



Estácio

SERVLETS – 2ª Parte

OBJETIVOS

- Compreender a Tecnologia de Servlets
- Criar aplicativos Web Baseados nessa Tecnologia
- Compreender as formas de recepção de dados, bem como o encaminhamento e redirecionamento no fluxo de páginas

ARQUITETURA WEB

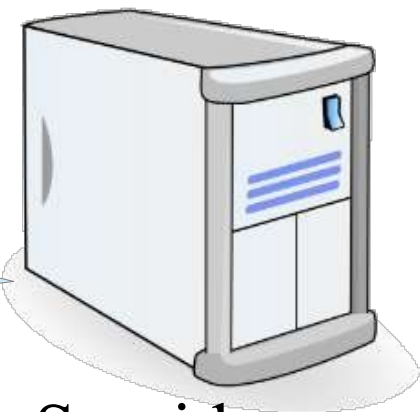
- **Arquitetura Web**
 - Nas aulas passadas ...

Cliente



REQUEST

RESPONSE



Servidor

TIPOS DE REQUISIÇÃO

- De maneira rápida, vimos também que as requests podem ser de dois tipos:
 - **POST** → Função é enviar dados para o servidor
 - usualmente por formulários (forms)
 - **GET** → Função é requisitar dados do servidor
 - usualmente por links/barra de url

REQUISIÇÕES GET

- Podemos passar parâmetros pela requisição do tipo **GET** através do seguinte esquema:

<http://servidor/servlet?param1=valor1>

- O “truque” é a interrogação: ?
 - Esse caractere indica que:
 - o endereço já acabou
 - tudo que vem em seguida é parâmetro

REQUISIÇÕES POST

- Na URL não aparecem os parâmetros!
 - Estes seguem na requisição logo após o cabeçalho HTTP
- Não há limite de tamanho numa requisição POST (GET → ~ 1KB)
 - Os servidores impõem o seu limite, o que é customizável

```
POST /index.html HTTP/1.0
Accept: text/html
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

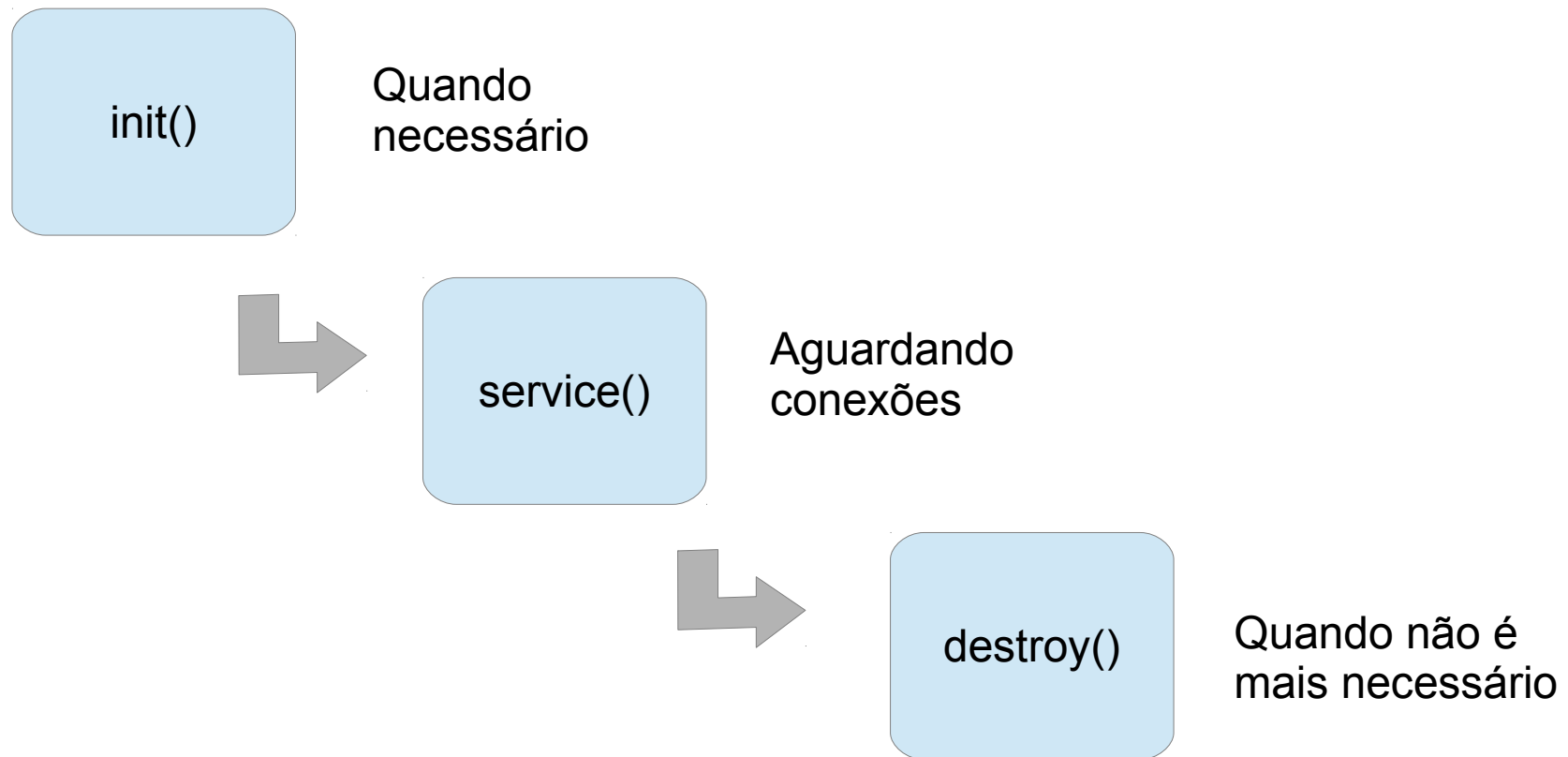
Nome=Daniel&Idade=23&Curso=SistemasInformacao
```

