05

Spring MVC com Spring Boot

Transcrição

[00:00] Aqui no slide nós temos uma aplicação Web utilizando o Servlet e já separado em um padrão de projeto chamado MVC. Separando as responsabilidades da lógica da aplicação, da visão da aplicação através de um "controller" que vai receber as requisições e vai redirecionar essas requisições para uma "action" especifica que sabe qual modelo deve ser utilizado e qual classe de serviços da nossa aplicação deve ser utilizada para responder o usuário. Então no caso, por exemplo: uma requisição para "localhost:8080/pedidos", um "get" para "/pedidos".

[00:41] Essa requisição chegaria na nossa Servlet, a Servlet verificaria qual é o "path" que está sendo chamado - por exemplo: "/pedidos" - e identificaria qual a "action" que sabe responder isso. Aí pegaria o "HTTP Servlet Request", mandaria para essa "action" que iria usar no modelo, por exemplo, o pedido service para pegar e dar um "[GET PEDIDO]" e aí retornar do banco de dados. O modelo vai no banco de dados e pegar a lista de pedidos que tem cadastrada.

[01:11] Então dado que a "action" já retornou essa informação, o que a Servlet precisa fazer é redirecionar isso para a "JSP" específica para processar essa requisição. Essa "JSP" tem a inteligência de pegar essa lista que foi recuperada pela "action" através do modelo, recuperar essa lista de pedidos e gerar um "HTML" que vai ser enviado para o usuário.

[01:35] Então aqui nós temos uma separação no "MVC". Nós vamos continuar utilizando essa mesma separação, só que usando Spring MVC que é um projeto do Spring. Só que hoje, a primeira coisa que vamos fazer é o Tomcat, que é al

que normalmente baixamos e integramos com o Eclipse, integramos o nosso projeto. Nós não precisamos nos preocupar porque o Spring Boot já vem com ele integrado.

[02:01] Então no setup do Spring Boot, ele já gera para nós a aplicação com Tomcat e com muitas coisas configuradas. O Spring Boot meio que gerencia o setup do projeto. E o Servlet é de um outro projeto da Spring chamado de Spring MVC.

[02:21] Então o Spring MVC tem o "Dispatcher" de Servlet, ele tem o Servlet dele, o que já foi configurado também junto com o Spring Boot. O Spring Boot configurou tanto o servidor de aplicação quanto configurou o "Spring MVC" e a Servlet está lá configurada também. É algo que vamos ver que está bem automático.

[02:39] O que acontece? As requisições batem no Spring MVC, mas ele não sabe chegar no modelo. Por exemplo: se fizermos um "[GET PEDIDO]" e a nossa requisição bater em uma aplicação Spring MVC, ele precisa delegar para alguém, para alguma "action", a responsabilidade de tratar essa requisição. Então nós vamos adicionar antes do modelo, para integrar o nosso modelo com o Spring MVC, nós vamos criar um "controller", que é uma classe.

[03:09] E cada método das nossas classes "controller", são "actions" diferentes. Então se eu faço um "localhost:8080/pedido", eu vou bater no pedido "controller" que tem um método que vai no modelo para pegar uma lista de pedidos.

[03:27] E ele vai adicionar isso na requisição do usuário para ser processado pela "view", para a "view" poder gerar uma interface para o usuário - que no nosso caso, nós não vamos utilizar "JSP", mas sim Thymeleaf. Vamos utilizar também uma Expression Language chamada de Spring Expression Language, que é bem parecida com Expression Language Default. E vamos utilizar uma biblioteca de CSS chamada Bootstrap, que vai dar o layout da nossa aplicação para vários dispositivos.

[03:59] Automaticamente, ele ajusta a tela à nossa interface para dispositivos diferentes. O Bootstrap tem uma biblioteca bem legal para isso, nós só precisamos saber configurá-la. Então o que nós vamos fazer nesse vídeo, dado esse entendimento, é criar o nosso projeto com Spring Boot, mas o nosso projeto utilizando Spring MVC.

[04:17] Como fazemos isso? A primeira coisa que eu vou fazer é entrar no site do Spring, no menu “projects” tem todos os projetos do Spring, Spring Security, Spring Mobile, AMQP, Spring Rest e tal. Nós vamos utilizar o Spring Boot, mas integrado no Spring Frameowork, o Spring MVC mais especificamente.

[04:40] Nós também utilizar mais para a frente o Spring Data e o Spring Security. Mas por agora. é o Spring Framework utilizando o Spring Boot. Eu vou clicar nesse aqui e na aba “Learn” eu vou clicar no “Reference Doc.”, a documentação do Spring. Na documentação eles dividem em categorias. Então, por exemplo: na categoria “Web Servlet”, nós temos uma documentação do Spring MVC. Você pode abrir e ler como é todos os detalhes disso.

[05:06] E tem vários outros projetos. Mas o que eu preciso é só desse aqui, o “getting started”. Como eu começo um projeto usando o Spring MVC com o Spring Boot? Então nós viemos para uma outra página que ele fala sobre a história do Spring e filosofia do design do Spring etc.

