Spring data: @Query Annotation

1. JPA with @Query, NamedQueries

Why?

- reuse existing JPQL
- use of complexe JPQL : join, grouping,...
- performance eager loading with “fetch” keyword

@Query Annotation

@Query Annotation

Reuse existing JPQL in your data access layer

Advanced query functionality

Eager loading control (“fetch”)

```
@Query("select m from Model m where m.name = ?1")
List<Model> queryByName(String name);
```



```
1  //modelJpaRepository.java
2
3
4      @Query("select m from Model m where m.price >= :lowest and m.price <= :h
5      List<Model> queryByPriceRangeAndWoodType(@Param("lowest") BigDecimal low
6                                               @Param("highest") BigDecimal hi
7                                               @Param("wood") String wood);
8
9
10 //ModelRepository.java
11
12         /**
13          * Custom finder
14          */
15         public List<Model> getModelsByPriceRangeAndWoodType(BigDecimal lowes
16 //              @SuppressWarnings("unchecked")
17 //              List<Model> mods = entityManager
18 //                                  .createQuery("select m from Model m where m.
19 //                                  .setParameter("lowest", lowest)
20 //                                  .setParameter("highest", highest)
21 //                                  .setParameter("wood", "%" + wood + "%").getR
22 //              return mods;
23                                  return modelJpaRepository.queryByPriceRangeA
24
25         }
26
27
28
```

@Query Annotation

Reuse existing JPQL in your data access layer

Advanced query functionality

Eager loading control (“fetch”)

```
@Query("select m from Model m where m.name = ?1")
List<Model> queryByName(String name);
```