REQUISIÇÕES GET e POST

- Do ponto de vista do Servlet:
 - Qual a implicação se for: **GET**?
 - Qual a implicação se for: **POST**?
- Ambos os métodos enviam parâmetros!!
- Mas como diferenciar então:
 - Será que temos como diferenciar?

REQUISIÇÕES GET e POST

- Lembram-se disto:



REQUISIÇÕES GET e POST

- Vamos olhar só para o método service()



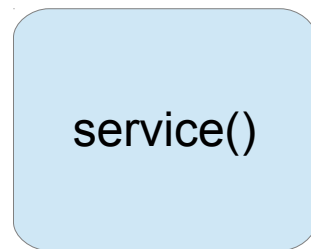
service()

- O service aguarda requisições

E se chega uma requisição: **POST**

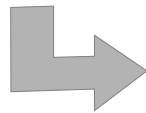
REQUISIÇÕES GET e POST

- Vamos olhar só para o método service()



- O service aguarda requisições

E se chega uma requisição: **POST**



REQUISIÇÕES GET e POST

- Vamos olhar só para o método service()



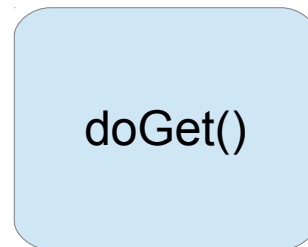
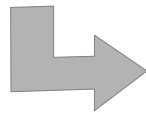
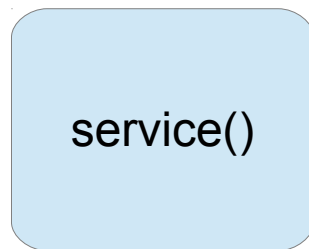
service()

- O service aguarda requisições

E se chega uma requisição: **GET**

REQUISIÇÕES GET e POST

- Vamos olhar só para o método `service()`



- O service aguarda requisições

E se chega uma requisição: **GET**


REQUISIÇÕES GET e POST

- Onde estão esses métodos: doGet e doPost ?

```

L  */
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<!DOCTYPE html><html><head>");
        out.println("<title>Servlet CalculaMedia</title>");
        out.println("</head><body><h1>Calculo Media </h1>");
        String valorA = request.getParameter("valorA");
        String valorB = request.getParameter("valorB");
        double dValorA = Double.valueOf(valorA);
        double dValorB = Double.valueOf(valorB);
        double media = ( dValorA + dValorB )/2;
        out.println("<p>A media &eacute;; " + media + "</p>");
        out.println("</body></html>");
    } finally {
        out.close();
    }
}

```



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

REQUISIÇÕES GET e POST

- Ambos chamam o método *processRequest* do template do Netbeans

```
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods"
+ /**...*/
@Override
protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {
-   processRequest(request, response);
+ }

+ /**...*/
@Override
protected void doPost(HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException {
-   processRequest(request, response);
+ }

+ /**...*/
@Override
public String getServletInfo() {
-   return "Short description";
+ }
}
```

REQUISIÇÕES GET e POST

- E se quisermos tratar de forma diferente estas requisições?
 - Basta sobreescrever os referidos métodos:
 - Podemos redirecionar para outra página
 - Podemos apresentar uma página de erros
 - Podemos ignorar e não executar nada
 - Apresentar resultados diferentes
 - ...

