spring boot

Java Persistence API

JPA

Project Lombok

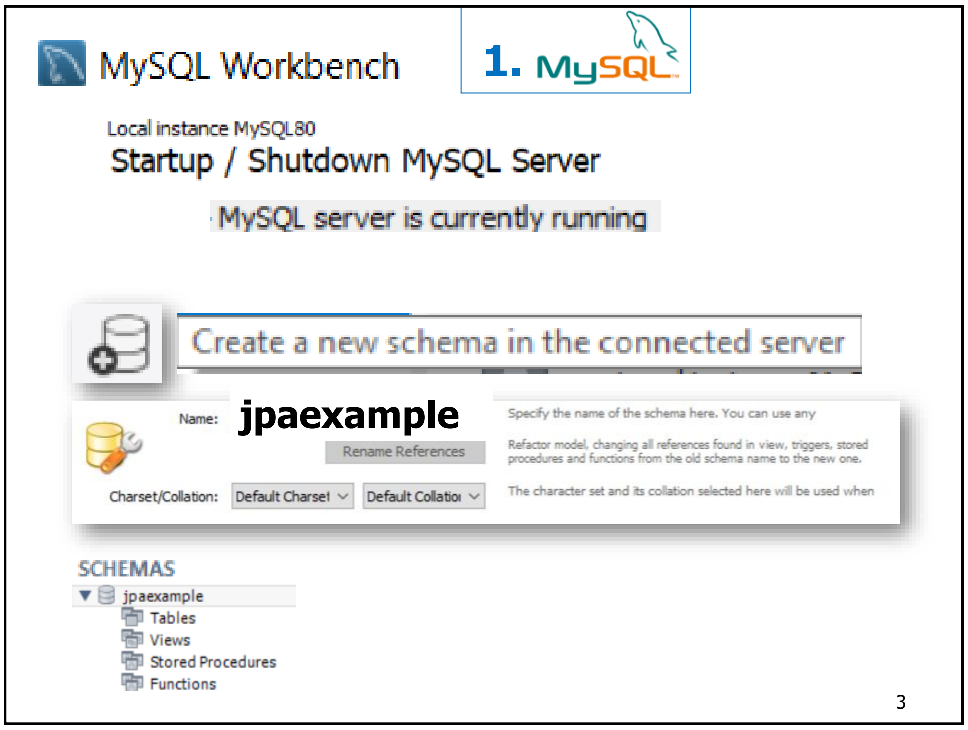
Spice up your java

MySQL

MySQL

Java Persistence API

JPA


MySQL Workbench

1. MySQL

Local instance MySQL80

Startup / Shutdown MySQL Server

MySQL server is currently running

Create a new schema in the connected server

Name: jpaexample

Rename References

Specify the name of the schema here. You can use any

Refactor model, changing all references found in view, triggers, stored procedures and functions from the old schema name to the new one.

