

FIAP GRADUAÇÃO

DESENVOLVIMENTO DE SISTEMAS WEB

PROF. THIAGO T. I. YAMAMOTO
thiagoyama@gmail.com

SERVLETS

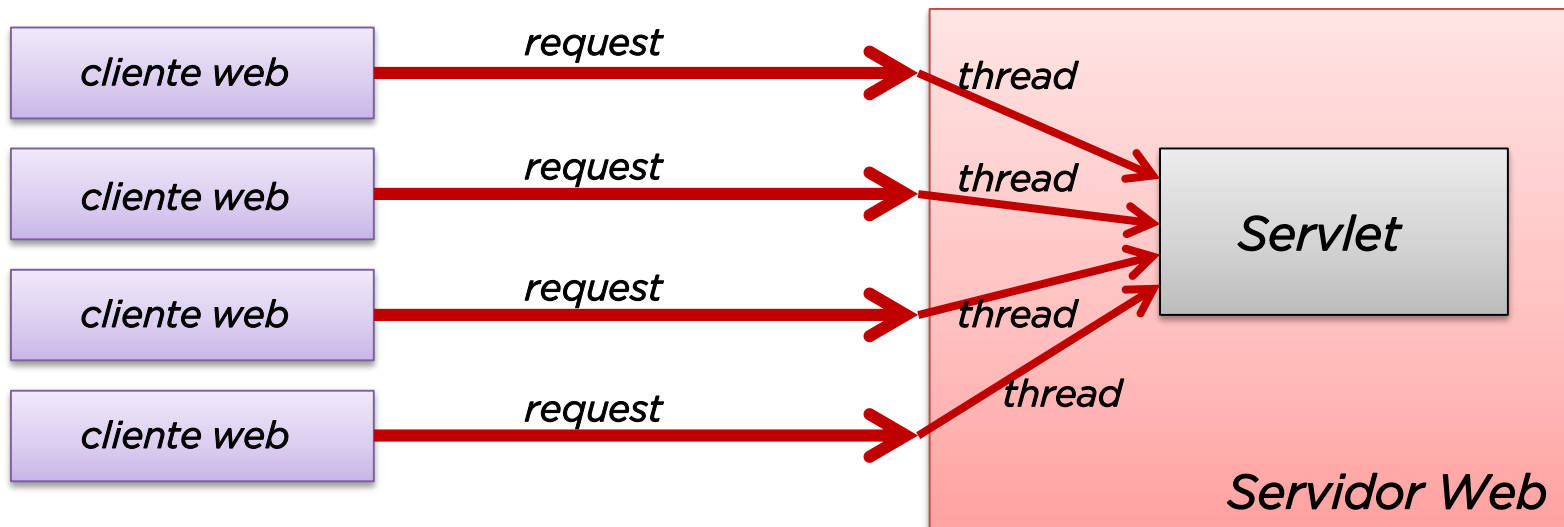
1. Introdução ao Servlets
2. HttpServletRequest e HttpServletResponse
3. Mapeando uma servlet: web.xml e Annotations
4. Exercício
5. Web.xml

INTRODUÇÃO AO SERVLETS

- É uma classe Java;
- Componente Java EE;
- Baseado no modelo Request/Response;
- Deve ser executado dentro de um web container;
- É controlado pelo servidor;