@Query Options

Named Parameters

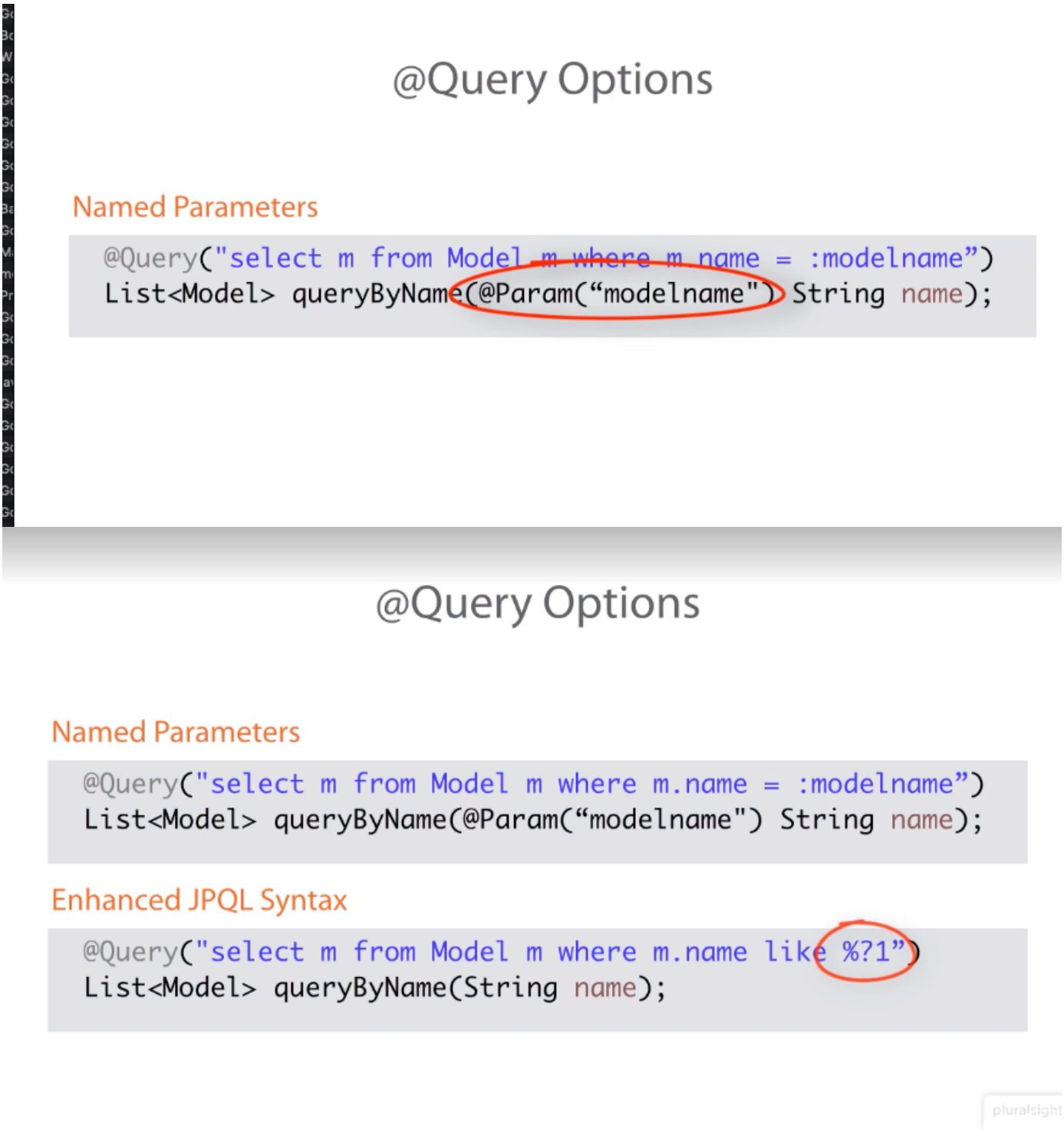
@Query Options

Named Parameters

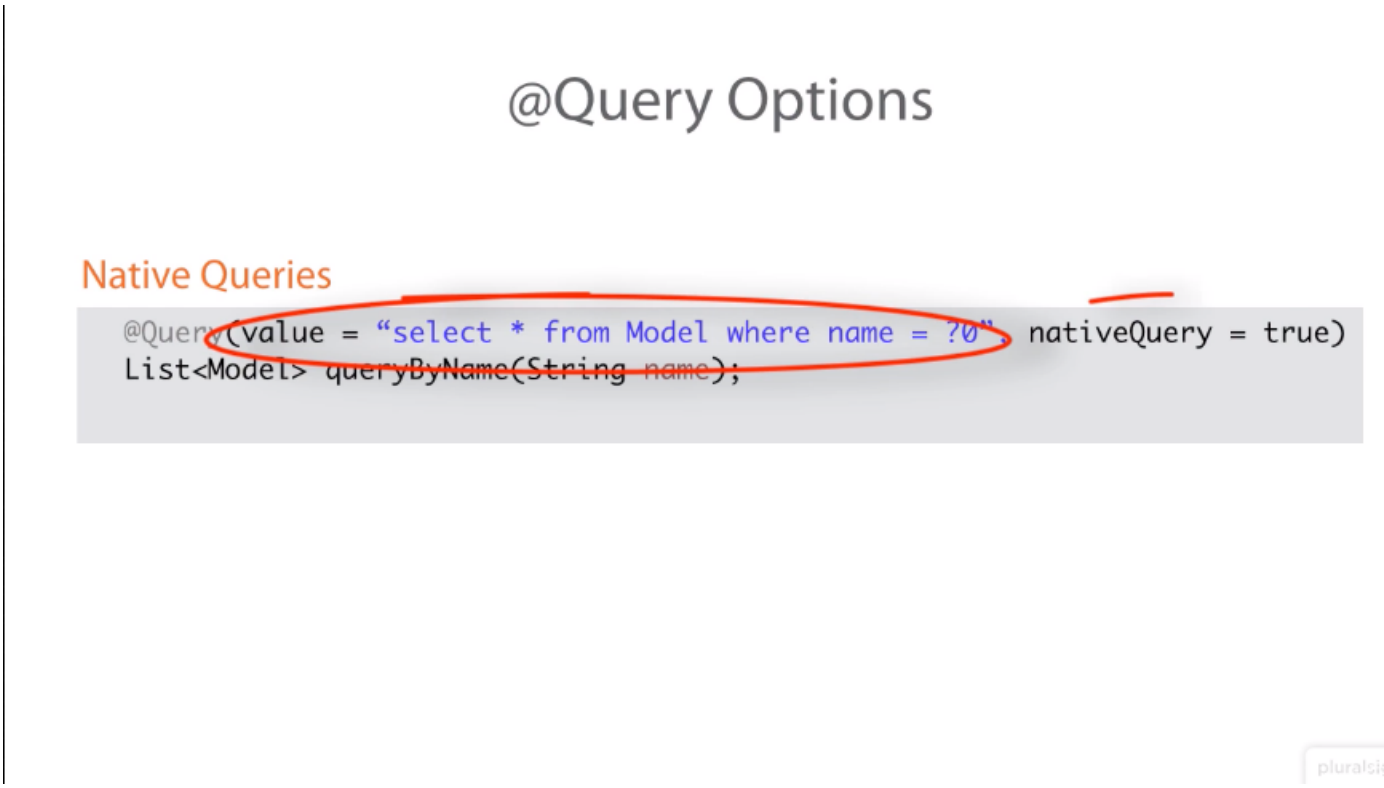
```
@Query("select m from Model m where m.name = :modelname”)
List<Model> queryByName(@Param(“modelname") String name);
```

```
1    List<Model> findAllModelsByType(@Param("name") String name);
2
3  ---
4          /**
5           * NamedQuery finder
6           */
7          public List<Model> getModelsByType(String modelType) {
8                  return  modelJpaRepository.findAllModelsByType(modelType);
```

```
9
10          }
11
```



