

# Desenvolvimento Web I

## Aula 03 - Manipulando requisições e respostas HTML com Servlets

# Apresentação

---

Olá! Continuamos agora o assunto da aula passada, com os chamados Servlets. Você já aprendeu que os Servlets são criados para processar as requisições web, gerando as páginas HTML de resposta que serão mostradas no navegador web do usuário. Tudo bem até aí?

Muito bem, continuaremos o assunto de Servlets agora apresentando mais detalhes sobre como processar as requisições dos clientes e sobre como identificá-los ao longo do tempo. Por exemplo, você já deve ter preenchido formulários na web, correto? Diversas são as situações em que precisamos fazer isso: quando cadastramos uma nova conta de e-mail, informando dados pessoais e e-mail desejado; quando nos autenticamos (logamos) em sistemas web, acessando nossa conta de e-mail ou conta bancária. Pois bem, esses sistemas web precisam receber e processar os dados digitados. Nesta aula, vamos ver como fazer isso com Servlets.

Além disso, um cadastro pode ser feito em mais de uma tela. Imagine um sistema em que na primeira tela seja necessário entrar com dados pessoais, clicar em um botão de nome “Próximo” e depois, em uma segunda tela, digitar os seus dados profissionais para só então clicar em um botão “Confirmar”. É bem provável que o sistema deva salvar os dados coletados apenas quando ambas as telas forem preenchidas. Dessa forma, como guardar a informação digitada na primeira tela enquanto o usuário digita informações na segunda tela? Também veremos uma possível solução para isso.



**Vídeo 01** - Apresentação

## Objetivos

- Processar parâmetros das requisições web.
- Repassar parâmetros de uma requisição para outra.

# Manipulando os parâmetros das requisições web

---

Na aula anterior, você percebeu que os Servlets são acessados através de suas URL (ex.: <http://localhost:8080/ProgramacaoWeb/ServletExemplo>). Esse acesso pode ser via método GET ou via método POST. Ou seja, se você apenas digitar uma URL em um navegador web ou clicar em um link usual numa página, estará usando o método GET. Já para se usar o método POST, precisamos definir e usar um formulário, ou seja, usar o marcador **<form>** na página HTML com atributo **method** configurado para POST. Daí, podemos acioná-lo, por exemplo, através do clique de um botão para envio dos dados. A escolha entre esses dois métodos é feita geralmente baseada na quantidade de informações (parâmetros da requisição) que você quer passar.

Por exemplo, para entrar na página da UFRN, você não precisa entrar com nenhuma informação além da URL do sítio (<http://www.ufrn.br>). Nesses casos, geralmente você faz uso do método GET, apesar de o método POST poder ser utilizado sem problemas.

Por outro lado, quando você faz uma consulta em um sítio de busca como o Google, precisa entrar com informações como as palavras que você quer buscar. Para isso, essa informação precisa ser transportada para o servidor web, o qual é responsável por encontrar as páginas que contenham aquelas palavras que você digitou na busca. Se fizermos uso do método GET, os parâmetros da requisição (palavras de busca etc.) serão transportados na própria URL de acesso ao Servlet. Por exemplo, a URL <http://localhost:8080/ProgramacaoWeb/ServletBusca?palavras=servlet%20java> representa o acesso ao ServletBusca passando o parâmetro de nome palavras, cujo valor é “servlet java”. Note que espaços em branco são substituídos automaticamente pelo código %20. Experimente fazer uma consulta em um sítio de busca e olhar a URL de resposta de uma consulta.

Já se você fizer uso do método POST, a URL de acesso ao Servlet será simplesmente <http://localhost:8080/ProgramacaoWeb/ServletBusca>, e seus parâmetros serão passados de forma oculta para o usuário do navegador web. Isso tem algumas consequências importantes. Primeiro, se você estiver transmitindo uma senha, não vai querer que ela fique trafegando na própria URL, pois URL são

armazenadas em históricos de acesso dos navegadores e dos servidores web. Além disso, existe um limite não muito grande para o tamanho de uma URL, que pode variar de acordo com o navegador e servidor web utilizados. Dessa forma, o método GET só deve ser utilizado quando a quantidade de informações for pequena e não for secreta.

Ok, mas apesar de os parâmetros de uma requisição serem trafegados via métodos HTTP diferentes, GET ou POST, o acesso a estes é praticamente o mesmo no código Java. De fato, a principal diferença no código Java você já aprendeu na aula passada. Requisições GET são processadas pelo método `doGet()`, e requisições POST são processadas pelo método `doPost()`. Dentro desses métodos, o acesso aos parâmetros tende a ser o mesmo: através do uso da classe `javax.servlet.http.HttpServletRequest`.

Ambos os métodos `doGet()` e `doPost()` possuem dois parâmetros. O primeiro deles, do tipo `HttpServletRequest`, irá lhe dar acesso aos parâmetros e a outras informações da requisição. Lembre-se de que o método será chamado de acordo com o evento acionado pelo usuário (clique em um link, digitação da URL no navegador, clique em um botão associado a um formulário etc.).

