Create Query Method

What is a createQuery()?

- We have studied the get and post services and how to use the fundamental operations to create and read data from databases up to this point. However, the method leaves unclear how to obtain the data for all the items and use them for a variety of other tasks.
- To understand this we can use the createQuery() method, an inbuilt method of EntityManager which will be helpful in writing more complex queries.
- JPA EntityManager is used to access a database in a particular application. It is used to manage entity instances, to find entities by their primary key, and to query over all entities.
- createQuery() is a method supported by EntityManager, it creates an instance of Query for executing a (JPQL) Java Persistence query language statement. We will study JPQL queries in further lessons.

```
Session currentSession = entityManager.unwrap(Session.class);

//Pass your query, and name of class on which you want to perform
the query in createQuery method.
Query<Entity> query = currentSession.createQuery("Your
Query",Entity.class);

// Save all item in the list
List<Entity> entitiesList = query.getResultList();
```

Example:

Task: We will create an API in our application to fetch all the Item entities from the database.

ItemController.java

```
package com.cn.ecommerce.controller;

import com.cn.ecommerce.entity.Item;
import com.cn.ecommerce.service.ItemService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api")
public class ItemController {

    @Autowired
    ItemService itemService;

    @GetMapping(path = "/item/{id}")
    public Item getItemById(@PathVariable int id) {
        return itemService.getItemById(id);
    }
```

We will create a getAllItems() method in ItemController class to fetch all item entities.

```
@GetMapping(path = "/item/getAllItems")
Public List<Item> getAllItems(){
    return itemService.getAllItems();
}
```

ItemService.java

```
package com.cn.ecommerce.service;

import com.cn.ecommerce.dao.ItemDAO;
import com.cn.ecommerce.entity.Item;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class ItemService {

    @Autowired
    private ItemDAO itemDAO;

    @Transactional
    public Item getItemById(int id) {
       return itemDAO.get(id);
    }
```

We will now implement the service called by the controller in ItemService class.

```
@Transactional
public List<Item> getItem(){
  return itemDAO.get();
}
```

ItemDAOImpl.java

```
package com.cn.ecommerce.dao;

import com.cn.ecommerce.entity.Item;
import com.cn.ecommerce.entity.ItemDetails;
import org.hibernate.Session;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;
import java.util.List;

@Repository
public class ItemDAOImpl implements ItemDAO{

    @Autowired
    private EntityManager entityManager;

    @Override
    public Item get(int id) {
        Session currentSession = entityManager.unwrap(Session.class);
        Item item = currentSession.get(Item.class, id);
        return item;
    }
```

In the ItemDAOImpl class we will use createQuery() method to fetch all item entities from the database.

```
@Override
public List<Item> get() {
  Session currentSession = entityManager.unwrap(Session.class);
  Query<Item> query = currentSession.createQuery("from Item",
Item.class);
  List<Item> list = query.getResultList();
    return list;
}
```

What’s more to learn?

So, till now we have learned to use the createQuery() method to fetch all entities from the database, it can also be used to create dynamic queries. We can also pass variables in our queries and perform more operations:

- Update and delete operations on entities.
- Using aggregate functions like sum(), max() and min().
- Fetching records with pagination, like fetching records from 10th to 20th number.

To know more about their implementation, you can refer to this link.