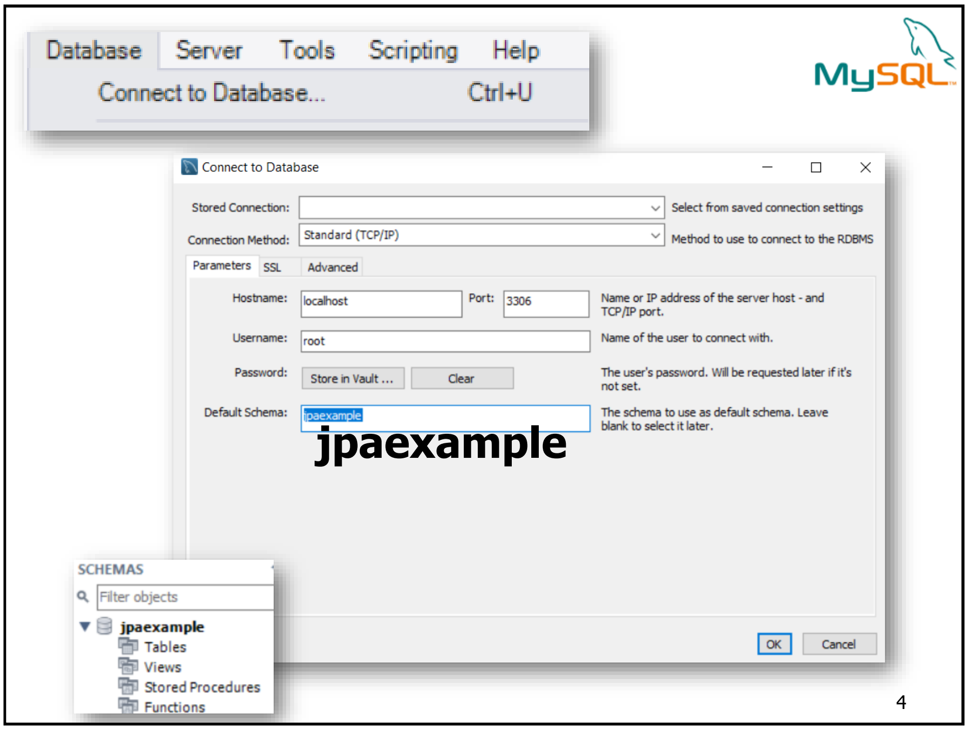
Charset/Collation: Default Charset | Default Collation

The character set and its collation selected here will be used when

SCHEMAS
- jpaexample
  - Tables
  - Views
  - Stored Procedures
  - Functions

3

3


Database Server Tools Scripting Help
Connect to Database... Ctrl+U
MySQL

Connect to Database

Stored Connection: | Select from saved connection settings

Connection Method: Standard (TCP/IP) | Method to use to connect to the RDBMS

Parameters | SSL | Advanced

Hostname: localhost Port: 3306 | Name or IP address of the server host - and TCP/IP port.

Username: root | Name of the user to connect with.

Password: Store in Vault ... Clear | The user's password. Will be requested later if it's not set.

Default Schema: jpaexample | The schema to use as default schema. Leave blank to select it later.

jpaexample

OK Cancel

SCHEMAS
Filter objects
- jpaexample
  - Tables
  - Views
  - Stored Procedures
  - Functions

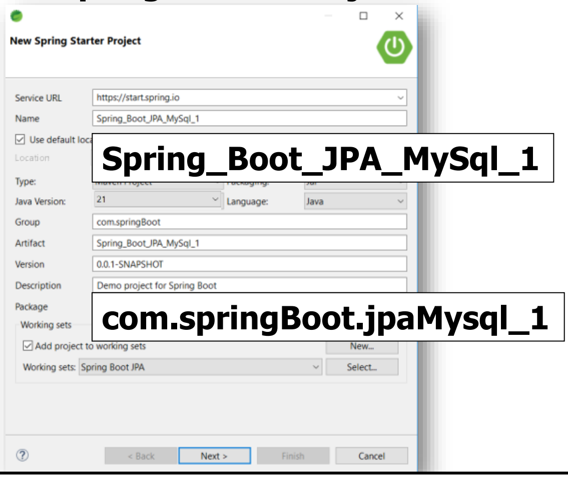
4

4

2. Example

Step 1: Create a Spring Boot Project

File → New → Spring Starter Project



New Spring Starter Project

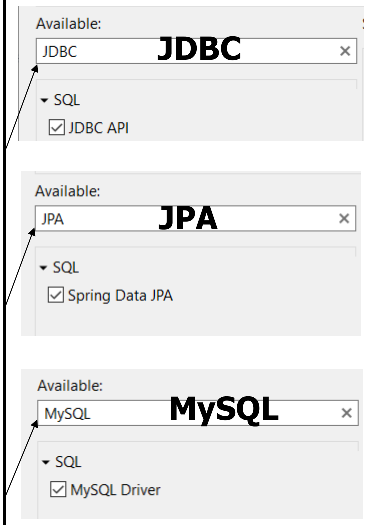
Service URL	https://start.spring.io
Name	Spring_Boot_JPA_MySql_1
☑ Use default loc	
Location	
Type:	
Java Version:	21
Language:	Java
Group	com.springBoot
Artifact	Spring_Boot_JPA_MySql_1
Version	0.0.1-SNAPSHOT
Description	Demo project for Spring Boot
Package	

