

Sistemas Cliente-Servidor

**Arquiteturas e Tecnologias
Cliente Servidor
(Servlets)**

OBJETIVOS

- Apresentar os dois principais tipos de requisição
- Apresentar os dois tipos mais comuns de redirecionamento
- Armazenamento de Dados na **request**

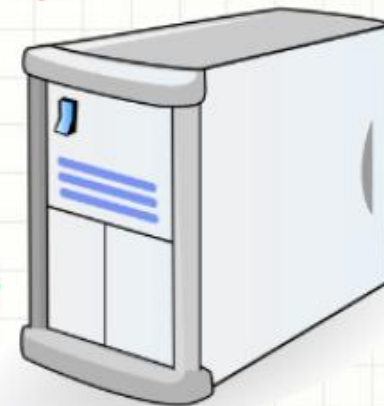
RELEMBRANDO!!!!

Arquitetura Web e Serviços



Cliente

REQUEST



Servidor

RESPONSE

Tipos de Requisição

De maneira rápida, vimos também que as **requests** podem ser de dois tipos:

POST → Função é enviar dados para o servidor

GET → Função é requisitar dados do servidor

- POST: usualmente por forms
- GET: usualmente por links/barra de url

NOTA: Ambas podem ser geradas por AJAX

Tipos de Requisição

Na aula passada:

Formulário **index.html**

→ Cria a request

→ Envia para a servlet **Calc** por POST

Podemos fazer o mesmo com GET?

– Digite na barra de endereços:

`http://localhost:8080/Calc?valor1=75&valor2=5&operacao=somar`

O que acontece?

Requisição com GET

Podemos passar alguns **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

Requisição com GET

`http://servidor/servlet?param1=valor1`

Depois da ?

- Nome do parâmetro (no exemplo, **param1**)
- Sinal de igualdade
- Valor do parâmetro (no exemplo, **valor1**)

E se quiser passar mais de um parâmetro?

Requisição com GET

`http://servidor/servlet?param1=valor1¶m2=valor2`

Para indicar valores de mais parâmetros, basta separá-los com o uso de um **&**

Podemos passar “infinitos” parâmetros?

NÃO com o GET: até cerca de 1KB

Requisição e Resposta HTTP

Os dois principais, e mais usados, tipos de requisições são o GET e POST. Eles possuem algumas diferenças básicas, porém muito importantes:

GET

- Só pode enviar até 255 caracteres de informações
- As informações vão como parte da URL (não indicado para senha)
- O browser ou proxy faz o cache da página pela URL
- Feito quando uma URL é digitada, via um link ou por um form de método GET

POST

- Pode enviar conteúdo ilimitado de informações
- Pode enviar texto e binário (ex: arquivos)
- O browser ou proxy não fazem o cache da página pela URL
- Feito por um form de método POST

Requisição com GET

Do ponto de vista do **servlet**, o que muda?

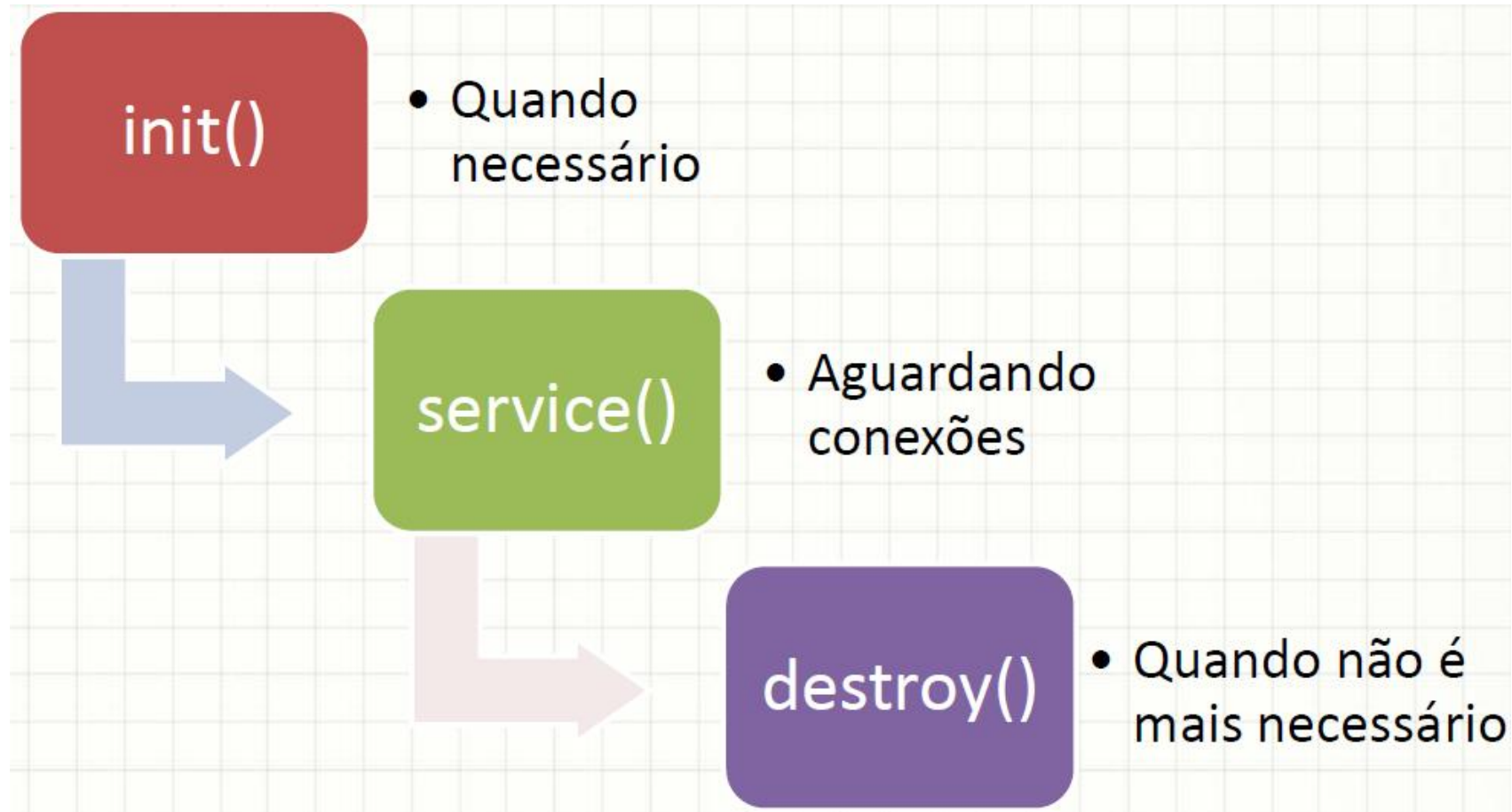
Usando NetBeans, **NADA**, os dados chegam, com o **GET**, da mesma forma que com o **POST**

Quer dizer que não temos como diferenciar um do outro no **servlet**?

Recebendo GET e POST no Servlet

Na verdade, é possível

Lembram-se deste esquema?



Recebendo GET e POST no Servlet

Vamos pensar só no **service()**



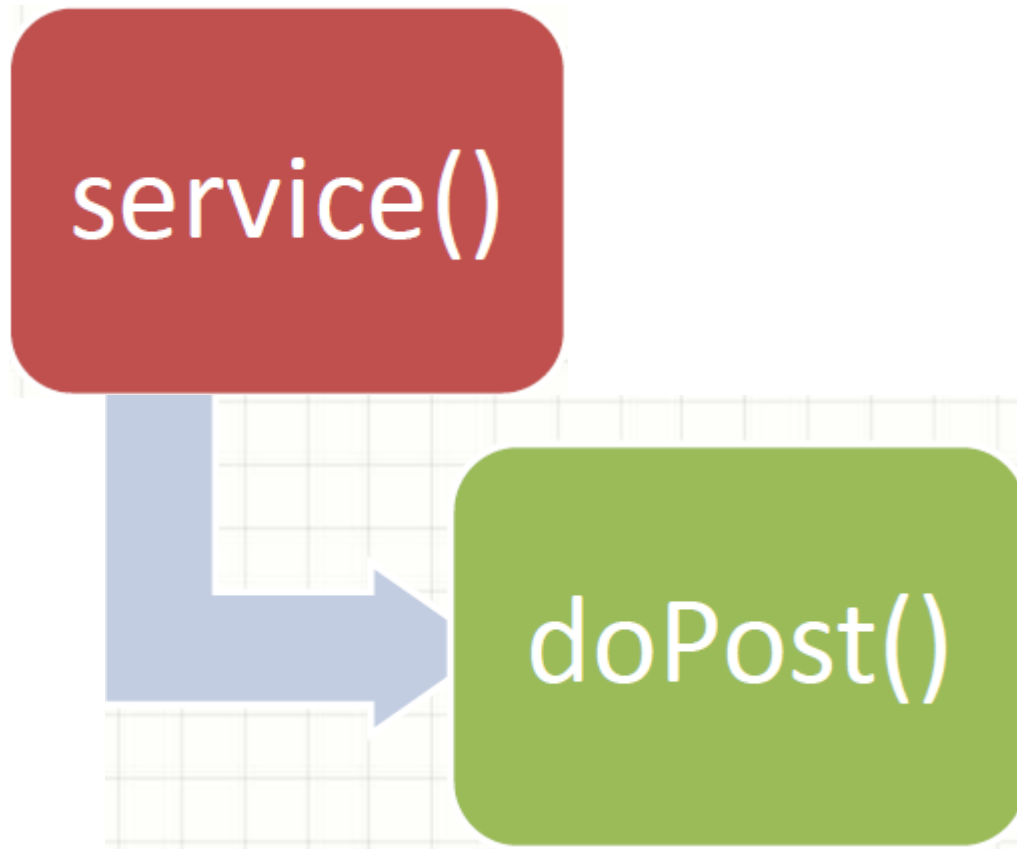
service()

**Aguardando
conexões**

- O **service** aguarda requisições...
- Se uma **POST** chega...

