**Hibernate JPQL Maven MySQL log4j**

*последнее изменение 16 июля 2020 г.*

В руководстве Hibernate JPQL SELECT показано, как выполнять операторы JPQL SELECT в Hibernate. Мы используем базу данных MySQL.

**Спящий режим**

*Hibernate* - это инструмент объектно-реляционного сопоставления для языка программирования Java. Он обеспечивает основу для сопоставления объектно-ориентированной модели предметной области с реляционной базой данных.

Java Persistence Query Language (JPQL) - это платформенно-независимый объектно-ориентированный язык запросов, определенный как часть спецификации Java Persistence API (JPA). JPQL используется для выполнения запросов к сущностям, хранящимся в реляционной базе данных. JPQL вдохновлен SQL.

JPQL объектно-ориентированный. В JPQL мы работаем с сущностями и коллекцией сущностей, а в SQL мы работаем со столбцами и строками.

Оператор JPQL SELECT имеет следующий синтаксис:

SELECT {select clause}

FROM {from clause}

[WHERE {where clause}]

[ORDER BY {order by clause}]

[GROUP BY {group by clause}]

[HAVING {having clause}]

**Пример Hibernate JPQL SELECT**

Следующее приложение выполняет пару операторов JPQL SELECT.

SQL для создания таблицы cities в MySQL:

**cities\_mysql.sql**

DROP TABLE IF EXISTS cities;

CREATE TABLE cities(id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100), population INT);

INSERT INTO cities(name, population) VALUES('Bratislava', 432000);

INSERT INTO cities(name, population) VALUES('Budapest', 1759000);

INSERT INTO cities(name, population) VALUES('Prague', 1280000);

INSERT INTO cities(name, population) VALUES('Warsaw', 1748000);

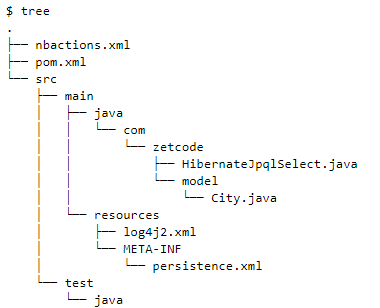
INSERT INTO cities(name, population) VALUES('Los Angeles', 3971000);

INSERT INTO cities(name, population) VALUES('New York', 8550000);

INSERT INTO cities(name, population) VALUES('Edinburgh', 464000);

INSERT INTO cities(name, population) VALUES('Berlin', 3671000);

Структура проекта:



Файл сборки Maven. Это mysql-connector-java драйвер MySQL,  hibernate-core обеспечивающий базовую функциональность Hibernate, а log4j-slf4j-impl также простой фасад регистрации в библиотеке Log4j2:

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>com.zetcode</groupId>

<artifactId>HibernateJpqlSelect</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.45</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.8.Final</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-slf4j-impl</artifactId>

<version>2.10.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>1.5.0</version>

<configuration>

<mainClass>com.zetcode.HibernateJpqlSelect</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

Файл log4j2.xml конфигурации для Log4j2:

**log4j2.xml**

<?xml version="1.0" encoding="UTF-8"?>

<Configuration>

<Appenders>

<Console name="Console" target="SYSTEM\_OUT">

<PatternLayout pattern="%d [%t] %-5level %logger - %m%n" />

</Console>

</Appenders>

<Loggers>

<!-- Log everything in Hibernate -->

<Logger name="org.hibernate" level="error" additivity="false">

<AppenderRef ref="Console" />

</Logger>

<!-- Log SQL statements -->

<Logger name="org.hibernate.SQL" level="error" additivity="false">

<AppenderRef ref="Console" />

</Logger>

<!-- Log custom packages -->

<Logger name="com.zetcode" level="error" additivity="false">

<AppenderRef ref="Console" />

</Logger>

<Root level="error">

<AppenderRef ref="Console" />

</Root>

</Loggers>

</Configuration>

Стандартный файл конфигурации в формате JPA - persistence.xml. Он должен быть включен в META-INF каталог внутри файла JAR, который содержит объектные компоненты.  Файл persistence.xml должен определить блок- сохраняемости с уникальным именем в текущем контекстном загрузчике классов. Атрибут provider определяет базовую реализацию JPA EntityManager.

