VÍDEO 4.2: Integrando uma classe Servlet com a JSP

Nesta videoaula, você dará continuidade ao estudo da JSP. Aqui, você irá ajustar o seu projeto para que a classe FilmeServlet possa trabalhar possuindo apenas a lógica de montar uma lista de filmes e realizar um filtro para exibir apenas alguns filmes, que foi criada em Java, além de deixar que a apresentação fique a cargo de uma página JSP.

Em primeiro lugar, você irá remover tudo que foi criado em formato HTML dentro da classe. Você deve deixar apenas o filtro dos filmes. Contudo, irá alterá-lo para que gere uma nova lista, contendo apenas a nova relação de filmes filtrados. Para isso, você utilizará o Collect do java 8. Dessa maneira, a nova instrução ficará assim: ArrayList<Filme>listaFiltrada = filmes.stream().filter(filme ->filme.getGenero().toUpperCase().equals(genero.toUpperCase())) .collect(Collectors.toCollection(ArrayList::new));

```java
@WebServlet("/filme")
public class FilmeServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        Filme coringa = new Filme("Coringa", "Drama", 2019);
        Filme matrix = new Filme("Matrix", "Ação", 1999);
        Filme forrestGump = new Filme("Forrest Gump", "Drama", 1994);

        ArrayList<Filme> filmes = new ArrayList<Filme>();

        filmes.add(coringa);
        filmes.add(matrix);
        filmes.add(forrestGump);

        String genero = req.getParameter("genero");

        ArrayList<Filme> listaFiltrada = filmes.stream().filter(filme -> filme.getGenero().toUpperCase().equals(genero.toUpperCase()))
                                        .collect(Collectors.toCollection(ArrayList::new));
    }
}
```

#Audiodescricao: A imagem mostra o código:

“@WebServlet("/filme")
public class FilmeServlet extends HttpServlet {
@Override protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
Filme coringa = new Filme("Coringa", "Drama", 2019);
Filme matrix = new Filme("Matrix", "Ação", 1999);
Filme forrestGump = new Filme("Forrest Gump", "Drama", 1994);

```
ArrayList<Filme> filmes = new ArrayList<Filme>();

filmes.add(coringa);
filmes.add(matrix);
\filmes.add(forrestGump);

String genero = req.getParameter("genero");

ArrayList<Filme> listaFiltrada = filmes.stream().filter(filme ->
filme.getGenero().toUpperCase().equals(genero.toUpperCase()))
.collect(Collectors.toCollection(ArrayList::new));
}
}”
```

Com o objeto lista filtrada, você irá precisar, agora, passar essa lista para uma página JSP que irá se encarregar de exibir esses dados. Para isso, você deve adicionar esse objeto na requisição, pois, dessa maneira, a página JSP irá conseguir ter acesso a esse objeto. Para realizar a requisição você deve utilizar o objeto req do comando HttpServletRequest, que o método service recebeu por parâmetro. É através do req que você conseguirá manipular dados da requisição.

Tudo bem até aqui? Calma! No próximo passo, você compreenderá como isso funciona. Você deve adicionar o objeto de lista que precisa ser exibido na requisição. Utilize o método setAttribute do objeto req, passando por parâmetro uma String, que servirá de identificador para o seu objeto. A instrução deve ficar da seguinte maneira:

```
req.setAttribute("listaFiltrada", listaFiltrada);
```

Pronto, agora você já tem o objeto filtrado, já o adicionou à requisição, porém precisa chamar a sua página JSP. Para isso, utilize o RequestDispatcher. Esse elemento irá possibilitar despachar a requisição, como uma espécie de redirecionamento. Você se utilizará dessa característica para redirecionar a requisição que chegou até a classe Servlet para a página JSP.

Isso tudo funcionará da seguinte maneira: você deve declarar um objeto do tipo RequestDispatcher e atribuir, a esse objeto, o conteúdo da chamada ao método req.getRequestDispatcher("/lista-filmes.jsp"), passando como parâmetro o nome do arquivo JSP. Em seguida, realize a chamada ao método forward do objeto criado, passando por parâmetro os mesmos objetos que o método service recebeu por parâmetro.