Recebendo GET e POST no Servlet

Vamos pensar só no **service()**



- O **service** aguarda requisições...
- Se uma **POST** chega...

Recebendo GET e POST no Servlet

Vamos pensar só no **service()**



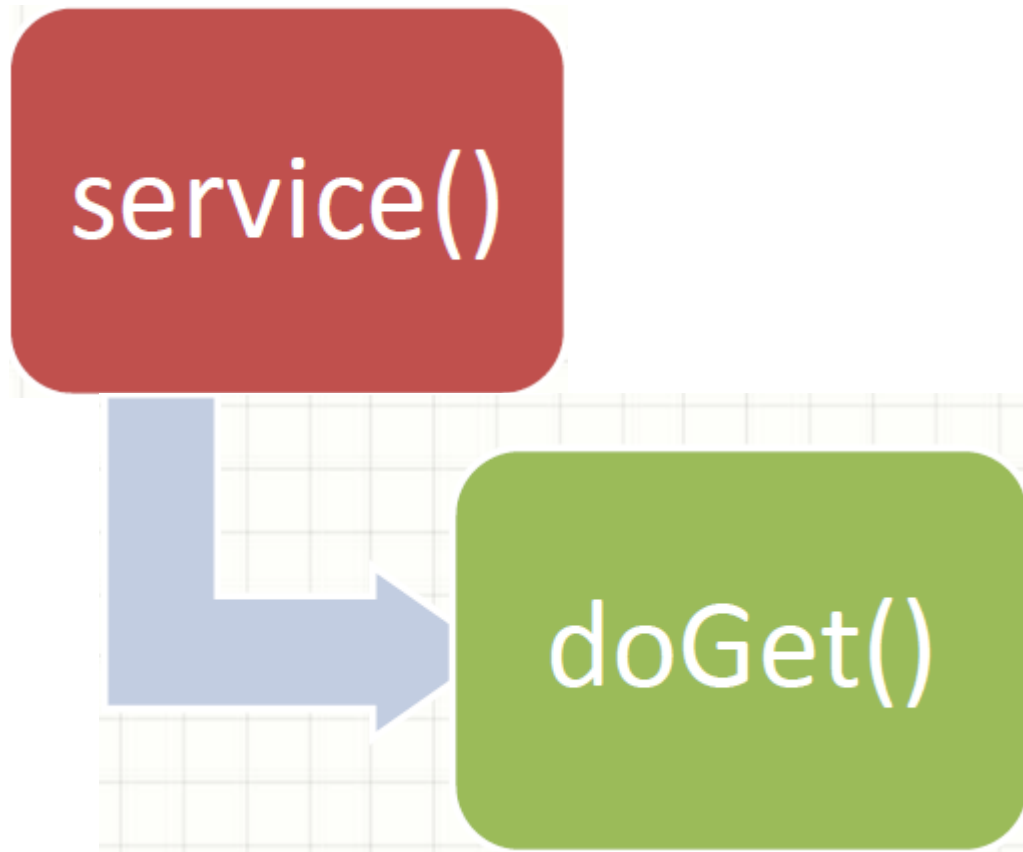
service()

**Aguardando
conexões**

- O **service** aguarda requisições...
- Mas... E se uma **GET** chega?

Recebendo GET e POST no Servlet

Vamos pensar só no **service()**



- O **service** aguarda requisições...
- Mas... E se uma **GET** chega?

Recebendo GET e POST no Servlet

Vamos pensar só no `service()`

`service()`

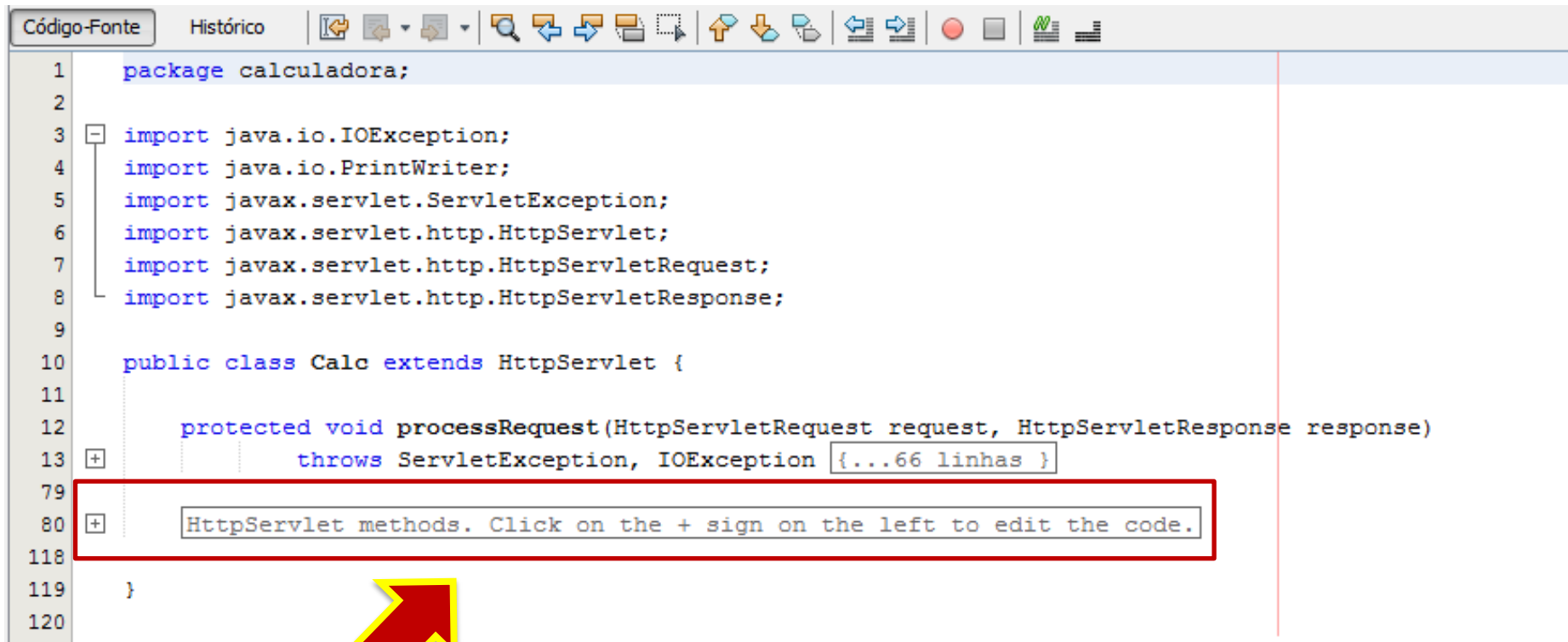
**Como não vimos
isso na Servlet???**

...es...

GET chega?

Entendendo o Servlet

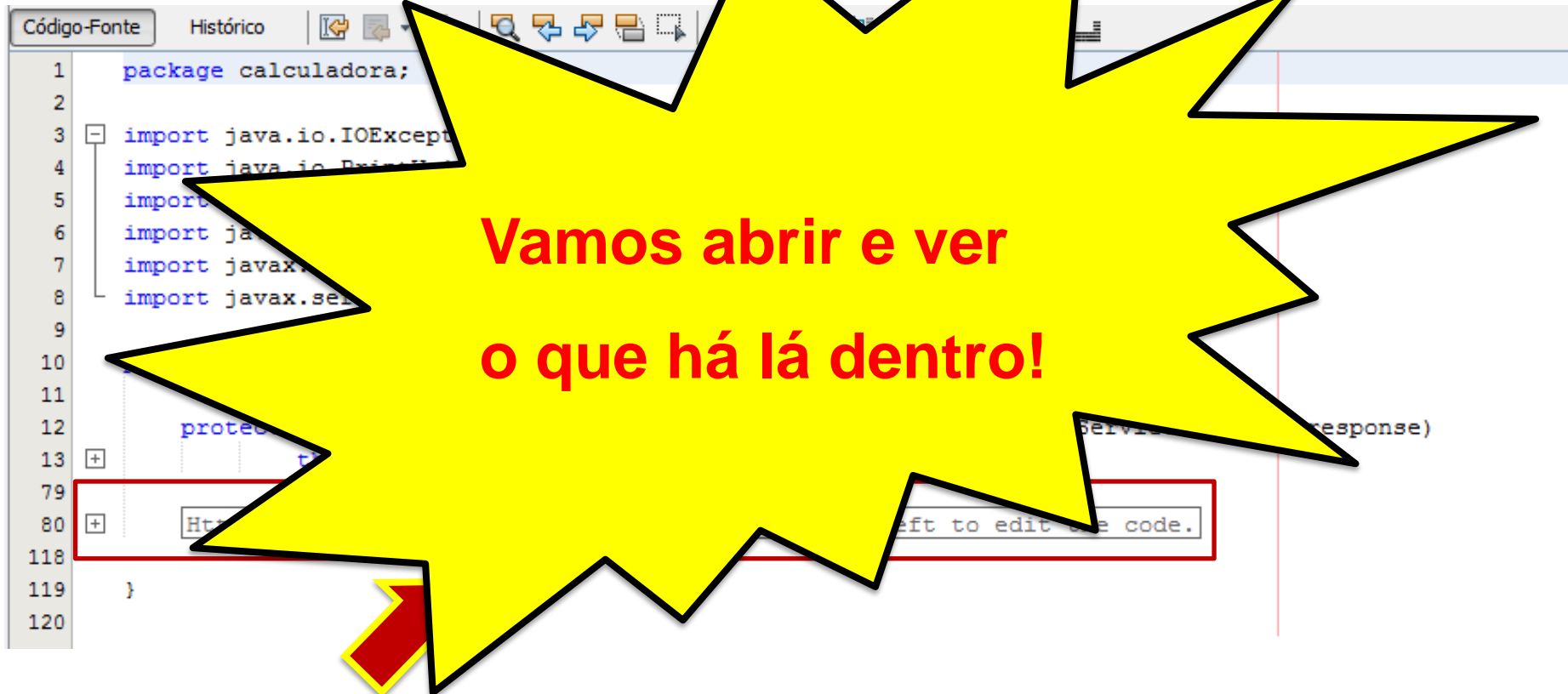
Lembra-se desta parte aqui?



```
1 package calculadora;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 public class Calc extends HttpServlet {
11
12     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
13         throws ServletException, IOException {...66 linhas }
14
15     // ...
16
17     // HttpServlet methods. Click on the + sign on the left to edit the code.
18
19     // ...
20
21 }
```

