Java Web

AULA 05 – SERVLETS PARTE III

1

Objetivos e Conceitos

- Objetivos:
  - Entender os redirecionamentos com forward(), include() e sendRedirect(). Apresentar as diferenças entre os redirecionamentos. Apresentar a passagem de parâmetros via escopo da requisição e via URL. Mostrar as diferenças.
- Conceitos:
  - Redirecionamento, forward, include, sendRedirect. Parâmetros. Escopo da requisição.

2

Tópicos

- Redirecionamentos
- Parâmetros

3

Redirecionamentos

4

Redirecionamentos.

Transferir o controle para outro recurso (servlet/jsp/html)

- Fluxo de páginas
- Páginas de erro
- Respostas a requisições
- MVC

Opção de incluir a saída na resposta original

Três tipos:

`RequestDispatcher.forward`
`RequestDispatcher.include`
`response.sendRedirect`

Para os dois primeiros deve-se obter um objeto do tipo **`RequestDispatcher`**

O terceiro é invocado direto de um **`HttpServletResponse`**

5

RequestDispatcher.

Dois tipos de redirecionamento

- **`RequestDispatcher.forward`**: redireciona (altera o fluxo), para outro recurso, efetuando uma requisição
- **`RequestDispatcher.include`**: efetua a requisição ao outro recurso e adiciona a saída do recurso chamado à saída do chamador

Deve-se obter um objeto do tipo **`RequestDispatcher`**

- A partir do **`ServletContext`**
- A partir do **`ServletRequest`**

6

O que é ServletContext?

Classe que define métodos que uma Servlet usa para se comunicar com o Contêiner

- Obter o MIME type de um arquivo
- Obter um **`RequestDispatcher`**
- Escrever um log
- Manipular o escopo da aplicação

Existe somente um *Context* por aplicação Web por máquina virtual

ServletContext existe dentro de um **`ServletConfig`**, que o servidor provê quando a Servlet é inicializada

Para obter um **`ServletContext`** usa-se:

```
ServletContext ctx = request.getServletContext();
ServletContext ctx = getServletContext();
ServletContext ctx = getServletConfig().getServletContext();
ServletContext ctx = request.getSession().getServletContext();
```


7

O que é ServletContext?

Alguns métodos de **`ServletContext`**:

- **`ctx.getContextPath()`** : retorna a raiz de contexto da aplicação
- **`ctx.getServletContextName()`** : retorna o nome da aplicação indicado no arquivo web.xml no atributo display-name
- **`ctx.getRealPath("/")`** : retorna o caminho real do recurso passado (caminho da web)
- **`ctx.getServerInfo()`** : retorna o nome e a versão do servlet conteiner
- **`ctx.getMajorVersion()`** : retorna o número (major) da versão da API servlet
- **`ctx.getMinorVersion()`** : retorna o número (minor) da versão da API servlet
- **`ctx.getRequestDispatcher("recurso")`** : retorna um RequestDispatcher para o recurso passado

8

Exemplo: InfoServlet.java.

```
package servlets;
// importações
@WebServlet(name = "InfoServlet", urlPatterns = {"/InfoServlet"})
public class InfoServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
                        throws ServletException, IOException {

        ServletContext ctx = request.getServletContext();

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html><html><head>");
            out.println("<title>Servlet InfoServlet</title></head><body>");
            out.println("<p>getContextPath() = " + ctx.getContextPath() + "</p>");
            out.println("<p>getServletContextName() = " +
                                        ctx.getServletContextName() + "</p>");
            out.println("<p>getRealPath(\"/\") = " +
                                        ctx.getRealPath("/") + "</p>");
            out.println("<p>getServerInfo() = " + ctx.getServerInfo() + "</p>");
            out.println("<p>getMajorVersion().getMinorVersion() = " +
                                ctx.getMajorVersion() + "." +
                                ctx.getMinorVersion() + "</p>");
            out.println("</body></html>");
        }
    }
+ HttpServlet methods. Click on the + sign on the left to edit the code.
}
```