Working sets
☑ Add project to working sets
Working sets: Spring Boot JPA

Spring_Boot_JPA_MySql_1

com.springBoot.jpaMysql_1

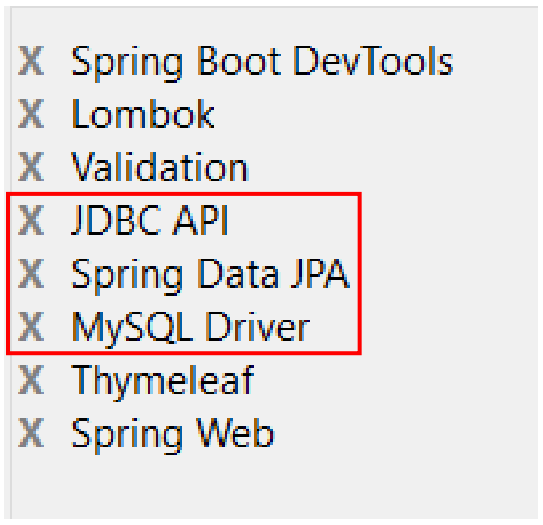
5



Available:
JDBC **JDBC**
▾ SQL
☑ JDBC API

Available:
JPA **JPA**
▾ SQL
☑ Spring Data JPA

Available:
MySQL **MySQL**
▾ SQL
☑ MySQL Driver



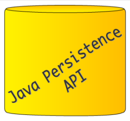
X Spring Boot DevTools
X Lombok
X Validation
X JDBC API
X Spring Data JPA
X MySQL Driver
X Thymeleaf
X Spring Web

6

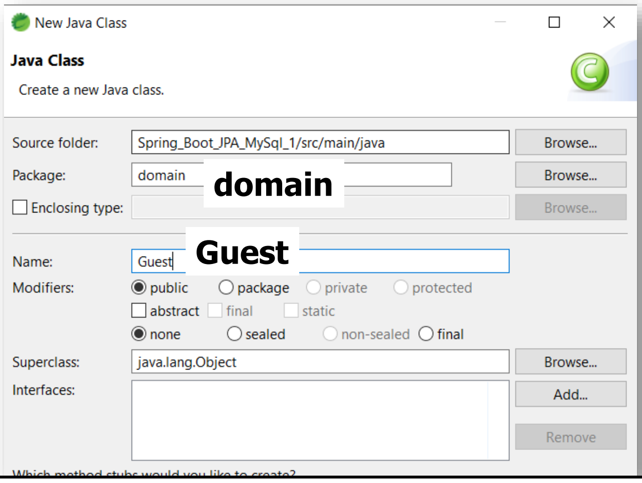
```
@SpringBootApplication
public class SpringBootJpaMySql1Application
                              implements WebMvcConfigurer{
    ...

   @Override
   public void addViewControllers(ViewControllerRegistry registry) {
          registry.addRedirectViewController("/", "/guest");
   }

}
```

7

Step 2: Entity class



New → Class



Java Class
Create a new Java class.

Source folder: Spring_Boot_JPA_MySql_1/src/main/java
Package: domain **domain**
Enclosing type:
Name: Guest **Guest**
Modifiers: [✓ public] package private protected
abstract final static
[✓ none] sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?

8

8

Step 2: Entity class

```
package domain;
import java.io.Serializable;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.ToString;
import lombok.AccessLevel;
import lombok.EqualsAndHashCode;

@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode(exclude = "id")
@ToString(exclude = "id")
public class Guest implements Serializable {
```

protected default constructor

9

```
@Entity @Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode(exclude = "id")
@ToString(exclude = "id")
public class Guest implements Serializable {

   private static final long serialVersionUID = 1L;
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Getter(AccessLevel.NONE)
  private Long id;

   private String name;

   private String firstname;

   public Guest(String name, String firstname) {
                  this.name = name;
                  this.firstname = firstname;
   }
```

equals, hashCode and toString methods for name and firstname, but not the id

10

Suppose you want the methods equals and hashcode with name,
without id and first name.