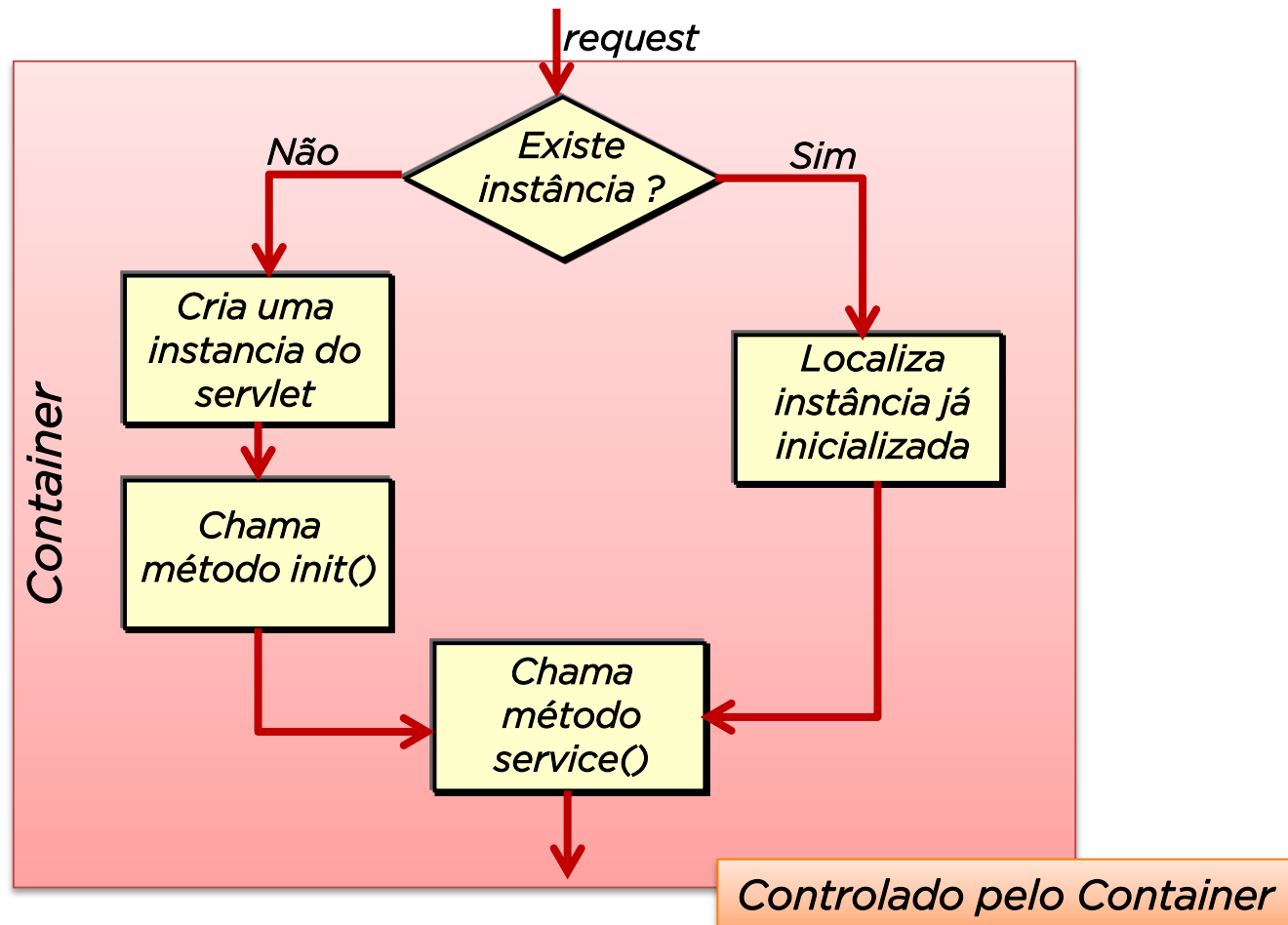


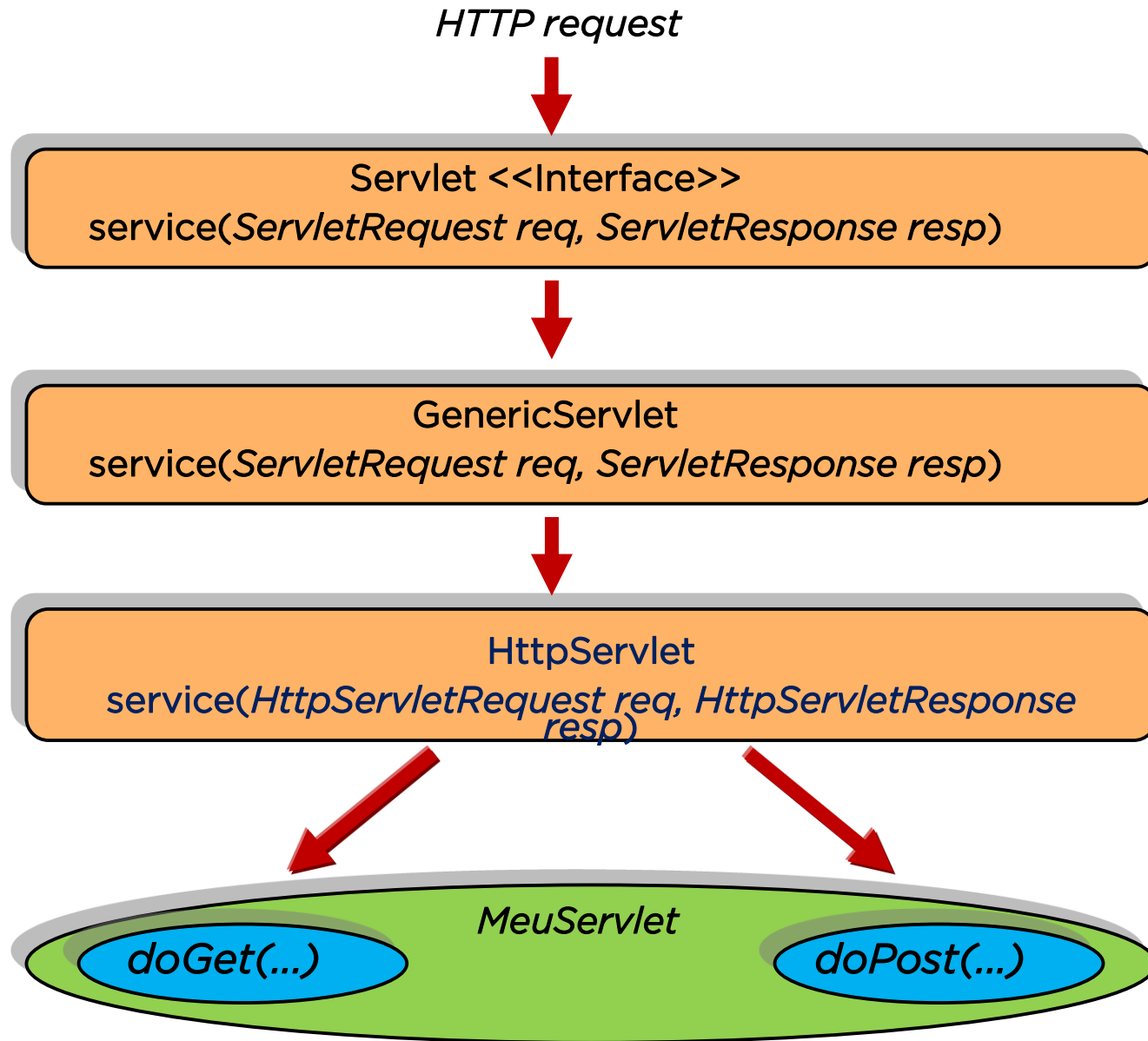
Objetivo é receber requisições HTTP, processá-las e devolver uma resposta ao cliente.



- Cada requisição a uma servlet é executada em uma thread;
- O objeto servlet é único na aplicação;

- Os servlets são instanciados pelo container web;
- Após iniciados, os servlets podem atender as requisições;
- O container decide a hora de destruir os servlets;





- Classe Java que estende de **HTTPServlet**;
- Responsável por atender requisições HTTP;
- Pode sobrescrever os métodos **doGet()** e **doPost()**;

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                  throws ServletException, IOException{

}

public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
                   throws ServletException, IOException {

}
```

```
public class MeuServlet extends HttpServlet {  
  
    public void init(ServletConfig config) throws  
        ServletException {  
        Inicialização do Servlet  
    }  
  
    public void destroy() {  
        Destruição do Servlet  
    }  
  
    protected void doGet(HttpServletRequest request,  
        HttpServletResponse response) throws ServletException,  
        IOException {  
        Requisição tipo GET  
    }  
  
