



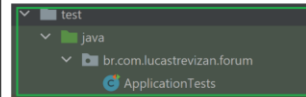
[REST-API]



## Testes automatizados

A importância de testes automatizados já nem é mais uma discussão, eles precisam existir. O **Spring Boot** já integra o módulo de teste que utiliza o **Junit**. O **Spring** inicializa a aplicação toda vez que precisa testar, ele faz isso pra poder fornecer contexto e infra que a aplicação precisa para ser testada:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```



```
@SpringBootTest
class ApplicationTests {

    @Test
    void contextLoads() {

    }

}
```

## Testar Repository

O **Spring Boot Test** fornece anotações específicas para o tipo de classe de teste, por exemplo, no teste de um repositório dá pra usar **@DataJpaTest**, essa anotação prove recursos pra teste de repository (*como o entitymanager específico pra teste, etc..*), testes do Spring permitem injeção de dependência:

```
@DataJpaTest
public class CursoRepositoryTest {

    @Autowired
    private CursoRepository cursoRepository;

    @Test
    public void deveriaRetornarUmCursoQuandoBuscadoPeloNome(){
        String nomeCurso = "HTML 5";
        Curso curso = cursoRepository.findByNome(nomeCurso);

        Assertions.assertNotNull( condition: curso != null);
        Assertions.assertEquals(nomeCurso, curso.getNome());
    }

    @Test
    public void deveriaRetornarNullQuandoCursoNaoEncontradoPeloNome(){
        String nomeCurso = "Jobster";
        Curso curso = cursoRepository.findByNome(nomeCurso);

        Assertions.assertNotNull( condition: curso == null);
    }

}
```

**Importante:** O **Spring Boot Tester** utiliza por padrão **bancos de dados em memória** (*usando os dados do arquivo data.sql*), então se você não tiver um banco em memória para teste ou quiser **usar um banco próprio**, é necessário avisar ao **Spring** através de uma **anotação**:

```
@DataJpaTest
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
public class CursoRepositoryTest {

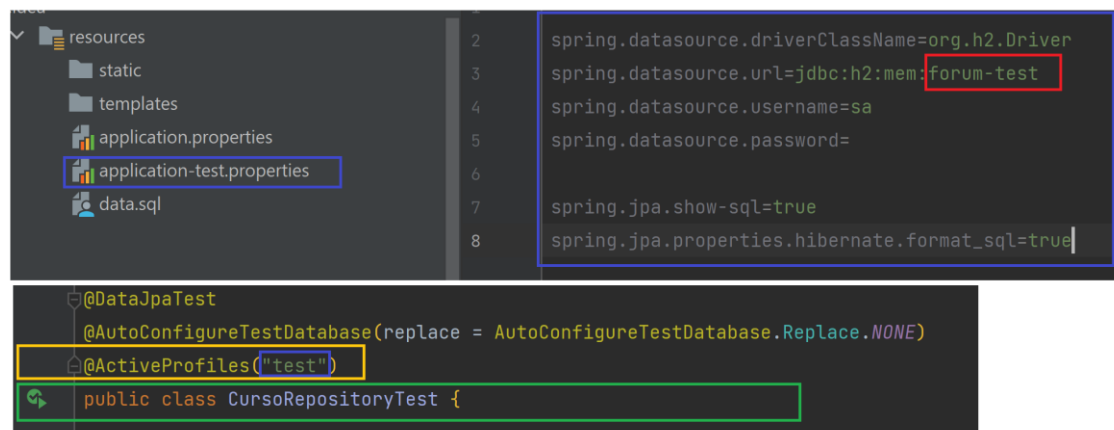
    @Autowired
    private CursoRepository cursoRepository;

    @Test
    public void deveriaRetornarUmCursoQuandoBuscadoPeloNome() {
```

## Usando Profile de Teste

Um bom uso do **Profile** é no caso de testes, por exemplo, eu posso ter um **banco de dados específico para executar os testes** de maneira isolada.

A ideia é criar um **novo arquivo properties** para representar o **perfil de teste**, esse que aponta para outro **banco de dados**, e na **classe de teste** avisar **QUAL PROFILE ela será executada** (*essa anotação em específico força o Spring a ativar o profile application-test*):



```
2 spring.datasource.driverClassName=org.h2.Driver
3 spring.datasource.url=jdbc:h2:mem:forum-test
4 spring.datasource.username=sa
5 spring.datasource.password=
6
7 spring.jpa.show-sql=true
8 spring.jpa.properties.hibernate.format_sql=true

@DataJpaTest
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
@ActiveProfiles("test")
public class CursoRepositoryTest {
```