@EqualsAndHashCode(exclude = {"id", "firstname"})

OR

@EqualsAndHashCode(of = "name")

@AllArgsConstructor with exclude does not exist.

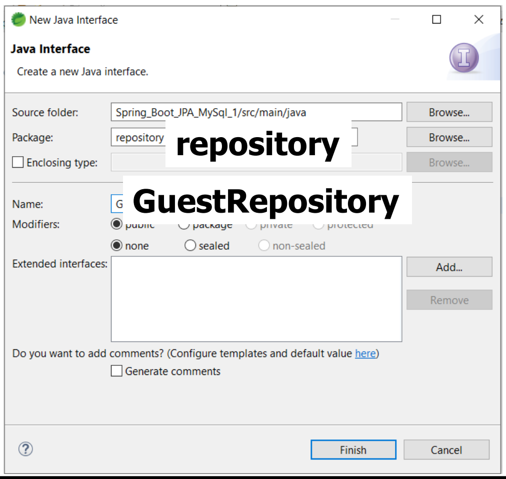
```
public Guest(String name, String firstname) {
                this.name = name;
                this.firstname = firstname;
}
```

11

11

Step 3: Define a Repository interface

New → Interface



New Java Interface

Java Interface

Create a new Java interface.

Source folder: Spring_Boot_JPA_MySql_1/src/main/java Browse...

Package: repository Browse...

repository

Enclosing type: Browse...

Name: G

GuestRepository

Modifiers: [✓ public] package private protected

[✓ none] sealed non-sealed

Extended interfaces: Add... Remove

Do you want to add comments? (Configure templates and default value here)

Generate comments

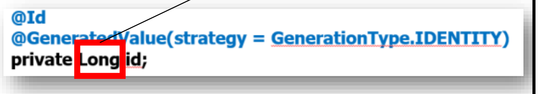
Finish Cancel

12

12

```
package repository;
import org.springframework.data.repository.CrudRepository;
import domain.Guest;

public interface GuestRepository
                extends CrudRepository<Guest, Long> {
```

Entity class Guest



```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

CrudRepository is a part of the **Spring Data JPA module**.

CrudRepository is an **interface** that provides a set of methods for performing basic **CRUD** (Create, Read, Update, Delete) operations on entities in a database.

```
package repository;
import org.springframework.data.repository.CrudRepository;
import domain.Guest;

public interface GuestRepository
                extends CrudRepository<Guest, Long> {
```

The CrudRepository interface provides the following methods:

▪ **save(entity)**	saves the given entity and returns it
▪ **findById(id)**	finds an entity by its id and returns it as an Optional
▪ **findAll()**	returns all entities as a List
▪ **count()**	returns the number of entities
▪ **deleteById(id)**	deletes an entity by its id
▪ **delete(entity)**	deletes the given entity
▪ Etc.	

```
package repository;
import org.springframework.data.repository.CrudRepository;
import domain.Guest;

public interface GuestRepository
                extends CrudRepository<Guest, Long> {

        List<Guest> findByName(String name);

        List<Guest> findByFirstname(String firstname);
}
```

By extending the CrudRepository interface, **you can define your own repositories with custom methods**, for example **findByName** and **findByFirstname**.

The Spring Data JPA module will automatically generate the implementation of the repository interface at runtime, providing you with a ready-to-use repository for your entities.

15

15

findBy, deleteBy, countBy, existsBy, and findFirstBy

```
@Entity ...
public class Example implements Serializable { ...
   private int age; private String lastname; ...

public interface ExampleRepository extends CrudRepository<Example, Long> {
     List<Example> findByAgeAndLastname(int age, String lastname);
     List<Example> findByAgeLessThan(int age);
     List<Example> findByAgeGreaterThan(int age);
     List<Example> findByAgeLessThanOrEqual(int age);
     ...
     void deleteByAge(int age);
     void deleteByAgeLessThan(int age);
     ...
     long countByAge(int age);
     boolean existsByAge(int age);
     boolean existsByAgeGreaterThan(int age);
     boolean existsByAgeAndLastname(int age, String lastname);
     Example findFirstByAge(int age);
     ...
```

16

16

findBy...orderBy, findTop...By

