

Advanced Topics in JPA/Spring

Martin Ledvinka, Miroslav Blaško

KBSS

Winter Term 2021



Contents

1 JPA

2 Spring

3 Querydsl



JPA



Criteria API example

```
SELECT p FROM Product p WHERE p.name LIKE '%p%'
```

Static Metamodel

```
CriteriaBuilder cb =
    em.getCriteriaBuilder();
CriteriaQuery<Product> cq =
    cb.createQuery(Product.class);
Root<Product> r =
    cq.from(Product.class);
cq.where(
    cb.like(
        r.get(Product_.name)
        , "%p%")
);
return
    em.createQuery(cq).getResultList();
```

Metamodel

```
Metamodel m = em.getMetamodel();
CriteriaBuilder cb =
    em.getCriteriaBuilder();
CriteriaQuery<Product> cq =
    cb.createQuery(Product.class);
Root<Product> r =
    cq.from(Product.class);
cq.where(
    cb.like(
        r.get(
            m.entity(Product.class)
                .getSingularAttribute("name",
                    String.class))
        , "%p%")
);
return
    em.createQuery(cq).getResultList();
```



Metamodel

Static Metamodel

- Generated before build (usually using a Maven plugin)
- Classes with static fields representing entities and their attributes
- Compile-time checking

Dynamic Metamodel

- Generated by persistence provider at runtime
- Use names of attributes
- More verbose, typos cause runtime errors



Criteria API - Main classes

- 1 **CriteriaBuilder** used to construct
 - Criteria queries, e.g., `cb.createQuery(User.class)`
 - Expressions, e.g., “test that a concrete student is member of a studygroup” – `cb.isMember(student, studygroup.get("students"))`
 - Predicates, e.g., `cb.and(expression1, expression2)`
- 2 **CriteriaQuery** defines top-level structure of a query such as
 - `Root<User> = cq.from(User.class)`
 - `cq.select(..).where(..)`
- 3 **Root** – root type in the from clause that references an entity. It can be used within
 - Expressions e.g., `dog.get(Dog_.color).in("brown", "black")`
 - Selections e.g., `q.select(dog.get("color"))`
 - ...



Spring



Spring Data Repositories

You have seen in the lecture.

```
interface UserRepository extends JpaRepository<User, Long> {  
  
    User findByUsername(String username);  
  
    List<User> findByLastName(String lastName, Sort sort);  
  
    Page<User> findByFirstName(String firstName, Pageable pageable);  
}
```

Enable

```
@EnableJpaRepositories("cz.cvut.kbss.ear.eshop.dao")  
public class PersistenceConfig {  
    ...  
}
```



Specifications – Reusable Predicates

Specification interface

```
interface Specification<T> {
    Predicate toPredicate(
        Root<T> root,
        CriteriaQuery query,
        CriteriaBuilder cb);
}
```

Implementation of specifications

```
public static Specification<Product>
    hasNameLike(String name) {
    return (root, query, cb) ->
        cb.like(root.get(Product_.name),
            name);
    }
};
```

Definition of a repository

```
interface CustomerRepository
    extends JpaRepository<Customer>,
        JpaSpecificationExecutor
{
    // Your query methods here
}
```

Usage of repository

```
dao.findAll(hasBirthday());
dao.findAll(isLongTermCustomer());
dao.findAll(hasBirthday(),
    isLongTermCustomer());
```



Querydsl



Querydsl

- Unified query framework for Java
- Support for SQL, JPA, JDO, Mongo etc.
- Nice and succinct syntax
- Static metamodel for type safety and readability

```
List<Person> persons = queryFactory.selectFrom(person)
    .where(
        person.firstName.eq("John"),
        person.lastName.eq("Doe"))
    .orderBy(person.lastName.asc())
    .fetch();
```



Resources

- <https://docs.oracle.com/javase/6/tutorial/doc/gjrij.html>
- <https://www.baeldung.com/spring-data-criteria-queries>
- <https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl>
- <http://www.querydsl.com/>