Entendendo o Servlet

Lembra-se desta parte aqui?



Entendendo o Servlet

Observe os métodos `doGet` e `doPost`

```
public class Calc extends HttpServlet {  
  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...66 linhas }  
  
    @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";  
    }  
}
```

Entendendo o Servlet

Observe os métodos `doGet` e `doPost`

```
public class Calc extends HttpServlet {  
  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException { ...66 linhas }  
  
    @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";  
    }  
}
```

Eles simplesmente
redirecionam a chamada
para o método
`processRequest()`!

Rejeitando GET

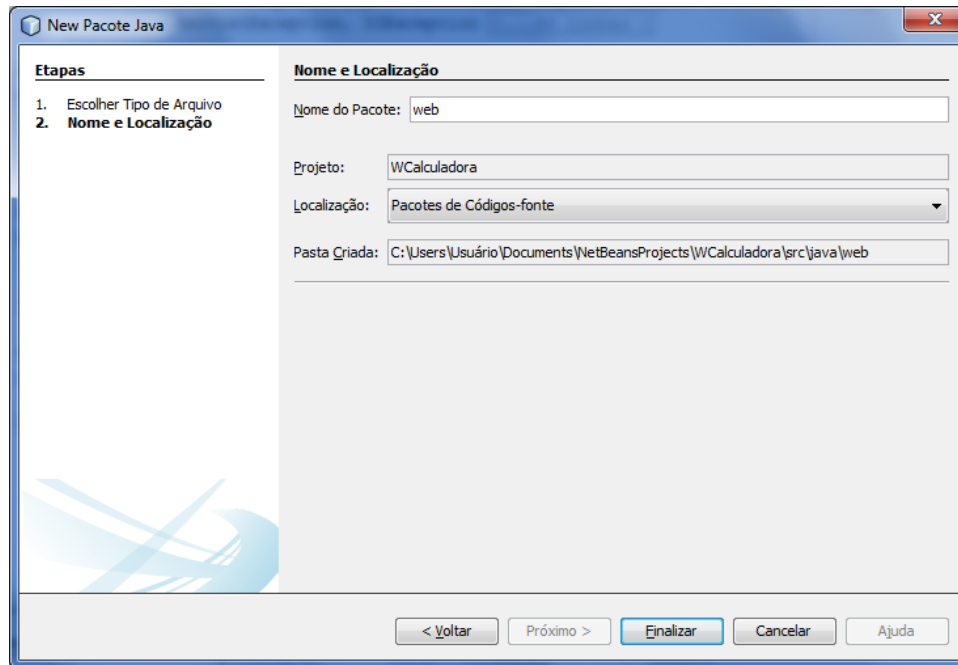
E se não quisermos que nosso **servlet** responda com requisições GET?

Podemos ir até o método **doGet()** e remover a chamada ao **processRequest()**

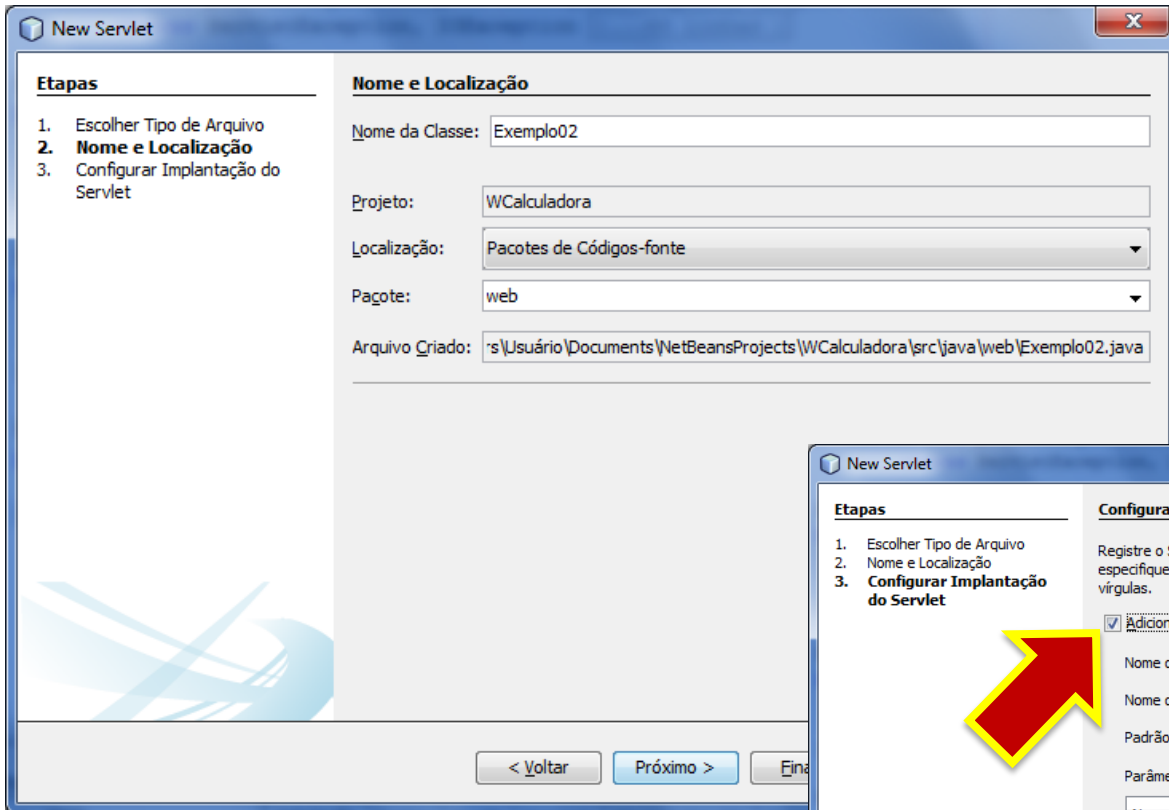
É possível encaminhar para página de erro...

Ou para a tela correta de preenchimento!

No projeto Calculadora, crie uma pacote com nome web



No pacote web, crie uma servlet com nome Exemplo02



New Servlet

Etapas

1. Escolher Tipo de Arquivo
- 2. Nome e Localização**
3. Configurar Implantação do Servlet

Nome e Localização

Nome da Classe:

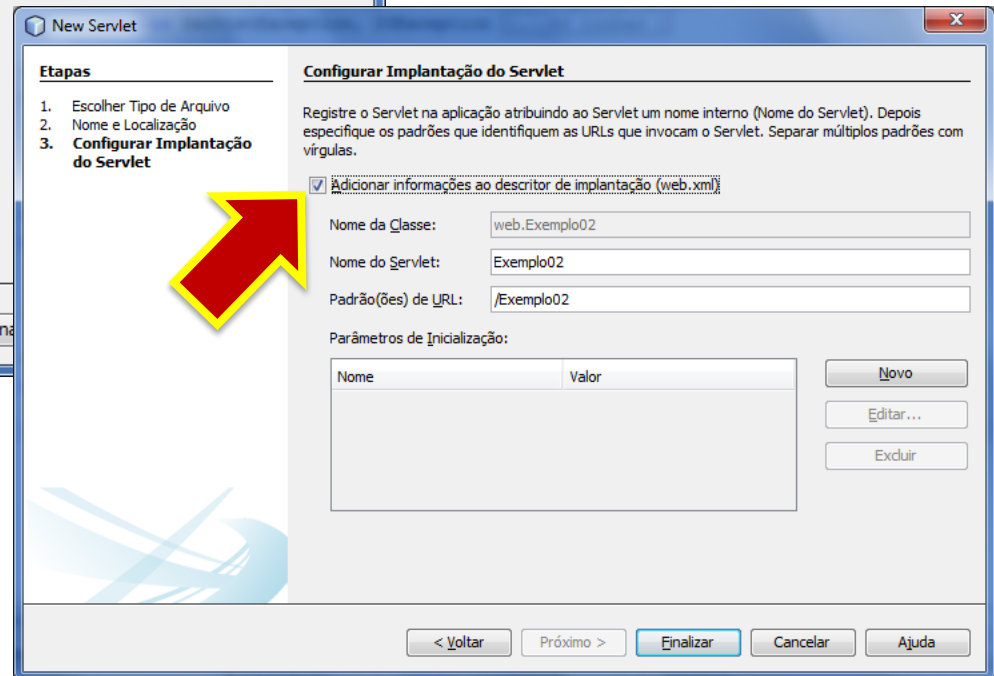
Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > Finalizar



New Servlet

Etapas

1. Escolher Tipo de Arquivo
2. Nome e Localização
- 3. Configurar Implantação do Servlet**

Configurar Implantação do Servlet

Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.