9

RequestDispatcher

1. Obtendo a partir do **`ServletContext`**

```
RequestDispatcher rd = getServletContext().getRequestDispatcher("/principal.jsp");
```

- Recurso precisa começar com "/"
- É relativo ao *Context Root* (**`http://www.servidor.com.br:8080/app`**)
- Pode redirecionar para recursos fora da aplicação/servidor atual (outros contextos)

```
RequestDispatcher rd = getServletContext().
                           getContext("/OutraAplicacao").
                           getRequestDispatcher("/principal.jsp");
```

- Para isso:
  - Configurar **crossContext** no Tomcat, e **crossContextAllowed** no Glassfish

10

RequestDispatcher

2. Obtendo a partir do **`ServletRequest`**

```
RequestDispatcher rd = request.getRequestDispatcher("principal.jsp");
```

- Recurso pode ser relativo ao contexto atual (pastas)
- Não redireciona para outras aplicações (outros contextos)
- Se usar "/" se torna relativo ao *Context Root* (aplicação)
  - Se comporta como o retornado do **`ServletContext`**

11

forward()

Método de **`RequestDispatcher`**

```
rd.forward(request, response);
```

- Deve ser informado um objeto *request* e um *response*
- Podem ser os mesmos da Servlet que está chamando
- Usados para passar parâmetro
- Ao ser executado, redireciona o Servlet para outra página (JSP/Servlet)

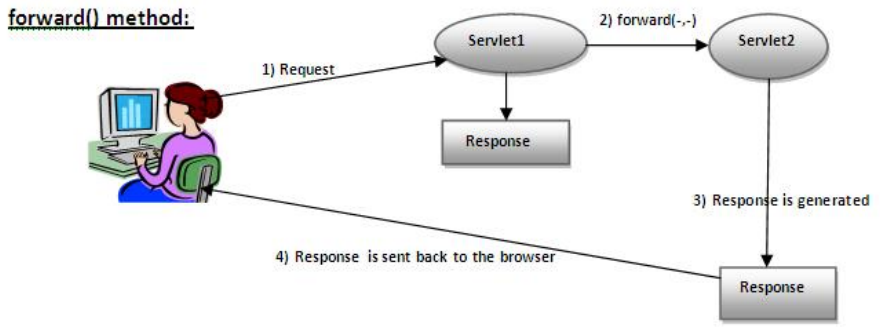
Uso:

```
RequestDispatcher rd = getServletContext().
                    getRequestDispatcher("/principal.jsp");
rd.forward(request, response);
```

- O Servlet que chamou este método NÃO pode inserir dados na saída (**`IllegalStateException`**)
- O controle é redirecionado para o novo recurso

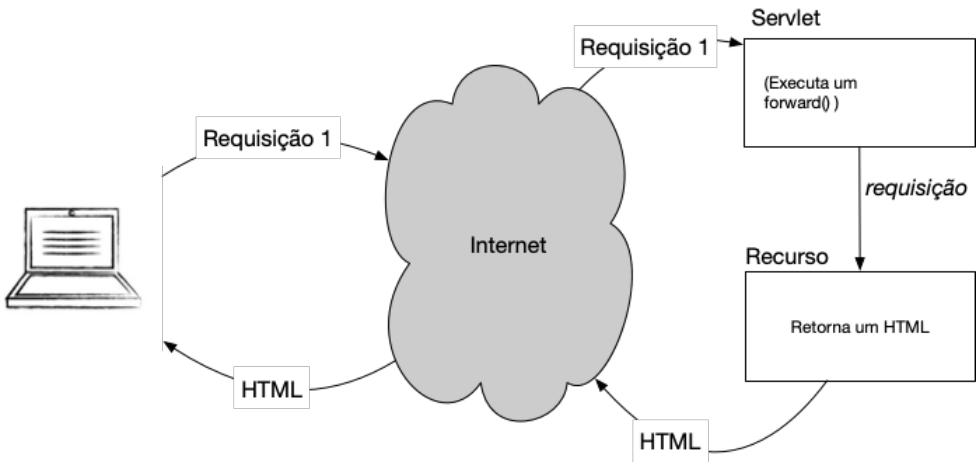
12

forward()



13

forward().