(Bom exercício para ser feito em Laboratório)

REDIRECIONANDO REQUISIÇÕES

REDIRECIONAMENTO

- O Redirecionamento consiste numa resposta para o cliente indicando uma nova URL a ser chamada:
 - **Implicações:**
 - Uma nova requisição será feita pelo cliente
 - Será alterado endereço na barra de navegação do browser
 - Você pode redirecionar para um domínio externo
 - **Sintaxe**
 - `response.sendRedirect("http://www.endereco.com/");`

(OBS: Como irá gerar uma resposta é um método de HttpServletResponse)

REDIRECIONAMENTO

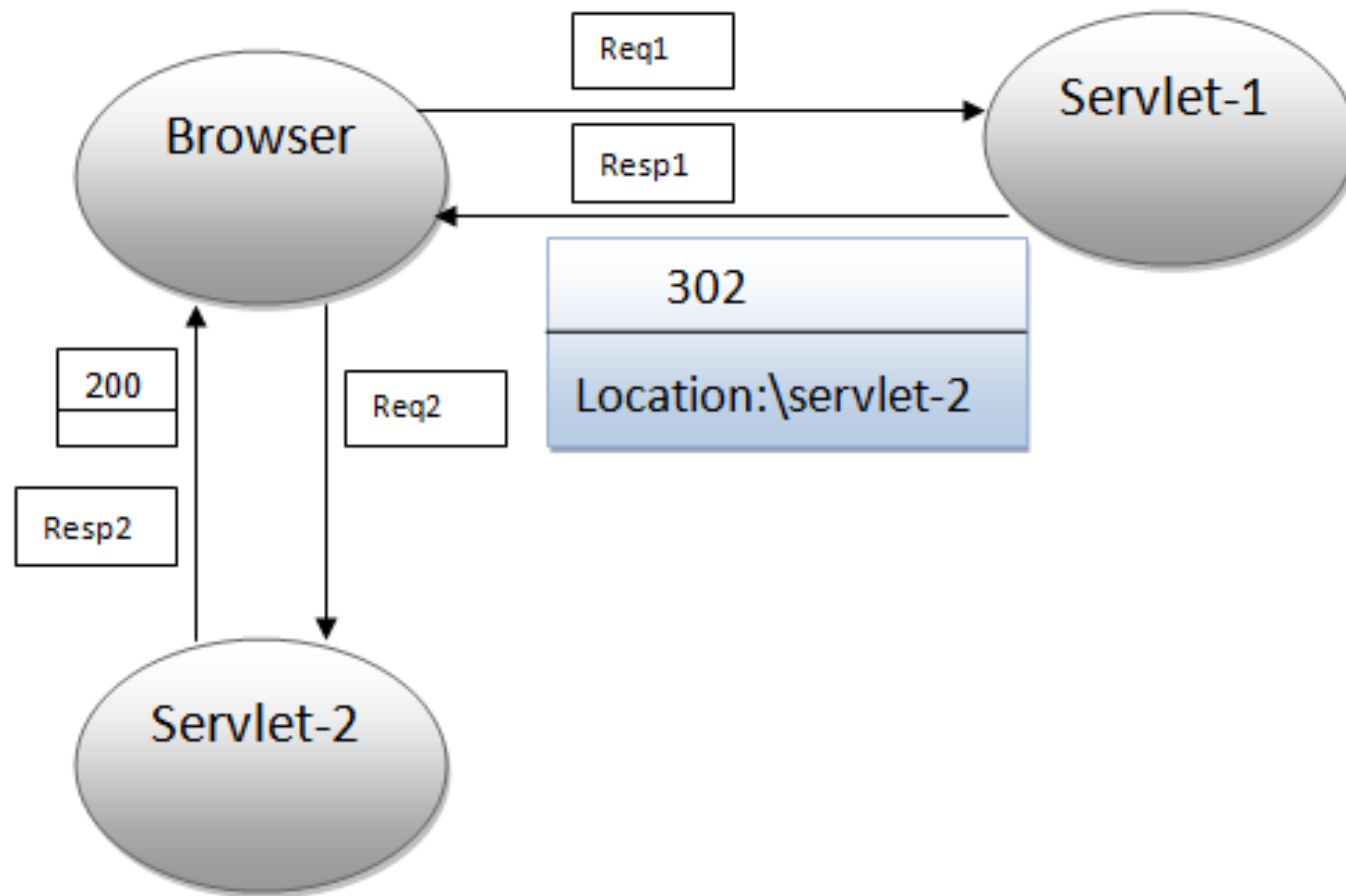
- Ex.:
 - Tratar diferenciado requisições GET e POST

```
+ /**...*/
@Override
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
    response.sendRedirect("http://endereco.com.br?id=123");
}

+ /**...*/
@Override
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
    processRequest(request, response);
}
```

