Hibernate/ JPA

Topics:

What is Hibernate?

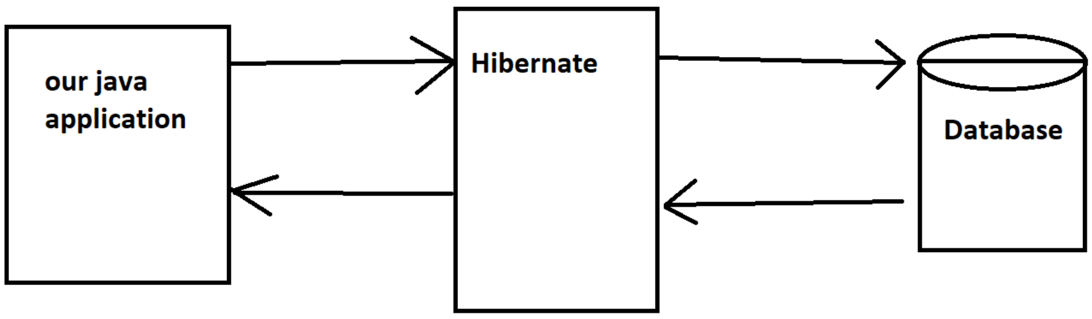
Benefits of using hibernate?

What is JPA?

Benefits of using JPA?

What is Hibernate?

- A framework which is used for persisting or saving java objects in a database.
- We can use it to retrieve the data from database



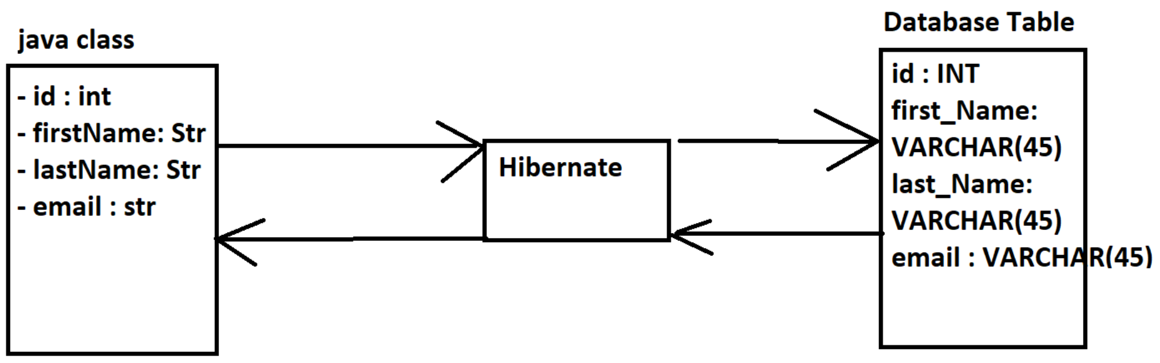
Benefits of Hibernate:

- It handles all low-level sql codes
- Minimizes the amount of JDBC code we have to develop

- Hibernate also provides the object-to-relational-mapping(ORM)

What is ORM?

- As a developer we have to just tell hibernate how our java class or object maps to the database.



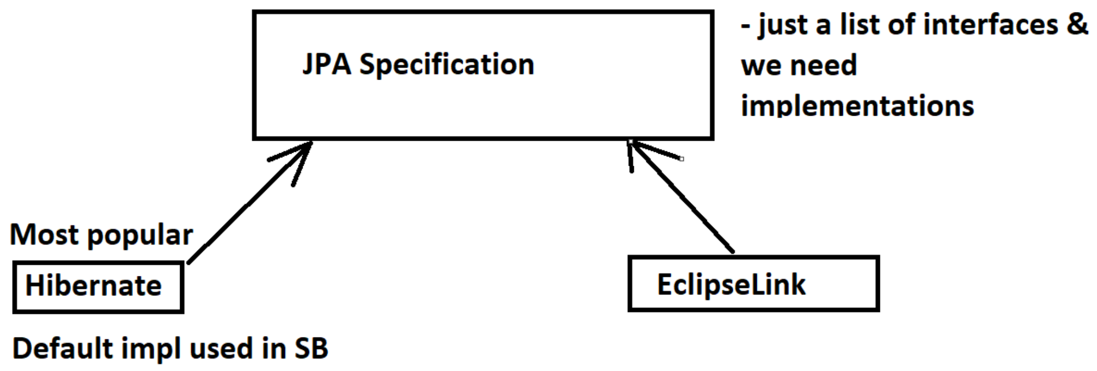
- What we do is Map java class to the table
- We have set one-to-one mapping between the fields and actual columns in database
- We can set up this mapping via configuration file using XML, but we are going to use Java Annotations.

What is JPA?

