02

Testes automatizados com Spring Boot

Transcrição

[00:00] Olá, bem-vindos de volta o curso de Spring Boot na Alura. Na aula de hoje eu quero discutir um assunto bem importante com vocês que também não conseguimos ver nos outros dois cursos de Spring Boot, que é sobre testes automatizados.

[00:13] Essa é uma excelente prática que os times de desenvolvimento utilizam no mundo todo para garantir a qualidade, garantir que o código que você escreveu está funcionando conforme o esperado, conforme atende às necessidades de negócio; e principalmente para garantir também a parte de manutenção. Quando você vai fazer alguns ajustes no código você não quer correr o risco de estragar alguma coisa que estava funcionando.

[00:37] Então no início do projeto, quando você tem poucas classes é mais simples de detectar problemas quando você faz mudança. Mas à medida que o projeto vai crescendo isso se torna mais complicado. Cada vírgula que você muda no projeto é um risco grande que você leva de estragar e quebrar, causar um bug em alguma funcionalidade que estava funcionando anteriormente.

[00:59] Então teste automatizado é algo extremamente importante que poderíamos fazer na nossa API, criar testes para garantir que aquelas funcionalidades estão funcionando conforme o esperado.

[01:11] No caso do Java nós temos o JUnit, que é a principal biblioteca para fazer testes automatizados. E o Spring Boot também tem um módulo específico para fazer testes automatizados.

[01:22] No caso do Spring Boot ele já fornece algumas classes, algumas anotações focadas para você escrever testes automatizados das classes, dos componentes do Spring, como o Controller, Service, Repositories, dentre outros componentes. Vamos aprender nessa aula como funcionam esses testes no Spring Boot.

[01:40] Uma coisa importante, o foco não é em teste automatizado, não é no JUnit. Então não vou ensinar o básico de testes automatizados, o que é um teste automatizado, quais são as vantagens, como funciona, como eu escrevo um teste automatizado em Java, como eu uso o JUnit.

[01:56] Nós vamos focar específico no Spring Boot, então vamos considerar que você já sabe o que é teste automatizado e já sabe escrever teste automatizado em Java utilizando JUnit.

[02:06] Inclusive, na Alura nós temos um treinamento de testes, o treinamento de TDD com Java. Então caso você não tenha conhecimento nós recomendamos que você faça esse treinamento e aprenda um pouco sobre como funcionam testes automatizados em Java para que possamos focar no Spring Boot e não desviar muito a atenção com outros assuntos. Então o foco sempre será no Spring Boot em si.

[02:30] Antes de começarmos a escrever os testes eu queria mostrar uma coisa importante para vocês. Não sei se vocês lembram, mas nós criamos o nosso projeto naquele Spring Initializr, naquele site “start.spring.io”.

[02:41] Naquele site nós informamos algumas coisas do projeto e escolhemos quais dependências queremos baixar no nosso projeto. Eu vou abrir o `pom.xml` . Assim que você baixa o projeto no “start.spring.io” e abre o `pom.xml` do seu projeto você verá que ele já configurou tudo para você.

[02:59] Tem a dependência do Spring Boot, tem as informações do `groupId` , do `artifactId` e as dependências que você adicionou no projeto. Então tem o

módulo web, o módulo do data JPA, módulo de cache, segurança, tem todos os módulos.

[03:14] Mas se você for lá embaixo você verá uma coisa curiosa. Veio também uma dependência que é um módulo de testes do Spring Boot.

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

COPIAR CÓDIGO

[03:25] Então o Spring Boot tem um módulo específico focado em testes automatizados. Inclusive, uma das dependências desse módulo é o JUnit. Então não precisaremos colocar a dependência de testes do Spring Boot e nem o JUnit, porque elas já estão adicionadas no projeto.

[03:43] E nós não escolhemos essa dependência. Quando você cria o projeto no site do Spring Initializr você não escolhe. Mas quando você baixa o projeto, ele já automaticamente vem com esse módulo de testes.

[03:56] Então o Spring está assumindo que você vai escrever testes automatizados, uma excelente prática, usada no mundo inteiro. Ele vai assumir que você vai escrever testes automatizados. Então ele já vem incluído automaticamente como uma dependência.

[04:16] Então mesmo que você não queira, vai ter essa dependência. Claro, depois você pode tirar.

