Olá, tudo bem?

Nesta videoaula, você estudará sobre: migração do comportamento de um servlet para o controller do Spring MVC, criação de um recurso (ou funcionalidade) para a consulta de filmes, compreender a utilidade da classe ModelAndView do Spring MVC e enviar dados para uma JSP a partir do ModelAndView.

Essa aula objetiva uma abordagem mais prática, pois sua finalidade é integrar todos conceitos aprendidos de JPA com o Spring MVC. Para isso, serão utilizados recursos de aulas anteriores como EntityManager, Controller, RequestMapping e RequesParam para te ajudar a migrar o comportamento do servlet PersisteFilmeServlet para o controller FilmeController.

Primeiramente, você irá criar a classe FilmeController no pacote br.com.lead.controller (br ponto com ponto lead ponto controller). Essa classe ficará encarregada pelo cadastro de novos filmes e autores. Ela conterá a anotação (ou diretiva) @Controller (arroba controller) usada para indicar que se trata de um controlador do Spring MVC. No código descrito a seguir é apresentado o conteúdo da classe FilmeController.*java* sem nenhum comportamento, pois nela está contida uma declaração da classe sem método algum e uma anotação @Controller (arroba controler) informando que se trata de controlador do Spring MVC.

#Audiodescrição: A imagem mostra uma captura de tela do Eclipse Workspace em que a aba “FilmeController.java” está selecionada. Logo abaixo, temos o seguinte código:

```
package br.com.lead.controller;

import org.springframework.stereotype.Controller;

@Controller

public class FilmeController {

}
```

No controller FilmeController você irá disponibilizar o recurso (ou funcionalidade) persistirFilme que pode ser acessado a partir do caminho /persistir-filme (barra persistir traço filme) em uma requisição HTTP do tipo GET. O método persistirFilme deve receber como parâmetro nome, genero, ano, nomeAutor e dataNascimentoAutor todos do tipo String e antes do tipo serem anotados com @RequestParam (arroba request param). A anotação (ou

diretiva) @RequestMapping (arroba request mapping) do Spring MVC declara um recurso (ou funcionalidade) para ser acessada a partir de um caminho, enquanto a @RequestParam (arroba request param) é responsável por capturar os parâmetros enviado na requisição. O código a seguir apresenta o conteúdo da classe FilmeController.*java* com o método persistirFilme responsável por realizar o cadastro de novos filmes e autores.

# Audiodescrição: A imagem mostra uma captura de tela do Eclipse Workspace em que a aba "FilmeController.java" está selecionada. Logo abaixo, temos o seguinte código:

```
package br.com.lead.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class FilmeController {

	@RequestMapping(value = "/persistir-filme", method = RequestMethod.GET)

	public String persistirFilme(@RequestParam String nome, @RequestParam String genero, @RequestParam String ano, @RequestParam String nomeAutor, @RequestParam String dataNascimentoAutor) {

		return null;

	}

}
```

Nessa etapa você irá migrar todas as instruções do servlet PersisteFilmeServlet para o método persistirFilme que se encontra no controller FilmeController. Com isso, ele terá os comandos necessários para persistir (ou salvar) no banco de dados um novo filme e autor. Navegue até a classe PersisteFilmeServlet localizada em

src/br/com/lead/servlet/PersisteFilmeServlet.java (no pacote br ponto com ponto lead ponto servlet na classe PersisteFilmeServlet.java) e copie todos os comandos do método service e depois os cole no corpo do método persistirFilme no controller FilmeController. O retorno dele será uma String contendo o nome do JSP encarregado por exibir uma página em HTML.

Por padrão, o Spring MVC avalia o conteúdo repassado ao cliente, verificando se deve ou não encaminhar os dados processados a uma view. Nesse caso, o retorno será o nome do JSP que será exibido após os dados terem sido salvos na base de dados. O código a seguir apresenta a versão final da classe FilmeController.*java*.

# Audiodescrição: A imagem mostra uma captura de tela do Eclipse Workspace em que a aba “persistir-filme-view.jsp” está selecionada. Logo abaixo, temos o seguinte código:

```
package br.com.lead.controller;

import java.time.LocalDate;

import javax.persistence.EntityManager;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import br.com.lead.modelo.Autor;

import br.com.lead.modelo.Filme;

import br.com.lead.util.JPAUtil;

@Controller

public class FilmeController {

    @RequestMapping(value = "/persistir-filme", method = RequestMethod.GET)

    public String persistirFilme(@RequestParam String nome, @RequestParam String genero, @RequestParam String ano, @RequestParam String nomeAutor, @RequestParam String dataNascimentoAutor) {

        Autor autor = new Autor();

        autor.setDataNascimento(LocalDate.parse(dataNascimentoAutor));
```

```
            autor.setNome(nomeAutor);
            Filme filme = new Filme(nome, genero, Integer.valueOf(ano));
            filme.setAutor(autor);
        EntityManager entityManager = JPAUtil.getEntityManager();
       entityManager.getTransaction().begin();
        entityManager.persist(autor);
        entityManager.persist(filme);
        entityManager.getTransaction().commit();
        entityManager.close();
        return "persistir-filme-view";
    }
}
```

Para concluirmos a etapa de migração da funcionalidade que cadastra filmes do servlet PersisteFilmeServlet para o controller FilmeController, você irá criar o JSP persistir-filme-view.jsp (persistir traço filme traço view ponto jsp) no diretório WebContent/WEB-INF/views (webcontent barra web inf barra views). Ele apresentará a mensagem "Salvo com sucesso" no formato HTML. A página vai conter um título chamado mensagem e no corpo o informativo de que o filme foi salvo com sucesso. O código a seguir apresenta o conteúdo do JSP persistir-filme-view.jsp (persistir traço filme traço view ponto jsp).

