

Universidade Presbiteriana Mackenzie Faculdade de Computação e Informática

## Abordagem MVC utilizando servlets e JSP

NOTAS DE AULA - Teoria 18

Linguagem de Programação II 2º semestre de 2015 Versão 1.0

### Referências

As referências para esta aula é o capítulo 15 de:

HALL, M.; et al. Core Servlets and Javaserver Pages: vol. 1: Core Technologies. New jersey: Prentice-Hall, 2003.

A versão on-line deste livro está disponível em <a href="http://pdf.coreservlets.com">http://pdf.coreservlets.com</a>

#### Observações:

- As notas de aula são material de apoio para a aula e não têm o objetivo de apresentar o assunto de maneira exaustiva. Não deixe de ler o material de referência da disciplina.
- Os trechos de código fornecidos são simplistas e curtos para podermos focar no assunto que está sendo estudado, por isso não devem ser utilizados diretamente em produção.

# Introdução

• Nesta aula estudaremos a aplicação da abordagem MVC (*Model-View-Controller*) em aplicações web implementadas com servlets e JSP.

### Servlet

- Implementado totalmente em código Java.
- Eventualmente, pode gerar conteúdo HTML.

# Servlet (cont.)

- Adequado quando bastante programação é necessária para a implementação de uma funcionalidade.
- Não é adequado para gerar conteúdo HTML, pois é trabalhoso e leva a resultados que são difíceis de modificar.

### **JSP**

• Código HTML que pode conter trechos de código Java.

```
<
```

# JSP (cont.)

- Adequado para definir o conteúdo HTML e gerar dinamicamente apenas alguns itens da página.
- Não é adequado inserir longos trechos de código Java em uma página JSP.

### Model View Controller em aplicações web

- A combinação adequada de servlets e páginas JSP permite aplicar a abordagem MVC ao desenvolver aplicações web.
- A página JSP é utilizada para a apresentação (view).
- Os objetos que encapsulam os dados são o modelo (*model*).
- O servlet é o *controller*, pois decide qual lógica de negócios aplicar e qual página deve apresentar os resultados.

# RequestDispatcher

• Uma instância de **RequestDispatcher** é utilizada para encaminhar os resultados do processamento feito pelo servlet para serem apresentados em uma página JSP.

# Uso de RequestDispatcher

Passos usuais para o encaminhamento da apresentação da página utilizando **RequestDispatcher**:

- Definição de beans para representar os dados.
- Uso de servlets para tratar requisições.
- Instanciação de beans com os resultados.
- Armazenamento dos beans na requisição (*request*), sessão (*session*) ou no contexto do servlet (*servlet context*).
- Encaminhamento da requisição para uma página JSP.
- Extração dos dados armazenados nos beans para apresentação na página JSP.

# Uso de RequestDispatcher (cont.)

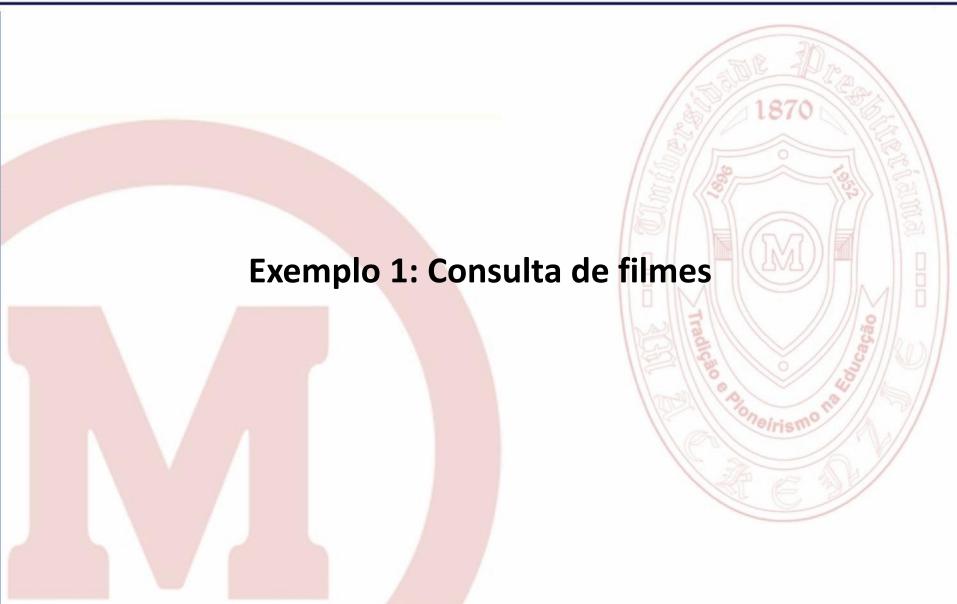
A abordagem padrão é utilizar o método **forward** de **RequestDispatcher**:

- O controle é transferido no web server, sem transações adicionais de rede.
- O usuário não vê o endereço da página JSP de destino em seu browser.

