

INSTITUTO FEDERAL  
ESPIRITO SANTO

# POO2: Comunicação em Redes de Computadores



INSTITUTO FEDERAL  
ESPIRITO SANTO

# Agenda



- Servlets
  - GET
  - POST
- JAVASCRIPT JSON

# Servlets

Na plataforma Java, a primeira e principal tecnologia capaz de gerar páginas dinâmicas são as **Servlets**, que surgiram no ano de 1997.

Permite a criação de páginas dinâmicas com Java, ou seja HTML. O nome "servlet" vem da ideia de um pequeno servidor (*servidorzinho*, em inglês) cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente.

Cada servlet é, portanto, um objeto Java que recebe tais requisições (**request**) e produz algo (**response**), como uma página HTML dinamicamente gerada.

# Servlets

A interface Servlet é a que define como uma servlet funciona, mas não é o que vamos utilizar, uma vez que ela possibilita o uso de qualquer protocolo baseado em requisições e respostas, e não especificamente o HTTP.

Vamos usar a classe HttpServlet. Vamos herda-la e sobrescrever um método chamado doPost ou doGet. Esse método será o responsável por atender requisições e gerar as respostas adequadas.

Sua assinatura:

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

    System.out.println("DO POST");
    processRequest(request, response);
}
```



# POST

“The POST method is used to”

- Annotation of existing resources;
- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
- Providing a block of data, such as the result of submitting a form, to a data-handling process;
- Extending a database through an append operation.

part of [Hypertext Transfer Protocol -- HTTP/1.1](#) - RFC 2616

# GET

“GET methods SHOULD NOT have the significance of taking an action other than retrieval”

- The GET method means retrieve whatever information

part of [Hypertext Transfer Protocol -- HTTP/1.1](#) - RFC 2616

# Método POST VS GET

Página HTML  
GET

```
<form action="adicionaContato" method="GET">
```

Página  
HTML POST

```
<form action="adicionaContato" method="POST">
```

```
void doGet(HttpServletRequest req, HttpServletResponse res);  
void doPost(HttpServletRequest req, HttpServletResponse res);
```

Servlet

# Código de Exemplo

Comunicação Servidores – Código JAVA



# Exemplo

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Minha Primeira Página</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Meu Primeiro Servlet");
        out.println("</body>");
        out.println("</html>");
    } finally {
```

Exibi uma mensagem  
HTML simples

<http://localhost:8084/ServletHTML/OiMundo>

# Web.xml

Para que funcione é necessário que seja configurado sua classe e o seu caminho no web.xml

```
<servlet>
  <servlet-name>OiMundo</servlet-name>
  <servlet-class>OiMundo</servlet-class>
</servlet>
```

```
<servlet-mapping>
  <servlet-name>OiMundo</servlet-name>
  <url-pattern>/OiMundo</url-pattern>
</servlet-mapping>
```

[https://github.com/felipefo/poo2/blob/master/Padroes\\_de\\_Projeto/Estrutural/Decorator/ServletHTML/web/WEB-INF/web.xml](https://github.com/felipefo/poo2/blob/master/Padroes_de_Projeto/Estrutural/Decorator/ServletHTML/web/WEB-INF/web.xml)

# Enviando Parâmetros

Ao desenvolver uma aplicação Web sempre precisamos realizar operações no lado do servidor, com dados informados pelo usuário, seja através de formulários ou através da URL.

Vamos criar uma página HTML, chamada formulario.html

Servlet que vai  
responder  
(RecebeDados.java)

Nome:

E-mail:


Endereço:

```
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
  <form action="RecebeDados">
    Nome: <input type="text" name="nome" /><br />
    E-mail: <input type="text" name="email" /><br />
    Endereço: <input type="text" name="endereco" /><br />
    <input type="submit" value="Enviar" />
  </form>
</body>
</html>
```

# Pegando Parâmetros

Para recebermos os valores que foram preenchidos na tela e submetidos, criaremos uma Servlet, cuja função será receber de alguma maneira esses dados e convertê-los, se necessário.

```
String valorDoParametro = request.getParameter("nomeDoParametro");
```



```
out.println("<h1>Nome: " + request.getParameter("nome") + "</h1>");  
out.println("<h1>Email: " + request.getParameter("email") + "</h1>");  
out.println("<h1>Endereco: " + request.getParameter("endereco") + "</h1>");
```

# Exemplo de Comunicação entre Servidores e Cliente

NÚMERO DA PORTA DE  
ACORDO COM O SEU  
SERVIDOR LOCAL

<http://localhost:8084/ServletsHTML/formulario.html>

Vamos adicionar method="GET" e depois  
method="POST"

# Exercício

Adicione o campo “identidade” a página formulario.html e faça o Servlet imprimir esse campo igual os outros parâmetros.

# Vamos deixar mais bonito?

<http://materializecss.com/>

- Um framework front-end moderno e responsivo baseado em Material Design]
- Baseado no <https://material.io/guidelines/> do google

Existem outras implementações? Sim existem n...

