1. إدارة البيانات العلاقية باستخدام JPA

1.1 مفهوم Object Relational Mapping (ORM)

ORM هو أسلوب يستخدم في إنشاء "جسر" بين البرامج الكائنية وقواعد البيانات العلائقية. بعبارة أخرى، يمكنك رؤية ORM على أنها الطبقة التي تربط البرمجة الموجهة للكائنات (OOP) بقواعد البيانات العلائقية.

عند التفاعل مع قاعدة بيانات باستخدام لغات OOP، سيتعين عليك إجراء عمليات مختلفة مثل إنشاء بيانات من قاعدة بيانات وقراءتها وتحديثها وحذفها (CRUD). يمكنك استخدام SQL لإجراء هذه العمليات في قواعد البيانات العلائقية. استخدام SQL لهذا الغرض ليس بالضرورة فكرة سيئة، ولكن ORM وادواته يساعد في تبسيط التفاعل بين قواعد البيانات العلائقية ولغات OOP المختلفة.

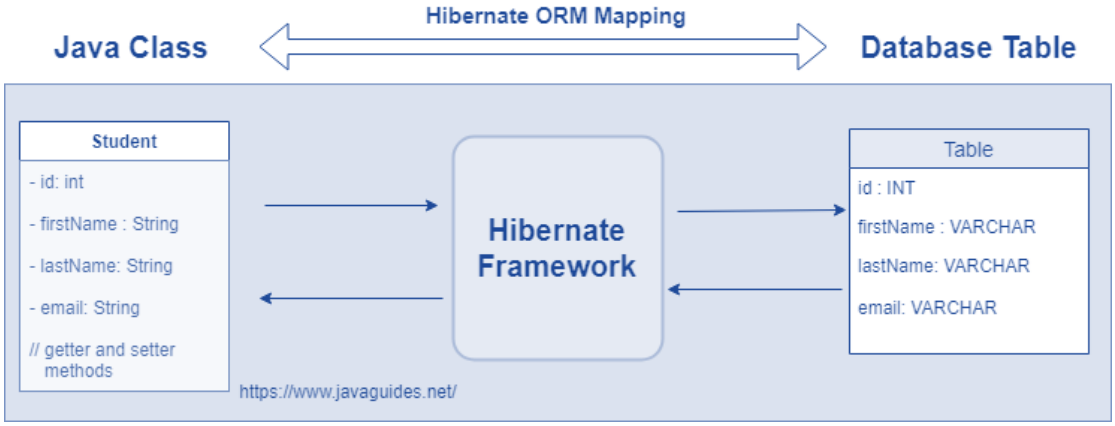
1.2 أداة ORM Tool - ORM

أداة ORM هي برنامج مصمم لمساعدة مطوري OOP على التفاعل مع قواعد البيانات العلائقية. لذا بدلاً من إنشاء برنامج ORM خاص بك من البداية، يمكنك الاستفادة من هذه الأدوات.

1.3 Hibernate

Hibernate هي أداة مفتوحة المصدر لـ ORM توفر إطار عمل لتعيين نماذج المجال الكائنية (object-oriented domain models) إلى قواعد البيانات العلاقية لتطبيقات الويب. يرشد إطار عمل Hibernate ORM لتعيين Java classes إلى جداول قاعدة البيانات، وأنواع بيانات Java لأنواع بيانات SQL ويوفر الاستعلام والاسترجاع.

يقلل Hibernate سطور الاوامر البرمجية من خلال الحفاظ على تعيين جدول الكائنات نفسه ويعيد النتيجة إلى التطبيق في شكل كائنات Java. إنه يعفي المبرمج من المعالجة اليدوية للبيانات المستمرة، وبالتالي تقليل وقت التطوير وتكلفة الصيانة.



1.4 مفهوم JPA (Java Persistence API)

JPA هي إحدى مواصفات Java لإدارة البيانات العلاقية في تطبيقات Java. يسمح لنا بالوصول إلى البيانات واستمرارها بين كائن Java وقاعدة البيانات العلاقية. يعمل JPA كجسر بين نماذج المجال الكائنية وأنظمة قواعد البيانات العلاقية (ORM).

نظرًا لأن JPA هي مجرد مواصفات، فهي لا تقوم بأي عملية بمفردها ;يتطلب التنفيذ. لذا، فإن أدوات ORM مثل Hibernateتنفذ مواصفات JPA لاستمرار البيانات.

1.5 ربط مشروع بقاعدة البيانات Connect Spring Project to DB

يوفر Spring Boot مكتبات (Dependencies) لتوصيل تطبيق Spring بقاعدة البيانات العلاقية بكفاءة. وهي:

- Spring Data JPA
- MySQL Driver

وللربط مع قاعدة البيانات يجب إضافة بعض التكوينات الى ملف application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/schoolDB
spring.datasource.username=root
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=create-drop

server.error.include-message=always
server.error.include-stacktrace=always
```

ولتعريف كيان بسيط بقاعدة بيانات، قم بإضافة تعليقات توضيحية لكائن Java:

```
@Data @NoArgsConstructor @AllArgsConstructor @Entity
public class Student {

    @Id
    private Integer id;
    private String name;
}
```

تم إضافة تعليق توضيحي على Student Class باستخدام @Entity ، للإشارة إلى أنه أحد كيانات JPA(نظرًا لعدم وجود التعليق التوضيحي @Table ، فمن المفترض أن هذا الكيان قد تم تعيينه لجدول يسمى student).

تمت إضافة التعليق التوضيحي @Id على id حتى يتعرف JPA عليها على أنها معرف الكائن.