Spring Data Series: Part 3 TypeSafe Query

# Type safe Query

In this article we will discuss about type-safe query. We are aware of that Spring Data can derives a query based on Bean property. But this type of convention has certain shortcomings.

**Shortcomings**:

1. When we need to filter a query based on many properties method name will look ugly and pretty big. Like **findPersonByNameAndHobbyAndCountry().**
2. If we want to filter a query based on runtime strategy say country and hobby, country and name, hobby and country etc. We can’t do it through Query Derivation technique.

It will be good if we have a method which takes a **Predicate**/**Strategy** and we can build the predicate runtime and pass it to the same method, so a single method act as **runtime query generator**.

One can relate the same techniques with JPA CriteriaAPI, Yes we it is like Criteria API.

The good part is that Spring Data incorporate querydsl project to support this technique.

The querydsl tries provides a fluent API by which client can build query based on bean property.The API is derived from persistence store or Object model but it is store- and model-agnostic at the same time, so it allows client to create and use the query with very ease.

This API supports variety of stores like JPA, Hibernate, JDO, native JDBC,

Lucene, Hibernate Search, and MongoDB etc.

**Due to this vast support Spring Data project integrates with Querydsl project.**

**How querydsl works?**

querydsl generates metamodel objects from Entity classes. Client/developer uses this metamodel Object to do type safe queries. Querydsl generates this metamodel by APT processor which uses java6 Annotation Processing Tool(APT). This tool hook with compiler and based on the store specific processor like for JPA **JpaAnnotationProcessor** it finds the annotation for jpa **@Entity and @Embeddable and** generates the metamodel sources in target destination**. It retains the same package structure.**

**Steps for integrating querydsl with Spring Data for type-safe query**

1. In pom.xml we need to add qurrydsl-apt and querydsl-jpa dependency.

<dependency>

<groupId>com.querydsl</groupId>

<artifactId>querydsl-apt</artifactId>

<version>4.1.3</version>

</dependency>

<dependency>

<groupId>com.querydsl</groupId>

<artifactId>querydsl-jpa</artifactId>

<version>4.1.3</version>

</dependency>

1. Now in plugin section we need to add JpaAnnotationProcessor.

<plugin>

<groupId>com.mysema.maven</groupId>

<artifactId>apt-maven-plugin</artifactId>

<version>1.1.3</version>

<executions>

<execution>

<phase>generate-sources</phase>

<goals>

<goal>process</goal>

</goals>

<configuration>

<outputDirectory>target/generated-sources</outputDirectory>

<processor>com.querydsl.apt.jpa.JPAAnnotationProcessor</processor>

</configuration>

</execution>

</executions>

<dependencies>

<dependency>

<groupId>com.querydsl</groupId>

<artifactId>querydsl-apt</artifactId>

<version>4.1.3</version>

</dependency>

</dependencies>

</plugin>

1. We define the output directory as target/generated-sources where metamodel classes are generated. Bound this plugin to generate-sources phase.
2. Now execute command

mvn -X generate-sources

5.It will create metamodel classes in target/generated-sources directory.

6. As package structure is same copy the metamodel classes under src directory maintaining same package structure.

7. Extend QueryDslPredicateExecutor interface in Custom repository.

8. Write a Query based on metamodel classes.

Example : We will find a person based on the name using type-safe query.

1. Pom.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>cabBook</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>CabBookingSystem</name>

<description>CabBookingSystem Using Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.1.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.7</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.querydsl</groupId>

<artifactId>querydsl-apt</artifactId>

<version>4.1.3</version>

</dependency>

<dependency>

<groupId>com.querydsl</groupId>

<artifactId>querydsl-jpa</artifactId>

<version>4.1.3</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<plugin>

<groupId>com.mysema.maven</groupId>

<artifactId>apt-maven-plugin</artifactId>

<version>1.1.3</version>

<executions>

<execution>

<phase>generate-sources</phase>

<goals>

<goal>process</goal>

</goals>

<configuration>

<outputDirectory>target/generated-sources</outputDirectory>

<processor>com.querydsl.apt.jpa.JPAAnnotationProcessor</processor>

</configuration>

</execution>

</executions>

<dependencies>

<dependency>

<groupId>com.querydsl</groupId>

<artifactId>querydsl-apt</artifactId>

<version>4.1.3</version>

</dependency>

</dependencies>

</plugin>

</plugins>

</build>

</project>

1. After execution mvn -X generate-sources. Following meta model Objects are created

package com.example.person;

import static com.querydsl.core.types.PathMetadataFactory.\*;

import com.querydsl.core.types.dsl.\*;

import com.querydsl.core.types.PathMetadata;

import javax.annotation.Generated;

import com.querydsl.core.types.Path;

import com.querydsl.core.types.dsl.PathInits;

/\*\*

\* QPerson is a Querydsl query type for Person

\*/

@Generated("com.querydsl.codegen.EntitySerializer")

public class QPerson extends EntityPathBase<Person> {

private static final long serialVersionUID = 955983357L;

public static final QPerson person = new QPerson("person");

public final StringPath country = createString("country");

public final StringPath gender = createString("gender");

public final ListPath<Hobby, QHobby> hobby = this.<Hobby, QHobby>createList("hobby", Hobby.class, QHobby.class, PathInits.DIRECT2);

public final NumberPath<Long> id = createNumber("id", Long.class);

public final StringPath name = createString("name");

public QPerson(String variable) {

super(Person.class, forVariable(variable));

}

public QPerson(Path<? extends Person> path) {

super(path.getType(), path.getMetadata());

}

public QPerson(PathMetadata metadata) {

super(Person.class, metadata);

