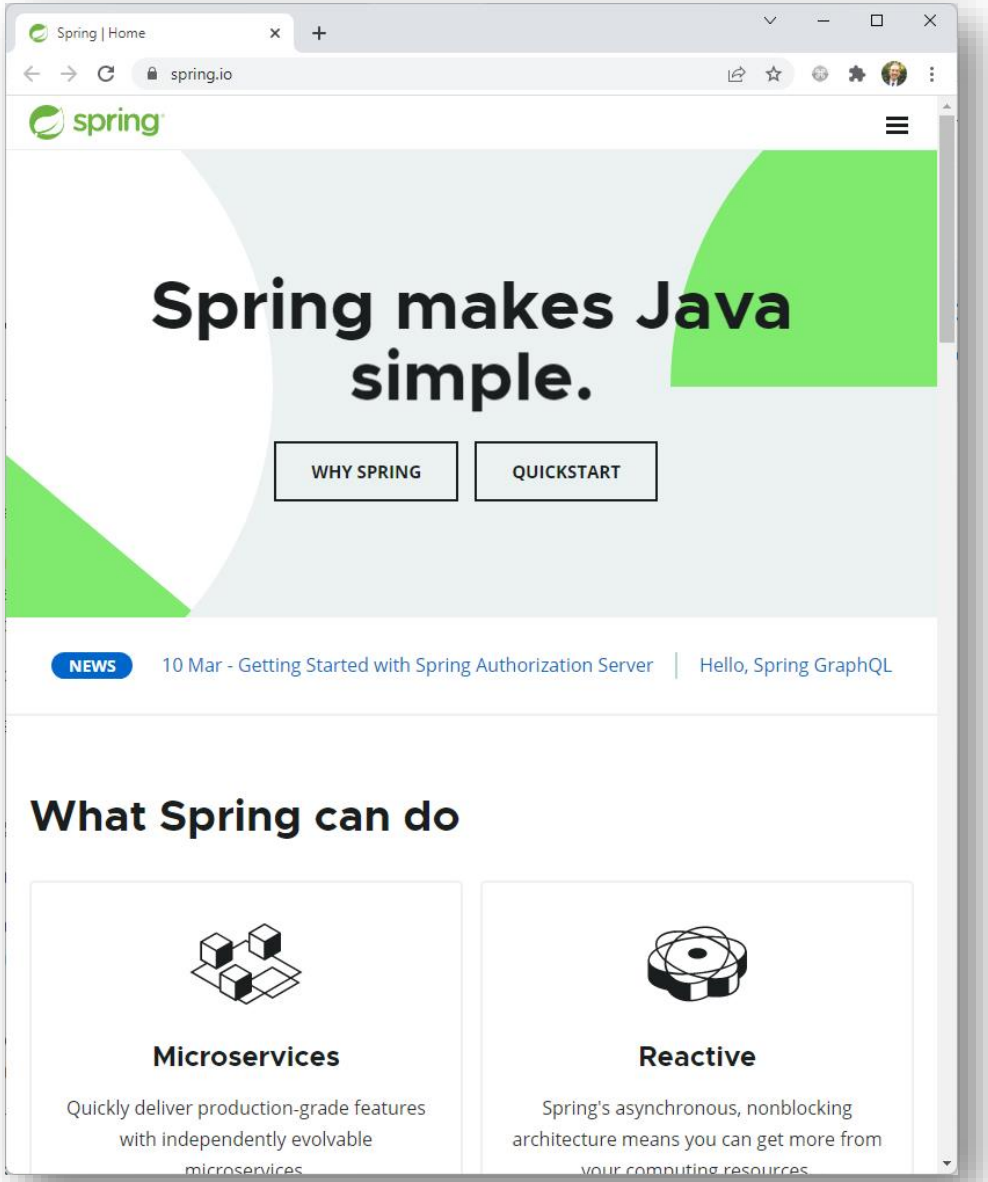
Spring Boot Primer

Paulo Gandra de Sousa

pag@isep.ipp.pt

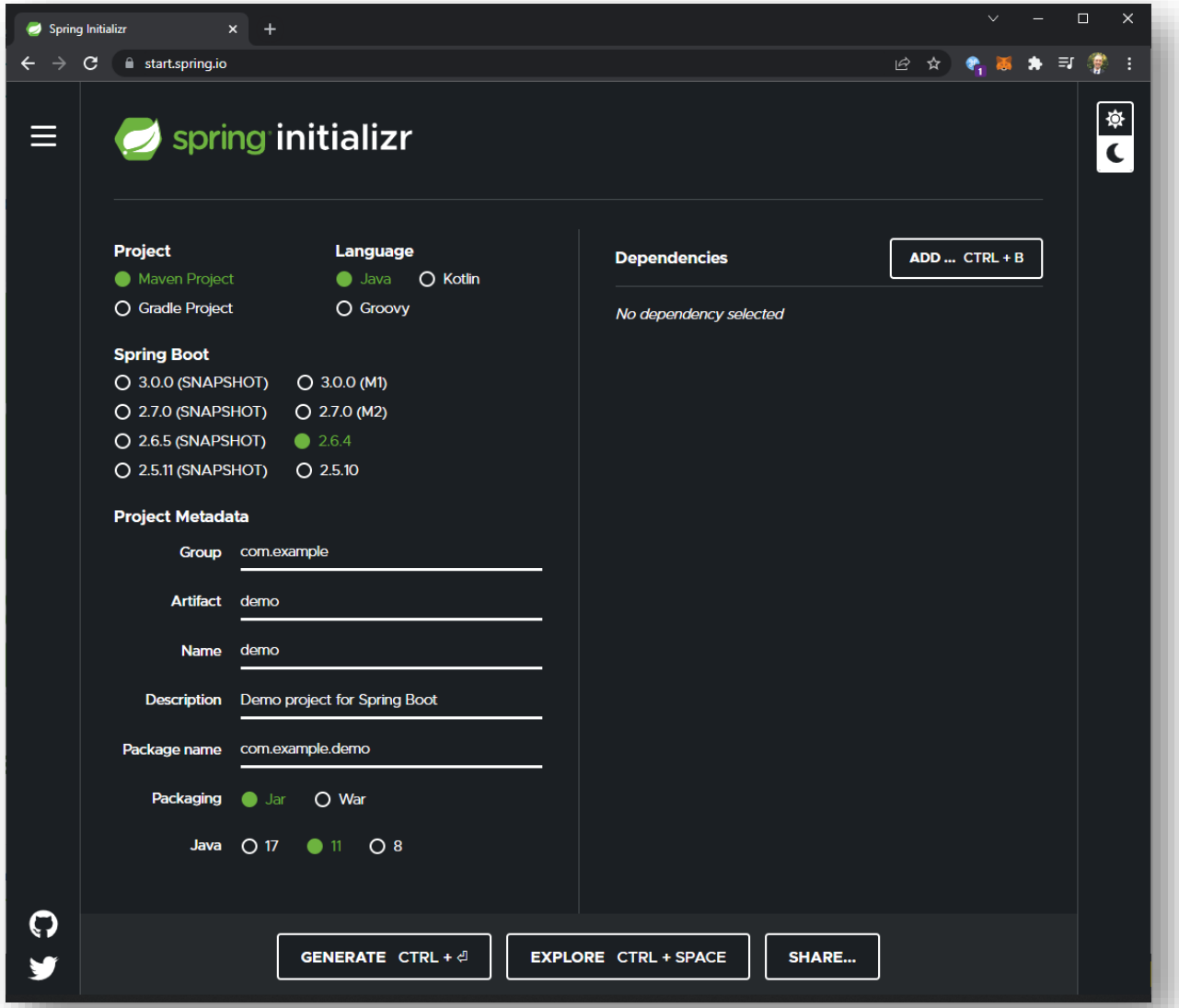
What is Spring Boot?

- A framework to ease application Development
- Starter dependencies
- Embed application server
- Convention over configuration



Spring Initialzr

- https://start.spring.io/
- Parent
- Starters
  - E.g., Web
- Other dependencies
  - Lombok



Generated

- Parent pom
- Web starter
- Lombok
- Test starter

Spring Initializr

start.spring.io

demo.zip

- .gitignore
- .mvn
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml
- src
  - main
    - java
      - com
        - example
          - demo
            - DemoApplication.java
    - resources
      - application.properties
      - static
      - templates
  - test
    - java
      - com
        - example

DOWNLOAD COPY

```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/XMLSche
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
```

