ bản về cơ sở dữ liệu quan hệ, JDBC, SQL sẽ rất hữu ích trong bài hướng dẫn này.

**MỤC LỤC**

*ORM – Overview 3*

*Hibernate – Overview 8*

*Architecture 10*

*Environment*

*Configuration*

*Sessions*

*Persistent Class*

*Mapping Files*

*Mapping Types*

*Example*

*O/R Mappings*

*Annotations*

*Query Language*

*Critetia Queries*

*Native SQL*

*Caching*

*Batch Processing*

*Interceptors*

***ORM Overview***

***JDBC là gì ?***

JDBC là viết tắt của Java Database Connectivity. Nó cung cấp một bộ Java API để kế nối đến cơ sở dữ liệ quan hệ từ chương trình Java. Các Java APIs này cho phép chương trình Java thực thi câu lệnh SQL và tương tác với bất kỳ cơ sở dữ liệu tuân thủ SQL.

JDBC cung cấp một kiến trúc linh hoạt để viết một ứng dụng độc lập cơ sở dữ liệu có thể chạy trên các nền tảng khác nhau và tương tác với các DBMS khác nhau mà không cần sửa đổi.

***Ưu và nhược điểm:***

|  |  |
| --- | --- |
| ***Ưu điểm*** | ***Nhược điểm*** |
| - Xử lí CSDL đơn giản và gọn gàng  - Hiệu suất tốt với dữ liệu lớn  - Tốt cho các chương trình nhỏ  - Cú pháp đơn giản, dễ hiểu | ***-*** Phức tạp khi sử dụng trong dự án lớn.  - Chi phí lập trình cao.  - Không đóng gói.  - Khó để thực hiện MVC concept.  - Lệnh truy vấn mỗi DBMS là riêng biệt. |

***Tại sao sử dụng Object Relation Mapping (ORM) ?***

Khi chúng ta làm việc với một hệ thống hướng đối tượng, có một sự không phù hợp giữa mô hình đối tượng và cơ sở dữ liệu quan hệ. Các RDBMS biểu thị dữ liệu theo định dạng bảng trong khi các ngôn ngữ hướng đối tượng, chẳng hạn như Java hoặc C # biểu thị nó dưới dạng biểu đồ liên kết của các đối tượng.

Hãy xem xét các hàm Java sau:

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

public int getId() {

return id;

}

public String getFirstName() {

return first\_name;

}

public String getLastName() {

return last\_name;

}

public int getSalary() {

return salary;

}

}

Hãy xem xét các đối tượng trên sẽ được lưu trữ và truy xuất vào bảng RDBMS sau

create table EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

Vấn đề đầu tiên, nếu chúng ta cần sửa đổi thiết kế cơ sở dữ liệu của mình sau khi đã phát triển một vài trang hoặc ứng dụng của chúng ta thì sao? Thứ hai, tải và lưu trữ các đối tượng trong cơ sở dữ liệu quan hệ cho chúng ta thấy năm vấn đề không khớp sau đây.

|  |  |
| --- | --- |
| Sr.No. | Điều không phù hợp và mô tả |
| 1 | ***Độ chi tiết***  Đôi khi bạn sẽ có mô hình đối tượng, có nhiều Class hơn số lượng Table  tương ứng trong CSDL. |
| 2 | ***Kế thừa***  Các RDBMS không định nghĩa bất cứ điều gì tương tự như Kế thừa,  đó là một mô hình tự nhiên trong các ngôn ngữ lập trình hướng đối tượng***.*** |
| 3 | ***Định danh***  Một RDBMS định nghĩa chính xác một khái niệm về 'sự giống nhau':  khóa chính. Tuy nhiên, Java định nghĩa cả danh tính đối tượng (a == b)  và đẳng thức đối tượng (a.equals (b)). |
| 4 | ***Liên kết***  Các ngôn ngữ hướng đối tượng biểu thị các liên kết bằng cách sử dụng các  tham chiếu đối tượng trong khi RDBMS biểu thị một liên kết dưới dạng  khóa ngoài. |
| 5 | ***Cách truy cập***  Các cách bạn truy cập các đối tượng trong Java và trong RDBMS về cơ bản  là khác nhau. |

Object Relation Mapping (ORM) là giải pháp để xử lý tất cả các sự không phù hợp trở ở trên.