<https://github.com/react-materialize/react-materialize>

```
<head>
  <meta charset="UTF-8">
  <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <link type="text/css" rel="stylesheet" href="materialize-v0.98.1/css/materialize.min.css">
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</head>
<body>
  <form style='margin:20px;' action="RecebeDados">
    Nome: <input type="text" name="nome" /><br />
    E-mail: <input type="text" name="email" /><br />
    Endereço: <input type="text" name="endereco" /><br />
    <button class="btn waves-effect waves-light" type="submit" name="action">Enviar
      <i class="material-icons right">send</i>
    </button>
  </form>
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script type="text/javascript" src="materialize-v0.98.1/js/materialize.min.js"></script>
```



# Exemplo com o get

<http://localhost:8080/ServletHTML/exemploget?id=001>

```
    usuarios.put("001" , "Felipe");
    usuarios.put("002" , "Joao");
}

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Minha Primeira Página</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Meu Primeiro Servlet");
        out.println("<h1>Id: " + request.getParameter("id") + "</h1>");
        out.println("<h1>Nome: " + usuarios.get(request.getParameter("id")) + "</h1>");

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

# Enviando Parâmetros

## Enviando informações com JSON

```
<html>
<head>
  <meta charset="UTF-8">
  <script src="ajax.js"></script>
</head>
<body onload="onloadBody()">
  <form id="form" action="RecebeJSON" method="POST">
    Nome: <input type="text" name="nome" /><br />
    E-mail: <input type="text" name="email" /><br />
    Endereço: <input type="text" name="endereco" /><br />
    <input type="submit" value="Enviar" />
  </form>
</body>
</html>
```

Nome:

E-mail:

Endereço:

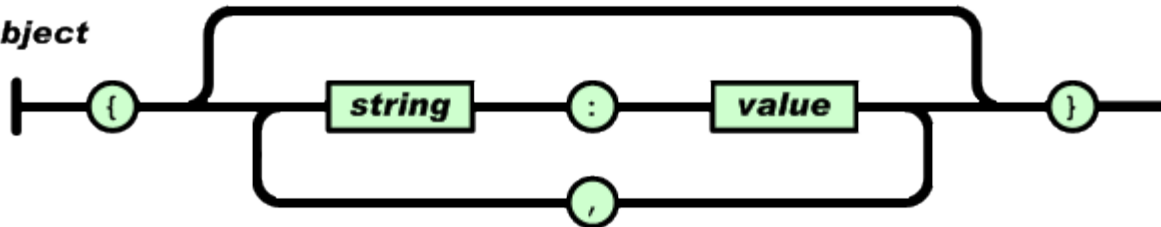
# JavaScript JSON

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

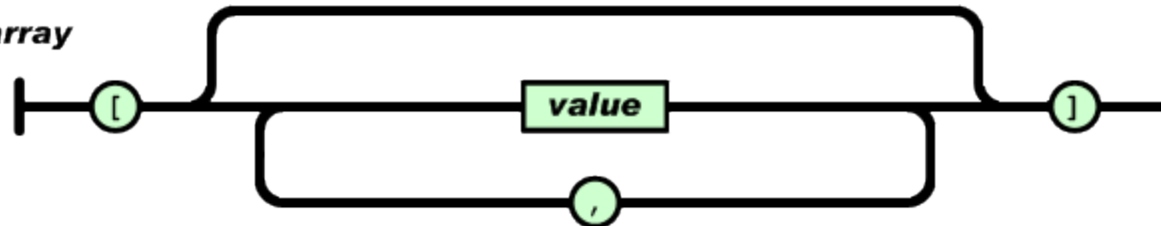
Uma lista de Empregados

Sintaxe

**object**



**array**



<http://www.json.org/>



# JavaScript JSON

```
function onloadBody() {  
    var form;  
    form = document.getElementById("form");  
    form.onsubmit = function (e) {  
        e.preventDefault();  
        var data = {};  
        for (var i = 0; i < form.length; i++) {  
            var input = form[i];  
            console.log(input);  
            if (input.name) {  
                data[input.name] = input.value;  
            }  
        }  
        var xhr = new XMLHttpRequest();  
        xhr.open(form.method, form.action, true);  
        xhr.setRequestHeader('Content-Type', 'text/plain; charset=UTF-8');  
        xhr.send(JSON.stringify(data));  
        xhr.onloadend = function () {  
        };  
    };  
}
```

# Exercício - JSON

Acrescente o campo identidade ao  
formulario\_json.html

# JSON

## Exemplo

[https://www.w3schools.com/js/tryit.asp?filename=tryjson\\_server\\_sql\\_style](https://www.w3schools.com/js/tryit.asp?filename=tryjson_server_sql_style)

URL DE RETORNO DO JSON:

[http://www.w3schools.com/website/Customers\\_MYSQL.php](http://www.w3schools.com/website/Customers_MYSQL.php)

```
[ { "Name" : "Alfreds Futterkiste", "City" : "Berlin", "Country" : "Germany" }, {  
  "Name" : "Berglunds snabbköp", "City" : "Luleå", "Country" : "Sweden" }, {  
  "Name" : "Centro comercial Moctezuma", "City" : "México D.F.", "Country" :  
    "Mexico" }, { "Name" : "Ernst Handel", "City" : "Graz", "Country" :.....
```

# Referências

<https://www.caelum.com.br/apostila-java-web/servlets/>

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>