Tôi đã giới thiệu với các bạn đa số các thành phần cần có cho việc test trong Spring Boot.

Trong bài này, tôi sẽ tập trung vào việc giới thiệu với các các bạn cách chuẩn bị dữ liệu test.

Bài viết có sử dụng:

1. [Hibernate là gì?](https://loda.me/hibernate-la-gi-loda1554623701594)
2. [Spring JPA](https://loda.me/spring-boot-11-huong-dan-spring-boot-jpa-my-sql-loda1558687596060)
3. [Lombok](https://loda.me/general-huong-dan-su-dung-lombok-giup-code-java-nhanh-hon-69-loda1552789752787)
4. [Spring Boot](https://loda.me/spring-boot-0-series-lam-chu-spring-boot-zero-to-hero-loda1558963914472)

**Cài đặt**

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

<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0https://maven.apache.org/xsd/maven-4.0.0.xsd"><modelVersion>4.0.0</modelVersion><parent><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-parent</artifactId><version>2.0.5.RELEASE</version><relativePath/> <!-- lookup parent from repository -->

</parent><groupId>me.loda.spring</groupId><artifactId>example-independent-maven-spring-project</artifactId><version>0.0.1-SNAPSHOT</version><name>example-independent-maven-spring-project</name><description>Demo project for Spring Boot</description><properties><java.version>1.8</java.version></properties><dependencies><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-web</artifactId></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-devtools</artifactId><scope>runtime</scope><optional>true</optional></dependency><dependency><groupId>org.projectlombok</groupId><artifactId>lombok</artifactId><optional>true</optional></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-test</artifactId><scope>test</scope></dependency><!--spring jpa-->

<dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-data-jpa</artifactId></dependency><!--in memory database-->

<dependency><groupId>com.h2database</groupId><artifactId>h2</artifactId><scope>runtime</scope></dependency></dependencies><build><plugins><plugin><groupId>org.springframework.boot</groupId><artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins></build></project>

**Prepare**

Trước hết, giả sử chúng ta có một hệ thống quản lý công việc

*Todo.java*

@Data

@AllArgsConstructor

@NoArgsConstructor

@Entity

public class Todo {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String title;

private String detail;

}

*TodoRepository.java*

public interface TodoRepository extends JpaRepository<Todo, Long> {

List<Todo> findAll();

Todo findById(int id);

}

*TodoController.java*

@RestController

@RequestMapping("/api/v1")

@RequiredArgsConstructor

public class TodoRestController {

private final TodoService todoService;

@GetMapping("/todo")

public List<Todo> findAll() {

return todoService.getAll();

}

}

**@DataJpaTest**

Về cơ bản, chúng ta không thể mock hay làm giả dữ liệu của Database mãi được, nó sẽ là một lỗ hổng lớn trong hệ thống. Ngoài ra, bạn cũng sẽ không thể test được quá trình thao tác với Database của dự án.

Vậy nên sau cùng, chúng ta cũng sẽ phải test các thao tác vận hành với Database,

Nhưng chớ lo, để tập trung vào việc test các thao tác với Database, Spring Boot đã hỗ trợ chúng ta bằng @DataJpaTest

Kết hợp giữa @DataJpaTest và h2-database (Memory database) là Combo hoàn hảo.

@RunWith(SpringRunner.class)

/\*\*

\* Khi đánh dấu một class là @DataJpaTest.

\* Spring Boot sẽ load lên tất cả các Bean có liên quan tới tầng Repository

\* Thay vì load hết tất cả Bean như là @SpringBootTest

\*/

@DataJpaTest

public class DataJpaAnnotationTest {

@Autowired

private DataSource dataSource;

@Autowired

private JdbcTemplate jdbcTemplate;

@Autowired

private EntityManager entityManager;

@Autowired

private TodoRepository todoRepository;

@Test

public void allComponentAreNotNull() {

Assertions.assertThat(dataSource).isNotNull();

Assertions.assertThat(jdbcTemplate).isNotNull();

Assertions.assertThat(entityManager).isNotNull();

Assertions.assertThat(todoRepository).isNotNull();

}

}

@DataJpaTest có tác dụng khởi tạo các Bean cần thiết cho tầng Repository. Ngoài ra, nó không khởi tạo thêm các Bean thừa thãi nào khác. (Chuyên biệt hơn @SpringBootTest)

**Tạo Data**

Cách Fake dữ liệu đơn giản nhất là tự insert bằng repository

import org.assertj.core.api.Assertions;

import org.junit.After;

@Test

public void testTodoRepositoryByCode() {

todoRepository.save(new Todo(0, "Todo-1", "Detail-1"));

todoRepository.save(new Todo(0, "Todo-2", "Detail-2"));

Assertions.assertThat(todoRepository.findAll()).hasSize(2);

Assertions.assertThat(todoRepository.findById(1).getTitle()).isEqualTo("Todo-1");

}

@After

public void clean() {

todoRepository.deleteAll();

}

**@Sql**

Một cách làm hay hơn để chuẩn bị dữ liệu cho Test đó là sử dụng annotation @Sql

@Sql có tác dụng load một hoặc nhiều file scripts sql lên và thực thi.

ví dụ:

*createToDo.sql*

INSERT INTO todo (title, detail) VALUES ('Todo-1', 'Do homework');

INSERT INTO todo (title, detail) VALUES ('Todo-2', 'Walking');

Để chạy scripts này trong class Test, bạn làm như sau:

@RunWith(SpringRunner.class)

@DataJpaTest

public class SqlAnnotationTest {

@Autowired

private TodoRepository todoRepository;

@Test

@Sql("/createTodo.sql")

public void testTodoRepositoryBySqlSchema() {

Assertions.assertThat(todoRepository.findAll()).hasSize(2);

Assertions.assertThat(todoRepository.findById(1).getTitle()).isEqualTo("Todo-1");

}

}

Để chạy nhiều file sql cùng lúc, bạn sử dụng @SqlGroup

**Kết**