REDIRECIONAMENTO

- O browser recebe uma resposta solicitando redirecionamento e gera uma nova requisição



ENCAMINHANDO REQUISIÇÕES

ENCAMINHAMENTO

- O Encaminamento consiste em encaminhar, internamente na aplicação, uma requisição para que outro arquivo a trate (Servlet, HTML, JSP, ...)
 - **Implicações:**
 - Transparente para o cliente
 - Não será alterado o endereço na barra de navegação do browser
 - Você só pode redirecionar dentro da mesma aplicação
 - **Sintaxe**
 - `RequestDispatcher rd = request.getRequestDispatcher("/Servlet");`
 - `rd.forward(request, response);`

(OBS: No encaminhamento as informações na requisição original não são perdidas)

ENCAMINHAMENTO

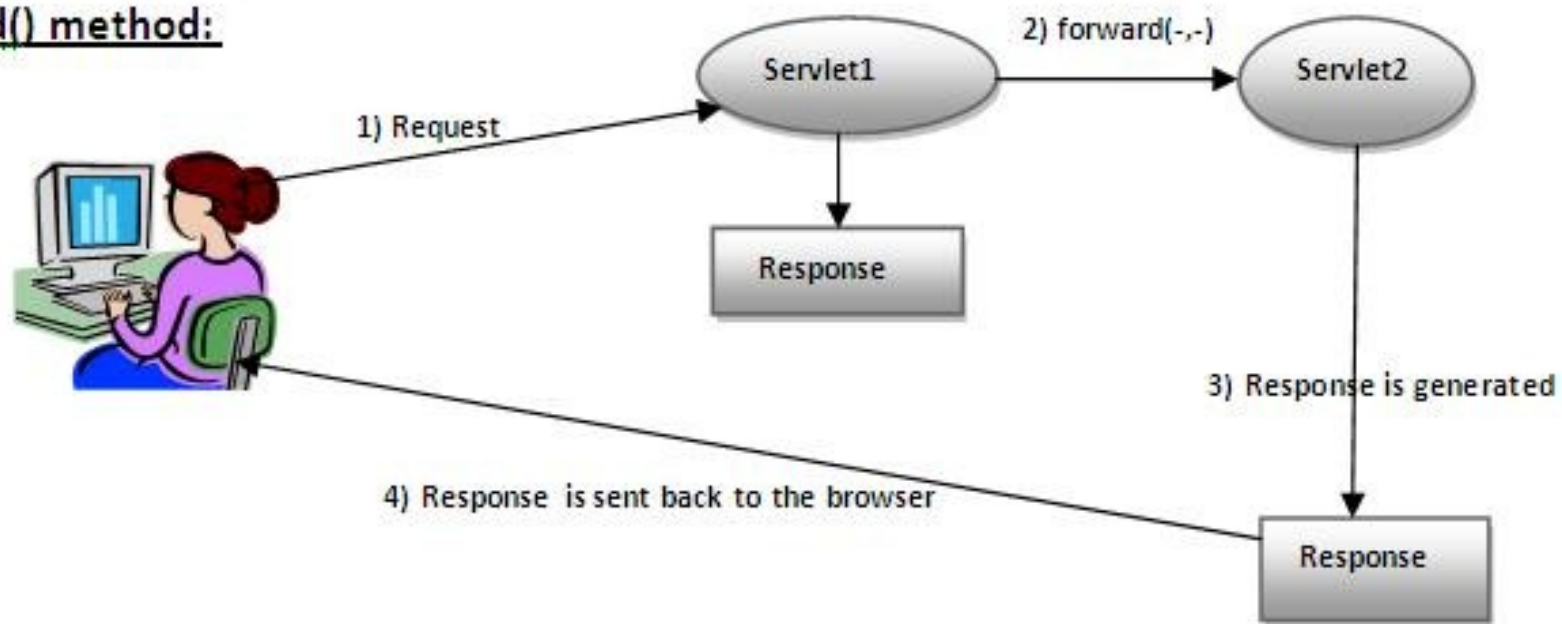
- Ex.:
 - Encaminhar requisição para uma página de erro!

```
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<!DOCTYPE html>");
        out.println("<html><head>");
        out.println("<title>Pagina de Teste</title>");
        out.println("</head><body>");
        out.println("Pode dar erro: "
            + request.getParameter("nogfgme").toUpperCase());
        out.println("</body></html>");
    } catch (Exception e) {
        RequestDispatcher rd = request.getRequestDispatcher("/Erro");
        rd.forward(request, response);
    } finally {
        out.close();
    }
}
```