14

include()

Método de **`RequestDispatcher`**

```
rd.include(request, response);
```

- Deve ser informado um objeto *request* e um *response*
- Podem ser os mesmos da Servlet que está chamando
- Usados para passar parâmetro
- Ao ser executado, requisita o recurso e adiciona a saída na sua própria
- O controle não é desviado para o outro recurso

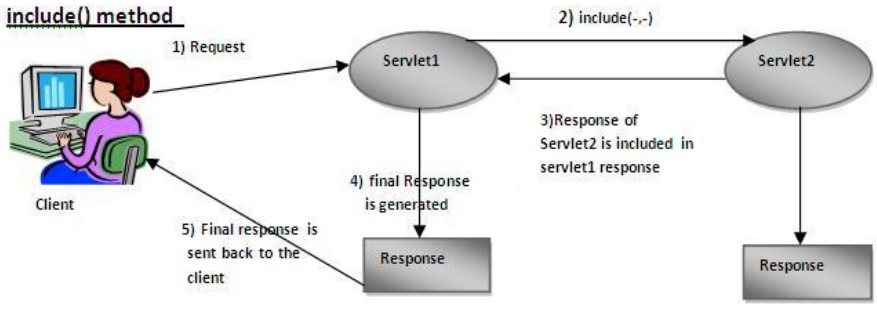
Uso:

```
RequestDispatcher rd = getServletContext().
                       getRequestDispatcher("/principal.jsp");
rd.include(request, response);
```

O recurso (**`principal.jsp`**) será requisitado e processado, e toda sua saída incluída na saída do Servlet chamador

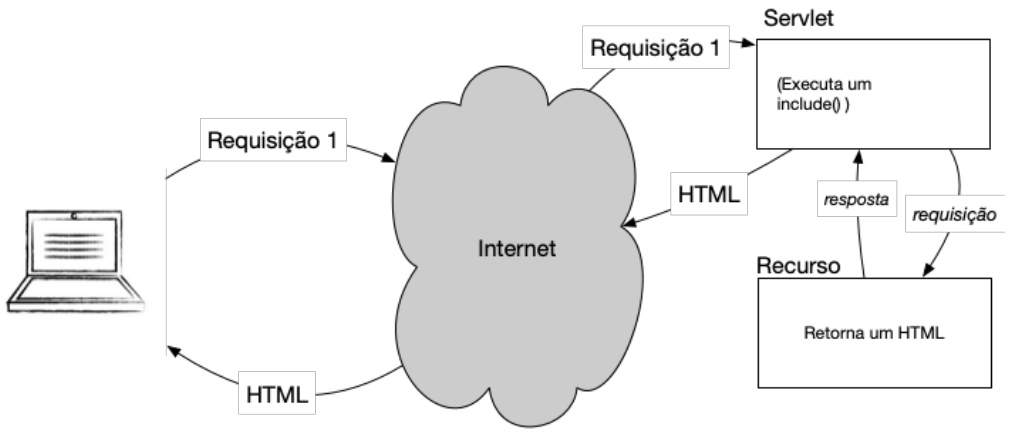
15

include()



16

include().



17

sendRedirect()

