Hibernate, JPA – laboratorium

Uladzislau Tumilovich

II. Basics

1) Udało się uruchomić server i do nie go podłączyć się.

Command Prompt - ij

```
C:\Users\vladi\Desktop\hibernate\db-derby-10.14.2.0-bin\db-derby-10.14.2.0-bin\bin>ij
ij version 10.14
ij> connect 'jdbc:derby://127.0.0.1/UTumilovichJPA';
ij> show tables
> ;
TABLE_SCHEM         |TABLE_NAME                    |REMARKS
------------------------------------------------------------------------
SYS                 |SYSALIASES                    |
SYS                 |SYSCHECKS                     |
SYS                 |SYSCOLPERMS                   |
SYS                 |SYSCOLUMNS                    |
SYS                 |SYSCONGLOMERATES              |
SYS                 |SYSCONSTRAINTS                |
SYS                 |SYSDEPENDS                    |
SYS                 |SYSFILES                      |
SYS                 |SYSFOREIGNKEYS                |
SYS                 |SYSKEYS                       |
SYS                 |SYSPERMS                      |
SYS                 |SYSROLES                      |
SYS                 |SYSROUTINEPERMS               |
SYS                 |SYSSCHEMAS                    |
SYS                 |SYSSEQUENCES                  |
SYS                 |SYSSTATEMENTS                 |
SYS                 |SYSSTATISTICS                 |
SYS                 |SYSTABLEPERMS                 |
SYS                 |SYSTABLES                     |
SYS                 |SYSTRIGGERS                   |
SYS                 |SYSUSERS                      |
SYS                 |SYSVIEWS                      |
SYSIBM              |SYSDUMMY1                     |
APP                 |PRODUCT                       |

24 rows selected
ij>
```

2) Następnie stworzyłem klasę Product z polami ProductId, ProductName oraz UnitsOnStock

```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int ProductId;
    public String ProductName;
    public int UnitsOnStock;

    public Product(String productName, int unitsOnStock) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsOnStock;
    }

    public Product() {
    }
}
```

3) Uzupełniłem potrzebne property w konfiguracji hibernate'a

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/UTumilovichJPA</property>
        <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
        <property name="format_sql">true</property>
        <property name="show_sql">true</property>
        <property name="use_sql_comments">true</property>
        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
    </session-factory>
</hibernate-configuration>
```

4) I na koniec tego punktu sworzyłem przykładowy produkt i dodałem go do bazy

Product.java | hibernate.cfg.xml | Main.java

```java
import ...

public class Main {
    private static final SessionFactory ourSessionFactory;

    static {
        try {
            Configuration configuration = new Configuration();
            configuration.configure();

            ourSessionFactory = configuration.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session getSession() throws HibernateException {
        return ourSessionFactory.openSession();
    }

    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        Transaction transaction = session.beginTransaction();
        session.save(new Product( productName: "Laptop",  unitsOnStock: 10));
        transaction.commit();
        try {
            System.out.println("querying all the managed entities...");
            final Metamodel metamodel = session.getSessionFactory().getMetamodel();
            for (EntityType<?> entityType : metamodel.getEntities()) {
                final String entityName = entityType.getName();
                final Query query = session.createQuery( s: "from " + entityName);
                System.out.println("executing: " + query.getQueryString());
                for (Object o : query.list()) {
                    System.out.println("  " + o);
                }
            }
        } finally {
            session.close();
        }
    }
}
```

5) Wygląd tabeli z datagrip’a

APP.PRODUCT [UTumilovichJPA]

1 row | Tx: Auto | DDL

<Filter Criteria>

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Laptop	10

PRODUCT

PRODUCTID	int
PRODUCTNAME	varchar(255)
UNITSONSTOCK	int

IV.

1) Stworzyłem klasę Supplier z polami SupplierID, CompanyName, Street, City

Product.java | Supplier.java | hibernate.cfg.xml | Main.java

```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int SupplierId;
    public String CompanyName;
    public String Street;
    public String City;

    public Supplier(String companyName, String street, String city) {
        this.CompanyName = companyName;
        this.Street = street;
        this.City = city;
    }

    public Supplier() {
    }
}
```

2) Następnie zmodyfikowałem klasę Product, dodając do niej pole Supplier oraz konstruktorze

Product.java | Supplier.java | hibernate.cfg.xml | Main.java

```java
import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int ProductId;
    public String ProductName;
    public int UnitsOnStock;

    @ManyToOne
    public Supplier supplier;

    public Product(String productName, int unitsOnStock) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsOnStock;
    }

    public Product() {
    }

    public Product(String productName, int unitsOnStock, Supplier supplier) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsOnStock;
        this.supplier = supplier;
    }

    public void setSupplier(Supplier supplier) {
        this.supplier = supplier;
    }
}
```

3) Do pliku konfiguracyjnemu hibernate.cfg.xml dodałem Supplier’a

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/UTumilovichJPA</property>
        <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
        <property name="format_sql">true</property>
        <property name="show_sql">true</property>
        <property name="use_sql_comments">true</property>
        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
        <mapping class="Supplier"></mapping>
    </session-factory>
</hibernate-configuration>
```

4) I na koniec tego zadania zmieniłem klasę Main dla uruchomienia i sprawdzania działalności programu

```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction transaction = session.beginTransaction();
    Supplier supplier = new Supplier( companyName: "Somebody",  street: "Somewhere",  city: "Anywhere");
    Product prodToUpdate = session.get(Product.class,  serializable: 1);
    prodToUpdate.setSupplier(supplier);
    session.save(supplier);
    session.save(prodToUpdate);
    transaction.commit();
    try {
        System.out.println("querying all the managed entities...");
        final Metamodel metamodel = session.getSessionFactory().getMetamodel();
        for (EntityType<?> entityType : metamodel.getEntities()) {
            final String entityName = entityType.getName();
            final Query query = session.createQuery( s: "from " + entityName);
            System.out.println("executing: " + query.getQueryString());
            for (Object o : query.list()) {
                System.out.println("  " + o);
            }
        }
    } finally {
        session.close();
    }
}
```

5) Wygląd bazy z datagrip’a

console_2 [UTumilovichJPA]

```
SELECT * FROM PRODUCT;
```

Output | APP.PRODUCT

1 row

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID
1	1	Laptop	10	3

console_2 [UTumilovichJPA]

```
SELECT * FROM SUPPLIER;
```

Output | APP.SUPPLIER

2 rows

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Anywhere	Somebody	Somewhere
2	3	Anywhere	Somebody	Somewhere