☒ Adicionar informações ao descritor de implantação (web.xml)

Nome da Classe:

Nome do Servlet:

Padrão(ões) de URL:

Parâmetros de Inicialização:

Nome	Valor
------	-------

< Voltar Próximo > Finalizar Cancelar Ajuda

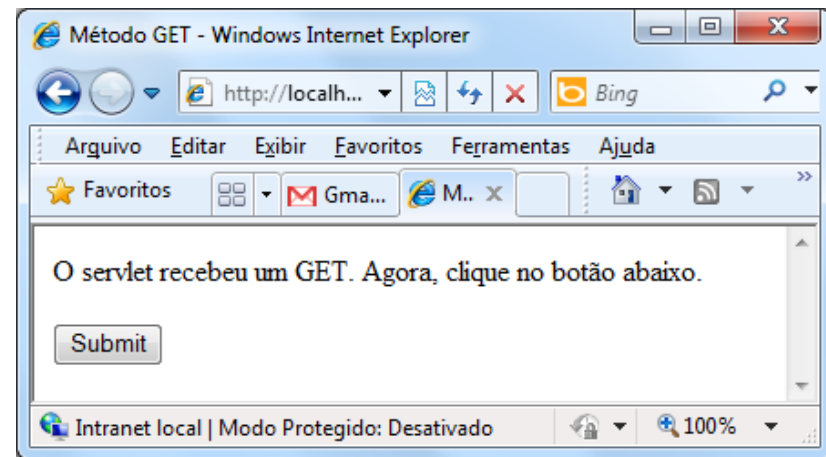
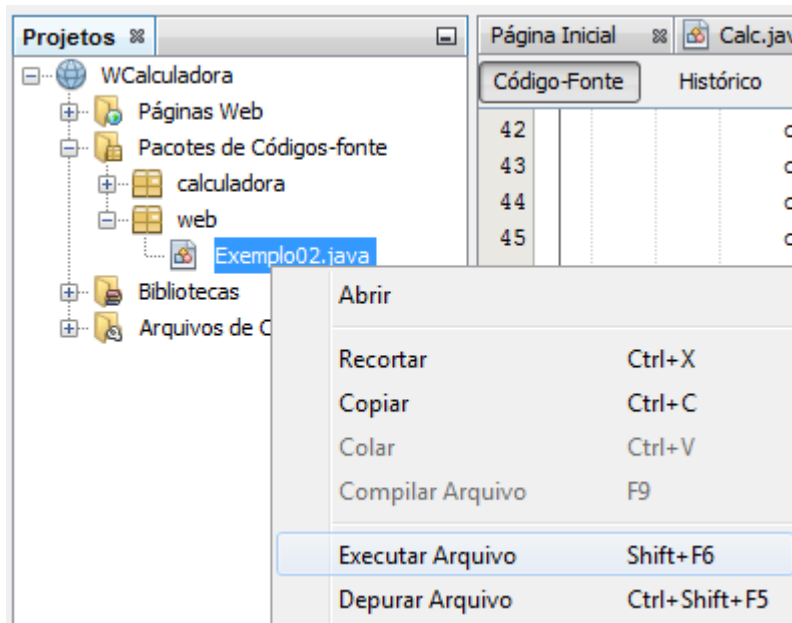
@Override

```
protected void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException {
    try (PrintWriter out = response.getWriter()) {
        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<TITLE>Método GET</TITLE>");
        out.println("</HEAD>");
        out.println("<BODY>");
        out.println("O servlet recebeu um GET. " +
            "Agora, clique no botão abaixo.");
        out.println("<BR>");
        out.println("<FORM METHOD=POST>");
        out.println("<INPUT TYPE=SUBMIT VALUE=Submit>");
        out.println("</FORM>");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```

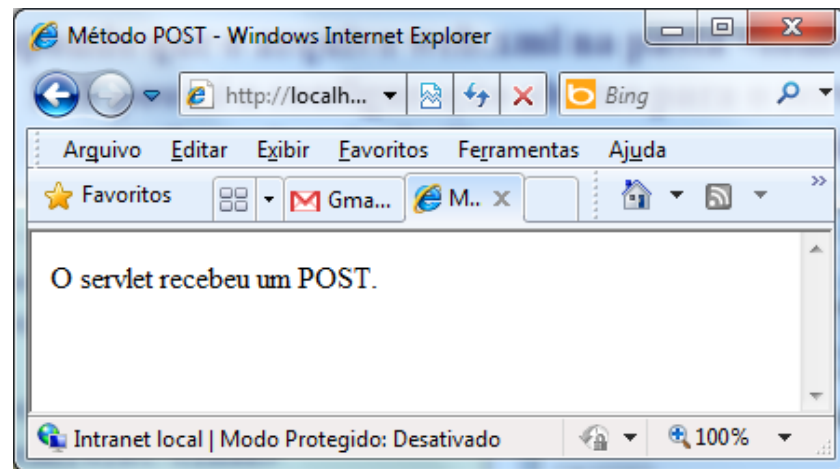
@Override

```
protected void doPost(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException {
    try (PrintWriter out = response.getWriter()) {
        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<TITLE>Método POST</TITLE>");
        out.println("</HEAD>");
        out.println("<BODY>");
        out.println("O servlet recebeu um POST.");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```


Exemplo



GET



POST

Solicitando Redirecionamento

O jeito mais fácil é modificar a **response**

Na resposta, vamos dizer ao navegador que ele deve ir para outra página

Isso é feito com o seguinte comando:

```
response.sendRedirect("http://www.endereco.com/");
```

SendRedirect para Rejeitar GET

Observe o novo método **doGet**

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.sendRedirect("http://www.uol.com.br/");
}

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

Se for tentado um acesso por GET, o navegador redireciona para a página da UOL

Encaminhando uma Requisição

- Pode ser que, por alguma razão, seu **servlet** detecte que aquela requisição não pode ser processada por ele



- Neste caso, se ele souber qual **servlet** é responsável pela execução, ele pode **encaminhar a requisição**

Encaminhando uma Requisição

- Para isso, é preciso arrumar um “entregador”... Quem sabe dele é a própria **request**

Nome do Servlet Destino!

```
request.getRequestDispatcher("/Servlet");
```

- Isso **não faz redirecionamento**, apenas devolve um objeto **RequestDispatcher** (ou, em bom português, um **entregador de requisições**)



Encaminhando uma Requisição

Assim, devemos guardar esse objeto em uma variável

RequestDispatcher rd =

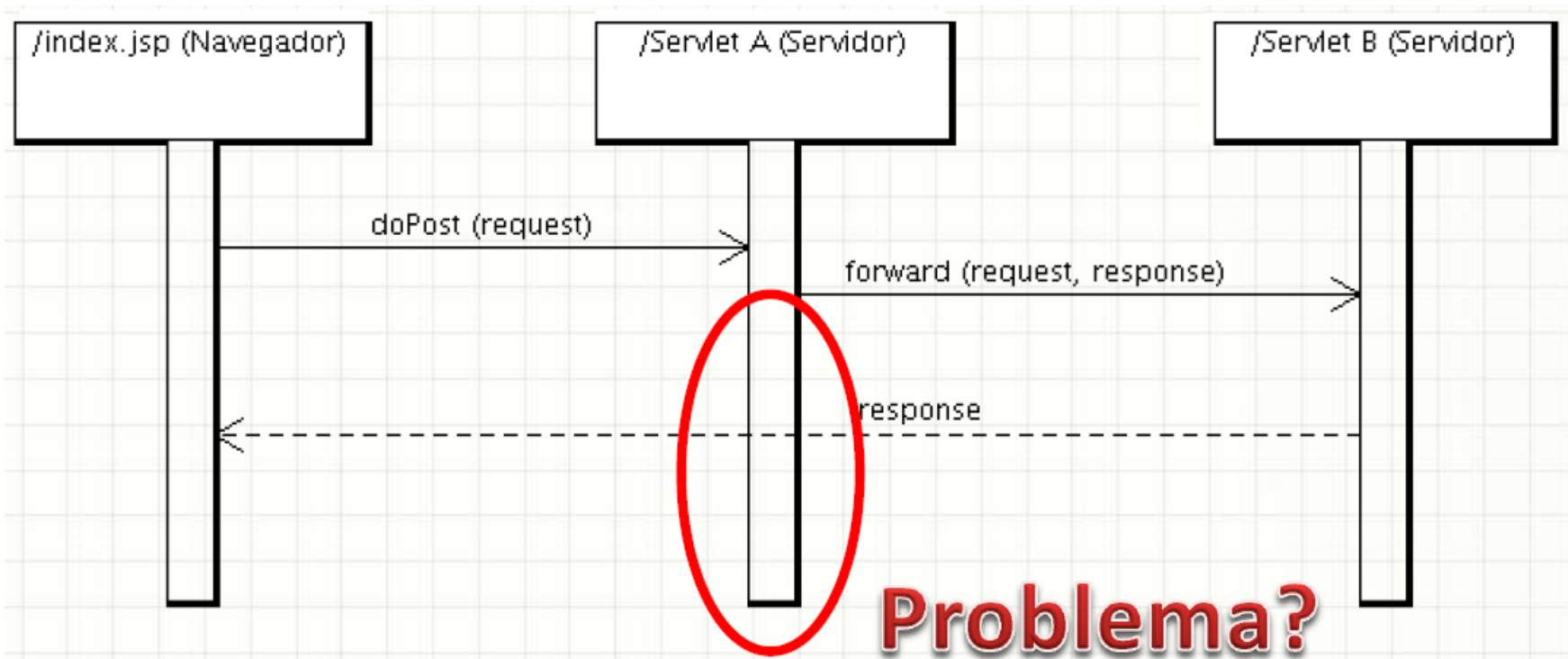
request.getRequestDispatcher("/Servlet");

E, para redirecionar, damos o comando **forward()** para o entregador:

rd.forward(request, response);

Encaminhando uma Requisição

Diagrama de Sequência (forward)



Execução do Servlet A continua!

