JPA Introduction

Java (Jakarta) Persistence API

Overview

Topics Covered:

- What is JPA?
- JPA vs Classic Hibernate
- persistence.xml Configuration
- EntityManager Usage
- Core JPA Methods: persist(), find(), remove()

After this guide and related exercises, you should be able to implement JPA in your project, and create, retrieve, and remove data from a database.

What is JPA?

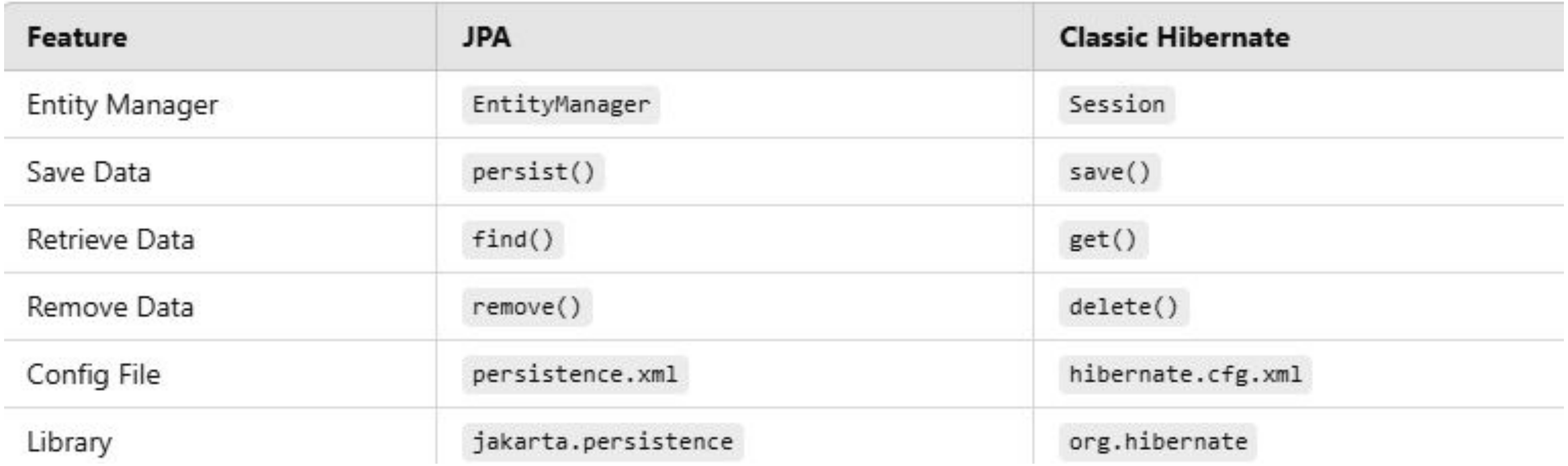
The Java Persistence API (JPA) is a Java EE (Jakarta) specification for managing relational data in enterprise applications.

JPA Implementations:

- eclipselink
- openJPA (apache software foundation)
- hibernate (JBOSS)

Although JPA is similar to Hibernate, JPA is an official standard, whereas classic Hibernate was a non-standard ORM solution. Hibernate now includes a JPA implementation.

JPA vs Classic Hibernate



Feature	JPA	Classic Hibernate
Entity Manager	`EntityManager`	`Session`
Save Data	`persist()`	`save()`
Retrieve Data	`find()`	`get()`
Remove Data	`remove()`	`delete()`
Config File	`persistence.xml`	`hibernate.cfg.xml`
Library	`jakarta.persistence`	`org.hibernate`

persistence.xml

The xml file, persistence.xml is placed in a folder called META-INF under the src folder

This file is a configuration file and has all the connections to the database

```
<property name="hibernate.connection.driver_class"
                    value="org.apache.derby.jdbc.ClientDriver"/>
 <property name="hibernate.connection.url"
              value="jdbc:derby://localhost:50000/hibernate"/>
```

persistence.xml-Transaction Type Options

```
<persistence-unit name="databaseConfig"
                    transaction-type="RESOURCE_LOCAL">
```

This is in the persistence.xml file, which gets a name and transaction-type.

The value of this transaction-type can be JTA (Java Transaction API) or RESOURCE_LOCAL.

RESOURCE_LOCAL is typically used in standalone Java applications, such as desktop applications or small-scale applications.

JTA transactions are typically used in enterprise applications deployed on Java EE application servers (such as WildFly, WebLogic, or GlassFish) where there is a need for distributed transactions across multiple transactional resources.

persistence.xml, Additional Configuration Options

We can have these settings in the file:

```
<property name="hibernate.show_sql" value="false" />

<property name="hibernate.format_sql" value="false" />

<property name="hibernate.hbm2ddl.auto" value="create" />
```

persistence.xml, Additional Configuration Options

Explanation of hibernate.hbm2ddl.auto values:

create → Drops and recreates tables every time the application runs.

update → Updates schema without losing data.

EntityManager and Transactions

The EntityManager is used to interact with the database. Transactions are managed using EntityTransaction.

The transaction is created by getTransaction method in EntityManager class

```
EntityManagerFactory emf =
             Persistence.createEntityManagerFactory("databaseConfig");
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();

// The code that interacts with the database will be written here.

tx.commit();
em.close();
```

Required Imports

These are the libraries you need to import:

```java
import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.EntityTransaction;
import jakarta.persistence.Persistence;
```

Core JPA Methods

Hibernate	JPA
session.save()	em.persist()
session.get()	em.find()
session.delete()	em.remove()

Persisting data

An example of creating objects using JPA:

```
tx.begin();

Tutor t1 = new Tutor("ABC123", "Teacher 1", 290000);
em.persist(t1);

Student s1 = new Student("Student1", "1-STU-2019");
t1.addStudentToTeachingGroup(s1);

em.persist(s1);

tx.commit();
em.close();
```

find method

find method in EntityManager will retrieve an object by its id.

```
Tutor tutor = em.find(Tutor.class, 1);
System.out.println(tutor);
```

find() is the same as get() in hibernate.

It gets two parameters: class name and the id of the object that we want to retrieve.

remove method

To remove an object from the table, we call the remove method in EntityManager.

For example:

```
Student student = em.find(Student.class, 4);
em.remove(student);
```

What is next?

In the next slide, we will look at the cascading.

Entity relationships often depend on the existence of another entity. For example, the Student - Tutor relationship.

To establish a dependency between related entities, JPA provides jakarta.persistence.CascadeType enumerated types that define the cascade operations.