# Audiodescrição: A imagem mostra uma captura de tela do Eclipse Workspace em que a aba “FilmeController.java” está selecionada. Logo abaixo, temos o seguinte código:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "https://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
        <meta charset="UTF-8">
        <title>Mensagem</title>
</head>
<body>
        <h1> Salvo com sucesso </h1>
</body>
</html>
```

Pronto! Agora que você concluiu a migração do comportamento do servlet PersisteFilmeServlet para o controller FilmeController. O próximo passo é verificar se o controller FilmeController está respondendo às requisições, mas antes de inicializar o Tomcat remova o servlet PersisteFilmeServlet do projeto, apague a classe PersisteFilmeServlet.java, para que não exista dois encarregados por responder a /persistir-filme (barra persistir traço filme). Acesse o Eclipse e navegue até o menu Run e escolha a opção Run As. Em seguida, acione Run on Server, isso irá inicializar o Tomcat e a aplicação de catálogo de livros.

Agora abra o seu navegador e digite a URL localhost:8080/CatalogoDeFilmes/persistir-filme?nome=Avatar&genero=Ficção&ano=2009&nomeAutor=Sam%20Worthington&dataNascimentoAutor=1976-08-02 (localhost dois pontos 8080 barra CatalogoDeFilmes barra persistir traço filme interrogação nome igual a Avatar e comercial genero igual a Ficção e comercial ano igual a 2009 e comercial nomeAutor a igual Sam Worthington e comercial dataNascimentoAutor igual a 1976-08-02). Isso vai disparar uma requisição para o recurso persistir-filme que irá cadastrar um novo filme.

Agora, você aprenderá a adicionar o recurso consulta de filmes ao controller FilmeController. Esse recurso (ou funcionalidade) pode ser acessado através do caminho /consultar-filme (barra consultar traço filme) em uma requisição HTTP do tipo GET. É importante ressaltar que o recurso para a consulta de filmes deve receber um único parâmetro que é o código do filme a ser consultado. Primeiramente, você deve realizar o resgate das informações do filme na base dados utilizando o método find da classe EntityManager. Após obter o resultado crie uma instância de ModelAndView e a partir do método setViewName informe para qual JSP vai encaminhar o resultado. Envie para consultar-filme-view (consultar traço filme traço view) o JSP que será criado para exibir os dados referentes ao filme. Após isso, informe os dados que irá repassar ao JSP consultar-filme-view (consultar traço filme traço view) que são nome,

genero e ano, isso deve ser feito utilizando o método addObject do objeto ModelAndView. Dessa forma, os dados que constituem o filme buscado serão transmitidos e exibidos na view. O código a seguir apresenta a versão final da classe FilmeController.*java* com o método consultarFilme responsável por pesquisar os filmes cadastrados.

# Audiodescrição: A imagem mostra uma captura de tela do Eclipse Workspace em que a aba “FilmeController.java” está selecionada. Logo abaixo, temos o seguinte código:

```
import javax.persistence.EntityManager;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.servlet.ModelAndView;

import br.com.lead.modelo.Autor;

import br.com.lead.modelo.Filme;

import br.com.lead.util.JPAUtil;

@Controller

public class FilmeController {

	@RequestMapping(value = "/persistir-filme", method = RequestMethod.GET)

	public String persistirFilme(@RequestParam String nome, @RequestParam String genero, @RequestParam String ano, @RequestParam String nomeAutor, @RequestParam String dataNascimentoAutor) {

		Autor autor = new Autor();

		autor.setDataNascimento(LocalDate.parse(dataNascimentoAutor));

		autor.setNome(nomeAutor);