**persistence.xml**

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.1" 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">

<persistence-unit name="my-pu" transaction-type="RESOURCE\_LOCAL">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

<properties>

<property name="javax.persistence.jdbc.url"

value="jdbc:mysql://localhost:3306/testdb?useSSL=false"/>

<property name="javax.persistence.jdbc.user" value="testuser"/>

<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>

<property name="javax.persistence.jdbc.password" value="test623"/>

<property name="hibernate.cache.provider\_class"

value="org.hibernate.cache.NoCacheProvider"/>

<property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect"/>

</properties>

</persistence-unit>

</persistence>

, где как я понял "my-pu" - это мы сами задаем имя блока-сохраняемости, чтобы потом вызвав его создать “emf”, а потом “em” (“entityManager”).

Мы предоставляем варианты конфигурации для подключения к MySQL. Установлен hibernate.dialect диалект MySQL Hibernate.

Сущность City:

**City.java**

package com.zetcode.model;

import java.util.Objects;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "cities")

public class City {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String name;

private int population;

public City() {

}

public City(String name, int population) {

this.name = name;

this.population = population;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getPopulation() {

return population;

}

public void setPopulation(int population) {

this.population = population;

}

@Override

public int hashCode() {

int hash = 7;

hash = 79 \* hash + Objects.hashCode(this.id);

hash = 79 \* hash + Objects.hashCode(this.name);

hash = 79 \* hash + this.population;

return hash;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final City other = (City) obj;

if (this.population != other.population) {

return false;

}

if (!Objects.equals(this.name, other.name)) {

return false;

}

return Objects.equals(this.id, other.id);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append("City{id=").append(id).append(", name=")

.append(name).append(", population=")

.append(population).append("}");

return builder.toString();

}

}

В следующим примере будут выполняется пять операторов JPQL SELECT:

**HibernateJpqlSelect.java**

package com.zetcode;

import com.zetcode.model.City;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

import javax.persistence.Query;

public class HibernateJpqlSelect {

private static final String PERSISTENCE\_UNIT\_NAME = "my-pu";

public static void main(String[] args) {

EntityManagerFactory emf = Persistence.createEntityManagerFactory(

PERSISTENCE\_UNIT\_NAME);

EntityManager entityManager = emf.createEntityManager();

try {

entityManager.getTransaction().begin();

String qlQuery = "SELECT c FROM City c";

Query query = entityManager.createQuery(qlQuery);

List<City> cities = query.getResultList();

cities.stream().forEach((x) -> System.out.println(x));

String qlQuery2 = "SELECT COUNT(c) FROM City c";

Query query2 = entityManager.createQuery(qlQuery2);

Long nOfCities = (Long) query2.getSingleResult();

System.out.printf("There are %d cities%n", nOfCities);

String qlQuery3 = "SELECT c.name FROM City c WHERE c.population=1759000";

Query query3 = entityManager.createQuery(qlQuery3);

List<String> names = query3.getResultList();

names.stream().forEach((x) -> System.out.println(x));

String qlQuery4 = "SELECT c FROM City c ORDER BY population DESC";

Query query4 = entityManager.createQuery(qlQuery4);

List<City> cities2 = query4.getResultList();

cities2.stream().forEach((x) -> System.out.println(x));

String qlQuery5 = "SELECT SUM(c.population) FROM City c";

Query query5 = entityManager.createQuery(qlQuery5);

Long popSum = (Long) query5.getSingleResult();

System.out.printf("Sum of city populations %d%n", popSum);

entityManager.getTransaction().commit();

} finally {

entityManager.close();

emf.close();

}

}

}

Результат:

$ mvn exec:java -q

City{id=1, name=Bratislava, population=432000}

City{id=2, name=Budapest, population=1759000}

City{id=3, name=Prague, population=1280000}

City{id=4, name=Warsaw, population=1748000}

City{id=5, name=Los Angeles, population=3971000}

City{id=6, name=New York, population=8550000}

City{id=7, name=Edinburgh, population=464000}

City{id=8, name=Berlin, population=3671000}

There are 8 cities

Budapest

City{id=6, name=New York, population=8550000}

City{id=5, name=Los Angeles, population=3971000}

City{id=8, name=Berlin, population=3671000}

City{id=2, name=Budapest, population=1759000}

City{id=4, name=Warsaw, population=1748000}

City{id=3, name=Prague, population=1280000}

City{id=7, name=Edinburgh, population=464000}

City{id=1, name=Bratislava, population=432000}

Sum of city populations 21875000