Encaminhando uma Requisição

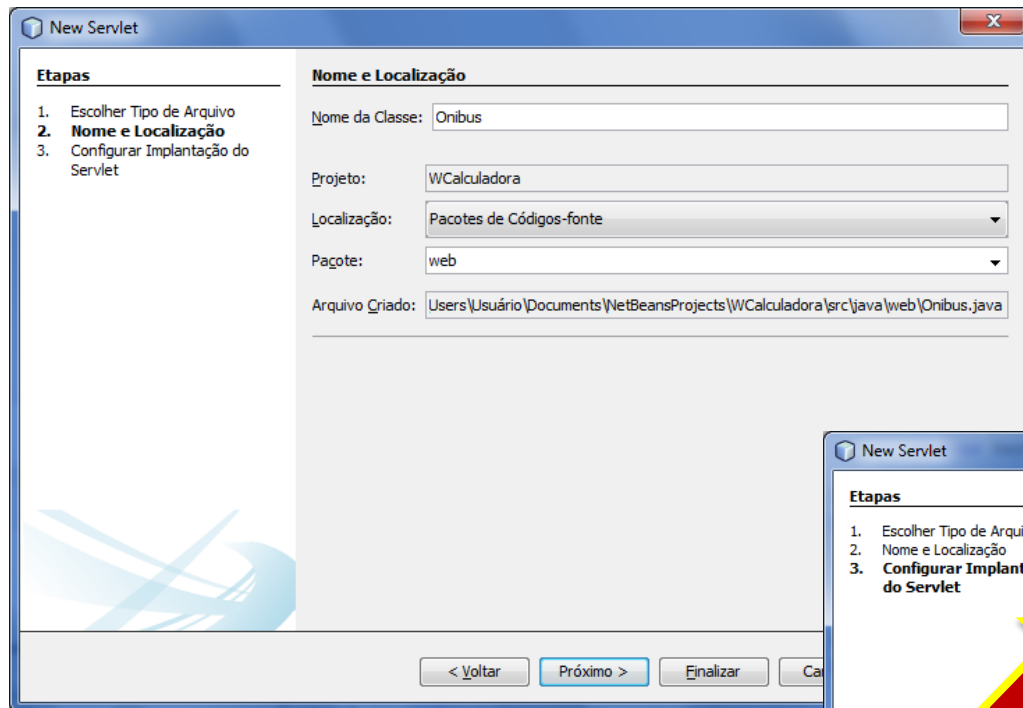
Como resolver?

Usando um **return** após o forward!

Assim, a sequência de despacho fica...

```
RequestDispatcher rd =  
request.getRequestDispatcher("/Servlet");  
rd.forward(request, response);  
return;
```


No pacote web, crie uma servlet com nome Onibus



New Servlet

Etapas

1. Escolher Tipo de Arquivo
2. **Nome e Localização**
3. Configurar Implantação do Servlet

Nome e Localização

Nome da Classe:

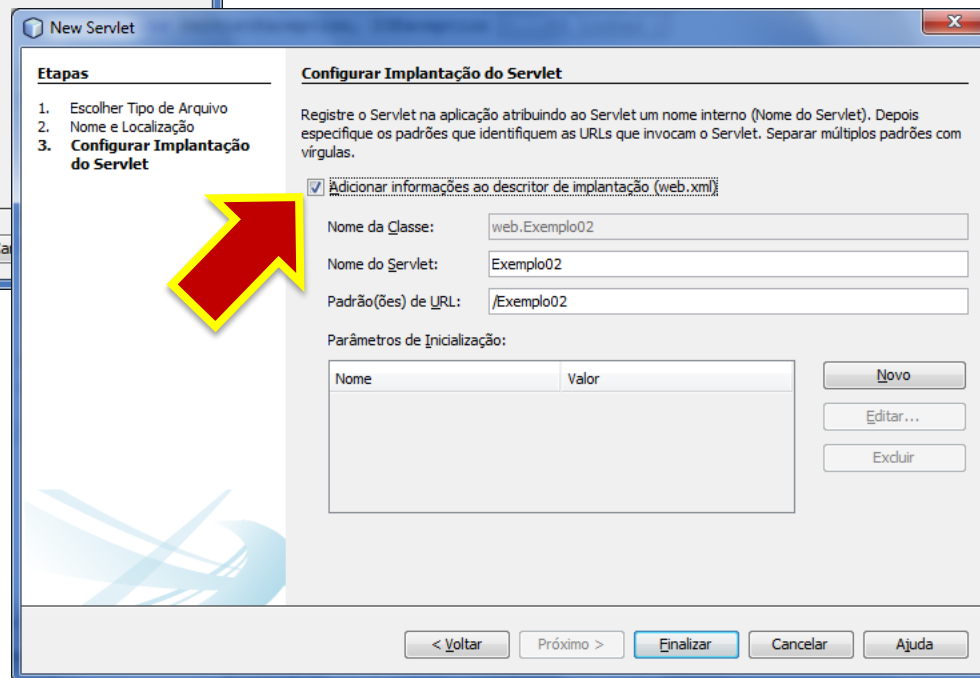
Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > Finalizar Cancelar



New Servlet

Etapas

1. Escolher Tipo de Arquivo
2. Nome e Localização
3. **Configurar Implantação do Servlet**

Configurar Implantação do Servlet

Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.

☒ Adicionar informações ao descritor de implantação (web.xml)

Nome da Classe:

Nome do Servlet:

Padrão(ões) de URL:

Parâmetros de Inicialização:

Nome	Valor
------	-------

Novo Editar... Excluir

< Voltar Próximo > Finalizar Cancelar Ajuda

No corpo da classe iremos criar uma variável do tipo Array de Strings com nome “cidades” contendo os valores abaixo:

```
public class Onibus extends HttpServlet {  
    private String cidades[]={ "Araraquara","Bertioga","Caraguatatuba","Natal","Belém","Santarém","Belo Horizonte","Óbidos"};
```

```
public class Onibus extends HttpServlet {  
  
    private String cidades[]={ "Araraquara","Bertioga","Caraguatatuba","Natal",  
                                "Belém","Santarém","Belo Horizonte","Óbidos"};
```

Acrescente o import abaixo na classe

```
import javax.servlet.RequestDispatcher;
```

@Override

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=rs.getWriter();
    out.println("<FORM METHOD=POST ACTION='Onibus'>");

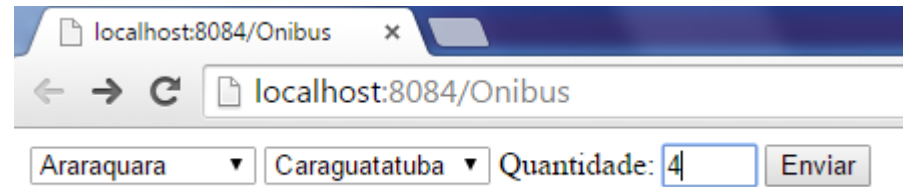
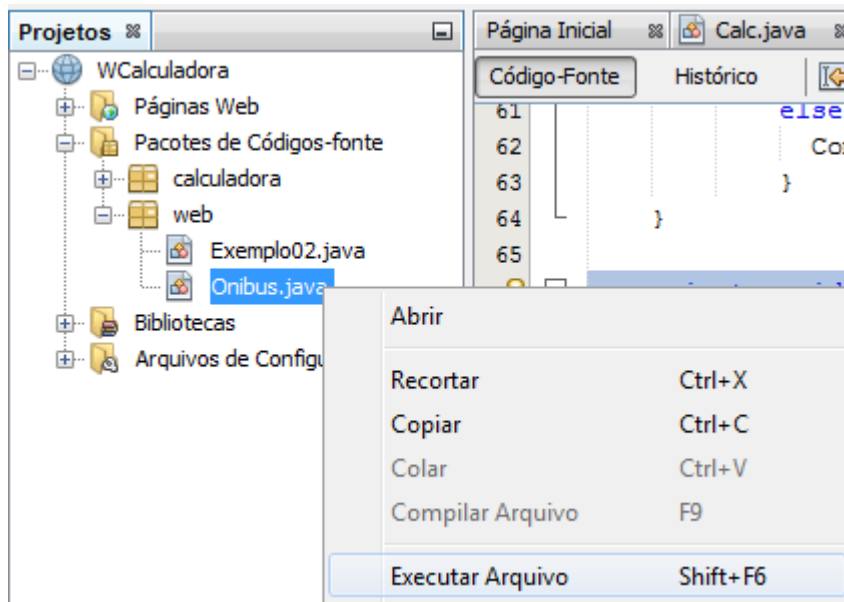
    for (int j=0;j<2;j++)
    {
        out.println("<SELECT NAME='OPCOES'+j+'>");
        for (int i=0;i<idades.length;i++)
            out.println("<OPTION VALUE="+i+">"+idades[i]+"</OPTION>");
        out.println("</SELECT>");
    }
    out.println("Quantidade: <INPUT TYPE='TEXT' SIZE=2 NAME='QTD'>");
    out.println("<INPUT TYPE='SUBMIT'>");
}
```

@Override

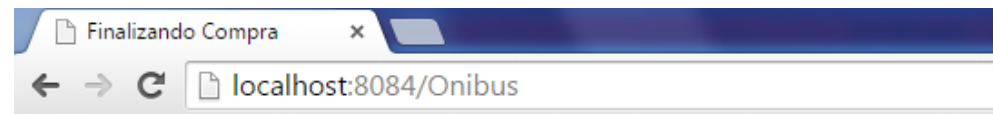
```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String origem = request.getParameter("OPCOES0");
    String destino = request.getParameter("OPCOES1");
    if (origem.equals(destino))
    {
        RequestDispatcher rd = request.getRequestDispatcher("OnibusErro");
        rd.forward(request, response);
        return;
    }
    else {
        Comprar(request, response);
    }
}
```