[05:24] E no “Getting Started” tem o seguinte: você pode usar o “start.spring.io” para gerar o projeto base para você. Esse “start.spring” é uma aplicação Web que nos permite já criarmos um projeto inicial com as dependências do que precisamos na nossa aplicação.

[05:44] Ele já vem pré-configurado com Maven e não Gradle, mas tem essas opções. Você pode utilizar Java, Kotlin e Groovy. Nós vamos utilizar Java. O Spring Boot a última versão é o 2.2.2 e no Project Metadata eu vou colocar o seguinte: “br.com.alura.mvc” e no nome do projeto eu vou colocar “mudi”.

[06:06] Aqui tem as dependências que nós vamos adicionar na nossa aplicação. A primeira dependência eu vou colocar “Web”, que é o Spring Web - que é o quê? É o Spring MVC configurado.

[06:17] Também tem “Rest” configurado utilizando como “default embedded container”, o Apache Tomcat. Então eu vou selecionar esse aqui. O segundo é o Thymeleaf, que vai gerenciar as nossas “views”, que vai gerar o HTML. Então eu vou adicionar o Thymeleaf e o DevTools também. O Spring DevTools é um projeto que faz o restart da nossa aplicação enquanto estamos programando. Então você vai alterando as classes, vai criando funcionalidades novas, ele vai detectando que tem mudança e já vai aplicando isso.

[06:48] Então ele facilita bastante o nosso trabalho e agiliza a programação. Não temos que ficar derrubando servidor, subindo de novo etc. Então, “Generate”. Gerei um projeto. Já tinha gerado um. Vou pegar esse “mudi” e jogar dentro do “workspace-spring-mvc”.

[07:09] Vou abrir esse aqui e jogar esse projeto aberto dentro do “workspace-spring-mvc”. Beleza! Eu vou fechar esse aqui, vou abrir o Eclipse. O Eclipse já está apontando para essa pasta “workspace-spring-mvc”, a minha pasta. Vou dar um “Launch”.

[07:33] O que eu preciso fazer é importar o meu projeto e você precisar saber como importar um projeto Maven. Então, provavelmente você vai estar com ele aberto desse jeito, ou seja, essa janela de “import” vai estar assim. Se você vier em “General”, ele tem um “import”, “Existing Projects into Workspace”.

[07:52] Só que o nosso projeto tem uma característica a mais porque ele é um projeto Maven que tem um “pom.xml”, um arquivo com as nossas dependências que devem ser processadas. Ele tem uma estrutura um pouco diferente.

[08:02] Um projeto Maven tem um padrão diferente de pastas. Então “Existing Maven Projects” é o que nós temos que selecionar. Aí você vai selecionar o

diretório que é só clicar na pasta “mudi” que está dentro do “workspace-spring-mvc”. Seleciona a pasta, “finish”. Ele vai baixar nesse momento todas as dependências que estão no “pom.xml”. No seu caso pode demorar um pouco mais. Como eu já fiz isso aqui antes, as dependências já foram baixadas para o meu computador.

[08:30] Então como dependência, o que nós temos? Primeiro, temos um “parent” aqui com a versão do Spring Boot, ou seja, todas as dependências do Spring que nós vamos adicionar no nosso projeto estão relacionadas à versão 2.2.2 do Spring Boot, porque nós vamos utilizar basicamente dependências dele.

[08:46] Alguma ou outra não são do Spring Boot especificamente, mas a maioria sim e elas já vão estar associadas à essa versão 2.2.2. Por isso que na hora que eu adiciono uma dependência, “eu quero o Spring Boot Starter Thymeleaf”, eu não preciso dizer qual é a versão, ele já vai pegar na versão 2.2.2 do Spring Boot.

[09:04] Nós temos as três dependências que nós adicionamos, o Starter Thymeleaf, o Starter Web e o Spring Boot DevTools. Ele adicionou a dependência para teste que não vamos utilizar nesse treinamento. Então, pronto! Ele também criou para nós uma classe chamada de “MudiApplication.java”. Essa classe tem o método “main” e se você rodar essa classe, a nossa aplicação vai subir já com o Tomcat configurado e a nossa aplicação deployada nesse Tomcat]. Vamos ver isso!

[09:34] Vou dar um “Run As”, “Java Application”, nós abrimos o console e ele printou que está usando o Spring Boot na versão 2.2.2 e na penúltima linha aparece “Tomcat started on port(s): 8080 (http)”, ou seja, ele já subiu o Tomcat e também adicionou as configurações do Thymeleaf. Ele até diz que ele vai ficar na pasta “templates”. Então em “Source/main/resources” tem a pasta “templates” - que é onde vai ficar as nossas “views” futuramente.

[10:10] Mas é isso que temos para esse vídeo! É só criarmos e entendermos quais as tecnologias que nós vamos usar aqui - como baixar, como configurar e como criar um projeto utilizando Spring Boot.

[10:20] Na próxima aula nós vamos fazer um “Hello World” utilizando já o Thymeleaf e passando informações para o Thymeleaf utilizando o Spring Boot. Vamos ver também como se faz um “controller” e como essa integração é feita. Até o próximo vídeo!