Native queries




```
1
2
3       List<Manufacturer> getAllThatSellAcoustics(String name);
4
5   ---
6   **
7            * Native Query finder
8            */
9           public List<Manufacturer> getManufacturersThatSellModelsOfType(Strin
10                  return manufacturerJpaRepository.getAllThatSellAcoustics(mod
11  //              @SuppressWarnings("unchecked")
12  //              List<Manufacturer> mans = entityManager
13  //                              .createNamedQuery("Manufacturer.getAllThatSe
14  //                              .setParameter(1, modelType).getResultList();
15  //              return mans;
16  //              List<Manufacturer> mans = manufacturerJpaRepository.findByAc
17  //              return mans;
18
```

@Query Options

Native Queries

```
@Query(value = “select * from Model where name = ?0”, nativeQuery = true)
List<Model> queryByName(String name);
```


@Query Options

Native Queries

```
@Query(value = “select * from Model where name = ?0”, nativeQuery = true)
List<Model> queryByName(String name);
```

@Query Options

Native Queries

```
@Query(value = “select * from Model where name = ?0”, nativeQuery = true)
List<Model> queryByName(String name);
```

Modifiable Queries

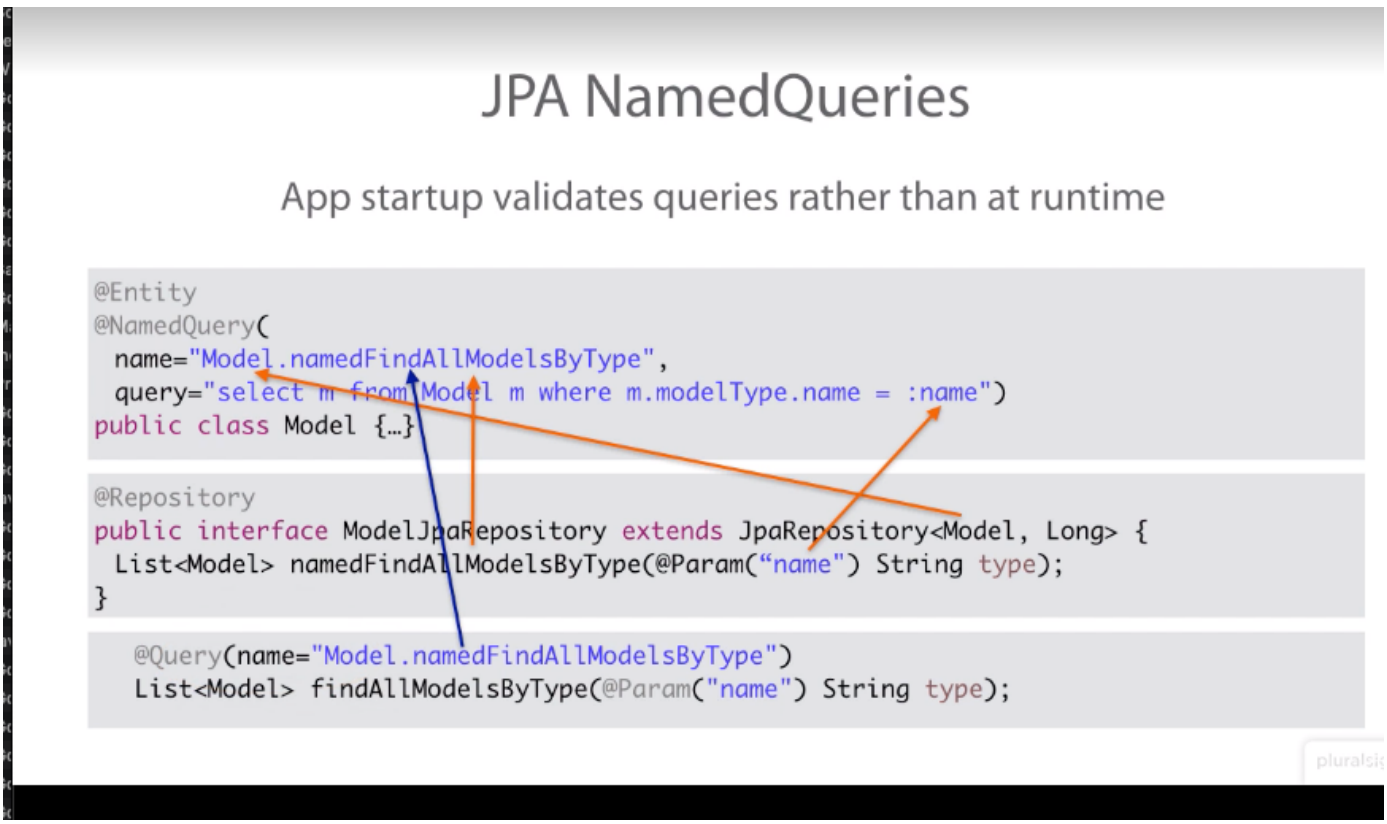
```
@Modifying
@Query(“update Model m set m.name = ?1”)
int updateByName(String name);
```