dispatcher.forward(req, resp);

```java
@WebServlet("/filme")
public class FilmeServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        Filme coringa = new Filme("Coringa", "Drama", 2019);
        Filme matrix = new Filme("Matrix", "Ação", 1999);
        Filme forrestGump = new Filme("Forrest Gump", "Drama", 1994);

        ArrayList<Filme> filmes = new ArrayList<Filme>();

        filmes.add(coringa);
        filmes.add(matrix);
        filmes.add(forrestGump);

        String genero = req.getParameter("genero");

        ArrayList<Filme> listaFiltrada = filmes.stream().filter(filme -> filme.getGenero().toUpperCase().equals(genero.toUpperCase()))
                                        .collect(Collectors.toCollection(ArrayList::new));

        req.setAttribute("listaFiltrada", listaFiltrada);

        RequestDispatcher dispatcher = req.getRequestDispatcher("lista-filmes.jsp");
        dispatcher.forward(req, resp);
    }
}
```

#Audiodescricao: A imagem mostra o código:

“@WebServlet("/filme")
public class FilmeServlet extends HttpServlet {

@Override
protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
Filme coringa = new Filme("Coringa", "Drama", 2019);
Filme matrix = new Filme("Matrix", "Ação", 1999);
Filme forrestGump = new Filme("Forrest Gump", "Drama", 1994);

ArrayList<Filme> filmes = new ArrayList<Filme>();

filmes.add(coringa);
filmes.add(matrix);
filmes.add(forrestGump);

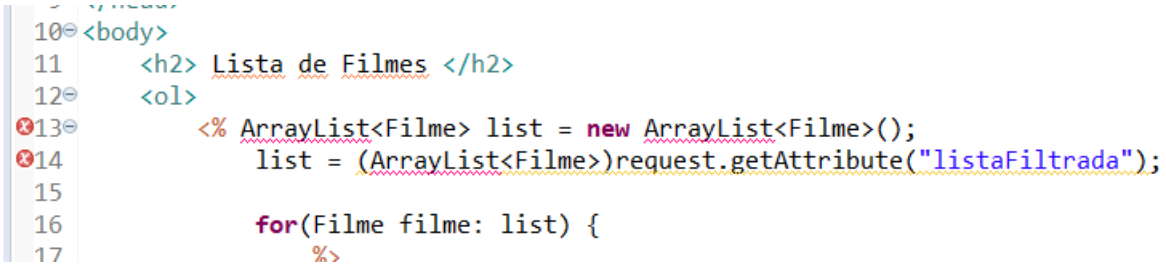
```
String genero = req.getParameter("genero");

ArrayList<Filme> listaFiltrada = filmes.stream().filter(filme ->
filme.getGenero().toUpperCase().equals(genero.toUpperCase()))
.collect(Collectors.toCollection(ArrayList::new));

req.setAttribute("listaFiltrada", listaFiltrada);

RequestDispatcher dispatcher = req.getRequestDispatcher("lista-filmes.jsp")
dispatcher.forward(req, resp);
}
}".
```

Pronto, a sua Servlet está bem completa, recebendo uma requisição, aplicando uma lógica Java e redirecionando a requisição, para uma página JSP, para que sejam exibidos os dados necessários. Agora, você precisa criar a sua página JSP que será a responsável por exibir essa lista. Lembre-se de que, na Servlet, a requisição deverá ser redirecionada para o arquivo “lista-filmes.jsp”. Logo, esse será o nome do seu arquivo JSP a ser criado. Então, com a ajuda do Eclipse novamente, você irá criar o arquivo lista-filmes.jsp dentro da pasta WebContent, como vimos anteriormente.

Com o arquivo lista-filmes.jsp criado, você irá editá-lo. Localizar a tag title, que, como em um arquivo HTML, geralmente fica na parte superior do arquivo. Portanto, delete o seu conteúdo através do teclado e digite a informação Lista de FIlmesNatagbody, você irá imprimir os dados da sua lista. Antes disso, você deve inserir a informação Lista de Filmes, entre as tags H2 e /H2, que são uma das tags do HTML, pois existem outras, para definir o título de uma informação. Feito isso, você irá abrir a tagol, a qual indica que você está criando uma lista.

Agora, será necessário capturar a informação da sua lista, aquela que foi adicionada à requisição, ou seja, deve-se abrir uma Scriptlet, utilizando <% e %>, e, em seguida, é preciso escrever um código Java para acessar essa informação da sua lista. Dentro do bloco, entre as marcações do Scriptlet, você irá declarar um objeto

do tipo ArrayList de Filme e irá chamá-lo de list. Com isso, você perceberá que o Eclipse irá exibir uma advertência em vermelho ao lado esquerdo do arquivo, informando que é necessário realizar o import da classe ArrayList e da classe Filme.