***ORM là gì ?***

ORM là viết tắt của **O**bject-**R**elative **M**apping (ORM) là một kỹ thuật lập trình để chuyển đổi dữ liệu giữa cơ sở dữ liệu quan hệ và ngôn ngữ lập trình hướng đối tượng như Java, C #, v.v.

Một hệ thống ORM có những ưu điểm sau so với JDBC:

|  |  |
| --- | --- |
| ***Sr.No*** | ***Ưu điểm*** |
| 1 | Sử dụng code để truy cập đối tượng hơn là các bảng của DB |
| 2 | Ẩn chi tiết các truy vấn SQL |
| 3 | Dựa trên JDBC |
| 4 | Không phải đối phó với sự thay đổi của cơ sở dữ liệu |
| 5 | Các entities dựa trên các business concept hơn là cấu trúc cơ sở  dữ liệu. |
| 6 | Quản lí transaction và tạo key tự động |
| 7 | Phát triển ứng dụng nhanh |

Một giải pháp ORM bao gồm bốn thực thể sau:

|  |  |
| --- | --- |
| ***Sr.No*** | ***Giải pháp*** |
| 1 | Một API để thực hiện các hoạt động CRUD cơ bản trên các đối tượng của các lớp. |
| 2 | Một ngôn ngữ hoặc API để chỉ định các truy vấn tham chiếu đến các lớp và thuộc tính của các lớp. |
| 3 | Một cơ sở cấu hình cho siêu dữ liệu ánh xạ. |
| 4 | Một kỹ thuật để tương tác với các đối tượng transaction để thực  hiện dirty checking, lazy association fetching và các chức năng  tối ưu hóa khác. |

***Java ORM Frameworks***

Có một số framework và các tùy chọn ORM trong Java.A persistent framework là một dịch vụ ORM lưu trữ và truy xuất các đối tượng vào cơ sở dữ liệu quan hệ.

* Enterprise JavaBeans Entity Beans
* Java Data Objects
* Castor
* TopLink
* Spring DAO
* Hibernate
* And many more

***Hibernate – Overview***

Hibernate là một giải pháp Object-Relation Mapping (ORM) cho JAVA. Nó là một framework tác bền vững mã nguồn mở được tạo ra bởi Gavin King vào năm 2001. Đây là một dịch vụ truy vấn và duy trì quan hệ đối tượng mạnh mẽ, hiệu suất cao cho bất kỳ ứng dụng Java nào.

Hibernate ánh xạ các lớp Java vào các bảng cơ sở dữ liệu và từ các kiểu dữ liệu Java thành các kiểu dữ liệu SQL và giải phóng nhà phát triển khỏi 95% các tác vụ lập trình liên quan đến dữ liệu phổ biến.

Hibernate nằm giữa các đối tượng Java truyền thống và máy chủ cơ sở dữ liệu để xử lý tất cả các công việc trong việc duy trì các đối tượng đó dựa trên các cơ chế và mẫu O / R thích hợp.



***Ưu điểm của Hibernate***

* Hibernate ánh xạ các lớp Java vào các bảng cơ sở dữ liệu bằng các tệp XML và không cần viết bất kỳ dòng mã nào.
* Cung cấp các API đơn giản để lưu trữ và truy xuất các đối tượng Java trực tiếp đến và từ cơ sở dữ liệu.
* Nếu có sự thay đổi trong cơ sở dữ liệu hoặc trong bất kỳ bảng nào, thì bạn chỉ cần thay đổi các thuộc tính tệp XML.
* Tóm tắt các loại SQL không quen thuộc và cung cấp một cách để làm việc xung quanh các Đối tượng Java quen thuộc.
* Hibernate không yêu cầu máy chủ ứng dụng hoạt động.
* Điều khiển sự lien kết phức tạp của các đối tượng của cơ sở dữ liệu.
* Giảm thiểu việc truy cập cơ sở dữ liệu với các cách tìm nạp thông minh.
* Cung cấp truy vấn dữ liệu đơn giản.

***Cơ sở dữ liệu hỗ trợ:***

Hibernate hỗ trợ hầu hết tất cả các RDBMS chính. Sau đây là danh sách:

* HSQL Database Engine
* DB2/NT
* MySQL
* PostgreSQL
* FrontBase
* Oracle
* Microsoft SQL Server Database
* Sybase SQL Server
* Informix Dynamic Server

***Công nghệ hỗ trợ:***

* XDoclet Spring
* J2EE
* Eclipse plug-ins
* Maven

# *Hibernate - Architecture*