```
private void Comprar(HttpServletRequest rq, HttpServletResponse rs) throws ServletException, IOException {  
    String origem = rq.getParameter("OPCOES0");  
    String destino = rq.getParameter("OPCOES1");  
    rs.setContentType("text/html");  
    PrintWriter out = rs.getWriter();  
    out.println("<HTML>");  
    out.println("<HEAD>");  
    out.println("<TITLE>Finalizando Compra</TITLE>");  
    out.println("</HEAD>");  
    out.println("<BODY>");  
  
    out.println("<BR>");  
    out.println("<BR>Você comprou " + rq.getParameter("QTD") + " Passagens de: " +  
        cidades[Integer.parseInt(origem)] + ", com destino a: " + cidades[Integer.parseInt(destino)]);  
    out.println("<input type='button' name='btnvoltar' value='Voltar' onClick='javascript:history.back(1)'>");  
    out.println("</BODY>");  
    out.println("</HTML>");  
}
```

Exemplo



GET

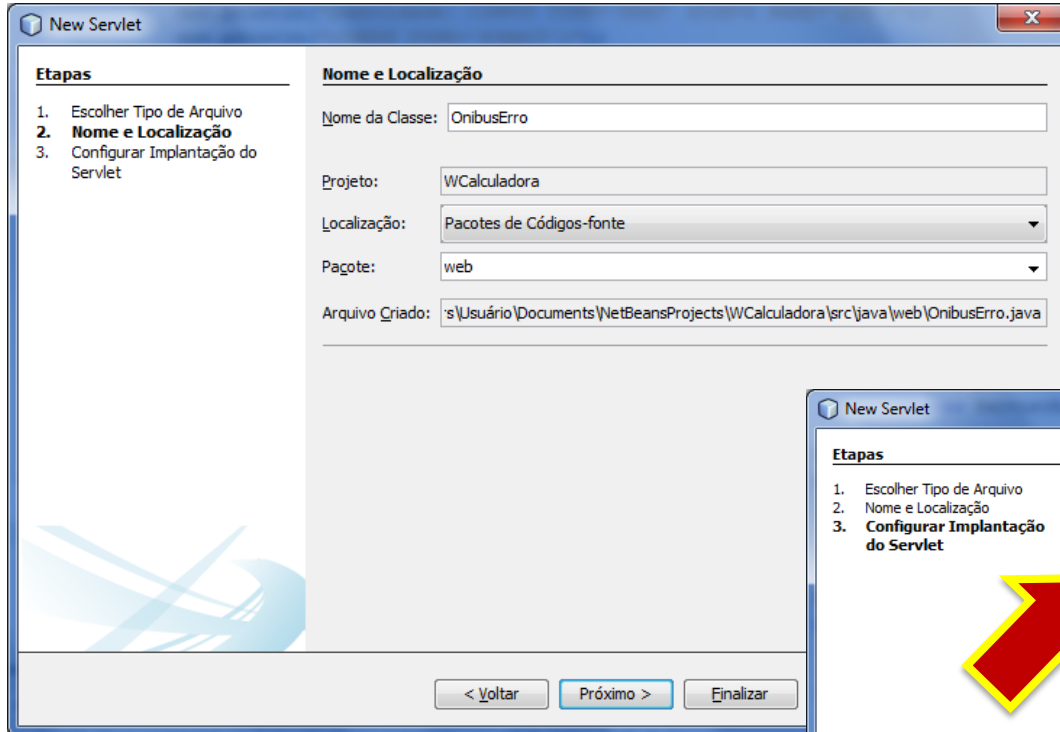


Você comprou 4 Passagens de: Araraquara, com destino a: Caraguatatuba [Voltar](#)

POST

Completando o exemplo:

No pacote web, crie uma servlet com nome OnibusErro



New Servlet

Etapas

1. Escolher Tipo de Arquivo
- 2. Nome e Localização**
3. Configurar Implantação do Servlet

Nome e Localização

Nome da Classe:

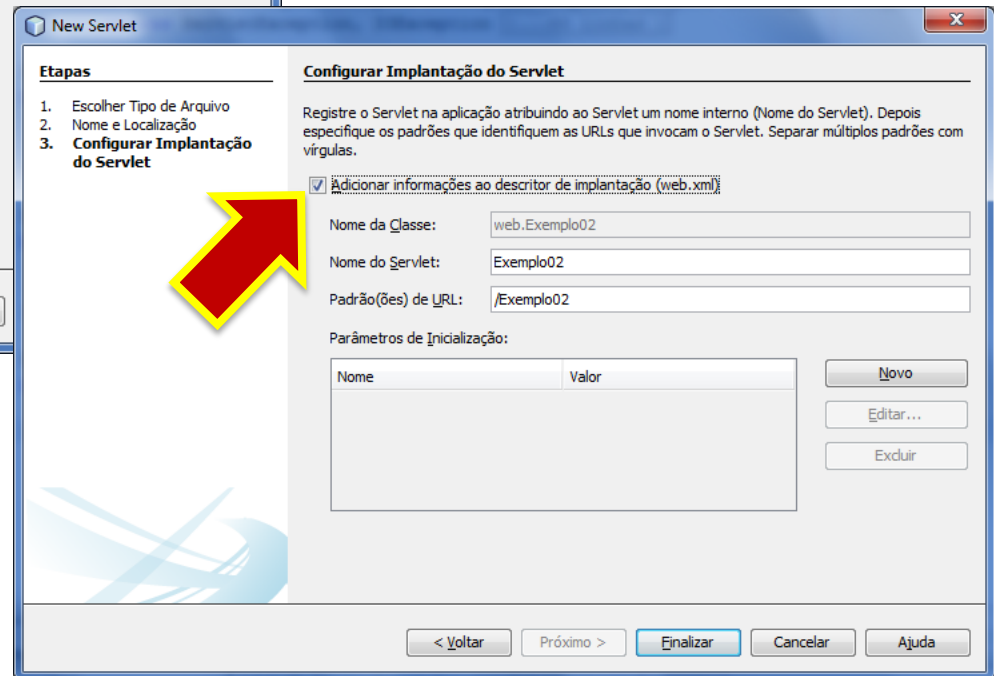
Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > Finalizar



New Servlet

Etapas

1. Escolher Tipo de Arquivo
2. Nome e Localização
- 3. Configurar Implantação do Servlet**

Configurar Implantação do Servlet

Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.

☒ Adicionar informações ao descritor de implantação (web.xml)

Nome da Classe:

Nome do Servlet:

Padrão(ões) de URL:

Parâmetros de Inicialização:

Nome	Valor
------	-------

Novo Editar... Excluir

< Voltar Próximo > Finalizar Cancelar Ajuda

No corpo da classe iremos criar uma variável do tipo Array de Strings com nome “cidades” contendo os valores abaixo:

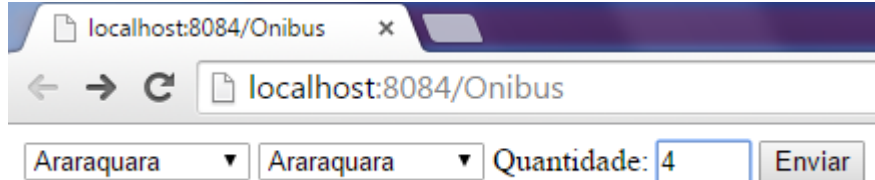
```
public class OnibusErro extends HttpServlet {  
    private String cidades[]={ "Araraquara","Bertioga","Caraguatatuba","Natal","Belém","Santarém","Belo Horizonte","Óbidos"};
```

```
public class OnibusErro extends HttpServlet {  
    private String cidades[]={"Araraquara","Bertioga","Caraguatatuba","Natal",  
                                "Belém","Santarém","Belo Horizonte","Óbidos"};
```


@Override

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String origem = request.getParameter("OPCOES0");
    String destino = request.getParameter("OPCOES1");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<TITLE>Pagina de erro</TITLE>");
    out.println("</HEAD>");
    out.println("<BODY>");
    out.println("<H1 style='font-face:arial; color:blue'>As cidades de origem " +
        cidades[Integer.parseInt(origem)] + " e destino nao podem ser iguais!!!!</H1>");
    out.println("</BODY>");
    out.println("</HTML>");
}
```

Agora execute com a cidade de Origem igual a de Destino



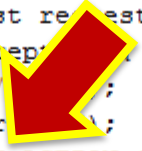
```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String origem = request.getParameter("OPCOES0");
    String destino = request.getParameter("OPCOES1");
    if (origem.equals(destino))
    {
        RequestDispatcher rd = request.getRequestDispatcher("OnibusErro");
        rd.forward(request, response);
        return;
    }
    else {
        Comprar(request, response);
    }
}
```

Perceba que método doPost da classe Onibus isso é verificado e caso seja é redirecionado para a classe OnibusErro que acabamos de criar.