#Audiodescricao: A imagem mostra o trecho de código:

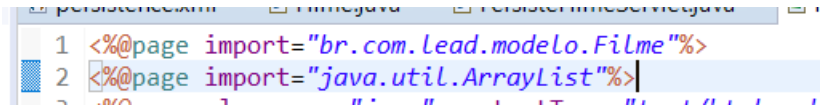
“<body>
<h2> Lista de Filmes </h2>

<% ArrayList<Filme> list = new ArrayList<Filme>();
list = (ArrayList<Filme>)request.getAttribute("listaFiltrada");
for(Filme filme: list) {
%>” .

Para que você possa manipular algumas classes Java, em uma página JSP, será necessário realizar o import delas. Como você geralmente faz em um código Java. Em JSP para declarar uma determinada classe, você precisa utilizar o seguinte modelo, <%@pageimport=”CaminhoDaClasse”%>. No exemplo utilizado aqui, você precisa realizar o import de ArrayList e de Filme. Logo, a declaração das classes ficou dessa forma:

<%@pageimport="br.com.lead.modelo.Filme"%>

<%@page import="java.util.ArrayList"%>



#Audiodescricao: A imagem mostra o trecho de código:

“<%@page import=”br.com.lead.modelo.Filme”%>
<%@page import=”java.util.ArrayList”%>".

Pronto! Com os imports realizados, o Eclipse não deve mais exibir as advertências em vermelho. Agora, você deve capturar o seu objeto listaFiltrada que foi adicionado à requisição. Para que você acesse os dados da requisição, a JSP fornece o objeto request do tipo HttpServletRequest. Com ele, você pode executar a instrução request.getAttribute, passando por parâmetro o nome do atributo que deseja, no caso é listaFiltrada. O retorno da chamada a esse método deve ser atribuído ao objeto list que foi criado anteriormente. Depois, será necessário realizar o cast para Arraylist de Filme.

```
<ol>
    <% ArrayList<Filme> list = new ArrayList<Filme>();
        list = (ArrayList<Filme>)request.getAttribute("listaFiltrada");
```

#Audiodescricao: A imagem mostra o trecho de código:

“
<% ArrayList<Filme> list = new ArrayList<Filme>() ;
list = (ArrayList<Filme>)request. getAttribute(“listaFiltrada”) ;".

Com o seu objeto list criado e populado, você deve, agora, percorrer essa lista e imprimir seus dados. Dentro do mesmo bloco de Scriptlet, você irá inserir apenas parte da instrução for, que corresponde da sua declaração até a abertura da chave. Após essa instrução, você deve finalizar o bloco de Scriptlet. Para isso, utilize a marcação %>. Concluído o fechamento do bloco de Scriptlet, você irá perceber que faltou fechar o bloco for com a outra chave, mas não iremos fazer isso nesse mesmo Scriptlet.

```
<% ArrayList<Filme> list = new ArrayList<Filme>();
    list = (ArrayList<Filme>)request.getAttribute("listaFiltrada");

    for(Filme filme: list) {
        %>
```

#Audiodescricao: A imagem mostra o trecho de código:

“<% ArrayList<Filme> list = new ArrayList<Filme>() ;
list = (ArrayList<Filme>)request. getAttribute(“listaFiltrada”) ;

for(Filme filme : list) {
%>".

Avançando mais um pouco, você, agora, irá criar um outro Scriptlet localizado mais abaixo do Scriptlet criado anteriormente. Para isso, abra, novamente, o Scriptlet e insira apenas a chave de fechamento do for. Então, o for foi montado, mas foi dividido em dois trechos de Scriptlets.

```
<% ArrayList<Filme> list = new ArrayList<Filme>();
    list = (ArrayList<Filme>)request.getAttribute("listaFiltrada");

    for(Filme filme: list) {
%>

<%
    }
%>
```

#Audiodescricao: A imagem mostra o trecho de código:

“<% ArrayList<Filme> list = new ArrayList<Filme>() ;
list = (ArrayList<Filme>)request. getAttribute(“listaFiltrada”) ;

for(Filme filme : list) {
%>
<%
}
%>”.