[04:20] E outra coisa também que talvez alguns de vocês já tenham visto é que no projeto, além do source folder, `src/main/java` e `src/main/resources`, veio junto o `src/test/java`, que é onde você coloca as classes, os pacotes de coisas relacionadas com testes automatizados.

[04:40] E se você clicar no “src/test/java”, você verá que ele já cria um pacote, que é o mesmo pacote raiz da aplicação, “br.com.alura.forum”, e cria uma classe de testes de exemplo para você. Então vamos abrir essa classe que ainda não tínhamos visto até então.

[04:54] Então é uma classe de teste Java, tem `public class ForumApplicationTests` . E depois ele cria um método de teste com `@Test` do pacote do JUnit, “org.junit.Test”. Só que é um teste vazio, que não faz nada.

[05:09] Mas já vem uma classe de testes de exemplo. Eu vou até colocar um teste só para rodar e ver o que acontece. Eu vou colocar `assert` e verificar se o `true` está valendo `true` , só par fazer um teste que vai passar: `Assert.assertTrue(true);` .

[05:25] E vamos rodar, clicando com o botão direito, indo na opção “Run As > JUnit Test”. O que será que acontece se eu rodar esse teste? Vai passar, só tem um método de teste, eu estou testando se true é true, e sim, vai dar verdadeiro.

[05:40] Vamos olhar na aba do JUnit. Perceba que eu rodei e não apareceu nada. E a aba do Console ficou em negrito. Se clicarmos nela, veremos que ele está imprimindo um monte de coisa.

[05:53] Então esse teste está subindo o projeto, ele está *startando* o Tomcat e inicializando nosso projeto. E por que ele fez isso? Porque se você reparar na classe de teste, ela é uma classe de testes Java usando o JUnit, mas ela tem duas coisas a mais em cima da classe, tem duas anotações e é aqui que vai entrar o Spring Boot.

```
package br.com.alura.forum;

import org.junit.Assert;

@RunWith(SpringRunner.class)
@SpringBootTest
```

```
public class ForumApplicationTests {

    @Test
    public void contextLoads() {
        Assert.assertTrue(true);
    }
}
```

COPIAR CÓDIGO

[06:17] Tem a anotação `@RunWith` , que vem passando como parâmetro esse `SpringRunner.class` , que é uma classe do pacote "org.springframework.test", do módulo de teste do Spring. E vem uma anotação `@SpringBootTest` . Então isso é o Spring Boot. Abaixo disso é Java e JUnit.

[06:38] O Spring Boot entra com algumas anotações para fornecer alguma infraestrutura para você escrever seus testes, porque quando quisermos testar um Controller, um Repository, vamos depender de recursos do servidor, recursos do Spring. E então precisamos dessa infraestrutura do Spring.

[06:56] Por isso que quando rodamos o teste ele subiu o servidor, inicializou o servidor, finalizou o teste. Está rodando ainda, pode ser que demore um pouco.

[07:05] Mas a ideia é justamente essa, para escrever os testes utilizando o módulo do Spring Boot, toda vez que você rodar o teste o Spring vai carregar o servidor. É como se você estivesse rodando o projeto. Só que ele roda o projeto, sobe tudo que tem que subir, inicializa tudo que tem que inicializar, executa os seus testes um por um, termina e derruba o servidor.

[07:29] E passou, porque meu teste é um teste falso, só para rodar e ver que ele vai passar.

[07:37] E na aba Console ele terminou e finalizou. No Console ele só vai dar uma *exception* por causa do Spring Boot Admin, porque eu não estou com um projeto do Spring Boot Admin rodando, então ele fica reclamando. Mas ele rodou normalmente.

[07:49] Então esse é o módulo do Spring Boot para nos ajudar a escrever testes automatizados. Então essa aula era só para mostrar para vocês, discutir essas questões. Mas no próximo vídeo vamos de fato começar a escrever o teste, por exemplo, da nossa classe Controller, da nossa classe Repository, como é que eu testo esses componentes do Spring.

[08:06] Então já vimos que vamos precisar usar algumas anotações, alguns recursos do Spring. Mas no próximo vídeo nós vemos como funciona e escrevemos o nosso primeiro teste automatizado com Spring Boot. Vejo vocês no próximo vídeo. Um abraço e até lá.