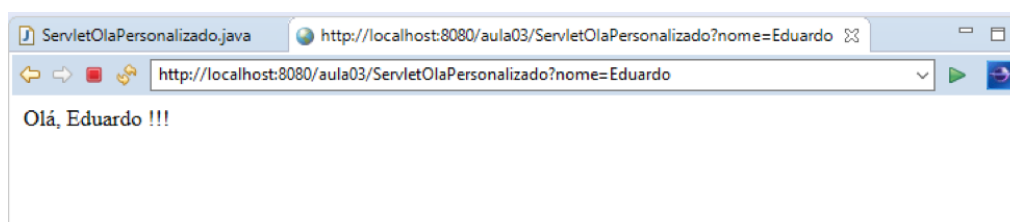
# Manipulando os parâmetros das requisições web II

Analise agora o código a seguir, cuja tela de resposta é mostrada na Figura 1.

```
1 package aula03;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet("/ServletOlaPersonalizado")
13 public class ServletOlaPersonalizado extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15
16     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
17         PrintWriter resposta = response.getWriter();
18         resposta.write("<html><body>");
19         resposta.write("Olá, " + request.getParameter("nome") + " !!!");
20         resposta.write("</html></body>");
21     }
22 }
```

**Listagem 1** - Acessando parâmetros de requisição em um Servlet

**Figura 01** - Resposta do ServletOlaPersonalizado



Como você pode perceber, a classe **ServletOlaPersonalizado** é um Servlet que implementa o método **doGet()**, ou seja, responde a requisições do tipo HTTP via método GET. A resposta desse Servlet irá depender da URL de invocação do Servlet, ou melhor, do valor do parâmetro nome que é passado através da URL. Por exemplo, a página de resposta desse Servlet, quando executado através da URL <http://localhost:8080/aula03/ServletOlaPersonalizado?nome=Eduardo>, é mostrada na Figura 1.

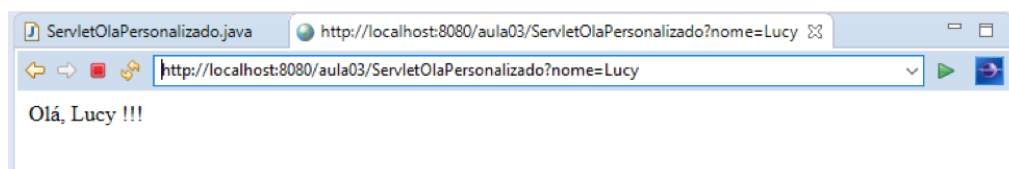
Caso você mude o valor do parâmetro para Lucy, teremos o resultado mostrado na Figura 2. Esse comportamento dinâmico, que muda de acordo com o valor do parâmetro nome, é dado pelo comando **request.getParameter("nome")** (linha 17 da Listagem 1). Esse comando aciona o método **getParameter()** da classe **HttpServletRequest**, o qual recebe como parâmetro uma String que representa o nome do parâmetro da requisição web cujo valor você deseja ter acesso. No caso, estamos interessados no valor do parâmetro identificado pela String "nome".



## Vídeo 02 - Primeiro Servlet

Observe a primeira URL de acesso ao Servlet (<http://localhost:8080/aula03/ServletOlaPersonalizado?nome=Eduardo>) e note que está indicado que o valor do parâmetro nome é igual ao texto Eduardo, justamente o texto que aparece na página HTML da Figura 1. Ao trocar o valor desse parâmetro para Lucy, o valor retornado pelo comando `request.getParameter("nome")` será o valor "Lucy" e não mais "Eduardo", fazendo com que a página resultante seja igual à da Figura 2. Fácil de entender, não é?

**Figura 02** - Resposta do ServletOlaPersonalizado ao mudar valor do parâmetro



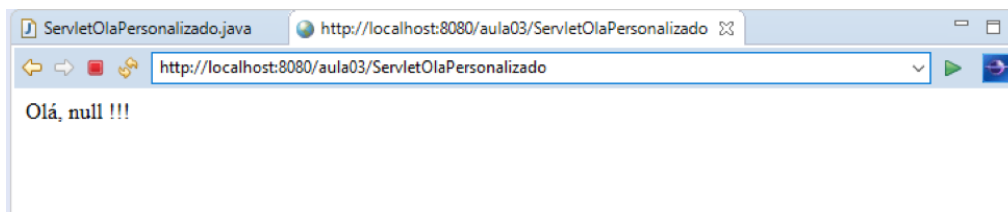
Ao trabalharmos com parâmetros nas requisições, temos que ter em mente que nem sempre os valores passados estão de acordo com o esperado. Considere, por exemplo, o caso das requisições tratadas pelo ServletOlaPersonalizado. Se o usuário digitar diretamente a URL no browser, ele pode:

1. esquecer de colocar na URL o valor do parâmetro "nome";
2. digitar errado o nome do parâmetro, como no caso da URL <http://localhost:8080/aula03/ServletOlaPersonalizado?noome=Eduardo>.

Observe que o nome do parâmetro passado pelo usuário é agora “noome” e não mais “nome”, como esperado pelo Servlet.

Nesses casos, o que o comando `request.getParameter("nome")` irá retornar? Basicamente o valor **null**. Sempre que o valor de um parâmetro não for encontrado ou o próprio parâmetro não existir, o valor padrão null será retornado. Veja na Figura 3 o resultado quando nos esquecemos de passar o valor do parâmetro (note que o valor null, quando convertido para String, é igual ao texto “null”).