```
@Entity ...
public class Example implements Serializable { ...
    private int age; private String lastname; …


import org.springframework.data.jpa.repository.JpaRepository;
public interface ExampleRepository extends JpaRepository<Example, Long>
{
        /*JpaRepository extends CrudRepository, inheriting its basic CRUD
           operations, and adds extra features like pagination, sorting, and
           JPA-specific query methods. */

        List<Example> findByOrderByAge(); //Asc
        List<Example> findByOrderByLastnameDesc();
        List<Example> findByOrderByLastnameDescAgeAsc();
        ...
        List<Example> findByAgeOrderByLastnameAsc(int age);
        ...
```

17

17

Step 4: Add a Controller Class

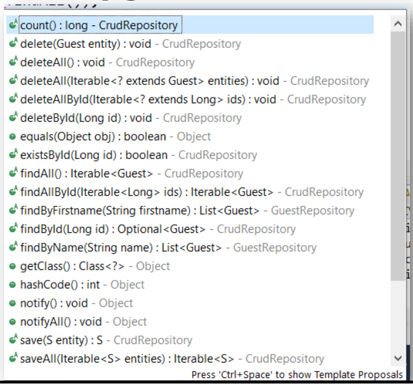
New → Java Class

```
package com.springBoot.jpaMysql_1;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import repository.GuestRepository;

@Controller
@RequestMapping("/guest")
public class GuestController {
```

18

18

```
@Controller
@RequestMapping("/guest")
public class GuestController {
        @Autowired
        private GuestRepository repository;

        @GetMapping
        public String listGuest(Model model) {

            model.addAttribute("guestList",
              repository.
```



```
count() : long - CrudRepository
delete(Guest entity) : void - CrudRepository
deleteAll() : void - CrudRepository
deleteAll(Iterable<? extends Guest> entities) : void - CrudRepository
deleteAllById(Iterable<? extends Long> ids) : void - CrudRepository
deleteById(Long id) : void - CrudRepository
equals(Object obj) : boolean - Object
existsById(Long id) : boolean - CrudRepository
findAll() : Iterable<Guest> - CrudRepository
findAllById(Iterable<Long> ids) : Iterable<Guest> - CrudRepository
findByFirstname(String firstname) : List<Guest> - GuestRepository
findById(Long id) : Optional<Guest> - CrudRepository
findByName(String name) : List<Guest> - GuestRepository
getClass() : Class<?> - Object
hashCode() : int - Object
notify() : void - Object
notifyAll() : void - Object
save(S entity) : S - CrudRepository
saveAll(Iterable<S> entities) : Iterable<S> - CrudRepository
                              Press 'Ctrl+Space' to show Template Proposals
```

19

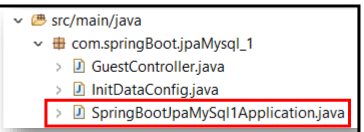
19

```
@Controller
@RequestMapping("/guest")
public class GuestController {

        @Autowired
        private GuestRepository repository;

        @GetMapping
        public String listGuest(Model model) {

            model.addAttribute("guestList", repository.findAll());
            model.addAttribute("guestName",
                                repository.findByName("Blondeel"));
            model.addAttribute("guestFirstname",
                                repository.findByFirstname("Sandra"));

            return "guest";
        }
}
```

20

20

Step 5.1: Set the Spring Config SpringBootApplication


src/main/java
com.springBoot.jpaMysql_1
GuestController.java
InitDataConfig.java
SpringBootJpaMySql1Application.java

