JPA with Hibernate 3.0

Lesson4- JPA Queries

Lesson Objectives

After completing this lesson, participants will be able to understand:

- JPQL
- JPA Query API
- Working with JPA Queries

Why JPQL?

Entity Manager’s find() method can be used to locate single entity only based on primary key value.

What if you want to load data based on complex criteria?

For example:

- Load employees residing in “Pune”
- Find orders placed between given two dates.
- Load and combine two entities data for reporting.

Solution ?

JAVA PERSISTENCE QUERY LANGUAGE!

What is JPQL?

JPQL is used to make queries against entities stored in a relational database.

JPQL is object-oriented as it automatically populates object with data received from database

Benefits of using JPQL:

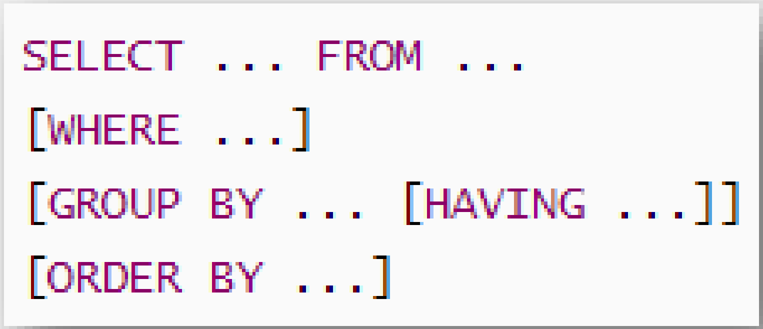
- JPQL is portable across implementer and database
- Uses SQL-like syntax to query entities, requires less effort to learn
- No manual conversion of row data into object and vice-versa

JPQL Query Syntax

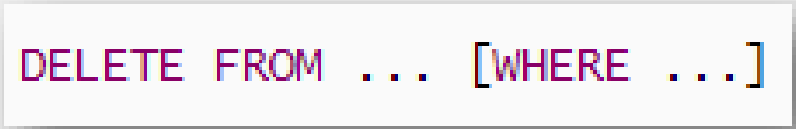
JPQL query syntax is similar to SQL syntax.

Difference between SQL and JPQL

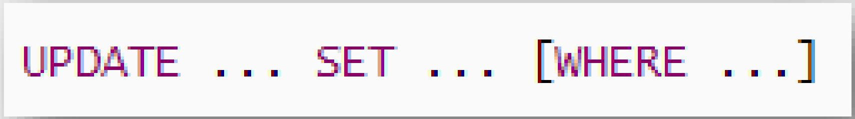
- JPQL works on Java Classes, Objects and Properties
- SQL works on Database tables, rows and columns



```
SELECT ... FROM ...
[WHERE ...]
[GROUP BY ... [HAVING ...]]
[ORDER BY ...]
```



```
DELETE FROM ... [WHERE ...]
```



```
UPDATE ... SET ... [WHERE ...]
```

JPA Query API

JPA provides two interfaces to query database:

- Query
- TypedQuery<T>

JPA 2 introduced TypedQuery interface which extends Query to query results in a type safe manner.

Below table list important methods of these interfaces:

Method	Purpose
getSingleResult()	Executes SELECT statement and returns a single object result
getResultList()	Executes SELECT statement and returns the query result as an list of objects.
executeUpdate()	Execute an UPDATE or DELETE statement.

Working with JPA Queries

Query work starts with an EntityManager, which serves as a factory for both Query and TypedQuery.

The EntityManager interface has an overloaded createQuery() method with the following signature:

- Query createQuery(java.lang.String query)
- TypedQuery<T> createQuery(java.lang.String query, T resultClass)

```
Query query = entityManager.createQuery("Select s FROM Student
s");
```

```
TypedQuery<Student> query = entityManager.createQuery
                            ("SELECT student from Student student",
Student.class);
```

Query Examples

The following queries to find single book with id ‘B13455’

```
String qStr = "SELECT book FROM Book book WHERE
book.id=‘B13455' ";
TypedQuery<Book> query = entityManager.createQuery(qStr ,
Book.class);
Book book = query.getSingleResult();
```

To list all books from store:

```
String qStr = "SELECT book FROM Book book ";
TypedQuery<Book> query = entityManager.createQuery(qStr,
Book.class);
List<book> bookList = query.getResultList();
```

Dynamic queries with parameters

JPA allows developer to create dynamic queries which are more efficient and can be built dynamically at runtime.

The following code retrieves a book object from the database by its title using named parameter “ptitle”:

```
String qStr = "SELECT book FROM Book book WHERE
book.title=:ptitle";
TypedQuery<Book> query =
entityManager.createQuery(qStr,Book.class);
query.setParameter("ptitle", “Introduction to JPA”);
Book book = query.getSingleResult();
```

Named Queries

JPA also provides a way for building static queries, as named queries, using the @NamedQuery and @NamedQueries annotations.

This approach is considered to be a good practice in JPA to prefer named queries over dynamic queries whenever possible.

Named queries are defined on Entity and created using createNamedQuery() method of EntityManager.

```
@Entity
@Table(name = "books")
@NamedQueries(
@NamedQuery(name = "getAllBooks", query = "SELECT book FROM
Book book"))
public class Book implements Serializable {  …………. }
```

4.3: Working with JPA queries

Demo

JPAQueries

Lab

Working with JPQL

Summary

In this lesson, you have learned about:

- What is JPQL
- JPA Query API
- Working with JPA queries

Review Question

Question 1: Which of the following method is used to create named query?

- EntityManager.createQuery()
- EntityManager.createNamedQuery()
- Both

Question 2: While working with JPQL, manual conversion of table row data into object is not needed.

- True/False