**Figura 03** - Esquecendo-se de passar o valor do parâmetro nome



Além disso, lembre-se de que existem pessoas na internet com intenções de “invadir e quebrar” os sistemas. Elas podem propositalmente deixar de passar parâmetros para o sistema ou então passar valores inválidos, buscando derrubar o sistema ou fazer com que ele tenha um comportamento inapropriado.

Quer ver um exemplo? Uma pessoa mal-intencionada pode criar uma página web com o link “clique aqui para ir para a empresa X”, onde X é a sua empresa e cujo link vai para o ServletOlaPersonalizado que você criou e hospedou no seu site empresarial. Basta que a URL do link criado contenha uma palavra inadequada para o parâmetro nome (ex: [http://localhost:8080/aula03/ServletOlaPersonalizado?noome=@\\_@\\_@\\$](http://localhost:8080/aula03/ServletOlaPersonalizado?noome=@_@_@$)) para se ter um possível estrago.

Ao clicar no link da página web da pessoa mal-intencionada, um possível cliente verá o resultado impresso na tela e provavelmente ficará ofendido. Sem entender de programação, o cliente em potencial achará que a grosseria veio do seu site e não do site da pessoa mal-intencionada, já que a URL que aparecerá no navegador web é a do ServletOlaPersonalizado (seu site!). Então, fique atento!



## Atividade 01

---

1. Implemente e execute o ServletOlaPersonalizado.
2. Crie um novo Servlet que receba dois parâmetros, primeiroNome e ultimoNome, e que faça uma saudação ao usuário imprimindo uma mensagem que contenha o primeiro e o último nome do usuário (parâmetros da requisição).
3. Releia o exemplo da calculadora mostrada na introdução da aula anterior e implemente-o, agora que você já sabe como acessar dados digitados pelo usuário.
4. Faça uma consulta em um sítio de busca e analise a URL de resposta de uma consulta para identificar o método de envio utilizado (GET ou POST). Caso seja GET, identifique o nome do parâmetro que representa as palavras de consulta. Confirme essa informação observando o formulário no código-fonte HTML da página inicial de consulta.
5. Faça uma consulta em um sítio de busca e analise a URL de resposta de uma consulta para identificar o método de envio utilizado (GET ou POST). Caso seja GET, identifique o nome do parâmetro que representa as palavras de consulta. Confirme essa informação observando o formulário no código-fonte HTML da página inicial de consulta.

## Manipulando os parâmetros das requisições web III

---

Além do método `getParameter()`, a classe `HttpServletRequest` possui vários outros métodos que podem ser utilizados para pegar informação sobre parâmetros da requisição web. Um exemplo é o método `getParameterNames()`, o qual você pode usar para pegar os nomes de todos os parâmetros da requisição. Esse método é geralmente útil quando você quer, por exemplo, listar para o usuário todos os parâmetros enviados. Vejamos o exemplo de código mostrado na Listagem 2.

```

1 package aula03;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import java.util.Enumeration;
6
7 import javax.servlet.ServletException;
8 import javax.servlet.annotation.WebServlet;
9 import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12
13 @WebServlet("/ServletListaNomeParametros")
14 public class ServletListaNomeParametros extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
18         PrintWriter resposta = response.getWriter();
19         resposta.write("<html><body>");
20         resposta.write("Olá, os parâmetros recebidos por essa requisição são: <BR>");
21         Enumeration<String> nomesParametros = request.getParameterNames();
22         while (nomesParametros.hasMoreElements()) {
23             resposta.write(nomesParametros.nextElement().toString());
24             resposta.write(", ");
25         }
26         resposta.write("</html></body>");
27     }
28
29     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
30         doGet(request, response);
31     }
32 }

```

**Listagem 2** - Lendo nome de todos os parâmetros da requisição

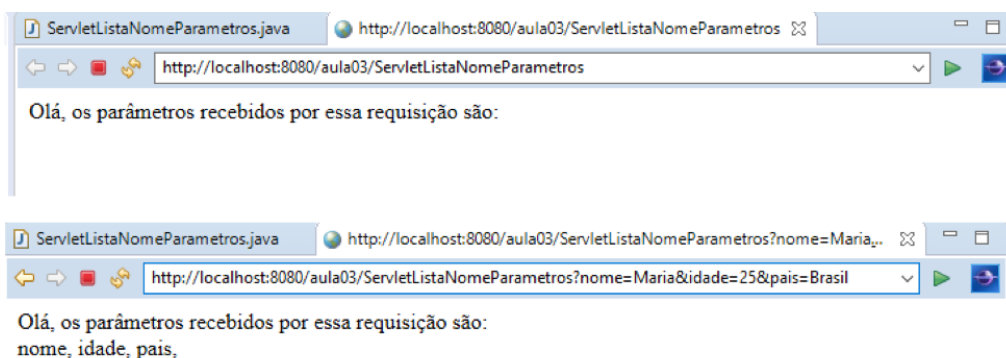
Ao executar o `ServletListarNomeParametros` sem passar nenhum parâmetro, será apresentada a página de resposta mostrada na parte superior da Figura 4. Já ao passar parâmetros para a requisição, como mostrado na parte inferior da Figura 4, os nomes desses parâmetros serão apresentados. Se você observou bem, as linhas 21 a 25 da Listagem 2 são as responsáveis por imprimir essa lista de nomes de parâmetros. O tipo de retorno do método `getParameterNames` é **Enumeration<String>**, uma classe utilizada para acessar os elementos de uma coleção com itens do tipo **String**. No caso, queremos acessar os nomes (elementos) de todos os parâmetros da requisição (coleção).