### Exemplo:

```
RequestDispatcher rd;
rd = request.getRequestDispatcher("/WEB-INF/pagina.jsp");
rd.forward(request, response);
```





# **Exemplo 1: Consulta de filmes**

```
package cinema.dominio;
public class Filme {
    private String titulo;
    private int ano;
    private String diretor;
    public Filme() {}
    public Filme(String titulo, int ano, String diretor) {
        this.titulo = titulo;
        this.ano = ano;
        this.diretor = diretor;
    // getters e setters
```

```
package cinema.servico.api;
import cinema.dominio.Filme;
import java.util.List;

public interface ConsultarFilmesServicoInterface {
   List<Filme> listarTodos();
   void adicionar(Filme f);
}
```

```
public class ConsultarFilmesServicoExemplo implements ConsultarFilmesServicoInterface {
    private List<Filme> filmes;
    public ConsultarFilmesServicoExemplo() {
        filmes = new ArrayList<Filme>();
        filmes.add(new Filme("Senhor dos Anéis", 2001, "Peter Jackson"));
        filmes.add(new Filme("Vingadores 2", 2015, "Joss Whedon"));
        filmes.add(new Filme("Batman vs Superman", 2016, "Zack Snyder"));
        filmes.add(new Filme("Forrest Gump", 1994, "Robert Zemeckis"));
        filmes.add(new Filme("Star Wars", 1977, "George Lucas"));
    }
    @Override
    public List<Filme> listarTodos() {
        return filmes;
    @Override
    public void adicionar(Filme f) {
        filmes.add(f);
```

#### Arquivo index.html

```
<!DOCTYPE html>
<html>
   <head>
       <title>Gerenciamento de filmes</title>
       <meta charset="UTF-8">
       <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
       <h1>Informações de filmes</h1>
       <a href="ConsultarFilmesServlet">Listagem de todos os filmes</a>
       Cadastro de um novo filme
   </body>
</html>
```

```
public class ConsultarFilmesServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
        ConsultarFilmesServicoInterface servico;
        servico = new ConsultarFilmesServicoExemplo();
        List<Filme> filmes;
        filmes = servico.listarTodos();
        request.setAttribute("lista_filmes", filmes);
        RequestDispatcher rd;
        rd = request.getRequestDispatcher("listar filmes.jsp");
        rd.forward(request, response);
```

Obs.: Note que a lista de filmes é salva como atributo da requisição.

#### Arquivo listar\_filmes.jsp

```
<%@page import="cinema.dominio.Filme,java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Listagem dos filmes</title>
   </head>
   <%
    List < Filme > filmes:
   filmes = (List<Filme>) request.getAttribute("lista_filmes");
   %>
   <body>
      <h1>Lista de filmes:</h1>
      NomeAnoDiretor
        for (Filme f: filmes) {
         </body>
</html>
```

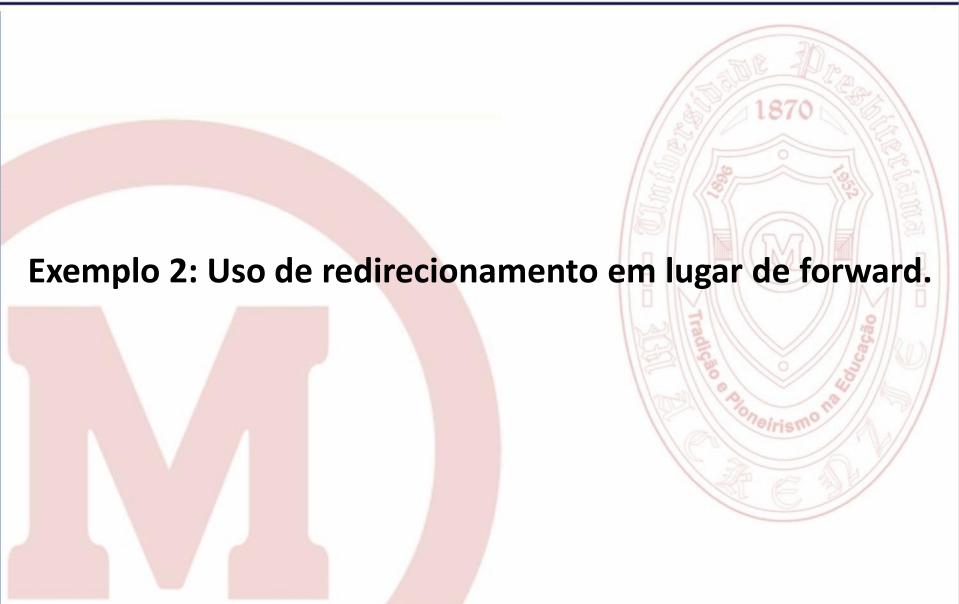
Obs.: Note que a lista de filmes é recuperada como atributo da requisição.

### Alternativa ao uso de RequestDispatcher

Uma alternativa a esta abordagem, quando os dados são armazenados na sessão, é utilizar o método **sendRedirect** de **HttpServletResponse**:

- O controle é transferido enviando para o browser uma resposta com o código de status 302 e o local de redirecionamento.
- O usuário vê o endereço da página de destino.





