JakartaEE (JPA)

pom.xml

packaging and properties

```xml
<packaging>war</packaging>

<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <jakarta.version>8.0.0</jakarta.version>
    <junit-jupiter.version>5.5.2</junit-jupiter.version>
</properties>
```

dependencies

```xml
<dependencies>
    <dependency>
        <groupId>jakarta.platform</groupId>
        <artifactId>jakarta.jakartaee-api</artifactId>
        <version>${jakarta.version}</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>${junit-jupiter.version}</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.assertj</groupId>
        <artifactId>assertj-core</artifactId>
        <version>3.14.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.derby</groupId>
        <artifactId>derbyclient</artifactId>
        <version>10.14.2.0</version>
    </dependency>
    <dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>javax.persistence</artifactId>
        <version>2.2.1</version>
    </dependency>
    <dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>org.eclipse.persistence.jpa</artifactId>
        <version>2.6.8</version>
    </dependency>
</dependencies>
```

model.java

```
@Entity
@Table(name = "MY_MODEL")
@NamedQueries({
        @NamedQuery(name = "Person.findAll", query = "select p from Person p"),
        @NamedQuery(name = "Person.findByName", query = "select p from Person p where
p.name = :NAME")
})
public class Model {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

    @ManyToOne
    private AnderesModel anderesModel;
}
```

assoziativTabelle.java

```
@ManyToOne
private Person person;
@ManyToOne
private Hobby hobby;
```

RestConfig.java

```
@ApplicationPath("api")
public class RestConfig extends Application {
}
```

InitBean.java

```
@ApplicationScoped
public class InitBean {

    @PersistenceContext
    EntityManager em;

    @Transactional
    private void init(@Observes @Initialized(ApplicationScoped.class) Object init) {

        em.persist(new Person("Susi", "Leonding"));

        em.persist(new PerformsHobby(
                em.find(Person.class, 1L),
                em.find(Hobby.class, 1L)
        ));
    }
}
```

Read CSV https://stuetzpunkt.wordpress.com/2016/12/28/how-to-access-file-in-resources-folder-javaee/

```
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(getClass()
.getResourceAsStream("/data.csv")));

            // erste Zeile überspringen falls es Header gibt
            br.readLine();

            String line;
            while ((line = br.readLine()) != null) {
                String[] row = line.split(";");
                // row[0] -> 1. Spalte aus CSV
                // row[1] -> 2. Spalte aus CSV
                // row[2] -> 3. Spalte aus CSV

                List<Weekday> weekDays = this.em
                        .createNamedQuery("Weekday.getByDay", Weekday.class)
                        .setParameter("DAY", row[0])
                        .getResultList();

                if (weekDays.size() != 1) {
                    currentDay = new Weekday(row[0]);
                    this.entityManager.persist(currentDay);
                } else {
                    currentDay = weekDays.get(0);
                }

                this.entityManager.persist(new Meal(row[1], currentDay));
                this.entityManager.persist(new Meal(row[2], currentDay));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Get Data from external REST

```java
@NamedQuery(name = "Driver.findByName", query = "select d from Driver d where d.name =
:NAME")
@PersistenceContext
EntityManager em;
public static final String RESULTS_ENDPOINT = "http://vm90.htl-leonding.ac.at/results
";
private Client client = ClientBuilder.newClient();
private WebTarget target = client.target(RESULTS_ENDPOINT);
public void readResultsFromEndpoint() {
  Response response = this.target.request(MediaType.APPLICATION_JSON).get();
  JsonArray payload = response.readEntity(JsonArray.class);
  persistResult(payload);
}
@Transactional
void persistResult(JsonArray resultsJson) {
  System.out.println(resultsJson.get(0).asJsonObject().getString("driverFullName"));
  System.out.println(resultsJson.get(0).asJsonObject().getInt("position"));
  for (JsonValue jsonValue : resultsJson) {
  String name = jsonValue.asJsonObject().getString("driverFullName");
  int pos = jsonValue.asJsonObject().getInt("position");
  int raceNo = jsonValue.asJsonObject().getInt("reaceNo");
  em.persist(new Result(em.find(Race.class,raceNo), pos, (java.sql.Driver) em
.createNamedQuery("Driver.findByName", Driver.class).setParameter("NAME", name)));
  }
}
```

DatabaseEndpoint.java

```java
@Path("")
public class DatabaseEndpoint {

    @PersistenceContext
    EntityManager em;
}
```

GET

```
    @GET
    @Path("getAll")
    @Produces(MediaType.APPLICATION_JSON)
    public List<Person> getAll() {
        return em
            .createNamedQuery("Person.findAll", Person.class)
            .getResultList();
    }

    @GET
    @Path("getById/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response getById(@PathParam("id") long id) {
        return Response.ok(em.find(Person.class, id)).build();
    }

    @GET
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Vehicle getOpel(@PathParam("id") long id) {
        return em.find(Vehicle.class,id);
    }
```

POST