### Vídeo 03 - Caracteres Especiais

A classe Enumeration possui métodos como `hasMoreElements()`, que retorna um booleano indicando se ainda existem elementos a serem pegos (ver condição do `while` na linha 22), e `nextElement()`, que retorna o próximo elemento da coleção (linha 23). Como Enumeration é uma classe de uso geral, o tipo de retorno do método `nextElement()` é o `Object`. Para montarmos uma String com sua representação de um objeto qualquer, basta usar o método `toString()`, não é? De fato, os nomes dos parâmetros são textos, ou seja, Strings, então uma alternativa ao uso do método `toString()` é o uso do `cast`: (String).

**Figura 04** - Páginas de resposta quando não se passam parâmetros (acima) e quando se passam parâmetros (abaixo) para a requisição



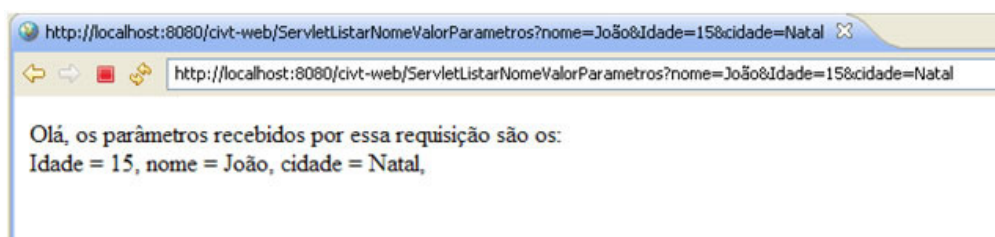
## Atividade 02

1. Sem olhar diretamente o código, implemente e execute o `ServletListarNomeParametros`. Relate se você encontrou problemas para realizar essa tarefa.
2. Altere o código do `ServletListarNomeParametros` para imprimir não só os nomes dos parâmetros, mas também o valor de cada um deles. Por exemplo, ao invés de só dizer que existe um parâmetro de nome idade, mostrar também o valor passado (ver exemplo da Figura 5).

3. Altere o código do ServletListarNomeParametros para imprimir somente os parâmetros que forem diferentes de nulo. Por exemplo, antes se o parâmetro idade não fosse informado, seria impresso na página o valor null, certo? Porém, agora, ele não deverá ser apresentado na tela de retorno das informações.

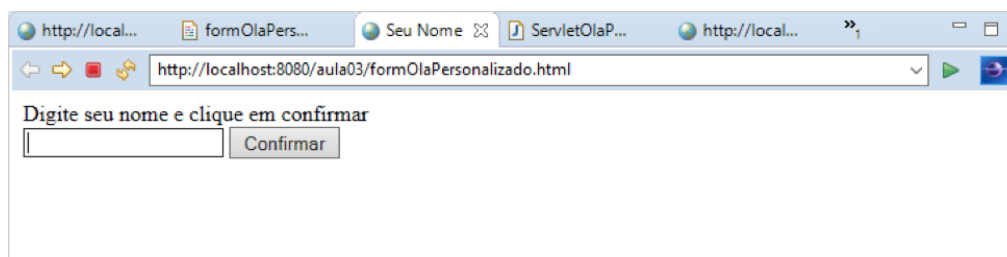
## Manipulando os parâmetros das requisições web IV

**Figura 05** - Nomes e valores dos parâmetros



Você já sabe como pegar parâmetros de requisições web, mas os exemplos que mostramos e que você praticou até agora usam basicamente o método GET, ou seja, os parâmetros foram passados na própria URL. Vamos agora praticar um pouco o uso do método POST, através da passagem de parâmetros via formulários web. Observe a página mostrada na Figura 6. Ela contém um formulário usado para chamar o ServletOlaPersonalizado, passando o nome da pessoa como parâmetro. O código HTML dessa página é mostrado na Listagem 3.

**Figura 06** - Formulário do olá personalizado



```

1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
4     <title>Seu Nome</title>
5   </head>
6   <body>
7     Digite seu nome e clique em confirmar
8     <form action="ServletOlaPersonalizado" method="post">
9       <input type="text" name="nome"/>
10      <input type="submit" value="Confirmar"/>
11    </form>
12  </body>
13 </html>

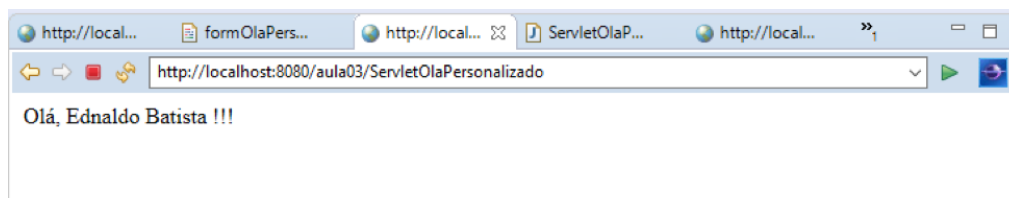
```

**Listagem 3** - Código HTML para criar formulário do ServletOlaPersonalizado

Se atualmente o ServletOlaPersonalizado só tiver o método **doGet** então ele não irá responder ao POST, portanto você deve criar um método **doPost** da forma tradicional (similar ao doGet) porém sua implementação deve conter somente o comando **doGet(request, response);** que simplesmente executa o método doGet. Dessa forma os dois métodos são suportados e executam o mesmo código.