		Filme filme = new Filme(nome, genero, Integer.valueOf(ano));

		filme.setAutor(autor);
```

```
            EntityManager entityManager = JPAUtil.getEntityManager();
            entityManager.getTransaction().begin();
            entityManager.persist(autor);
            entityManager.persist(filme);
            entityManager.getTransaction().commit();
            entityManager.close();

            return "persistir-filme-view";
        }
@RequestMapping(value = "/consultar-filme", method = RequestMethod.GET)
        @ResponseBody
        public ModelAndView consultarFilme(@RequestParam Integer id) {
                EntityManager entityManager = JPAUtil.getEntityManager();
            Filme filme = entityManager.find(Filme.class, id);
                ModelAndView modelAndView = new ModelAndView();
                modelAndView.setViewName("consultar-filme-view");
                modelAndView.addObject("nome", filme.getNome());
                modelAndView.addObject("genero", filme.getGenero());
                modelAndView.addObject("ano", filme.getAno());
            return modelAndView;
        }
}
```

No código anterior você deve ter observado o uso do ModelAndView para direcionar ao JSP consultar-filme-view (consultar traço filme traço view), os dados *nome, gênero e ano* do filme. Veja no trecho a seguir:

#Audiodescrição: A imagem mostra o seguinte trecho de código:

ModelandView modelandView = new ModelandView();

ModelandView.setViewName = (“consultar-filme-view”);

ModelandView.add0object = (“nome”, filme.getNome());

ModelandView.add0object = (“genero”, filme.getGenero());

ModelandView.add0object = (“ano”, filme.getAno());

Return modelAndView

Afinal, o que é ModelAndView? Qual a sua importância no Spring MVC? A ModelAndView é uma classe do Spring MVC que possibilita o repasse dos dados processados no controller para uma view (ou camada de apresentação), na maioria das vezes para um JSP. Na prática a classe ModelAndView atua como mediador, informando para quem você encaminhará os dados e quais deles serão enviados.

Por último, você deve criar o JSP consultar-filme-view.jsp no diretório WebContent/WEB-INF/views (webcontent barra web inf barra views). Nele estarão contidos os códigos em HTML e as diretivas ${nome} (dólar abre chave nome fecha chave), ${genero} (dólar abre chave genero fecha chave) e ${ano} (dólar abre chave ano fecha chave), usadas para relacionar os dados que serão enviados a partir do ModelAndView do recurso (ou funcionalidade) consultarFilme. O JSP consultar-filme-view.jsp (consultar traço filme traço view) vai conter em sua estrutura um título chamado consulta de filmes e os textos filme, gênero e ano. Além disso as diretivas ${nome} (dólar abre chave nome fecha chave), ${genero} (dólar abre chave genero fecha chave) e ${ano} (dólar abre chave ano fecha chave), que serão usadas para exibir os valores vindos do método consultarFilme do controller FilmeController. O código, a seguir, apresenta o conteúdo do JSP consultar-filme-view.jsp (consultar traço filme traço view) usado para exibir os dados dos filmes consultados:

#Audiodescrição: A imagem mostra uma captura de tela do Eclipse Workspace em que a aba “consultar-filme-view.jsp” está selecionada. Logo abaixo, temos o seguinte código:
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "https://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <meta charset="UTF-8">
        <title>Consulta de filmes</title>
</head>
<body>
        <h1> Filme: ${nome} </h1>
    <label> Gênero: ${genero} </label>
    <br/>
        <label> Ano: ${ano} </label>
</body>
</html>
```

Feito isso, você concluiu a migração da funcionalidade de cadastro de filmes do servlet PersisteFilmeServlet para o controller FilmeController. Além disso, incluiu um novo recurso que é encarregado de consultar um filme a partir do seu identificador. Pois bem, até o momento, você aprendeu como criar um controller, a criar recursos a partir da diretiva @RequestMapping (arroba request mapping) e como capturar dados vindos da URL com @RequestParam (arroba request param). Na próxima videoaula você irá conhecer o Spring Boot um framework voltado para a criação de microservices e APIs REST.

Até a próxima aula e bons estudos!

**Referências**


BRITO, Michelli. Spring MVC e o Dispatcher Servlet. Disponível em: <https://medium.com/@michellibrito/spring-mvc-e-o-dispatcher-servlet-15bd5575e2d8>. Acesso em: 8 jun 2020.

Spring 4 MVC HelloWorld Tutorial – Annotation/JavaConfig Example. Disponível:<http://websystique.com/springmvc/spring-4-mvc-helloworld-tutorial-annotation-javaconfig-full-example/>. Acesso em: 8 jun 2020.

Spring ModelAndView tutorial. Disponível em: <http://zetcode.com/spring/modelandview/>. Acesso em: 8 jun 2020.

Spring MVC Example. Disponível em: <https://www.journaldev.com/14476/spring-mvc-example>. Acesso em: 8 jun 2020.
Spring MVC Interview. Disponível: <https://www.baeldung.com/spring-mvc-interview-questions>. Acesso em: 8 jun 2020.
Spring MVC. Disponível em: <https://www.caelum.com.br/apostila-java-web/spring-mvc/>. Acesso em: 8 jun 2020.
Spring RequestMapping. Disponível em: <https://www.baeldung.com/spring-requestmapping>. Acesso em: 8 jun 2020.
Spring's RequestBody and ResponseBody Annotations. Disponível em: <https://www.baeldung.com/spring-request-response-body>. Acesso em: 8 jun 2020.
Web on Servlet Stack. Disponível em: <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>. Acesso em: 8 jun 2020.
What is Dispatcher Servlet in Spring?. Disponível: <https://stackoverflow.com/questions/2769467/what-is-dispatcher-servlet-in-spring>. Acesso em: 8 jun 2020.