How To Call Native SQL Queries With JPA

Create dynamic native queries

The *EntityManager* interface provides the method *createNativeQuery* to create dynamic native queries.

```
Query q = em.createNativeQuery(
        "SELECT a.firstname, a.lastname FROM Author a");
List<Object[]> authors = q.getResultList();
```

Create named native queries

Named native queries are defined via annotations and require a name an SQL query.

```
@NamedNativeQuery(name = "selectAuthorNames",
query = "SELECT a.firstname, a.lastname FROM Author a")
```

The name of the named native query is then used with the *createNamedQuery* method to get an instance of the query wich can be used in the same way as a named JPQL query.

```
Query q = em.createNamedQuery("selectAuthorValue");
List<Object[]> authors = q.getResultList();
```

Parameter binding

The JPA standard supports only positional parameter bindings for native queries.

```
Query q = em.createNativeQuery(
        "SELECT a.firstname, a.lastname FROM Author a “
        + “WHERE a.id = ?");
q.setParameter(1, 1);
```

Hibernate also supports named parameter bindings.

```
Query q = em.createNativeQuery(
        "SELECT a.firstname, a.lastname FROM Author a “
        + “WHERE a.id = :id");
q.setParameter("id", 1);
```

Result handling

Native queries return *Object[]* by default which can be easily mapped into managed entities.

```
Query q = em.createNativeQuery(
                    "SELECT * FROM Author a", Author.class);
List<Author> authors = q.getResultList();
```

More complex mappings can be defined with SQL Result Mappings:

- Result Set Mapping: The Basics
- Result Set Mapping: Complex Mappings
- Result Set Mapping: Constructor Result Mappings
- Result Set Mapping: Hibernate specific features