Depois de garantir que o método do doPost existe e simplesmente chama o doGet, ao preencher o seu nome no formulário e submetê-lo, ele será apresentado na página de resposta. Veja o exemplo de resposta da Figura 7 para o caso de digitarmos o nome "Ednaldo Batista".

**Figura 07** - Exemplo de resposta do ServletOlaPersonalizado invocado via formulário web



## Atividade 03

1. Sem olhar diretamente o código, tente criar o código da página HTML contendo um formulário para executar o ServletOlaPersonalizado. Teste e, em caso de dúvida, consulte o código mostrado.
2. Crie uma página HTML com um formulário de cadastro dos dados pessoais de uma pessoa (nome, sobrenome, endereço etc.) e faça-o invocar o

ServletListarNomeParametros adaptado para mostrar os nomes dos parâmetros e valores passados pelo usuário.

3. Além dos métodos para manipular parâmetros da requisição, a classe `HttpServletRequest` possui diversos outros métodos com função de retornar outros tipos de informação sobre a requisição web. Faça uma pesquisa na internet sobre essa classe e tente descobrir a função de alguns de seus outros métodos. Escreva aqui as funções que mais lhe chamaram a atenção.
4. Altere o `ServletListarNomeParametros` para imprimir na sua tela de resposta o resultado de alguns dos métodos de `HttpServletRequest` que você listou na resposta anterior.

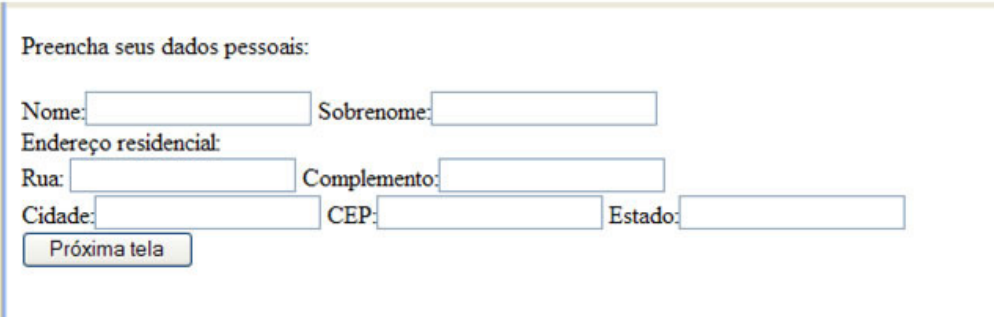
## Mantendo o estado do cliente web

---

Em aplicações desktop, ou seja, aquelas feitas para rodar na máquina do cliente, todas as informações que vão sendo digitadas pelo usuário vão ficando armazenadas ao longo das várias telas que um sistema pode ter. Por exemplo, em um sistema de cadastro, o usuário pode ter que cadastrar seus dados pessoais em uma tela e seus dados profissionais (local de trabalho, etc.) em outra, para só então confirmar o cadastro de todos os dados. Esse comportamento pode ser até natural em um sistema desktop, mas em um sistema web temos que seguir certos passos para isso poder funcionar (não quer dizer que também não seja fácil).

Vamos pensar primeiro no funcionamento geral desse sistema de cadastro de pessoas, no qual a primeira tela é mostrada na Figura 8.

**Figura 08** - Primeira tela do sistema de cadastro



Preencha seus dados pessoais:

Nome:  Sobrenome:

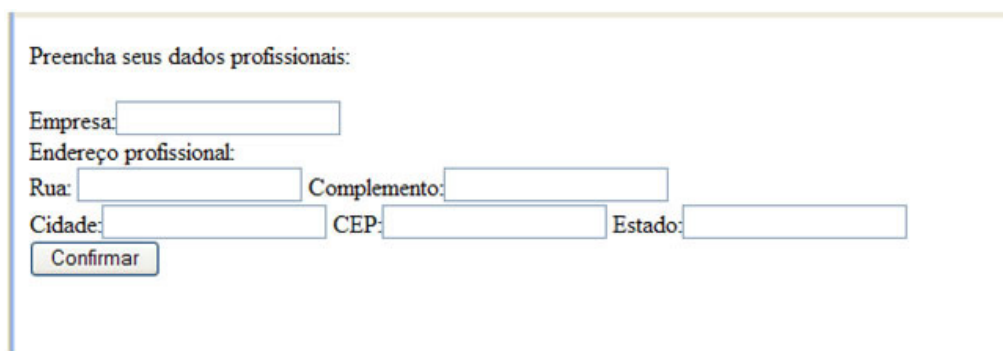
Endereço residencial:

Rua:  Complemento:

Cidade:  CEP:  Estado:

Após o usuário preencher seus dados pessoais e clicar no botão Próxima tela, a página da Figura 9 será apresentada.

