Creating Queries Using the Java Persistence Query Language

The `EntityManager.createQuery` and `EntityManager.createNamedQuery`methods are used to query the datastore by using Java Persistence query language queries.

The `createQuery` method is used to create **dynamic queries**, which are queries defined directly within an application's business logic:

```
public List findWithName(String name) {
return em.createQuery(
    "SELECT c FROM Customer c WHERE c.name LIKE :custName")
    .setParameter("custName", name)
    .setMaxResults(10)
    .getResultList();
}
```

The `createNamedQuery` method is used to create **static queries**, or queries that are defined in metadata by using the `javax.persistence.NamedQuery`annotation. The `name` element of `@NamedQuery` specifies the name of the query that will be used with the `createNamedQuery` method. The `query` element of`@NamedQuery` is the query:

```
@NamedQuery(
    name="findAllCustomersWithName",
    query="SELECT c FROM Customer c WHERE c.name LIKE :custName"
)
```

Here's an example of `createNamedQuery`, which uses the `@NamedQuery`:

```
@PersistenceContext
public EntityManager em;
...
customers = em.createNamedQuery("findAllCustomersWithName")
    .setParameter("custName", "Smith")
    .getResultList();
```

Named Parameters in Queries

Named parameters are query parameters that are prefixed with a colon (`:`). Named parameters in a query are bound to an argument by the following method:

```
javax.persistence.Query.setParameter(String name, Object value)
```

In the following example, the `name` argument to the `findWithName` business method is bound to the `:custName` named parameter in the query by calling`Query.setParameter`:

```
public List findWithName(String name) {
    return em.createQuery(
        "SELECT c FROM Customer c WHERE c.name LIKE :custName")
        .setParameter("custName", name)
        .getResultList();
}
```

Named parameters are case-sensitive and may be used by both dynamic and static queries.

Positional Parameters in Queries

You may use positional parameters instead of named parameters in queries. Positional parameters are prefixed with a question mark (`?`) followed the numeric position of the parameter in the query. The `Query.setParameter(integer position, Object value)` method is used to set the parameter values.

In the following example, the `findWithName` business method is rewritten to use input parameters:

```
public List findWithName(String name) {
    return em.createQuery(
        "SELECT c FROM Customer c WHERE c.name LIKE ?1")
        .setParameter(1, name)
        .getResultList();
}
```

Input parameters are numbered starting from 1. Input parameters are case-sensitive, and may be used by both dynamic and static queries.