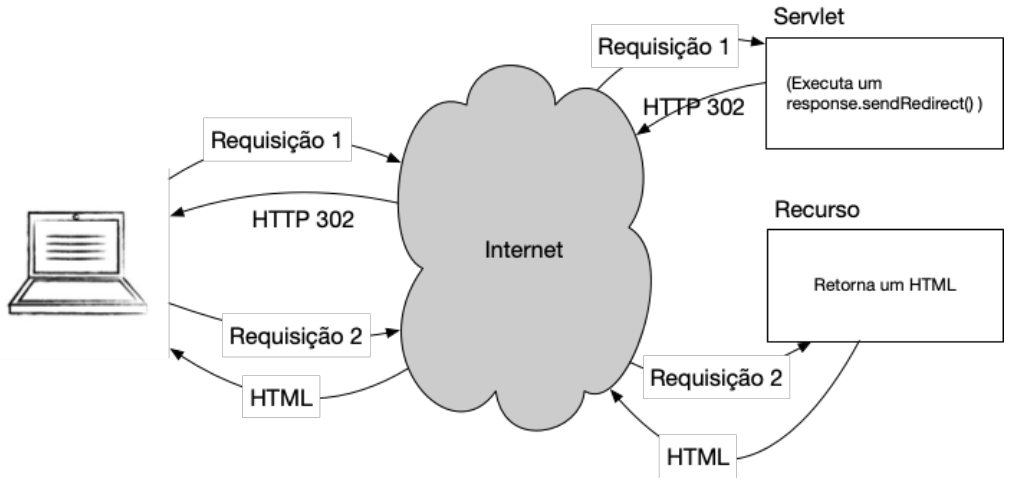
Método de **`HttpResponse`**

```
response.sendRedirect("principal.jsp");
```

- Solicita ao navegador que faça a requisição ao novo recurso
  - Retorna STATUS CODE do HTTP 302
- Browser executa 2 requisições: A original e a do redirecionamento
- A requisição atual é parada

18

sendRedirect().



19

Quando usar?

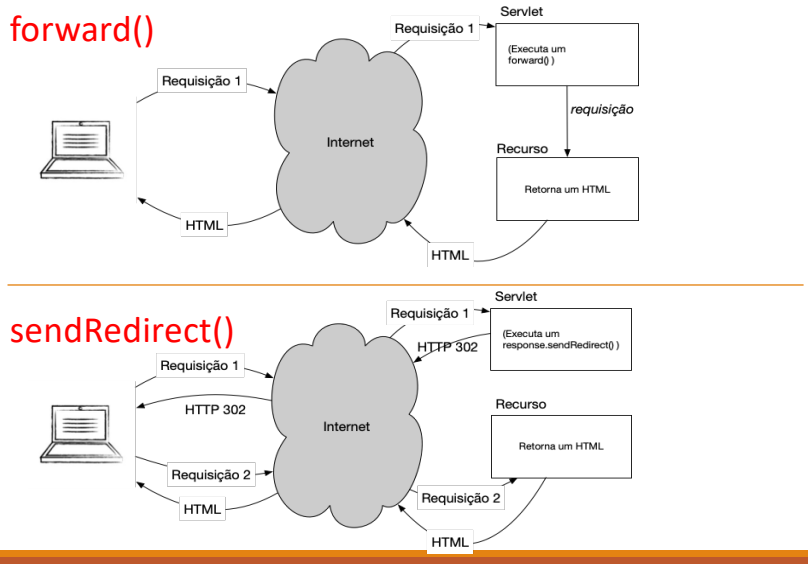
`forward()`
- Quando a requisição original pode ser repetida sem risco à integridade dos dados
- Refresh na página repete a primeira requisição, a requisição original
- Ex.: Uma consulta ao banco de dados

`sendRedirect()`
- Quando a requisição original NÃO pode ser repetida, pois gera risco à integridade dos dados
- Refresh na página repete a última requisição, só o redirecionamento
- Ex.: Uma inserção no banco de dados

`include()`
- Inclusão da saída de recursos na Servlet original

20

forward() x sendRedirect().



21

Exemplo: index.html

Alterar o **`index.html`** para conter

```
<html>
<head><title>Dados</title></head>
<body>
  <form action="Processa" method="post">
    <input type="radio" name="grupo" value="incl">Include
       <br/>
    <input type="radio" name="grupo" value="forw">Forward
       <br/>
    <input type="submit" value="Ok">
  </form>
</body>
</html>
```


22

Exemplo: include.jsp

```
<html>
<head><title>Include</title></head>
<body>
<h2>Isto foi incluído</h2>
<%
    for (int i=0; i<10; i++) {
       out.println(i + " - ");
    }
%>
</body>
</html>
```