#### **Exemplo 2: Uso de redirecionamento**

```
public class ConsultarFilmesServlet2 extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException, IOException {
        ConsultarFilmesServicoInterface servico;
        servico = new ConsultarFilmesServicoExemplo();
        List<Filme> filmes;
        filmes = servico.listarTodos();
        HttpSession session;
        session = request.getSession();
        session.setAttribute("lista filmes", filmes);
        response.sendRedirect("listar filmes2.jsp");
```

Obs.: Note que a lista de filmes é salva como **atributo da sessão**.

### **Exemplo 2: Uso de redirecionamento (cont.)**

#### Arquivo listar\_filmes2.jsp

```
<%@page import="cinema.dominio.Filme,java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Listagem dos filmes</title>
   </head>
   <%
    List < Filme > filmes:
    filmes = (List<Filme>) session.getAttribute("lista_filmes");
   %>
   <body>
      <h1>Lista de filmes:</h1>
      NomeAnoDiretor
        for (Filme f: filmes) {
         </body>
</html>
```

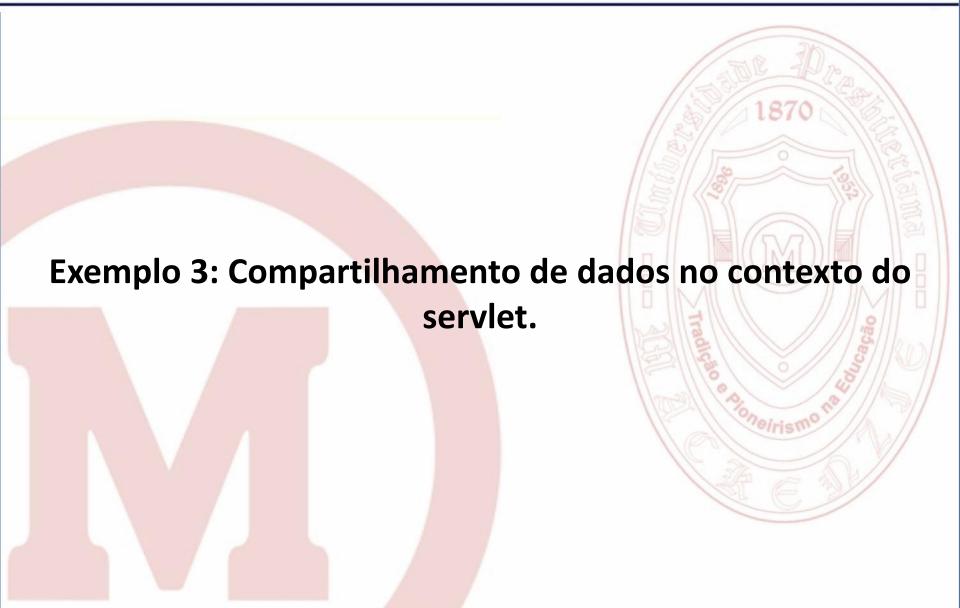
Obs.: Note que a lista de filmes é recuperada como atributo da sessão.

## Estratégias de compartilhamento de dados

Os servlets e páginas JSP de uma aplicação podem compartilhar dados:

- na requisição;
- na sessão;
- na aplicação (ou servlet context).





#### **Exemplo 3: Compartilhamento de dados no contexto do servlet**

### Arquivo definir\_cor\_fundo\_global.html

```
<html>
<head>
 <title>Cor de fundo</title>
 <meta charset="UTF-8">
</head>
<body>
 <h1>Escolha a cor de fundo global:</h1>
<form method="post" action="DefinirCorFundoGlobalServlet">
   Cor:
   <select name="cor fundo param">
     <option value="RED">vermelho</option>
     <option value="BLUE">azul</option>
     <option value="YELLOW">amarelo</option>
    </select>
   <input type="submit">
 </form>
</body>
</html>
```

#### **Exemplo 3: Compartilhamento de dados no contexto do servlet (cont.)**

```
public class DefinirCorFundoGlobalServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
             throws ServletException, IOException {
        String corFundo = request.getParameter("cor fundo param");
        if (corFundo == null) { corFundo = "WHITE"; }
        ServletContext aplicacao = getServletContext();
        aplicacao.setAttribute("cor fundo", corFundo);
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
             out.println("<!DOCTYPE html>");
             out.println("<html>");
             out.println("<head>");
             out.println("<title>Servlet DefinirCorFundoServlet</title>");
             out.println("</head>");
             out.println("<body>");
             out.println("<h1>Cor de fundo global definida com sucesso</h1>");
             out.println("</body>");
            out.println("</html>");
```

#### **Exemplo 3: Compartilhamento de dados no contexto do servlet (cont.)**

```
<%@page import="cinema.dominio.Filme,java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Listagem dos filmes</title>
   </head>
    List < Filme > filmes;
    filmes = (List<Filme>) session.getAttribute("lista_filmes");
  %>
   <body bgcolor="<%= application.getAttribute("cor_fundo")%>" >
      <h1>Lista de filmes (versão com redirecionamento):</h1>
      NomeAnoDiretor
          <%
         for (Filme f: filmes) {
      %>
          <%= f.getTitulo()%>
      </body>
</html>
```