    public void doPost(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        Requisição tipo POST  
    }  
}
```

- Representa a requisição feita pelo usuário;
- É possível obter dados enviados pelo browser, atributos, informações de endereço de IP, protocolo, etc..
- Alguns métodos importantes:
 - **getAttribute(String) / setAttribute(String, Object)** - Permite obter e armazenar objetos java temporariamente no request;
 - **HttpSession getSession()** - Obtém a sessão do usuário;
 - **String getParameter(String)** - Obtém informações submetidas para o servidor (GET ou POST);

Passagem de parâmetros:

<http://localhost:8080/JavaWeb/loginServlet?nome=Thiago&senha=123>

Nome
Thiago

Senha
...

Enviar

`<form action="loginServlet">`

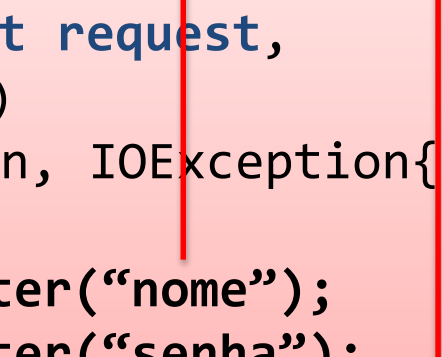
Login

Senha

`</form>`

Passagem de parâmetros:

http://localhost:8080/JavaWeb/loginServlet?nome=Thiago&senha=123



The diagram illustrates the mapping between a URL and a servlet method. Two red arrows originate from the query string of the URL 'http://localhost:8080/JavaWeb/loginServlet?nome=Thiago&senha=123'. One arrow points from the 'nome=Thiago' part to the 'request.getParameter("nome")' line in the code. The other arrow points from the 'senha=123' part to the 'request.getParameter("senha")' line. The code is enclosed in a light red box.