}

}

package com.example.person;

import static com.querydsl.core.types.PathMetadataFactory.\*;

import com.querydsl.core.types.dsl.\*;

import com.querydsl.core.types.PathMetadata;

import javax.annotation.Generated;

import com.querydsl.core.types.Path;

import com.querydsl.core.types.dsl.PathInits;

/\*\*

\* QHobby is a Querydsl query type for Hobby

\*/

@Generated("com.querydsl.codegen.EntitySerializer")

public class QHobby extends EntityPathBase<Hobby> {

private static final long serialVersionUID = -253362646L;

private static final PathInits INITS = PathInits.DIRECT2;

public static final QHobby hobby = new QHobby("hobby");

public final NumberPath<Long> id = createNumber("id", Long.class);

public final StringPath name = createString("name");

public final QPerson person;

public QHobby(String variable) {

this(Hobby.class, forVariable(variable), INITS);

}

public QHobby(Path<? extends Hobby> path) {

this(path.getType(), path.getMetadata(), PathInits.getFor(path.getMetadata(), INITS));

}

public QHobby(PathMetadata metadata) {

this(metadata, PathInits.getFor(metadata, INITS));

}

public QHobby(PathMetadata metadata, PathInits inits) {

this(Hobby.class, metadata, inits);

}

public QHobby(Class<? extends Hobby> type, PathMetadata metadata, PathInits inits) {

super(type, metadata, inits);

this.person = inits.isInitialized("person") ? new QPerson(forProperty("person")) : null;

}

}

1. Entity classes are as follows

package com.example.person;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

@Entity

public class Person {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

private Long id;

private String name;

private String country;

private String gender;

@OneToMany(mappedBy="person",targetEntity=Hobby.class,

fetch=FetchType.EAGER,cascade=CascadeType.ALL)

List<Hobby> hobby;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public List<Hobby> getHobby() {

return hobby;

}

public void setHobby(List<Hobby> hobby) {

this.hobby = hobby;

}

public void addHobby(Hobby ihobby)

{

if(hobby == null)

{

hobby = new ArrayList<Hobby>();

}

hobby.add(ihobby);

}

@Override

public String toString() {

return "Person [id=" + id + ", name=" + name + ", country=" + country + ", gender=" + gender + "]";

}

}

package com.example.person;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

@Entity

public class Hobby {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

private Long id;

@ManyToOne

@JoinColumn(name="person\_id")

private Person person;

private String name;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Person getPerson() {

return person;

}

public void setPerson(Person person) {

this.person = person;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

1. Extending QueryDslPredicateExecutor<T> by doing this we can pass Predicate in to the CRUD methods. Now findAll method look like

***Iterable<T> findAll(Predicate predicate);***

package com.example.repo;

import java.util.List;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.querydsl.QueryDslPredicateExecutor;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.example.person.Person;

@Repository

public interface PersonRepositary extends CrudRepository<Person, Long>,QueryDslPredicateExecutor<Person> {

@Query("select p from Person p where p.country like ?1")

List<Person> findByCountryContains(String country);

List<Person> findPersonByHobbyName(String name);

}

**package** com.example.repo;

**import** org.springframework.data.querydsl.QueryDslPredicateExecutor;

**import** org.springframework.data.repository.CrudRepository;

**import** com.example.person.Hobby;

**public** **interface** HobbyRepository **extends** CrudRepository<Hobby,Long>,QueryDslPredicateExecutor<Hobby> {

}

1. Create a method called findPersonByNameQueryDsl ,build a Predicate

based on person name property

BooleanExpression nameExpr = QPerson.***person***.name.contains("Shamik");

Pass this predicate in findAll(..) method

**package** com.example.person;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.data.jpa.repository.config.EnableJpaRepositories;

**import** com.example.repo.HobbyRepository;

**import** com.example.repo.PersonRepositary;

**import** com.querydsl.core.types.dsl.BooleanExpression;

@SpringBootApplication

@EnableJpaRepositories("com.example.repo")

**public** **class** PersonApplication {

@Autowired

HobbyRepository hRepo;

**private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(PersonApplication.**class**);

@Bean

**public** CommandLineRunner demo(PersonRepositary repository) {

//insertPerson(repository);

//findPersonByHobbyname(repository,"Photography");

findPersonByNameQueryDsl(repository,"Photography");

/\*List<Person> pesonList = repository.findByCountryContains("India");

for(Person p : pesonList)

log.info("Person " + p);\*/

**return** **null**;

}

**private** **void** findPersonByNameQueryDsl(PersonRepositary repository,String hobby)

{

BooleanExpression nameExpr = QPerson.***person***.name.contains("Shamik");

Iterable<Person> pList = repository.findAll(nameExpr);

**for**(Person p : pList)

***log***.info("Person " + p);

}

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(PersonApplication.**class**, args);

}

}

Output :

Hibernate:

select

person0\_.id as id1\_1\_,

person0\_.country as country2\_1\_,

person0\_.gender as gender3\_1\_,

person0\_.name as name4\_1\_

from

person person0\_

where

person0\_.name like ? escape '!'

Hibernate:

select

hobby0\_.person\_id as person\_i3\_0\_0\_,

hobby0\_.id as id1\_0\_0\_,

hobby0\_.id as id1\_0\_1\_,

hobby0\_.name as name2\_0\_1\_,

hobby0\_.person\_id as person\_i3\_0\_1\_

from

hobby hobby0\_

where

hobby0\_.person\_id=?

2016-10-04 18:34:18.520 INFO 7344 --- [ main] com.example.person.PersonApplication : Person Person [id=14, name=Shamik mitra, country=India, gender=male]