```
package com.springBoot.jpaMysql_1;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
…
```

used to enable JPA repositories in a Spring Boot application.

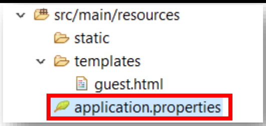
```
@SpringBootApplication
@EnableJpaRepositories("repository")
@EntityScan("domain")
public class SpringBootJpaMySql1Application
                              implements WebMvcConfigurer{
        …
}
```

to specify the package where the JPA entities are located.

21

21

Step 5.2: Set the Spring Config application.properties

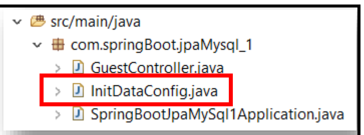


```
# MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/jpaexample?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=create-drop
#spring.jpa.hibernate.ddl-auto=create
#spring.jpa.hibernate.ddl-auto=none
```

22

22

Step 5.3: Set the Spring Config CommandLineRunner


src/main/java
com.springBoot.jpaMysql_1
GuestController.java
InitDataConfig.java
SpringBootJpaMySql1Application.java

```
package com.springBoot.jpaMysql_1;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import domain.Guest;
import repository.GuestRepository;
```

to mark the InitDataConfig class as a Spring Bean

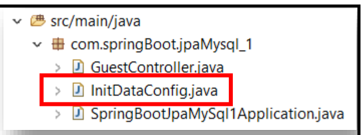
```
@Component
public class InitDataConfig implements CommandLineRunner {
```

CommandLineRunner is an interface that can be implemented by a Spring Bean to execute some code when the application starts up.

23

23

Step 5.3: Set the Spring Config CommandLineRunner

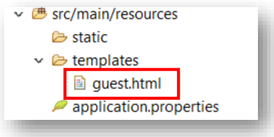

src/main/java
com.springBoot.jpaMysql_1
GuestController.java
InitDataConfig.java
SpringBootJpaMySql1Application.java

```
@Component
public class InitDataConfig implements CommandLineRunner {

        @Autowired
        private GuestRepository repository;

        @Override
        public void run(String... args) {
                repository.save(new Guest("Keters", "Sandra"));
                repository.save(new Guest("Blondeel", "Tania"));
                repository.save(new Guest("Blondeel", "Jurgen"));
                repository.save(new Guest("Blondeels", "Ann"));
        }
}
```

The CommandLineRunner interface has a single method called run.

24

24



Step 6: Add a Thymeleaf Page

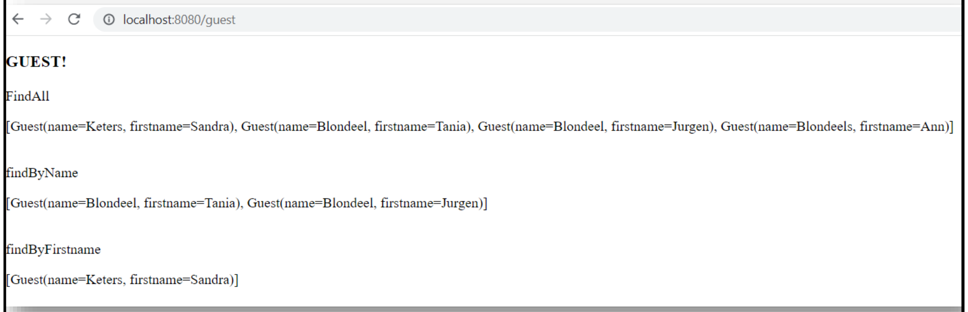
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
…
        <h3>GUEST!</h3>
        FindAll
        <p th:text="${guestList}"></p>
        <br>
        findByName
        <p th:text="${guestName}"></p>
        <br>
        findByFirstname
        <p th:text="${guestFirstname}"></p>
</body>
</html>
```