DOWNLOAD CTRL + ⏎ CLOSE ESC

Main application class

demo - demo/src/main/java/com/example/demo/DemoApplication.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- demo
  - src
    - main
      - java
        - com
          - example
            - demo
              - DemoApplication.java
      - resources
    - test
      - java
        - com
          - example
            - demo
              - DemoApplicationTests.java
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml

DemoApplication.java | demo/pom.xml | DemoApplicationTests.java

```
1  package com.example.demo;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class DemoApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(DemoApplication.class, args);
11     }
12
13 }
14
```

@SpringBootApplication

Outline | Task List

- com.example.demo
- DemoApplication
  - main(String[]) : void

Context loading test

demo - demo/src/test/java/com/example/demo/DemoApplicationTests.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- demo
  - src
    - main
      - java
        - com
          - example
            - demo
              - DemoApplication.java
    - resources
  - test
    - java
      - com
        - example
          - demo
            - DemoApplicationTests.java
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml

DemoApplication.java | demo/pom.xml | DemoApplicationTests.java

```
1  package com.example.demo;
2
3  import org.junit.jupiter.api.Test;
5
6  @SpringBootTest
7  class DemoApplicationTests {
8
9      @Test
10     void contextLoads() {
11     }
12
13 }
14
```

@SpringBootTest

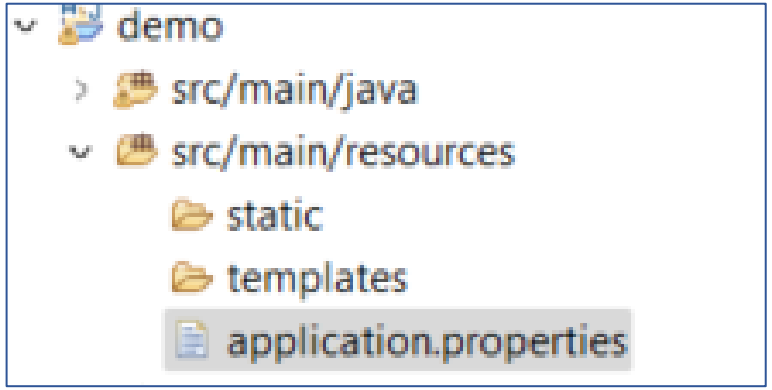
Outline | Task List

- com.example.demo
- DemoApplicationTests
  - contextLoads() : void

CLI Applications

- Implement **`CommandLineRunner`**

- Application.properties
  - *spring.main.web-application-type=NONE*




```java
@SpringBootApplication
public class SpringBootConsoleApplication
  implements CommandLineRunner {

    private static Logger LOG = LoggerFactory
      .getLogger(SpringBootConsoleApplication.class);

    public static void main(String[] args) {
        LOG.info("STARTING THE APPLICATION");
        SpringApplication.run(SpringBootConsoleApplication.class, args);
        LOG.info("APPLICATION FINISHED");
    }

    @Override
    public void run(String... args) {
        LOG.info("EXECUTING : command line runner");

        for (int i = 0; i < args.length; ++i) {
            LOG.info("args[{}]: {}", i, args[i]);
        }
    }
}
```

Startup execution thru CommandLineRunner

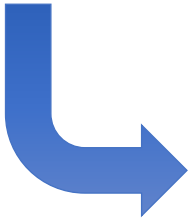
- Spring will execute any **`CommandLineRunner`** after starting up
- Example, bootstrap some data

```
@Configuration
class LoadDatabase {

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

  @Bean
  CommandLineRunner initDatabase(EmployeeRepository repository) {

    return args -> {
      log.info("Preloading " + repository.save(new Employee("Bilbo Baggins",
"burglar")));
      log.info("Preloading " + repository.save(new Employee("Frodo Baggins", "thief")));
    };
  }
}
```

Dependency Injection

- Modules declare their dependencies but do not create them explicitily
- Decreases coupling
- Allows to “plug in” alternative implementations of the needed servisse
- Spring is a powerful Dependency Injection engine

```
class Component {
        NeededService svc;

        Component() {
                svc = new ServiceImplementation();
        }
}
```



```
class Component {
        NeededService svc;

        Component(NeededService impl) {
                svc = impl;
        }
}
```

Dependency Injection - declaring

```
class FooController {

    @Autowired
    private FooService service;

    public List<Foo> findAll() {
        // check permissions
        …
        return service.findAll();
    }

    ...
```