- Jakarta-Persistence-API …………Previously known as Java Persistence API
- Standard API for Object-to-Relational mapping

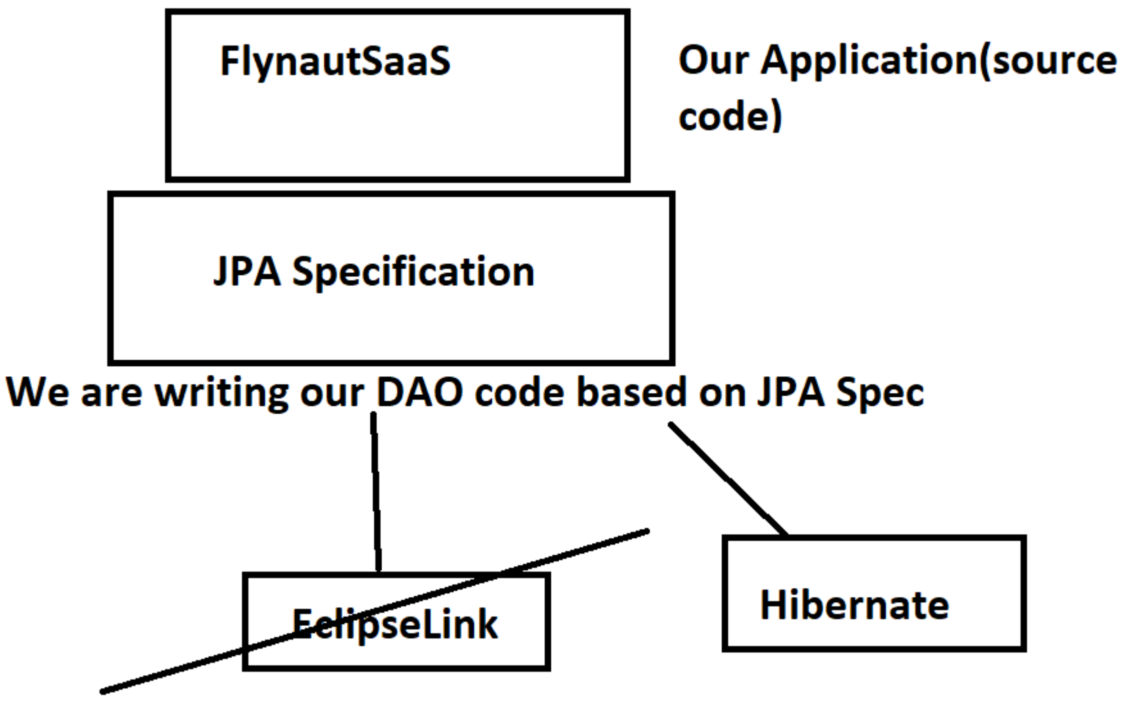
- It is only a Specification
- Defines a set of interfaces
- It requires the implementation to make it usable.

JPA vendor implementation



What are the benefits of using JPA?

- By having standard API, we are not bound/locked to vendor implementations
- We can switch vendor implementations
- Ex. If vendor EclipseLink is not supporting then we can switch to another vendor


FlynautSaaS
Our Application(source code)
JPA Specification
We are writing our DAO code based on JPA Spec
EclipseLink
Hibernate

Simply by changing the configuration we can change the vendor

2 ways to interact with database

1. entityManager
2. JpaRepository

Quick Example:

Saving java object with JPA

// Create a java object

Student theStudent = new Student(“Krishna”, “Jain”, “kj@gmail.com”)

// Saving it to db

entityManager.persist(theStudent);

- BTS hibernate is the implementation of JPA
  But here JPA with the hibernate does all the work for us in background.

- Retrieving the java object with JPA
  //create a java object
  //save it to db
  //now retrieving from the db using primary key
  int theId = 1;
  Student myStudent = entityManager.find(Student.class,theId);

- JPA/Hibernate
  Create Object
  Read Object
  Update Object
  Delete Object

Setting up the project

1. Spring Initializer
2. Dependencies: MySQL driver, Spring data JPA,web

AutoConfiguration:

- Sb will load DB connection information from application.properties

In application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/student_trac
ker
spring.datasource.username=springstudent
spring.datasource.password=springstudent
```

Entity class:

Java class that is mapped to db table

Musts:

1. It must be annotated with @Entity
2. It must have a public or protected no-arg constructor

Java annotations:

Step 1: Map class to the database table

Step 2: Map fields to db columns

Step 1:

@Entity

@Table(name=”student”)

public class Student{…}

Step2:

```
@Entity
@Table(name=”student”)
public class Student{
        @Id
        @GeneratedValue(strategy=GenerationType.IDENTITY)
        @Column(name= “id”)
        private int id;
        @Column(name=”first_name”)
        private String firstName;


}
```

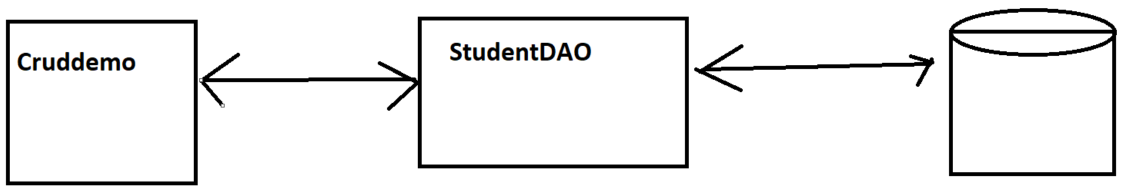
GenerationTypes:

AUTO

IDENTITY: assign a primary key using database identity column

TABLE
SEQUENCE

- Student Data Access Object
- Responsible for interacting with db
- This is a design pattern : Data Access Object(DAO)



Our DAO will have a number of methods

1. Save(…) -> for saving a student
2. findById(…)
3. findAll(…)
4. update()
5. delete()
6. deleteAll()

DEV process

StudentDAO

1. Define DAO interface
2. Define DAO implementation
- Inject Entity Manager

Spring @Transactional

-Automatically begin and end the transaction for our code

No need to do it explicitly in our code.