25

25

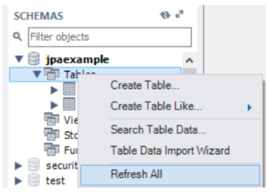
Step 7: Run the Spring Web App

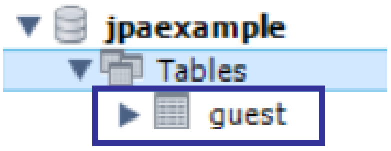
Run As → Spring Boot App



localhost:8080/guest

```
GUEST!

FindAll

[Guest(name=Keters, firstname=Sandra), Guest(name=Blondeel, firstname=Tania), Guest(name=Blondeel, firstname=Jurgen), Guest(name=Blondeels, firstname=Ann)]

findByName

[Guest(name=Blondeel, firstname=Tania), Guest(name=Blondeel, firstname=Jurgen)]

findByFirstname

[Guest(name=Keters, firstname=Sandra)]
```





26

26

3.1 Query

GuestRepository.java

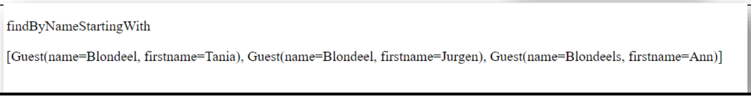
```
public interface GuestRepository extends
                    CrudRepository<Guest, Long> {
...
 @Query("SELECT g FROM Guest g WHERE g.name LIKE CONCAT(:username,'%')")
 List<Guest> findByNameStartingWith(
                    @Param("username") String username);
}
```

GuestController.java

```
   @GetMapping
    public String listGuest(Model model) { ...
          model.addAttribute("guestList2",
                 repository.findByNameStartingWith("blon"));
          return "guest";
    }
```

guest.html

```
findByNameStartingWith
<p th:text="${guestList2}"></p>
```



```
findByNameStartingWith

[Guest(name=Blondeel, firstname=Tania), Guest(name=Blondeel, firstname=Jurgen), Guest(name=Blondeels, firstname=Ann)]
```

27

27

3.2 NamedQuery

GuestRepository.java

```
public interface GuestRepository extends
                    CrudRepository<Guest, Long> {
...
//NamedQuery: Guest.findByNameStartingWith2
   List<Guest> findByNameStartingWith2(
                 @Param("username") String username);
```

Guest.java

```
@Entity
@NamedQueries({
   @NamedQuery(name="Guest.findByNameStartingWith2",
     query = """
               SELECT g
               FROM Guest g
               WHERE g.name LIKE CONCAT(:username,'%’)
               """)
})
@Getter @NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode(exclude = "id")
@ToString(exclude = "id")
public class Guest implements Serializable {
```

28

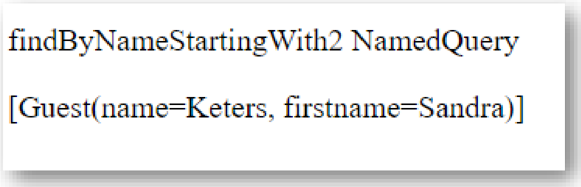
3.2 NamedQuery

GuestController.java

```
@GetMapping
 public String listGuest(Model model) { ...
        model.addAttribute("guestList3",
                  repository.findByNameStartingWith2("k"));
        return "guest";
 }
```

guest.html

```
findByNameStartingWith2 NamedQuery
<p th:text="${guestList3}"></p>
```



findByNameStartingWith2 NamedQuery

[Guest(name=Keters, firstname=Sandra)]

29

29

4. Composite primary key

```
@Embeddable
@AllArgsConstructor
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode
@Getter
@ToString
public class ComputerId implements Serializable {
          private static final long serialVersionUID = 1L;
          private String code;
          private int number;
}
```

Create a separate class that represents the composite key, annotated with @Embeddable

30

30

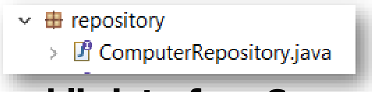
4. Composite primary key

```
@Entity
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode(of = "id")
@ToString
public class Computer {
        @EmbeddedId
        private ComputerId id;

        @Getter private String brand;

        public Computer(String code, int number, String brand)
        {
                this.id = new ComputerId(code, number);
                this.brand = brand;
         }
}
```

@EmbeddedId to reference the ComputerId class as the primary key

31

4. Composite primary key



```
public interface ComputerRepository
     extends CrudRepository<Computer, ComputerId> {
}
```

32

32