JPA NamedQueries

App startup validates queries rather than at runtime

```
@Entity
@NamedQuery(
  name="Model.namedFindAllModelsByType",
  query="select m from Model m where m.modelType.name = :name")
public class Model {…}
```

define in entity ands after use the query define in Entity class



2. Native SQL support (SQL)

Native SQL Support

Native Queries

```
@Query(value = “select * from Model where name = ?0”, nativeQuery = true)
List<Model> queryByName(String name);
```

Native SQL Support

Native Queries

```
@Query(value = “select * from Model where name = ?0”, nativeQuery = true)
List<Model> queryByName(String name);
```

@NamedNativeQuery

```
@NamedNativeQuery(name = "Manufacturer.getAllThatSellAcoustics",
query = "SELECT m.id, m.name, m.foundedDate, m.averageYearlySales, m.location_id as headquarters_id, m.active "
        + "FROM Manufacturer m "
        + "LEFT JOIN Model mod ON (m.id = mod.manufacturer_id) "
        + "LEFT JOIN ModelType mt ON (mt.id = mod.modeltype_id) "
        + "WHERE (mt.name = ?)", resultClass = Manufacturer.class)
public class Manufacturer { … }
```

Native SQL Support

Native Queries

```
@Query(value = “select * from Model where name = ?0”, nativeQuery = true)
List<Model> queryByName(String name);
```

@NamedNativeQuery

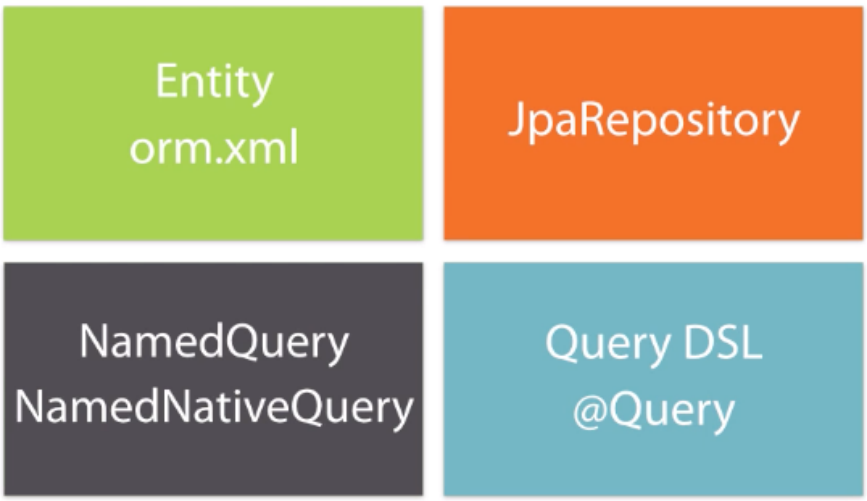
```
@NamedNativeQuery(name = "Manufacturer.getAllThatSellAcoustics",
query = "SELECT m.id, m.name, m.foundedDate, m.averageYearlySales, m.location_id as headquarters_id, m.active "
        + "FROM Manufacturer m "
        + "LEFT JOIN Model mod ON (m.id = mod.manufacturer_id) "
        + "LEFT JOIN ModelType mt ON (mt.id = mod.modeltype_id) "
        + "WHERE (mt.name = ?)", resultClass = Manufacturer.class)
public class Manufacturer { … }
```

Tip: Where Should I Put My Queries?

Entity
orm.xml

JpaRepository

Tip: Where Should I Put My Queries?



JpaRepository Query Precedence

Methods with @Query annotation take highest precedence

Methods that match a named or native named query "name"

Methods that follow the query DSL keyword naming structure