E como o método é Post....

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    out.println("<FORM METHOD=POST ACTION='Onibus'>");

    for (int j=0;j<2;j++)
    {
        out.println("<SELECT NAME='OPCOES"+j+"'>");
        for (int i=0;i<idades.length;i++)
            out.println("<OPTION VALUE="+i+">"+idades[i]+"</OPTION>");
        out.println("</SELECT>");
    }
    out.println("Quantidade: <INPUT TYPE='TEXT' SIZE=2 NAME='QTD'>");
    out.println("<INPUT TYPE='SUBMIT'>");
}
```



O método doPost da classe OnibusErro é disparado executando a mensagem abaixo.



As cidades de origem Araraquara e destino nao podem ser iguais!!!!

Encaminhando uma Requisição

Encaminhar requisições pode ser útil...

Mas se pudéssemos acrescentar informações na requisição...

Seria bem mais útil, não?

Guardando Coisas na Requisição

Vimos como ler **parâmetros** de uma requisição

– **request.getParameter(“nome”);**

Parâmetros são dados armazenados pelo **navegador** na requisição

Não podemos inserir parâmetros através da **servlet**

Guardando Coisas na Requisição

Quando nosso programa acrescenta dados na requisição, dá-se o nome de **atributo**

– **request.setAttribute(“nome”,objeto)**

Para ler esses valores, usamos...

– **request.getAttribute(“nome”);**

Guardando Coisas na Requisição



Guardando Coisas na Requisição



Guardando Coisas na Requisição



Guardando Coisas na Requisição



Por que isso é útil?

Por que é útil guardar coisas na requisição?

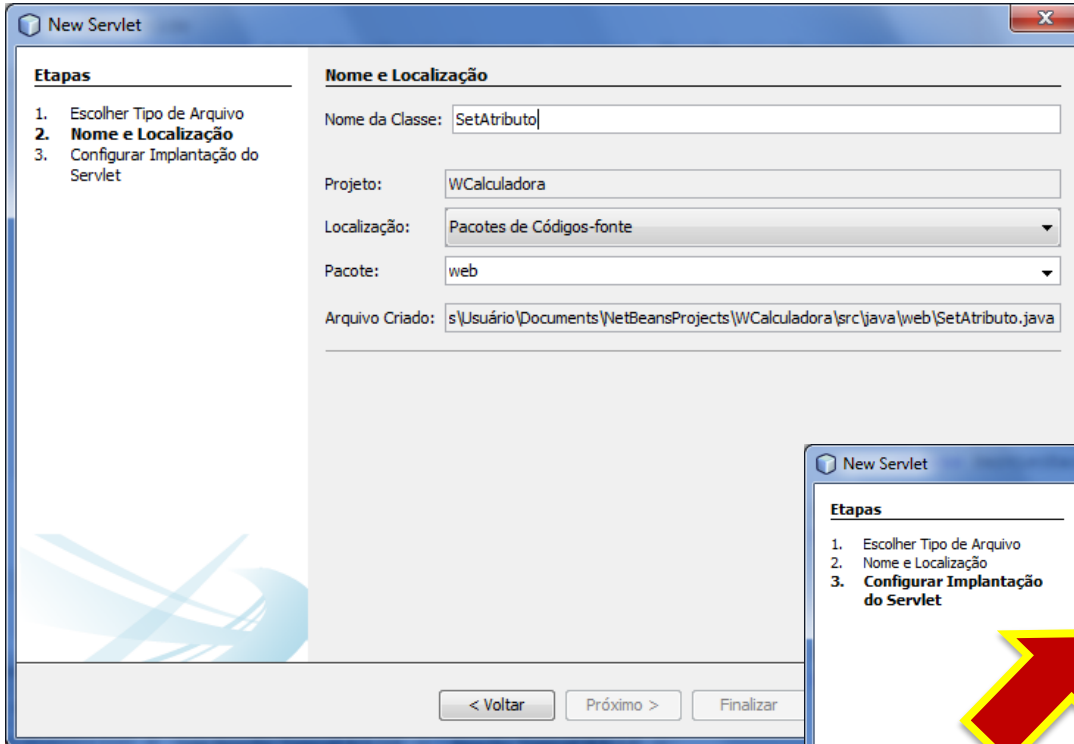
Que tal o nosso aplicativo **Calculadora** ser composto por dois **servlets**?

- Calc → Faz os cálculos
- CalcView → Apresenta os resultados

Por hoje: isso facilita o reuso de código, por separar processamento de apresentação

Posteriormente veremos mais razões para isso ser útil...

No pacote web, crie uma servlet com nome SetAtributo



New Servlet

Etapas

1. Escolher Tipo de Arquivo
- 2. Nome e Localização**
3. Configurar Implantação do Servlet

Nome e Localização

Nome da Classe:

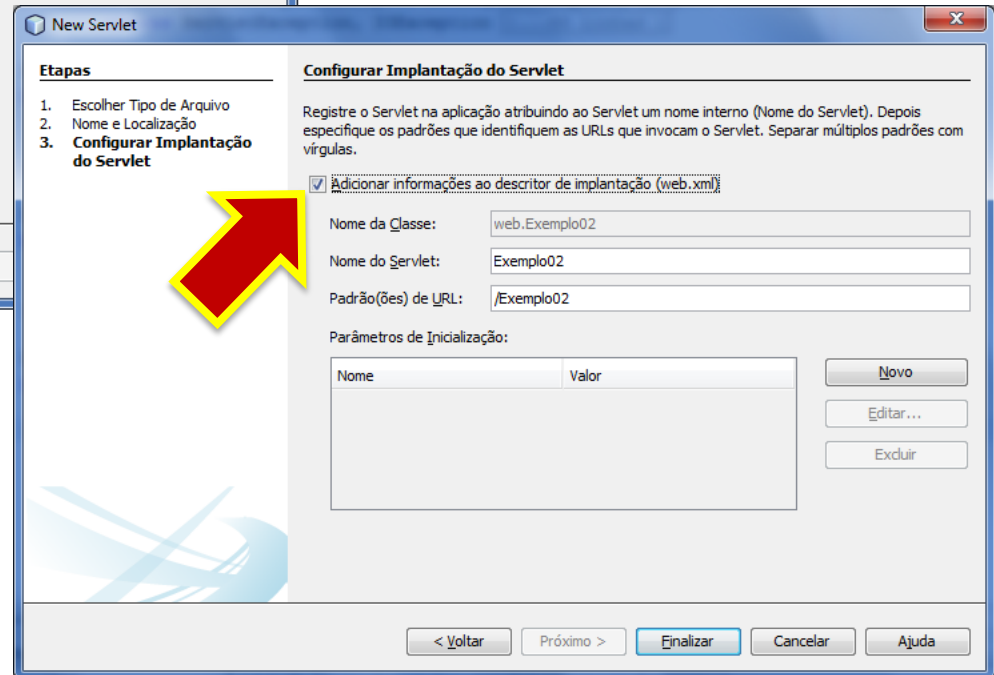
Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > Finalizar



New Servlet

Etapas

1. Escolher Tipo de Arquivo
2. Nome e Localização
- 3. Configurar Implantação do Servlet**

Configurar Implantação do Servlet

Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.

☒ Adicionar informações ao descritor de implantação (web.xml)

Nome da Classe:

Nome do Servlet:

Padrão(ões) de URL:

Parâmetros de Inicialização:

Nome	Valor
------	-------

< Voltar Próximo > **Finalizar** Cancelar Ajuda

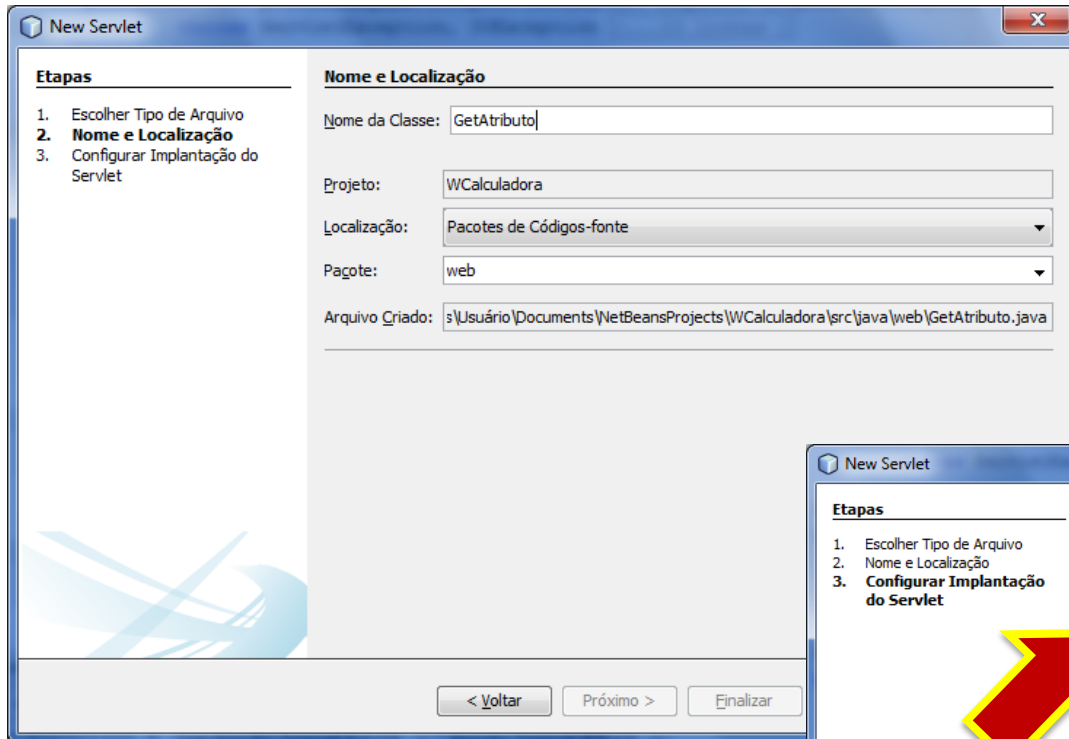
@Override

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setAttribute("usuario", "Usilva");
    request.setAttribute("senha", "balacobaco");

    RequestDispatcher rd=request.getRequestDispatcher("GetAtributo");
    rd.forward(request, response);

}
```


