08. Labwerk JPA

## JPA & Hibernate

In dit artikel bespreken we hoe we Java Persistence API kunnen gebruiken om CRUD-bewerkingen tegen de database uit te voeren. In dit voorbeeld gebruiken we de *MySQL* database.

### Wat zijn CRUD operaties in JPA?

De CRUD staat voor **Create, Retrieve (Lees), Update,** en **Delete** operaties op een Entiteit.

In JPA is de [EntityManager een interface] die wordt geassocieerd met een persistentiecontext. Het wordt gebruikt voor interactie met de persistentiecontext. Hier zijn enkele belangrijke methoden van [EntityManager](http://www.javaguides.net/2018/12/jpa-entitymanager-interface-with-example.html), em, die worden gebruikt om de CRUD-bewerkingen uit te voeren.

* *persist()*: Deze methode wordt gebruikt om een beheerde Entiteit te persisteren.
* *remove()*: Deze methode wordt gebruikt om een Entiteit uit de persistentiecontext te verwijderen.
* *merge()*: Deze methode wordt gebruikt om de huidige status van de Entiteit op te slaan in de persistentiecontext.
* *find()*: Deze methode wordt gebruikt om een Entiteit te laden op basis van zijn primaire sleutel.

De [Java Persistence API] is een Java-specificatie voor het beheren, persisteren en benaderen van gegevens tussen objecten en relationele databases. [Hibernate] is een ORM (Object Relational Mapping) tool die JPA specificaties implementeert. In dit voorbeeld gebruiken we Hibernate als JPA provider.

In deze post laten we zien hoe u een eenvoudige JPA-applicatie kunt maken of configureren met [Hibernate].

### Gebruikte technologieën en instrumenten

* Hibernate 5.3.7.Final
* JPA 2.2
* IDE - Eclipse Noen
* Maven 3.5.3
* JavaSE 1.8
* MySQL - 8.0.13

Laten we stap voor stap beginnen met ontwikkelen Hibernate-appliatie met behulp van Maven als project management en build tool.

### Ontwikkelingsstappen

1. Maak een eenvoudig Maven Project
2. Project Directory Structuur
3. Voeg jar afhankelijkheden toe aan pom.xml
4. Aanmaken van de JPA entiteitsklasse (Persistent class)
5. JPA CRUD Operaties
6. Een JPA configuratiebestand maken
7. Een JPA helper klasse maken
8. De hoofdklasse maken en een toepassing starten

### 1. Maak een eenvoudig Maven project

### 2. Structuur van de projectmap

De mapstructuur van het project ter referentie -

src/main/java - bevat alle Java broncode  
src/main/resources - bevat alle bronbestanden zoals XML-bestanden, afbeeldingen, enz.  
src/test/java - bevat alle testbroncode  
src/test/resources - bevat alle test-bronbestanden zoals XML-bestanden, afbeeldingen, enz.  
pom.xml - bevat alle Maven-afhankelijkheden en configuratie

### 3. Voeg jar afhankelijkheden toe aan pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>be.intecbrussel.notebooks</groupId>  
 <artifactId>jpa-crud</artifactId>  
 <version> 0.1.0-SNAPSHOT<version>  
 <packaging>jar</packaging>  
   
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>11</maven.compiler.source>  
 <maven.compiler.target>11</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.13</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>6.0.0.Alpha1</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <sourceDirectory>src/main/java</sourceDirectory>  
 <plugins>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>11</source>  
 <target>11</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>  
Copy

### 4. Creating the JPA Entity Class(Persistent class)

Het onderstaande diagram toont de *Student* JPA entiteit die is gekoppeld aan een database *studenten* tabel.

Wel, laten we een *Student* entiteitsklasse aanmaken onder *be.intecbrussel.notebooks.jpa.entity* pakket als volgt.

package be.intecbrussel.notebooks.jpa.entity;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@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;  
  
 @Column(name = "last\_name")  
 private String lastName;  
  
 @Column(name = "email")  
 private String email;  
  
 public Student() {  
  
 }  
  
 public Student(String firstName, String lastName, String email) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.email = email;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 @Override  
 public String toString() {  
 return "Student [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", email=" + email + "]";  
 }  
}  
  
Copy

### 5. JPA CRUD Operaties

Laten we de *EntityManager* interface methoden gebruiken om een CRUD Operatie tegen een database te maken.

package be.intecbrussel.notebooks.jpa;  
  
import javax.persistence.EntityManager;  
import javax.persistence.EntityTransaction;  
  
import be.intecbrussel.notebooks.jpa.entity.Student;  
import be.intecbrussel.notebooks.jpa.util.JPAUtil;  
  
/\*\*  
 \* JPA CRUD Operations  
 \* @author Ramesh Fadatare  
 \*  
 \*/  
public class CRUDOperations {  
 public void insertEntity() {  
 EntityManager entityManager = JPAUtil.getEntityManagerFactory().createEntityManager();  
 EntityTransaction entityTransaction = entityManager.getTransaction();  
 entityTransaction.begin();  
   
 Student student = new Student("Ramesh", "Fadatare", "rameshfadatare@javaguides.com");  
   
 entityManager.persist(student);  
 entityManager.getTransaction().commit();  
 entityManager.close();  
 }  
  
 public void findEntity() {  
 EntityManager entityManager = JPAUtil.getEntityManagerFactory().createEntityManager();  
 entityManager.getTransaction().begin();  
 Student student = entityManager.find(Student.class, 1);  
  