- Class **`FooController`** **needs** an object of type **`FooService`**
- Instead of creating the object, it **declares the need**, and asks Spring to **inject the dependency** thru **`@Autowired`**
- Spring **searches** for compatible classes and **creates and manages** the object

Dependency injection - implementing

@Service

```
public interface FooService {

  List<Foo> findAll();

  Foo findOne(Long id);

  Foo create(Foo resource);

  Foo upsert(Long id, Foo resource);

  Foo update(Long id, Foo resource);

  void deleteById(Long id);

}
```

```
@Service
public class FooServiceImpl implements FooService {

  @Override
  public List<Foo> findAll() {…}

  @Override
  public Foo findOne(Long id) {…}

  @Override
  public Foo create(Foo resource) {…}

  @Override
  public Foo upsert(Long id, Foo resource) {…}

  @Override
  public Foo update(Long id, Foo resource) {…}

  @Override
  public void deleteById(Long id) {…}
}
```

How does it all glue together?

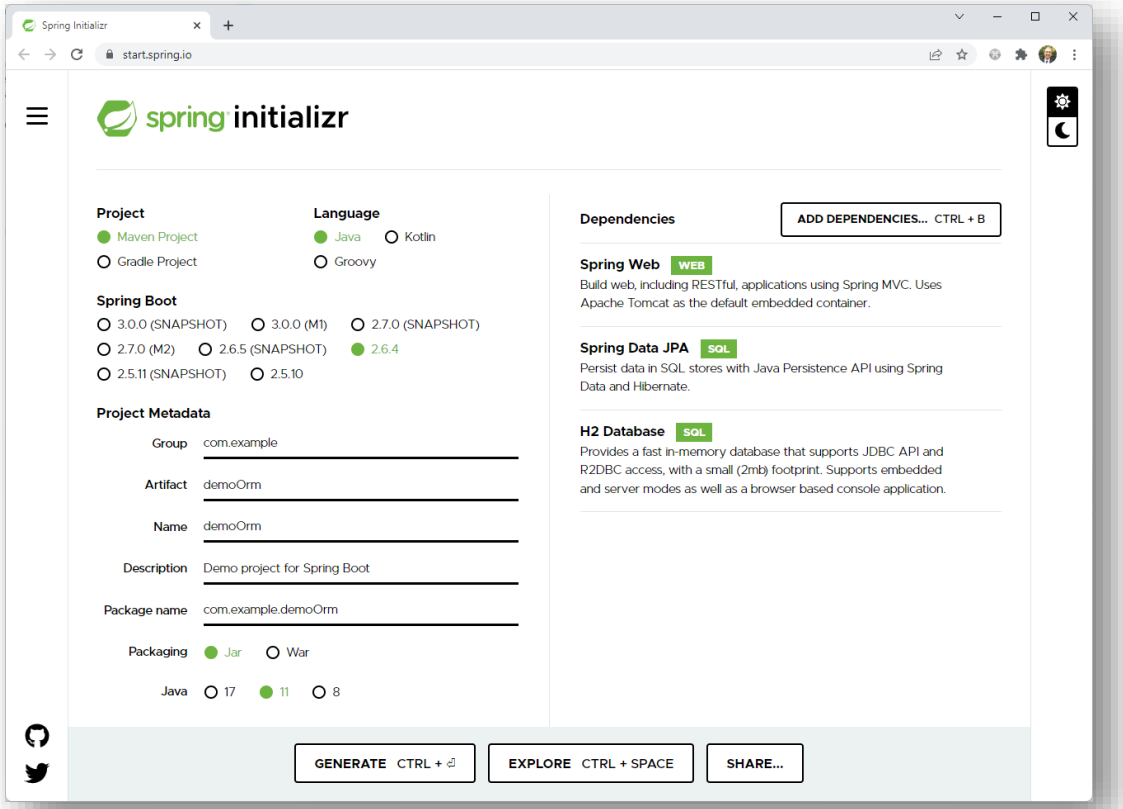
```java
@SpringBootApplication
public class DemoApplication implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @Autowired
    private FooUI fooUI;

    @Override
    public void run(String... args) throws Exception {
        fooUI.show();
    }
}
```