No pacote web, crie uma servlet com nome GetAtributo



New Servlet

Etapas

1. Escolher Tipo de Arquivo
- 2. Nome e Localização**
3. Configurar Implantação do Servlet

Nome e Localização

Nome da Classe:

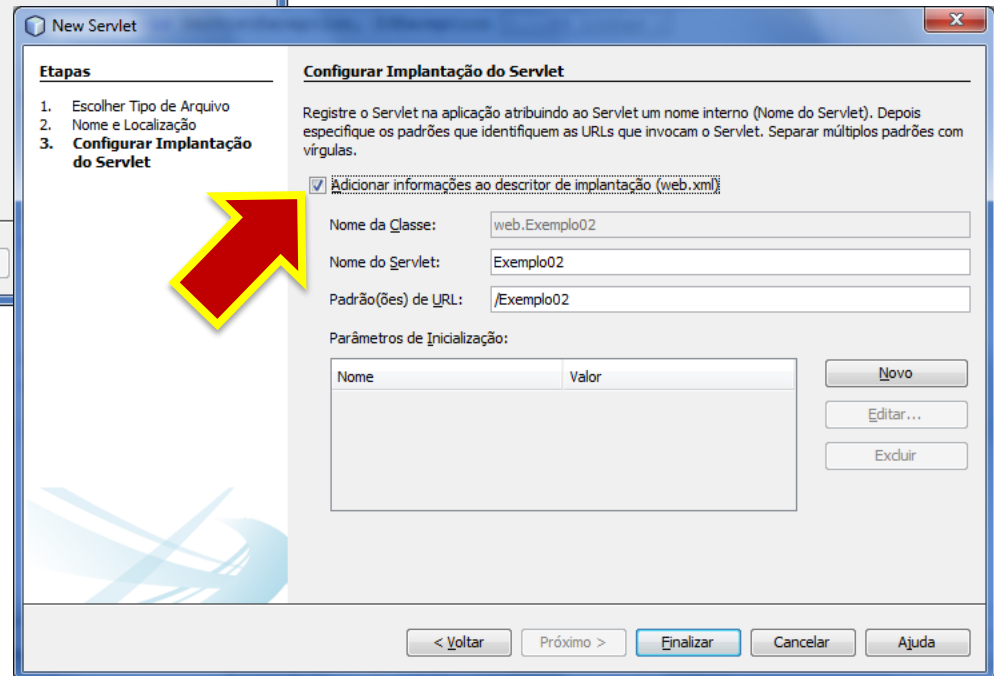
Projeto:

Localização:

Pacote:

Arquivo Criado:

< Voltar Próximo > Finalizar



New Servlet

Etapas

1. Escolher Tipo de Arquivo
2. Nome e Localização
- 3. Configurar Implantação do Servlet**

Configurar Implantação do Servlet

Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.

☒ Adicionar informações ao descritor de implantação (web.xml)

Nome da Classe:

Nome do Servlet:

Padrão(ões) de URL:

Parâmetros de Inicialização:

Nome	Valor
------	-------

< Voltar Próximo > Finalizar Cancelar Ajuda

@Override

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>Compartilha Dados</title></head><body>");

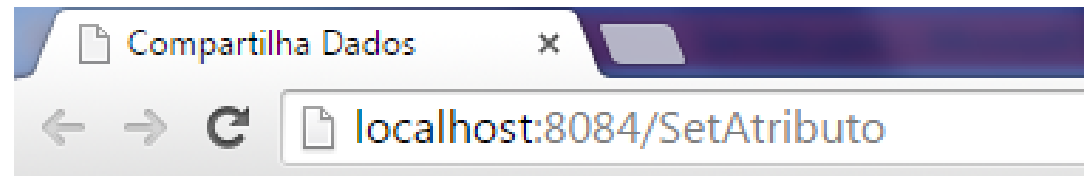
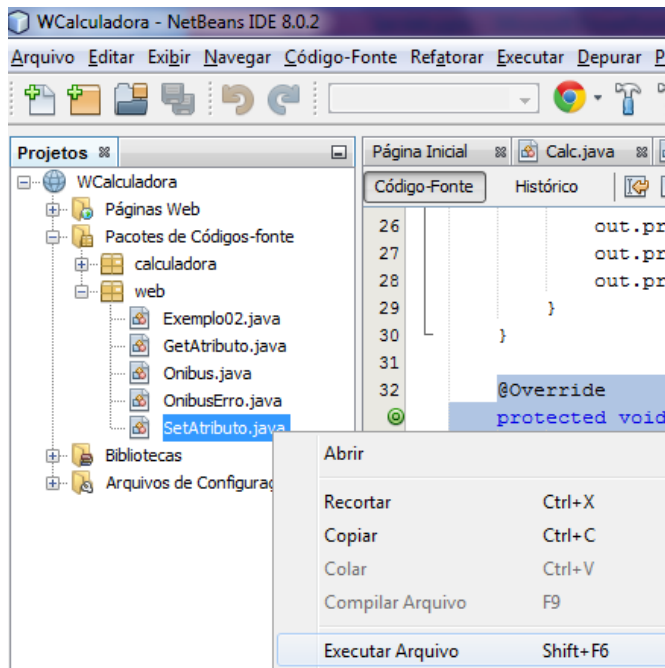
    out.println ("GetAtributo (Usuário): " + request.getAttribute("usuario")+ "<br>");
    out.println ("GetAtributo (Senha): " + request.getAttribute("senha")+ "<br>");

    request.setAttribute ("senha", "SenhaMudada");

    out.println ("<br>");
    out.println ("<br>");
    out.println ("GetAtributo (Usuário): " + request.getAttribute("usuario")+ "<br>");
    out.println ("GetAtributo (Senha): " + request.getAttribute("senha")+ "<br>");

    out.println("</body></html>");
}
```

Exemplo



GetAtributo (Usuário): Usilva
GetAtributo (Senha): balacobaco

GetAtributo (Usuário): Usilva
GetAtributo (Senha): SenhaMudada

Acompanhe o professor....

Exemplo Passagens

Conclusão

Há dois tipos de requisições: POST e GET

- Servlets podem diferenciar essas requisições
- Servlet pode redirecionar o navegador
- Servlet pode encaminhar requisição para outro
- Servlet pode acrescentar dados na requisição

Elaborar um formulário compra.html para compra de um veículos on-line, contendo as seguintes informações:

Nome:

CPF:

Endereço:

Telefone:

Itens da compra:

Um option para escolha dos modelos

↗ Ex - R\$ 10.000,00

↗ ELX - R\$ 20.000,00

↗ Super - R\$ 30.000,00

Um option para escolha das Portas

↗ 2 p

↗ 4 p – (+ R\$ 4.000,00)

Um Check para escolha dos acessórios

↗ Ar condicionado – (+ R\$ 3.000,00)

↗ Direção – (+ R\$ 2.000,00)

↗ VE/TE – (+ R\$ 1.000,00)

Exercício Carro

Nome:

CPF:

Endereço:

Telefone:

Modelos

☒ EX ☐ ELX ☐ Master

Portas

☐ 2 Portas ☐ 4 Portas

Acessórios

☐ Ar Condicionado ☐ Direção ☐ VE / TE

Ao clicar no botão Calcular, deverá chamar a Servlet Finalizar, e calcular o valor da compra conforme os critérios estabelecidos e exibir um formulário html

OBS.: Crie um objeto Pedido, com os dados do cliente, Modelo, Portas e Acessórios comprados.

Complicado fazer layout/html com Servlet!

- Será que não tem um jeito mais simples?
- Ah! Existem as Java Server Pages (JSPs!) (Próxima aula)