ENCAMINHAMENTO

- O processo é interno à aplicação, transparente ao cliente e preserva *request* e *response*

forward() method:



ATRIBUTOS

- Encaminhar requisições é útil...
 - Mas se pudéssemos acrescentar informações na requisição, antes de encaminhá-la, seria bem mais útil!
- Os parâmetros não podem ser setados pelo Servlet
 - Então, como acrescentar informações?
- Utilizamos o conceito de **atributos** para guardar/armazenar **objetos** na requisição

ATRIBUTOS

- Com atributos podemos armazenar:
 - Cálculos matemáticos
 - Resultados de **queries** em banco dados
 - Derivações lógicas
 - ...
- Atributos têm escopo de requisição
 - Terminada a requisição os atributos não estarão mais acessíveis!

ATRIBUTOS

- Atributos são identificados por um **nome** (**String**) e guardam um **valor** (**Objeto**)
- Para guardar/armazenar atributos na requisição:
 - **request.setAttribute(“nome”,objeto);**
- Para ler esse valores:
 - **request.getAttribute(“nome”);**

(OBS: Atributos são encontrados apenas nas requisições, não nas respostas!)

ENCAMINHAMENTO E ATRIBUTOS

- Ex.:
 - Passando atributos para a Página de Erros

```
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<!DOCTYPE html>");
        out.println("<html><head>");
        out.println("</head><body>");
        out.println("Pode dar erro: "
            + request.getParameter("nogfgme").toUpperCase());
        out.println("</body></html>");
    } catch (Exception e) {
        RequestDispatcher rd = request.getRequestDispatcher("/Erro");
        request.setAttribute("MensagemErro", e.getMessage());
        rd.forward(request, response);
    } finally {
        out.close();
    }
}
```

ENCAMINHAMENTO E ATRIBUTOS

- Outra utilidade:
 - Que tal o nosso aplicativo de Calculo de Media ser composto por dois Servlets?
 - ServletCalcula
 - Faz todo o processamento
 - ServletCalculaView
 - Responsável por apresentar a resposta
- Acreditem que isso facilita o reuso de código, por separar processamento de apresentação

ENCAMINHAMENTO E ATRIBUTOS

- Ex.:
 - ServletCalcula

```
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    try {
        String valorA = request.getParameter("valorA");
        String valorB = request.getParameter("valorB");
        double dValorA = Double.valueOf(valorA);
        double dValorB = Double.valueOf(valorB);
        double media = ( dValorA + dValorB )/2;
        request.setAttribute("media", new Double(media));
        RequestDispatcher rd =
            request.getRequestDispatcher("/CalculaView");
        rd.forward(request, response);
    } catch (Exception e) {
        request.setAttribute("MensagemErro", e.getMessage());
        RequestDispatcher rd =
            request.getRequestDispatcher("/Erro");
        rd.forward(request, response);
    }
}
```

ENCAMINHAMENTO E ATRIBUTOS

- Ex.:
 - ServletCalculaView

```
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    Double media = (Double)request.getAttribute("media");
    try {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head></head>");
        out.println("<body>");
        out.println("<p>A media é: " + media + "</p>");
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

Até a Próxima Aula!