```
public void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException{

    String n = request.getParameter("nome");
    String s = request.getParameter("senha");

}
```

- Representa a resposta que será enviada ao cliente
- Alguns métodos:
 - **addCookie(Cookie)** – Adiciona cookie no response que será retornado ao browser
 - **getOutputStream()** – Obtém um stream de saída para envio de conteúdo binário (imagens, pdf, etc)
 - **getWriter()** – Obtém objeto PrintWriter para envio de texto/html para o browser
 - **setContentType(String)** – Informa o tipo de retorno (html / arquivo) a ser enviado para o browser

Como enviar um conteúdo HTML para o browser da servlet através do **HTTPServletResponse**:

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {

    PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    out.print("<html><body>+\"Ola!\"+</body></html>");

}
```

Como vamos acessar uma servlet pelo browser?

Precisamos mapear uma url para a servlet!

É possível realizar a configuração de duas formas:

- web.xml;
- Annotations;

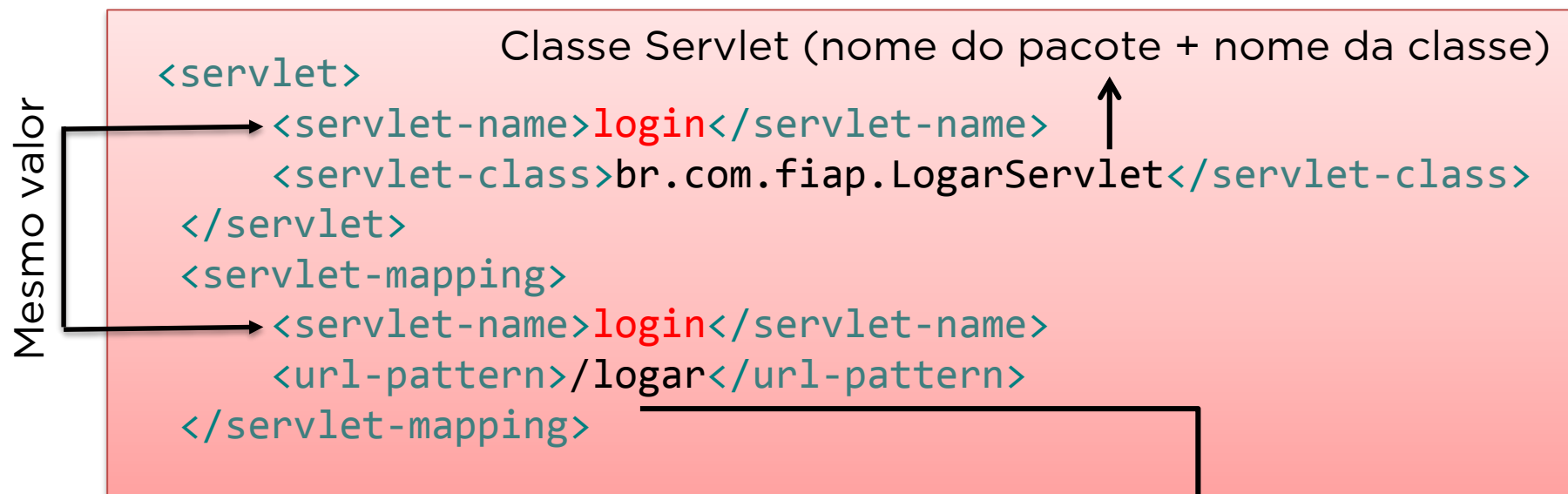
`http://localhost:8080/JavaWeb/logar`



Arquivo de configuração da aplicação web:

