Olá! Tudo bem?

Nesta videoaula, você dará continuidade ao estudo do Framework Spring. Agora, você irá conhecer um elemento bastante importante no uso desse framework para o desenvolvimento web, o Controller, que nada mais é que uma classe Java que possui alguns métodos responsáveis por executar ações, como enviar informações para o usuário, receber informações do usuário e processá-las.

No seu projeto, o Controller terá algumas responsabilidades, tais como:

- Interceptar as requisições HTTP que sua aplicação recebe, como a requisição que o usuário realiza através do navegador;
- Converter os dados recebidos de uma requisição HTTP para um modelo de objeto conhecido para dentro da sua aplicação;
- Enviar dados para a camada de persistência;
- Receber dados processados pela sua aplicação e retornar para a camada responsável pela apresentação dos dados para o usuário.

Para configurar uma classe Java, para atuar como um Controller, é bastante simples, basta adicionar a anotação @Controller do tipo org.springframework.stereotype.Controller logo acima do nome da classe que você está desenvolvendo. Para exemplificar o conteúdo, você irá criar a sua classe que atuará como um Controller. Então, com o Eclipse aberto, navegue até o seu projeto. Em seguida, acesse o menu de opções, utilizando o seu teclado através da tecla Aplicação. No menu, você deve navegar até a opção New e, posteriormente, até a opção Class. Com a janela de criação de classe aberto, você irá nomear a classe como SpringController. Também deve ser informado o pacote em que a classe irá pertencer. Para isso, adicione a informação br.com.lead.controller na caixa em que é solicitado o pacote. Por fim, finalize o processo acionando o botão Finish.

#Audiodescrição: A imagem mostra a caixa de diálogo “New Java Class”. Nela, há o texto “Java Class. Create a new Java class”. O campo “Source folder”, está preenchido por “CatalogoDeFilmes/src”; abaixo dele, temos o campo: “Package”, preenchido por “br.com.lead.controller”. Em seguida, o campo “Name” está preenchido por “SpringController”; a caixa de seleção “Modifiers” que está marcada em “public”. Em seguida, o campo “Superclass” que está preenchido por: “java.lang.Object” e a caixa de seleção que aparece em seguida “Which method stubs would you like to create?”, tem a opção “Inherited abstract methods” marcada. Abaixo, há o botão “Finish”, que está selecionado e o botão “Cancel”.

Com a classe criada, você deve adicionar a anotação @Controller logo acima do nome da classe.

#Audiodescrição: A imagem mostra o código:
“package br.com.lead.controller;
import org.springframework.stereotype.Controller;
@Controller
public class SpringController {
}”

Com a anotação @Controller adicionada, a sua classe SpringController estará apta a atuar como um Controller.

Embora a Controller tenha sido criada, ela ainda está incompleta. É importante lembrar que toda Controller necessita de algum método para que possa executar alguma ação, interceptar uma requisição e dar uma resposta a requisição recebida.

Agora, você deve criar um método que irá receber dados referentes a um filme, como nome, gênero e ano. O método criado irá se chamar adicionaFilme, que será um método público e deverá retornar uma String, recebendo três parâmetros: o primeiro, uma String nome; o segundo, uma String gênero; e o terceiro, um Integer ano. Depois, você deve definir o mapeamento da requisição que fará com que o método seja executado. Em seguida, realize esse mapeamento através da anotação @RequestMapping do tipo org.springframework.web.bind.annotation.RequestMapping. Essa anotação recebe uma String como parâmetro, que é referente ao mapeamento da URL que irá acionar o método adicionaFilme. Depois disso, você deve passar o valor adicionaFime nesse parâmetro e deve adicionar a anotação @RequestMapping logo acima do método desejado, ou seja, no método adicionaFilme.

#Audiodescrição: A imagem mostra o código: “@RequestMapping("adicionaFilme")

public String adicionaFilme(String nome, String genero, Integer ano)”

No corpo desse método, você deve criar um único objeto do tipo String, que irá concatenar cada um dos parâmetros recebidos em uma única String. Por isso, você deve separar cada elemento por espaço em branco. Em seguida, você deve utilizar esse objeto criado como o retorno do método.

Sua classe ficará da seguinte maneira:

#Audiodescrição: A imagem mostra a seguinte classe:

```
package br.com.lead.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class SpringController {
	@RequestMapping("adicionaFilme")
	public String adicionaFilme(String nome, String genero, Integer ano) {
		String resposta = nome + " " + genero + " " + ano;
		return resposta;
	}
}
```

Até aqui, você aprendeu a respeito do uso da Controller do Spring, entendeu como funciona e suas principais responsabilidades, dentro de um projeto, além de saber como mapear um método dentro da classe Controller para que este seja acionado, de acordo com uma URL informada. Contudo, o seu método ainda não está completo. Por isso, perceba que, se você deixar o método da maneira como ele está, o Spring procuraria por uma página do tipo JSP lá na pasta WebContent do seu projeto. Essa página possuiria o mesmo nome que o valor retornado pelo método, para assim poder exibir o retorno para o usuário que enviou a requisição. Entretanto, não é isso que você quer, pois deseja-se que o próprio Spring se encarregue de passar essa resposta para o usuário, sem a necessidade de ter um arquivo com o nome igual ao valor retornado. Para isso, você deve utilizar a anotação @ResponseBody de org.springframework.web.bind.annotation.ResponseBody.