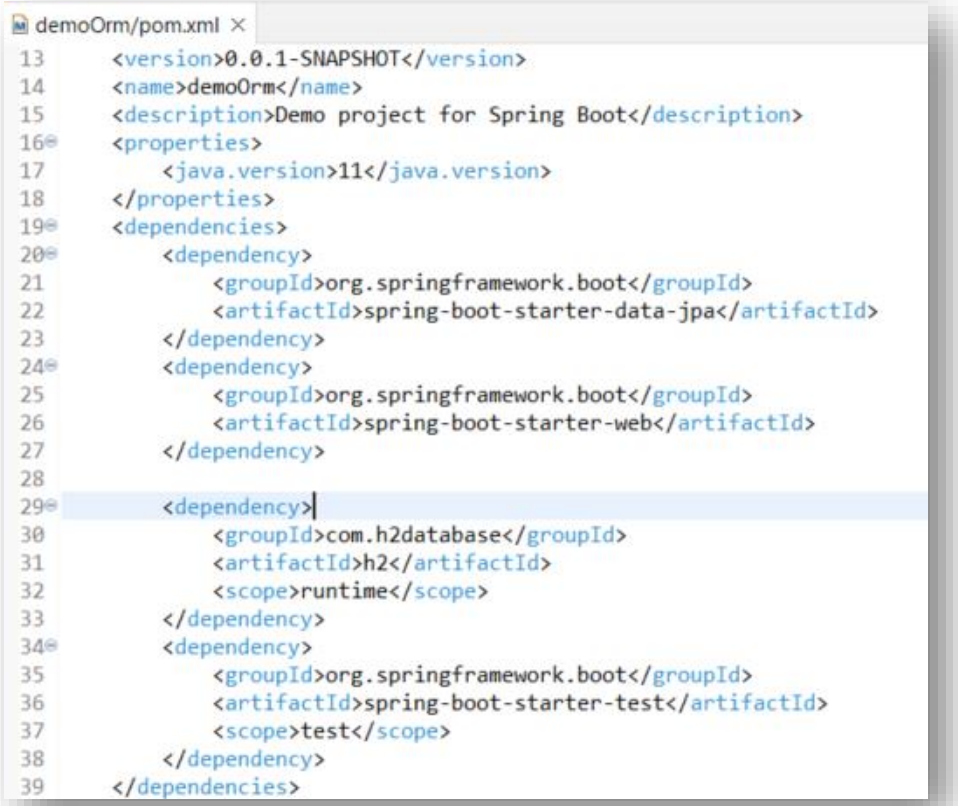
```java
@Component
public class FooUI {

    @Autowired
    private FooController controller;

    public void show() {
        …
        controller.submit(…);
    }

}
```

```java
@Component
public class FooController {
        ...
}
```

Acessing Data with Spring Data

Spring Data





demoOrm/pom.xml

```
13      <version>0.0.1-SNAPSHOT</version>
14      <name>demoOrm</name>
15      <description>Demo project for Spring Boot</description>
16      <properties>
17          <java.version>11</java.version>
18      </properties>
19      <dependencies>
20          <dependency>
21              <groupId>org.springframework.boot</groupId>
22              <artifactId>spring-boot-starter-data-jpa</artifactId>
23          </dependency>
24          <dependency>
25              <groupId>org.springframework.boot</groupId>
26              <artifactId>spring-boot-starter-web</artifactId>
27          </dependency>
28
29          <dependency>
30              <groupId>com.h2database</groupId>
31              <artifactId>h2</artifactId>
32              <scope>runtime</scope>
33          </dependency>
34          <dependency>
35              <groupId>org.springframework.boot</groupId>
36              <artifactId>spring-boot-starter-test</artifactId>
37              <scope>test</scope>
38          </dependency>
39      </dependencies>
```

Repositories

- Default CRUD operations
- Default implementations provided by Spring Data
- Use of java Generics

```java
public interface CrudRepository<T, ID> extends Repository<T, ID> {

    <S extends T> S save(S entity);

    <S extends T> Iterable<S> saveAll(Iterable<S> entities);

    Optional<T> findById(ID id);

    boolean existsById(ID id);

    Iterable<T> findAll();

    Iterable<T> findAllById(Iterable<ID> ids);

    long count();

    void deleteById(ID id);

    void delete(T entity);

    void deleteAllById(Iterable<? extends ID> ids);

    void deleteAll(Iterable<? extends T> entities);

    void deleteAll();
}
```

Specialized repositories

- Create a specialized repository for each aggregate of your model

Inherits default operations

Type of @Id field

```
public interface FooRepository extends CrudRepository<Foo, Long> {

}
```