**Figura 09** - Segunda tela de cadastro



Preencha seus dados profissionais:

Empresa:

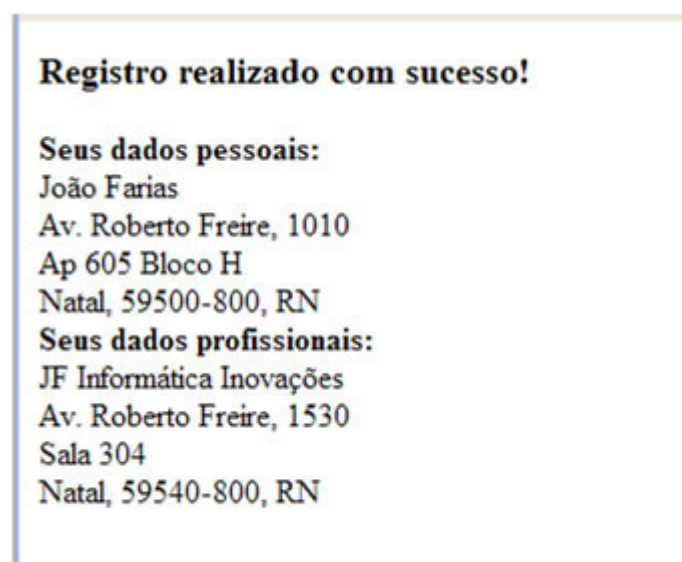
Endereço profissional:

Rua:  Complemento:

Cidade:  CEP:  Estado:

Ao preencher os dados profissionais e clicar no botão Confirmar, a tela de resposta apresentada é a da Figura 10.

**Figura 10** - Resposta obtida pelo sistema após preenchimento dos dados do usuário



**Registro realizado com sucesso!**

**Seus dados pessoais:**  
João Farias  
Av. Roberto Freire, 1010  
Ap 605 Bloco H  
Natal, 59500-800, RN

**Seus dados profissionais:**  
JF Informática Inovações  
Av. Roberto Freire, 1530  
Sala 304  
Natal, 59540-800, RN

## Mantendo o estado do cliente web II

---

Agora que você já tem uma boa ideia do que o sistema deve fazer, vamos ver como podemos implementá-lo. Começando pela primeira tela (Figura 8), precisamos criar um arquivo HTML (cadastro.html) com o formulário. O código dele é mostrado na Listagem 4. Note no código HTML a existência do formulário e de sua submissão para um Servlet de nome ServletTela1Cadastro.

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
4     <title>Dados Pessoais</title>
5   </head>
6   <body>
7     Preencha seus dados pessoais:
8     <form action="ServletTela1Cadastro">
9       Nome:<input type="text" name="nome">
10      Sobrenome:<input type="text" name="sobrenome"><br>
11      Endereço residencial:<br>
12      Rua: <input type="text" name="rua">
13      Complemento:<input type="text" name="complemento"><br>
14      Cidade:<input type="text" name="cidade">
15      CEP:<input type="text" name="cep">
16      Estado:<input type="text" name="estado"><br>
17      <input type="submit" value="Próxima tela"><br>
18    </form>
19  </body>
20 </html>
```

**Listagem 4** - Página HTML contendo formulário de cadastro dos dados pessoais

Página HTML contendo formulário de cadastro dos dados pessoais



```

1 package aula03;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet("/ServletTela1Cadastro")
13 public class ServletTela1Cadastro extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15
16     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
17         doPost(request, response);
18     }
19
20     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
21         PrintWriter resposta = response.getWriter();
22
23         resposta.write("<html>");
24         resposta.write("<head>");
25         resposta.write("<title>Dados Profissionais</title>");
26         resposta.write("</head>");
27         resposta.write("<body>");
28         resposta.write("Preencha seus dados profissionais:");
29         resposta.write("<form action=\"ServletTela2Cadastro\">");
30         resposta.write("Empresa:<input type=\"text\" name=\"empresa\"> <BR>");
31         resposta.write("Endereço profissional:<BR>");
32         resposta.write("Rua: <input type=\"text\" name=\"ruaEmpresa\">");
33         resposta.write("Complemento:<input type=\"text\" name=\"complementoEmpresa\"><BR>");
34         resposta.write("Cidade:<input type=\"text\" name=\"cidadeEmpresa\">");
35         resposta.write("CEP:<input type=\"text\" name=\"cepEmpresa\">");
36         resposta.write("Estado:<input type=\"text\" name=\"estadoEmpresa\"><BR>");
37         resposta.write("<input type=\"submit\" value=\"Confirmar\"><BR>");
38         resposta.write("</form>");
39         resposta.write("</body></html>");
40     }
41
42 }
43

```

**Listagem 5** - Implementação inicial do ServletTela1Cadastro

Note que a implementação do ServletTela1Cadastro mostrada na Listagem 5 monta uma página HTML de resposta referente à segunda tela de cadastro. Entretanto, os dados digitados na primeira tela não estão sendo armazenados. Temos então que pensar agora em como manter os dados de uma tela para outra.

Existem duas abordagens diferentes que podemos usar para isso: repassar parâmetros ou usar sessões. A primeira abordagem será apresentada a seguir, a outra será apresentada na nossa próxima aula.

## Repassando parâmetros

---

Uma primeira solução é usar os chamados campos ocultos (hidden) dos formulários web. Basicamente, nessa abordagem, todos os campos da primeira tela irão aparecer na segunda tela, mas agora como campos ocultos:

```
1 <input type="hidden" name="nome" value="">
2 <input type="hidden" name="sobrenome" value="">
3 <input type="hidden" name="rua" value="">
4 <input type="hidden" name="complemento" value="">
5 <input type="hidden" name="cidade" value="">
6 <input type="hidden" name="cep" value="">
7 <input type="hidden" name="estado" value="">
```

Os campos *value* devem ser preenchidos, claro, com os valores digitados pelo usuário. Isto é, feito inserindo-se o código mostrado na Listagem 6 logo após a linha 29 (dentro do form) da Listagem 5. Note que o código da Listagem 6 basicamente escreve os campos ocultos do formulário, definindo o valor do campo de acordo com o parâmetro recebido (`request.getParameter()`). Por exemplo, a linha 01 é responsável por escrever `<input type="hidden" name="nome" value=""`. Observe nessa linha 01 o uso de `\` para representar as aspas duplas dentro de uma String. Já o comando da linha 02 imprime o nome recebido como parâmetro e o fechamento das aspas e o marcador HTML input (`>`). Se o nome digitado pelo usuário for João, teremos:

```
1 <input type="hidden" name="nome" value=" + João + "> = <input type="hidden" name="nome" va
```

Esse processo se repete para os outros campos ocultos, um para cada parâmetro a ser recebido pelo Servlet.

```

1 resposta.write("<input type=\"hidden\" name=\"nome\" value=\"\"");
2 resposta.write(request.getParameter("nome") + "\">");
3 resposta.write("<input type=\"hidden\" name=\"sobrenome\" value=\"\"");
4 resposta.write(request.getParameter("sobrenome") + "\">");
5 resposta.write("<input type=\"hidden\" name=\"rua\" value=\"\"");
6 resposta.write(request.getParameter("rua") + "\">");
7 resposta.write("<input type=\"hidden\" name=\"complemento\" value=\"\"");
8 resposta.write(request.getParameter("complemento") + "\">");
9 resposta.write("<input type=\"hidden\" name=\"cidade\" value=\"\"");
10 resposta.write(request.getParameter("cidade") + "\">");
11 resposta.write("<input type=\"hidden\" name=\"cep\" value=\"\"");
12 resposta.write(request.getParameter("cep") + "\">");
13 resposta.write("<input type=\"hidden\" name=\"estado\" value=\"\"");
14 resposta.write(request.getParameter("estado") + "\">");

```

**Listagem 6** - Repassando parâmetros através dos campos ocultos

## Repassando parâmetros II

```

1 <html>
2   <head>
3     <title>Dados Profissionais</title>
4   </head>
5   <body>
6     Preencha seus dados profissionais:
7     <form action="ServletTela2Cadastro">
8       <input type="hidden" name="nome" value="João">
9       <input type="hidden" name="sobrenome" value="Farias">
10      <input type="hidden" name="rua" value="Av. Roberto Freire, 1010">
11      <input type="hidden" name="complemento" value="Ap 605 Bloco H">
12      <input type="hidden" name="cidade" value="Natal">
13      <input type="hidden" name="cep" value="59500-800">
14      <input type="hidden" name="estado" value="RN">
15      Empresa: <input type="text" name="empresa"> <br/>
16      Endereço profissional:<br/>
17      Rua: <input type="text" name="ruaEmpresa">
18      Complemento:<input type="text" name="complementoEmpresa"><br/>
19      Cidade: <input type="text" name="cidadeEmpresa">
20      CEP:<input type="text" name="cepEmpresa">
21      Estado:<input type="text" name="estadoEmpresa"><br/>
22      <input type="submit" value="Confirmar"><br/>
23    </form>
24  </body>
25 </html>

```

**Listagem 7** - Código HTML da segunda tela de cadastro contendo campos ocultos

Para receber e processar os dados da segunda tela de cadastro, podemos implementar um `ServletTela2Cadastro` de acordo com o código mostrado na Listagem 8. Observe que esse Servlet acessa da mesma forma tanto os dados ocultos (dados pessoais, repassados da primeira tela de cadastro) como os dados digitados pela segunda tela (dados profissionais).

```
1 package aula03;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet("/ServletTela2Cadastro")
13 public class ServletTela2Cadastro extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15
16     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
17         doPost(request, response);
18     }
19
20     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
21         PrintWriter resposta = response.getWriter();
22         resposta.write("<html>");
23         resposta.write("<head>");
24         resposta.write("<title>Confirmação de registro</title>");
25         resposta.write("</head>");
26         resposta.write("<body>");
27         resposta.write("<h3>Registro realizado com sucesso!</h3><br><br>");
28         resposta.write("<b>Seus dados pessoais:</b><br>");
29         resposta.write(request.getParameter("nome") + " " + request.getParameter("sobrenome"));
30         resposta.write("<br>");
31         resposta.write(request.getParameter("rua"));
32         resposta.write("<br>");
33         resposta.write(request.getParameter("complemento"));
34         resposta.write("<br>");
35         resposta.write(request.getParameter("cidade"));
36         resposta.write(", ");
37         resposta.write(request.getParameter("cep"));
38         resposta.write(", ");
39         resposta.write(request.getParameter("estado"));
40         resposta.write("<br>");
41         resposta.write("<b>Seus dados profissionais:</b><br>");
42         resposta.write(request.getParameter("empresa"));
43         resposta.write("<br>");
44         resposta.write(request.getParameter("ruaEmpresa"));
45         resposta.write("<br>");
46         resposta.write(request.getParameter("complementoEmpresa"));
47         resposta.write("<br>");
48         resposta.write(request.getParameter("cidadeEmpresa"));
49         resposta.write(", ");
50         resposta.write(request.getParameter("cepEmpresa"));
51         resposta.write(", ");
```