A anotação @ResponseBody possui a finalidade de deixar, para o Spring, a responsabilidade de capturar o retorno deste e escrevê-lo no corpo de uma resposta HTTP, retornando essa resposta para o usuário.

Dessa forma, deve-se adicionar a anotação @ResponseBody logo acima do método desejado, ou seja, o método adicionaFilme. Portanto, o seu método ficará assim:

#Audiodescrição: A imagem mostra o seguinte método:

```
“@RequestMapping("adicionaFilme")
@ResponseBody
public String adicionaFilme(String nome, String genero, Integer ano) {
String resposta = nome + " " + genero + " " + ano;
```

```
return resposta;
}
}"
```

Para que você possa verificar o resultado que acabou de criar, é preciso executar o seu projeto no Tomcat. Para isso, siga até a aba Servers do Eclipse, selecione o servidor configurado e navegue, no menu de opções, através do teclado, utilizando a tecla Aplicação. Em seguida, navegue até a opção Start. Com o Tomcat rodando, envie a requisição para o método que mapeou. Depois, adicione a seguinte URL no seu navegador: http://localhost:8080/CatalogoDeFilmes/adicionaFilme?nome=titanic&genero=drama&ano=1998 (http dois pontos barra barra localhost dois pontos 8080 barra catalogo de filmes barra adiciona filme interrogação nome igual titanic & gênero igual drama & ano igual 1998).
Feito isso, você deve perceber que a URL é composta, inicialmente, pelo caminho do projeto http://localhost:8080/CatalogoDeFilmes (http dois pontos barra barra localhost dois pontos 8080 barra catalogo de filmes), seguido do mapeamento que realizou para o método adicionaFilme, seguido pelos parâmetros passados para o método ?nome=titanic&genero=drama&ano=1998 (interrogação nome igual titanic & gênero igual drama & ano igual 1998). Ao inserir a URL no navegador, você deve se deparar com a resposta que definiu no método, ou seja, a informação do nome do filme, concatenado com o gênero e ano do filme. Logo, ficará da seguinte forma: titanic drama 1998.

#Audiodescrição: a imagem mostra um recorte de um navegador de internet. Nele, temos a barra de ferramentas, com as opções “Voltar”, “Avançar”, “Recarregar” e “Página inicial”, além da barra de endereço, com o seguinte endereço URL: “localhost:8080/CatalogoDeFilmes/adicionaFilme?nome=titanic&genero=drama&ano=1998”..
Logo abaixo, no corpo da página, temos o texto: “titanic drama 1998”.

Pronto! A requisição foi realizada e o método adicionaFilme conseguiu interceptá-la e executar sua ação, retornando, assim, uma resposta para o navegador.

A requisição que foi realizada ao inserir a URL passada no navegador. Por isso, foi uma requisição HTTP do tipo GET e o seu método conseguiu interceptar. Quando definiu o mapeamento do método, através da anotação @RequestMapping, você pode definir também sobre quais tipos de requisição HTTP esse método irá executar. Caso isso seja necessário, você pode passar a

seguinte informação por parâmetro na anotação method=RequestMethod.GET. Nesse caso, o método será executado apenas nas requisições do tipo GET.

Chegamos ao final de mais uma videoaula do Curso Java Web. Nela, você estudou a respeito do elemento Controller do framework Spring, entendendo um pouco do seu funcionamento e conhecendo como criar um Controller, mapear os métodos de um Controller e realizar o retorno a uma requisição, utilizando-se dos artifícios do Spring. Portanto, é importante que você explore o uso do framework Spring, navegue nas suas possibilidades e sempre coloque em prática o aprendizado adquirido até aqui para que suas habilidades aumentem diariamente.

Referências

Apostila do Curso FJ-21 Java para Desenvolvimento Web. Disponível em: <https://www.caelum.com.br/apostila-java-web/>. Acesso em: 23 maio 2020.

DUTTA, Prashant. **Spring Controllers**. Disponível em: <https://www.baeldung.com/spring-controllers>. Acesso em: 23 maio 2020.

PARASCHIV, Eugen. **Spring Request Mapping**. Disponível em: <https://www.baeldung.com/spring-requestmapping>. Acesso em: 23 maio 2020.