A instrução Java não precisa ficar toda em um único trecho de Scriptlet, você pode dividi-la. Com isso, o bloco for foi montado. Contudo, você irá precisar ainda imprimir os dados dos seus objetos de filme. Entre esses dois blocos que você acabou de criar, deve montar os itens da lista em HTML. Para isso, você deve começar inserindo as tags li e /li, essas são tags do HTMLque indicará à página que se trata de um item da lista, e, dentro delas, você criará um outro Scriptlet, só que agora contendo o objeto out e o método println. Iremos utilizar o método println para imprimir os dados dos filmes. Pronto, você finalizou a sua página JSP.

```
<%@page import="br.com.lead.modelo.Filme"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Lista de Filmes</title>
</head>
<body>
    <h2> Lista de Filmes </h2>
    <ol>
        <% ArrayList<Filme> list = new ArrayList<Filme>();
            list = (ArrayList<Filme>)request.getAttribute("listaFiltrada");

            for(Filme filme: list) {
                %>
            <li>
                <%
                out.println(filme.getNome());
                out.println(filme.getNome());
                out.println(filme.getNome());
                %>
            </li>
        <%
                }
        %>
    </ol>

</body>
</html>
```

Audiodescricao: A imagem mostra o código:

“<%@page import="br.com.lead.modelo.Filme"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Lista de Filmes</title>
</head>
<body>
<h2> Lista de Filmes </h2>

<% ArrayList<Filme> list = new ArrayList<Filme>();
list = (ArrayList<Filme>)request.getAttribute("listaFiltrada");

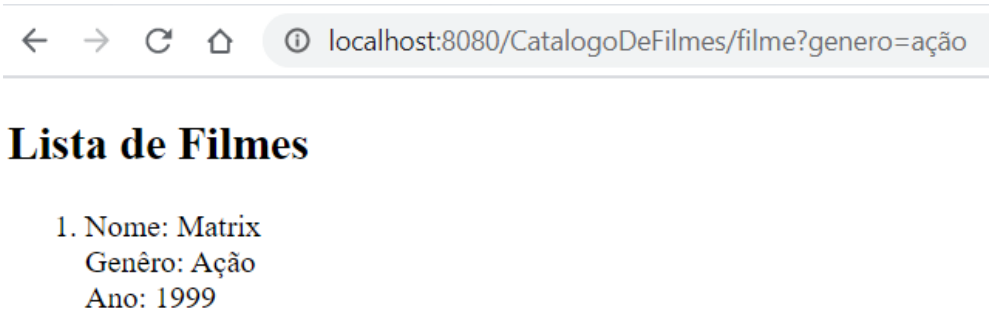
for(Filme filme: list) {
%>

<%
out.println(filme.getNome());
out.println (filme.getNome());
out.println (filme.getNome());
%>

<%
}
%>

</body>
</html>”

Para conferir a página que acabou de criar, você deve acessar, novamente, a URL que faz a requisição de filmes, filtrando pelo gênero ação. A URL informada no navegador deve ser http://localhost:8080/CatalogoDeFilmes/filme?genero=ação. Confira o resultado: a página deverá exibir apenas o filme Matrix.



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/lfilme?gênero=ação”. Logo abaixo, temos o texto “Lista de Filmes:
Nome: Matrix
Gênero: Ação
Ano: 1999.

Durante a videoaula, você estudou como a tecnologia JSP irá lhe ajudar no desenvolvimento das suas páginas web, servindo até de base de conhecimento para compreender melhor outros frameworks voltados para o desenvolvimento web. Além disso, você aprendeu como criar páginas utilizando a JSP, como passar atributos e como capturá-los. Isso lhe dará base para evoluir com sua aplicação e consequentemente se aprimorar no uso dessa ferramenta. Também conheceu acerca da existência do padrão de projeto MVC, em que seu projeto é divido em elementos que são responsáveis pela exibição da informação, no caso a JSP, em elementos responsáveis pelo controle de regras e fluxos da aplicação e elementos responsáveis pela persistência e comunicação com a base de dados.

Você viu como deixar cada componente com sua responsabilidade específica: a classe Servlet com suas regras e validações, como a filtragem dos filmes que devem ser exibidos, fazendo-se uso de parâmetros passados na URL, realizando também redirecionamentos da requisição; e a página JSP com a responsabilidade de realizar a apresentação dos dados, capturando a informação a ser exibida e fazendo-se uso de algumas tags para apresentar os dados para o usuário.

No próximo vídeo, você conhecerá sobre a persistência dos dados e como irá fazer para armazenar seus filmes em um banco de dados.

Até mais!

Referências

Introdução ao Java Server Pages – JSP. Disponível em: <https://www.devmedia.com.br/introducao-ao-java-server-pages-jsp/25602>. Acesso em: 2 maio 2020.

Java para desenvolvimento web. Disponível em: <https://www.caelum.com.br/apostila-java-web/>. Acesso em: 2 maio 2020.