```
    @POST
    @Path("postPerson")
    @Produces(MediaType.TEXT_PLAIN)
    @Consumes(MediaType.APPLICATION_JSON)
    public Response getJsonObject(JsonObject json) {
        String text = json.getString("name");
        return Response.ok(text).build();
    }
```

DELETE

```
    @Transactional
    @DELETE
    @Path("delete/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response delete(@PathParam("id") long id) {
        Person person = em.find(Person.class, id);
        em.remove(person);
        return Response.ok(person).build();
    }
```

PUT

```
    @Transactional
    @PUT
    @Path("put/{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Response updateById(@PathParam("id") long id, @QueryParam("name") String
name) {
        Person person = em.find(Person.class, id);
        person.setName(name);
        em.persist(person);
        return Response.ok(person).build();
    }
```

resources/META-INF/persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">
    <persistence-unit name="myPU" transaction-type="JTA">
        <jta-data-source>java:jboss/datasources/dbDS</jta-data-source>
        <properties>
            <!--
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
            <property name="hibernate.transaction.flush_before_completion"
value="true"/>
            <property name="hibernate.dialect"
value="org.hibernate.dialect.DerbyDialect"/>
            -->
            <property name="javax.persistence.schema-generation.database.action"
value="drop-and-create"/>
            <!--
            <property name="javax.persistence.schema-generation.scripts.action"
value="drop-and-create"/>
            <property name="javax.persistence.schema-generation.scripts.create-target"
value="sampleCreate.ddl"/>
            <property name="javax.persistence.schema-generation.scripts.drop-target"
value="sampleDrop.ddl"/>
            <property name="javax.persistence.sql-load-script-source"
value="insert.sql"/>
            -->
        </properties>
    </persistence-unit>
</persistence>
```

rest-requests/Person.http

```
### Send GET request for all persons
GET http://localhost:8080/person/api/person
Accept: application/json

###

GET http://localhost:8080/person/api/person/name?name=Susi
Accept: application/json

###

POST http://localhost:8080/person/api/person/jsondemo
Content-Type: application/json

{
  "name": "Sepp",
  "city": "New York"
}

###

DELETE http://localhost:8080/jpawithrest/api/person/delete/2
Content-Type: application/json

###

PUT http://localhost:8080/jpawithrest/api/person/put/1?name=Karl
Content-Type: application/json

###
```

JavaSE (assertJ)

pom.xml

packaging and properties

```
<packaging>jar</packaging>

<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <junit.jupiter.version>5.5.1</junit.jupiter.version>
</properties>
```

dependencies

```
<<dependencies>

        <dependency>
            <groupId>org.eclipse.persistence</groupId>
            <artifactId>javax.persistence</artifactId>
            <version>2.2.1</version>
        </dependency>
        <dependency>
            <groupId>org.eclipse.persistence</groupId>
            <artifactId>org.eclipse.persistence.jpa</artifactId>
            <version>2.6.8</version>
        </dependency>
        <dependency>
            <groupId>org.apache.derby</groupId>
            <artifactId>derbyclient</artifactId>
            <version>10.14.2.0</version>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
            <version>3.14.0</version>
            <scope>test</scope>
        </dependency>

    </dependencies>
```

Test.java

```
    static EntityManager em;

    @BeforeAll
    private static void init() {
        em = Persistence
                .createEntityManagerFactory("myPU")
                .createEntityManager();
    }
```

```java
    @Test
    void test01DatabaseConnection() {
        Vehicle opel = new Vehicle("Opel", "Kadett");
        em.getTransaction().begin();
        em.persist(opel);
        opel.setType("Commodore");
        em.getTransaction().commit();
    }

    @Test
    void test02readOneVehicle() {
        Vehicle opel = em.find(Vehicle.class,1L);
        assertThat(opel.getBrand(),is("Opel"));
        assertThat(opel.getType(),is("Kadett"));
    }
```

resources/META-INF/persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">
    <persistence-unit name="myPU" transaction-type="RESOURCE_LOCAL">
        <class>at.htl.jpasedemo.model.Vehicle</class>
        <exclude-unlisted-classes>true</exclude-unlisted-classes>
        <properties>
            <property name="eclipselink.logging.level" value="FINE"/>
            <property name="eclipselink.target-database" value="DERBY"/>
            <!--
            <property name="javax.persistence.jdbc.driver"
                      value="org.apache.derby.jdbc.EmbeddedDriver"/>
            <property name="javax.persistence.jdbc.url"
                      value="jdbc:derby:memory:myDb;create=true"/>
            <property name="javax.persistence.jdbc.user" value=""/>
            <property name="javax.persistence.jdbc.password" value=""/>
            -->
            <property name="javax.persistence.jdbc.driver"
                      value="org.apache.derby.jdbc.ClientDriver"/>
            <property name="javax.persistence.jdbc.url"
                      value="jdbc:derby://localhost:1527/db;create=true"/>
            <property name="javax.persistence.jdbc.user" value="app"/>
            <property name="javax.persistence.jdbc.password" value="app"/>
            <property name="javax.persistence.schema-generation.database.action"
                      value="drop-and-create"/>
        </properties>
    </persistence-unit>
</persistence>
```