23

Exemplo: forward.jsp

```
<html>
<head><title>Forward</title></head>
<body>
<h2>Foi REDIRECIONADO</h2>
<%
    for (int i=9; i>=0; i--) {
       out.println(i + " - ");
    }
%>
</body>
</html>
```


24

Exemplo: Processa.java

```
// importações

@WebServlet(urlPatterns = {"/Processa"})
public class Processa extends HttpServlet {
  public void doPost( HttpServletRequest request,
                      HttpServletResponse response)
                  throws ServletException, IOException {
     String str = request.getParameter("grupo");
     if (str.equals("incl")) {      // teste include
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.println("<html><head>");
        out.println("<title>Teste</title></head><body>");
        out.println("<h1>Teste de Include - ANTES</h1>");

        RequestDispatcher rd = getServletContext().
                           getRequestDispatcher("/include.jsp");
        rd.include(request, response);

        out.println("<h1>Teste de Include - DEPOIS</h1>");
        out.println("</body></html>");
     }
```


25

Exemplo: Processa.java (cont.)

```
     else {  // teste forward
         RequestDispatcher rd = getServletContext().
                     getRequestDispatcher("/forward.jsp");
         rd.forward(request, response);
         return;
     }
  }
}
```


26

Exercícios..

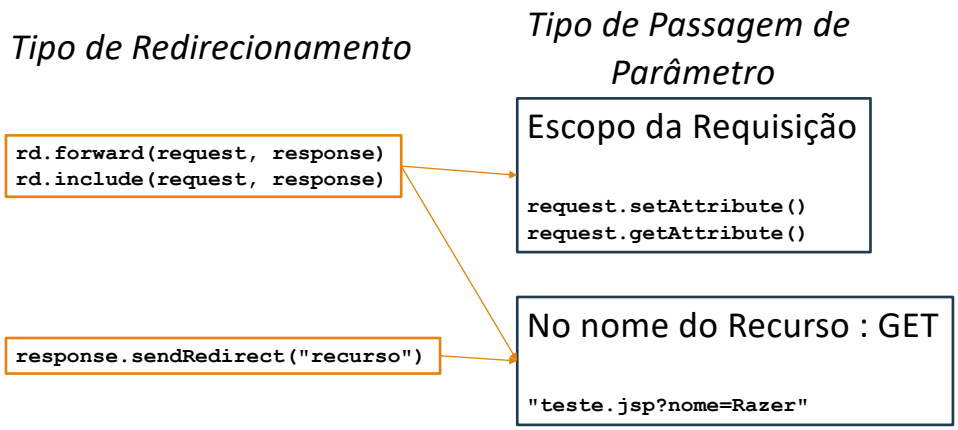
1. Executar a aplicação anterior

27

Parâmetros

28

Parâmetros no Redirecionamento



29

Parâmetros no Redirecionamento

Escopo da Requisição

A requisição termina quando a saída é enviada ao cliente (*browser*)

Enquanto estiver processando no servidor, todos os dados da requisição são mantidos:

- Parâmetros passados
- O método de invocação

Para redirecionamentos tipo **`forward()`** e **`include()`**:

- Pode-se passar parâmetros via *request*
- Parâmetros só existem nesta requisição

Usa-se os métodos em *request*: **`setAttribute()`** / **`getAttribute()`**

30

Parâmetros no Redirecionamento

Parâmetros são pares : Chave *String* / Dados *Object*

O parâmetro é identificado pela chave *String*, que deve ser a mesma ao passá-lo e ao recebê-lo

No chamador, para **passar** o parâmetro:

```
request.setAttribute(<String>, <Objeto>);
```

No chamado, para **receber** o parâmetro:

```
<Tipo> ret = (<Tipo>)request.getAttribute(<String>);
```


31

Parâmetros no Redirecionamento

No chamador, usa-se:

```
// AQUI SETAMOS OS ATRIBUTOS
request.setAttribute("nome", "razer");
RequestDispatcher rd = getServletContext().
            getRequestDispatcher("/principal.jsp");
rd.forward(request, response);
```