Type of @Entity class

Controller

Dependency injection

```
@Service
public class FooServiceImpl implements FooService {

    @Autowired
    FooRepository repository;

    @Override
    public Iterable<Foo> findAll() {
        return repository.findAll();
    }

    @Override
    public Foo findOne(Long id) throws MyResourceNotFoundException {
        return repository.findById(id).orElseThrow(
        () -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Foo Not Found")
        );
    }

    @Override
    public Foo create(Foo resource) {
        return repository.save(resource);
    }
```

Database properties

Application.properties

Database connection string

Package Explorer

- demo
- demo_orm_2022_a1_a2
- demoOrm
  - src/main/java
  - src/main/resources
    - static
    - templates
    - application.properties

application.properties

```
1 ## datasource
2 spring.datasource.url=jdbc:h2:~/foo;MV_STORE=FALSE;AUTO_SERVER=true;
3 spring.datasource.username=mysqluser
4 spring.datasource.password=mysqlpass
5
6 ## database schema generation
7 spring.jpa.generate-ddl=true
8 spring.jpa.hibernate.ddl-auto=update
```

Database schema generation

Resources folder

Queries by naming convention

- Create specialized queries by convention

```
public interface FooRepository extends CrudRepository<Foo, Long> {

  List<Foo> findByName(String name);
}
```

Spring automatically creates a query to search by attribute with name “name”

Queries by naming convention

Keyword	Sample	JPQL snippet
Distinct	findDistinctByLastnameAndFirstname	select distinct … where x.lastname = ?1 and x.firstname = ?2
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is, Equals	findByFirstname, findByFirstnameIs, findByFirstnameEquals	… where x.firstname = ?1
Between	findByStartDateBetween	… where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1

Queries by naming convention

Keyword	Sample	JPQL snippet
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull, Null	findByAge(Is)Null	… where x.age is null
IsNotNull, Not Null	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)

Queries by naming convention

Keyword	Sample	JPQL snippet
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> ages)	… where x.age not in ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	… where UPPER(x.firstname) = UPPER(?1)

Define Queries

```
public interface UserRepository extends JpaRepository<User, Long> {

@Query("select u from User u where u.emailAddress = ?1")
User findByEmailAddress(String emailAddress);
}
```

Positional Query parameter

Named Query parameter

```
public interface UserRepository extends JpaRepository<User, Long> {

@Query("select u from User u where u.firstname = :firstname or u.lastname = :lastname")
User findByLastnameOrFirstname(@Param("lastname") String lastname, @Param("firstname") String firstname);
}
```

Define Named Queries

Entity class with named query annotation
Query name must follow format>
«ClassName».«QueryName»

```
@Entity
@NamedQuery(name = "User.findByEmailAddress",
                    query = "select u from User u where u.emailAddress = ?1")
public class User {
            …
}
```

Will create query by naming convention

```
public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByLastname(String lastname);

User findByEmailAddress(String emailAddress);
}
```

Will use named query

Modifying Queries

```
public interface UserRepository extends JpaRepository<User, Long> {

@Modifying
@Query("update User u set u.firstname = ?1 where u.username = ?2")
int setFirstnameOfUser(String firstname, String username);
}
```

Can be used for UPDATE, DELETE, INSERT

Transactional

```
@Service
public class UserManagementImpl implements UserManagement {
    private final UserRepository userRepository;
    private final RoleRepository roleRepository;

    @Transactional
    public void addRoleToAllUsers(String roleName) {
        Role role = roleRepository.findByName(roleName);
        for (User user : userRepository.findAll()) {
            user.addRole(role);
            userRepository.save(user);
        }
    }
}
```

References

- Spring Boot Console Application, Baeldung, https://www.baeldung.com/spring-boot-console-app

- Introduction to Spring JPA, Baeldung, https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa

- Spring Data @Query, Baeldung. https://www.baeldung.com/spring-data-jpa-query

Additional References

- https://spring.io/
- Learn Spring Boot, Baeldung, https://www.baeldung.com/spring-boot
- Spring Data JPA Tutorial, Petri Kainulainen, https://www.petrikainulainen.net/spring-data-jpa-tutorial/
- Spring Persistence Tutorial, Baeldung, https://www.baeldung.com/persistence-with-spring-series
- Building REST services with Spring, Spring.io, https://spring.io/guides/tutorials/rest/