 System.out.println("student id :: " + student.getId());  
 System.out.println("student firstname :: " + student.getFirstName());  
 System.out.println("student lastname :: " + student.getLastName());  
 System.out.println("student email :: " + student.getEmail());  
  
 entityManager.getTransaction().commit();  
 entityManager.close();  
 }  
  
 public void updateEntity() {  
 EntityManager entityManager = JPAUtil.getEntityManagerFactory().createEntityManager();  
 entityManager.getTransaction().begin();  
  
 Student student = entityManager.find(Student.class, 1);  
  
 System.out.println("student id :: " + student.getId());  
 System.out.println("student firstname :: " + student.getFirstName());  
 System.out.println("student lastname :: " + student.getLastName());  
 System.out.println("student email :: " + student.getEmail());  
  
 // The entity object is physically updated in the database when the transaction  
 // is committed  
 student.setFirstName("Ram");  
 student.setLastName("jadhav");  
 entityManager.getTransaction().commit();  
 entityManager.close();  
 }  
  
 public void removeEntity() {  
 EntityManager entityManager = JPAUtil.getEntityManagerFactory().createEntityManager();  
 entityManager.getTransaction().begin();  
  
 Student student = entityManager.find(Student.class, 1);  
   
 System.out.println("student id :: " + student.getId());  
 System.out.println("student firstname :: " + student.getFirstName());  
 System.out.println("student lastname :: " + student.getLastName());  
 System.out.println("student email :: " + student.getEmail());  
  
 entityManager.remove(student);  
 entityManager.getTransaction().commit();  
 entityManager.close();  
 }  
}  
Copy

### 6. Create a JPA configuration file

Maak een XML-bestand met de naam *persistence.xml* onder de map *src/main/java/META-INF* en schrijf daarin de volgende code.

<persistence  
 xmlns="http://xmlns.jcp.org/xml/ns/persistence"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence  
 http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"  
 version="2.1">  
 <persistence-unit name="PERSISTENCE">  
 <description> Hibernate JPA DBConfig</description>  
 <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>  
 <class>be.intecbrussel.notebooks.jpa.entity.Student</class>  
 <properties>  
 <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver" />  
 <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/hibernate\_db" />  
 <property name="javax.persistence.jdbc.user" value="root" />  
 <property name="javax.persistence.jdbc.password" value="toor" />  
 <property name="hibernate.show\_sql" value="true" />  
 <property name="hibernate.hbm2ddl.auto" value="create-drop" />  
 </properties>  
 </persistence-unit>  
</persistence>  
  
Copy

### 7. Maak een JPA helper klasse

Maak een hulpklasse om een JPA *EntityManagerFactory* op te starten.

package be.intecbrussel.notebooks.jpa.util;  
  
import javax.persistence.EntityManagerFactory;  
import javax.persistence.Persistence;  
  
public class JPAUtil {  
  
 private static final String PERSISTENCE\_UNIT\_NAME = "PERSISTENCE";  
 private static EntityManagerFactory factory;  
  
 public static EntityManagerFactory getEntityManagerFactory() {  
 if (factory == null) {  
 factory = Persistence.createEntityManagerFactory(PERSISTENCE\_UNIT\_NAME);  
 }  
 return factory;  
 }  
  
 public static void shutdown() {  
 if (factory != null) {  
 factory.close();  
 }  
 }  
}  
  
Copy

### 8. Maak een hoofdklasse en draai een applicatie

Hier is de hoofdklasse om het *student* object te persisteren met behulp van de EntityManager#persist methode.

MainApp.java

package be.intecbrussel.notebooks.jpa;  
  
public class App {  
 public static void main(String[] args) {  
  
 CRUDOperations crudOperations = new CRUDOperations();  
 crudOperations.insertEntity();  
 crudOperations.findEntity();  
 crudOperations.updateEntity();  
 crudOperations.removeEntity();  
  
 }  
}  
  
Copy

### 9. Wat gebeurt er achter de schermen?

Hibernate:   
create table student (  
 id integer not null auto\_increment,  
 email varchar(255),  
 first\_name varchar(255),  
 last\_name varchar(255),  
 primary key (id)  
) engine=InnoDB  
  
Hibernate:  
insert   
into  
 student  
 (email, first\_name, last\_name)   
values  
 (?, ?, ?)  
  
student id :: 1  
student firstname :: John  
student lastname :: Doe  
student email :: john@doe.com  
  
Hibernate:  
update  
 student   
set  
 email=?,  
 first\_name=?,  
 last\_name=?   
where  
 id=?  
  
student id :: 1  
student firstname :: Nikola  
student lastname :: Doe  
student email :: nikola@doe.com  
  
Hibernate:  
delete  
from  
 student   
where  
 id=?  
  
Copy

### 10. Bronnen

[1] [Jakarta API Oracle Website](https://docs.oracle.com/javaee/7/tutorial/persistence-intro001.htm#BNBQA)

[2] [Jakarta Persistence API](https://jakarta.ee/specifications/persistence/2.2/apidocs/)

[3] [Hibernate ORM](https://hibernate.org/orm/)

[4] [Hibernate ORM 5.4.30.Final API](https://docs.jboss.org/hibernate/orm/5.4/javadocs/)

[5] [Hibernate ORM 5.4.30.Final User Guide](https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html)