32

Parâmetros no Redirecionamento

JSP: Para se obter os atributos usa-se:

```
<% String str = (String)request.getAttribute("nome"); %>
```

SERVLET: Para se obter os atributos usa-se:

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

   ...
   String str = (String)request.getAttribute("nome");
   ...
```


33

Parâmetros no Redirecionamento

- Exemplo com **String**:
  - Na Servlet que vai redirecionar

```
request.setAttribute("nome", "Razer");
rd.forward(request, response);
```

  - Na Servlet/JSP alvo do redirecionamento

```
String ret = (String)request.getAttribute("nome");
```


34

Parâmetros no Redirecionamento

- Exemplo com **Objeto**:
  - Na Servlet que vai redirecionar

```
Pessoa p = new Pessoa();
request.setAttribute("pessoa", p);
rd.forward(request, response);
```

  - Na Servlet/JSP alvo do redirecionamento

```
Pessoa ret = (Pessoa)request.getAttribute("pessoa");
```


35

Parâmetros no Redirecionamento

- Exemplo com **Lista**:
  - Na Servlet que vai redirecionar

```
ArrayList<String> list = new ArrayList<String>();
request.setAttribute("lista", list);
rd.forward(request, response);
```

  - Na Servlet/JSP alvo do redirecionamento

```
List<String> ret = (List<String>)request.getAttribute("lista");
```


36

Parâmetros no Redirecionamento.

Note que, ao passar um parâmetro foi usado:

```
request.setAttribute(...)
```

Portanto, ao se recuperar este parâmetro deve ser usado:

```
request.getAttribute(...)
```

Só use **`request.getParameter(...)`** quando for recuperar parâmetros passados para a página (GET/POST)

NÃO EXISTE **`request.setParameter(...)`**

37

Parâmetros no Redirecionamento

Nome do Recurso : via GET

Também é possível passar parâmetros no nome do recurso invocado

Sintaxe na URL adicionando: **`?chave1=valor1&chave2=valor2`**

```
RequestDispatcher rd = getServletContext().
        getRequestDispatcher("/principal.jsp?nome=Razer");
rd.forward(request, response);
```

Mas este parâmetro não é passado no escopo da requisição, mas sim via GET

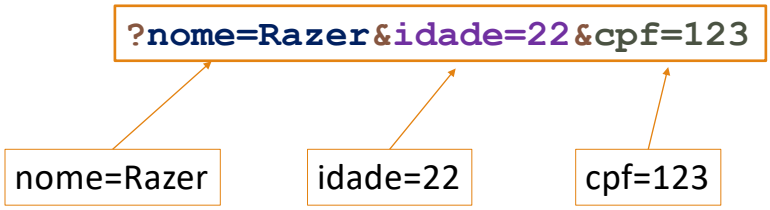
Deve-se obtê-lo com **`request.getParameter()`**:

```
String nome = request.getParameter("nome");
```


38

Parâmetros no Redirecionamento

Pode-se passar mais de um parâmetro separando-se por &



39

Parâmetros no Redirecionamento.

Pode-se passar mais de um parâmetro separando-se por &

```
String url = "/principal.jsp?nome=Razer&idade=22&cpf=123";
RequestDispatcher rd = getServletContext().
                       getRequestDispatcher(url);
rd.forward(request, response);
```

Deve-se obtê-los com **`request.getParameter()`**:

```
String nome  = request.getParameter("nome");
String idade = request.getParameter("idade");
String cpf   = request.getParameter("cpf");
```


40

Parâmetros no **sendRedirect()**

Nome do Recurso : via GET

O escopo da requisição não é persistido em uma chamada a **`sendRedirect()`**

- O escopo da requisição termina quando volta ao cliente
- O **`sendRedirect()`** retorna ao browser, que faz a segunda requisição

Parâmetros devem ser passados via GET (no nome do recurso)

```
response.sendRedirect("principal.jsp?nome=Razer");
```

Este parâmetro é passado via GET
Deve-se obtê-lo com **`request.getParameter()`**:

```
String nome = request.getParameter("nome");
```

Assim, não é possível passar objetos, somente dados Strings

41

Parâmetros no **sendRedirect().**

Para resolver problemas de codificação de caracteres usa-se **`URLEncoder`**

```
String msg = URLEncoder.encode("Erro de acesso à servlet", "UTF-8");
response.sendRedirect("ErrorServlet?msg=" + msg);
```


42

Exemplo 1: index.html

Alterar o **`index.html`** para conter

```
<html>
<head><title>Dados</title></head>
<body>
  <a href="PortalServlet">Portal</a>
</body>
</html>
```


43

Exemplo 1: PortalServlet.java

```
package servlets;

// importações

@WebServlet(name = "PortalServlet", urlPatterns = {"/PortalServlet"})
public class PortalServlet extends HttpServlet {
   protected void processRequest(HttpServletRequest request,
                                 HttpServletResponse response)
                       throws ServletException, IOException {

       RequestDispatcher rd = request.
                              getRequestDispatcher("ErroServlet");
       request.setAttribute("msg", "Erro acessando a Servlet");
       rd.forward(request, response);
    }

+ HttpServlet methods. Click on the + sign on the left to edit the code.

}
```


44

Exemplo 1: ErroServlet.java.

```
package servlets;

// importações

@WebServlet(name = "ErroServlet", urlPatterns = {"/ErroServlet"})
public class ErroServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
            throws ServletException, IOException {

        String mensagem = (String)request.getAttribute("msg");

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html><head>");
            out.println("<title>Servlet ErroServlet</title>");
            out.println("</head><body>");

            out.println("<h1>Mensagem</h1>");
            out.println("<h2>" + mensagem + "</h2>");

            out.println("</body></html>");
        }
    }
+ HttpServlet methods. Click on the + sign on the left to edit the code.
}
```


45

Exemplo 2: index.html

Alterar o index.html para conter

```
<html>
<head><title>Dados</title></head>
<body>
  <a href="PortalServlet">Portal</a>
</body>
</html>
```


46

Exemplo 2: PortalServlet.java

```
package servlets;

// importações

@WebServlet(name = "PortalServlet", urlPatterns = {"/PortalServlet"})
public class PortalServlet extends HttpServlet {
   protected void processRequest(HttpServletRequest request,
                                 HttpServletResponse response)
                       throws ServletException, IOException {

       String msg = URLEncoder.encode("Erro de acesso à servlet",
                                      "UTF-8");
        response.sendRedirect("ErrorServlet?msg=" + msg);
   }

+ HttpServlet methods. Click on the + sign on the left to edit the code.

}
```


47

Exemplo 2: ErroServlet.java.

```
package servlets;

// importações

@WebServlet(name = "ErroServlet", urlPatterns = {"/ErroServlet"})
public class ErroServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String mensagem = request.getParameter("msg");

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html><head><title>Servlet ErroServlet</title></head><body>");

            out.println("<h1>Mensagem</h1>");
            out.println("<h2>" + mensagem + "</h2>");

            out.println("</body></html>");
        }
    }
+ HttpServlet methods. Click on the + sign on the left to edit the code.
}
```


48

Exercícios..

1. Executar os exercícios anteriores.
2. Criar uma Servlet que, ao ser submetida de um formulário, verifica se o dado "nome" foi preenchido.
   a) Se foi preenchido, apresenta uma tela mostrando o nome digitado
   b) Se não foi preenchido, redireciona para um HTML (ou JSP) contendo uma mensagem de erro
3. Criar uma Servlet que é submetida a partir de um formulário de login (usuário/senha).
   a) Se usuário/senha estão no banco de dados, redireciona para uma Servlet chamada PortalServlet que mostra uma página de boas-vindas
   b) Se não, redireciona para uma Servlet chamada ErroServlet que mostra uma página fixa de erro
4. Melhorar a aplicação do exercício 3 para, ao ocorrer um erro, a mensagem de erro ser passada como parâmetro para a Servlet de erro
5. Melhorar a aplicação do exercício 4 para usar uma JSP ao invés de PortalServlet e outra JSP ao invés de ErroServlet

49