- Parâmetros iniciais do contexto (ServletContext)
- Configurações da Sessão (timeout)
- Declaração de servlets
- Mapeando de servlet (url-mapping)
- Listeners
- Filtros
- MIME Type
- Welcome File List
- Error Pages
- Locale e mapeamentos Encoding

- Primeiro defina a servlet, na tag <servlet> com um **nome** e o **pacote + classe**;
- Depois, na tag <servlet-mapping> defina a url para acessar a servlet, com o **mesmo nome** definido em <servlet> e a **url**;



Valor do endereço que a servlet será acessada:
<http://localhost:8080/localhost:8080/JavaWeb/logar>

Annotations: são textos inseridos diretamente no código fonte que expressam informações complementares sobre uma classe, seus métodos, atributos e etc..

Annotation para mapear a servlet

↑ <http://localhost:8080/localhost:8080/JavaWeb/logar>

```
@WebServlet("/logar")
public class LogarServlet extends HttpServlet {

}
```

Podemos configurar o mapeamento da servlet no arquivo web.xml e/ou com a annotation;

Crie uma página HTML e um Servlet para validar o login de um usuário:

- Usuário – FIAP e Senha – 2016;
- Exiba no browser se o usuário e senha estão corretos;

Passos sugeridos:

1. Crie uma página HTML com o formulário;
2. Crie a Servlet para recuperar os parâmetros do formulário e devolver a resposta;
3. Mapeie o servlet no web.xml e ajuste a propriedade action do formulário;

WEB.XML

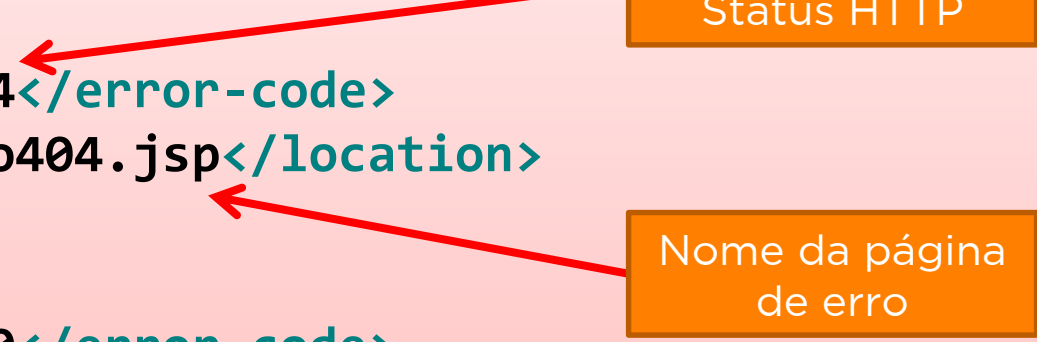
- É possível configurar a página inicial da aplicação;
- Os arquivos são procurados de cima para baixo, quando o sistema encontra o arquivo, ele o utiliza.

<http://localhost:8080/JavaWeb/>

```
<welcome-file-list>  
  <welcome-file>index.html</welcome-file>  
  <welcome-file>index.jsp</welcome-file>  
</welcome-file-list>
```

- É possível configurar páginas de erro para aplicação através do código do status HTTP ou o tipo da **exception** Java.
- **Status HTTP:**
 - 2xx – Sucesso
 - 200 - OK
 - 201 - Criado
 - 4xx – Erro Cliente
 - 401 – Não autorizado
 - 404 – Não encontrado
 - 5xx – Outros Erros
 - 500 – Erro interno do servidor

```
<error-page>
  <error-code>404</error-code>
  <location>/erro404.jsp</location>
</error-page>
<error-page>
  <error-code>500</error-code>
  <location>/erro500.jsp</location>
</error-page>
<error-page>
  <exception-type>
    java.lang.NullPointerException
  </exception-type>
  <location>/erroNullPointerException.jsp</location>
</error-page>
```



Código do Status HTTP

Nome da página de erro

- É possível configurar o tempo de timeout, ou seja, o tempo necessário de inatividade do usuário no sistema para que a sua sessão seja destruída.

```
<session-config>  
  <session-timeout>30</session-timeout>  
</session-config>
```



Tempo definido
em minutos

Copyright © 2013 - 2016 Prof. Me. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*“Para de perseguir o dinheiro e comece a perseguir o sucesso”
- Tony Hsieh*