```
52     resposta.write(request.getParameter("estadoEmpresa"));
53     resposta.write("</body></html>");
54 }
55 }
```

**Listagem 8** - Código da classe ServletTela2Cadastro

## Atividade 04

---

1. Implemente o ServletTela2Cadastro, conforme mostrado anteriormente, porém, tentando não olhar o código apresentado
2. Execute o sistema mostrado como exemplo, desde a primeira tela (processada pelo ServletTela1Cadastro) até a segunda tela (processada pelo ServletTela2Cadastro). Observe:
  - a. se o código fonte da segunda tela possui os campos ocultos. Para isso, clique com o botão direito na segunda tela e selecione a opção de exibir código fonte. Essa opção pode ter nome diferente de acordo com o navegador web utilizado.
  - b. se os valores digitados são mostrados corretamente na tela de confirmação de cadastro.
3. Altere o sistema mostrado como exemplo para ter uma terceira tela, responsável pela digitação de dados bancários do usuário (código do banco, agência e conta bancária). Só após a terceira tela é que a tela de confirmação dos dados deve aparecer, agora mostrando também os dados bancários do usuário.

## Limitações dessa abordagem

---

A abordagem mostrada (repasse de parâmetros de uma requisição para outra) permite que dados digitados em uma tela anterior não sejam perdidos, mas repassados para a próxima tela. Essa abordagem pode ser útil em diversas situações, mas possui, também, várias limitações. Em primeiro lugar, se o usuário digitou sua senha na primeira tela, não se deve repassar esse valor para a segunda tela. Por questões de segurança, senhas e outras informações confidenciais (número

de cartão de crédito etc.) não devem ser trafegadas como campos ocultos de páginas web, pois aumenta o risco de hackers conseguirem roubar essas informações.

Outro problema é que um sistema de cadastro pode ter mais telas do que o exemplo mostrado. Transitar os dados digitados entre todas as telas pode aumentar muito o código dos Servlets, além de aumentar a chance de erros de programação. Por fim, se a janela do navegador web do cliente for fechada, os dados serão perdidos, correto? Imagine um sistema de compras no qual um cliente escolhe cinco produtos para comprar e seu computador trava. Não seria interessante que, ao voltar à loja virtual, os cinco produtos escolhidos ainda estivessem no seu carrinho de compras? Na próxima aula, veremos uma abordagem que auxilia a resolver todas essas limitações. Até lá!

# Leitura Complementar

---

Para complementar seu entendimento sobre a manipulação de parâmetros de uma requisição, bem como o repasse desses parâmetros, analise os métodos disponibilizados pelas classes envolvidas. Isso pode ser feito observando-se a API do Java:

Java(TM) EE 8 Specification APIs

- <<https://javaee.github.io/javaee-spec/javadocs/>>

Em especial, olhar o pacote `javax.servlet` e as interfaces `HttpServletRequest` e `HttpServletResponse`.

## Resumo

---

Nesta aula, você aprendeu a receber e processar parâmetros de requisições web. Para isso, você usou métodos da classe `HttpServletRequest`. Você viu que o método mais utilizado nos exemplos foi o método `getParameter()`, responsável por retornar o valor de um dado parâmetro da requisição. Além disso, você aprendeu a repassar os valores de parâmetros de uma tela para outra. Apesar de essa abordagem ser útil em algumas situações, ela possui limitações, como o trabalho extra de codificação e o perigo de transitar entre as requisições dados sensíveis (senhas, dados bancários etc.).

## Autoavaliação

---

1. Descreva a(s) classe(s) e métodos utilizados para processar parâmetros de requisições web.
2. Descreva o funcionamento geral sobre a abordagem de repassar os valores digitados pelo usuário (parâmetros) de uma requisição para outra. Identifique também as limitações dessa abordagem.



# Referências

---

AHMED, K. Z.; UMRYSH, C. E. **Desenvolvendo aplicações comerciais em Java com J2EE J2EE e UML**. Rio de Janeiro: Ciência Moderna, 2003. 324 p.

CATTELL, Rick; INSCORE, Jim. **J2EE: criando aplicações comerciais**. Rio de Janeiro: Editora Campus, 2001.

HALL, Marty. **More Servlets and JavaServer Pages (JSP)**. New Jersey: Prentice Hall PTR (InformIT), 2001. 752 p. Disponível em: <<http://pdf.moreservlets.com/>>. Acesso em: 11 maio 2012.

HALL, Marty; BROWN, Larry. **Core Servlets and JavaServer Pages (JSP)**. 2nd ed. New Jersey: Prentice Hall PTR (InformIT), 2003. 736p. (Core Technologies, 1). Disponível em: <<http://pdf.coreservlets.com/>>. Acesso em: 11 maio 2012.

HYPertext Transfer Protocol: HTTP/1.1 – Methods Definition. Disponível em: <<http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>>. Acesso em: 11 maio 2012.

J2EE Tutorial. Disponível em: <<http://java.sun.com/javaee/reference/tutorials/>>. Acesso em: 11 maio 2012.