



# Desenvolvimento de Software para WEB

Aula 14 - Servlets + MVC + DAO

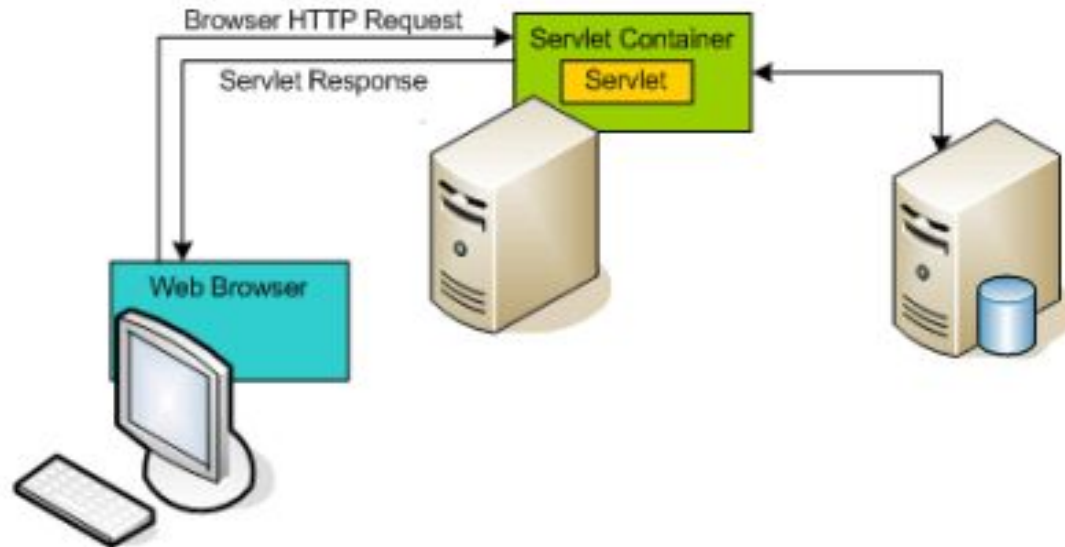
Professor: Anderson Almada

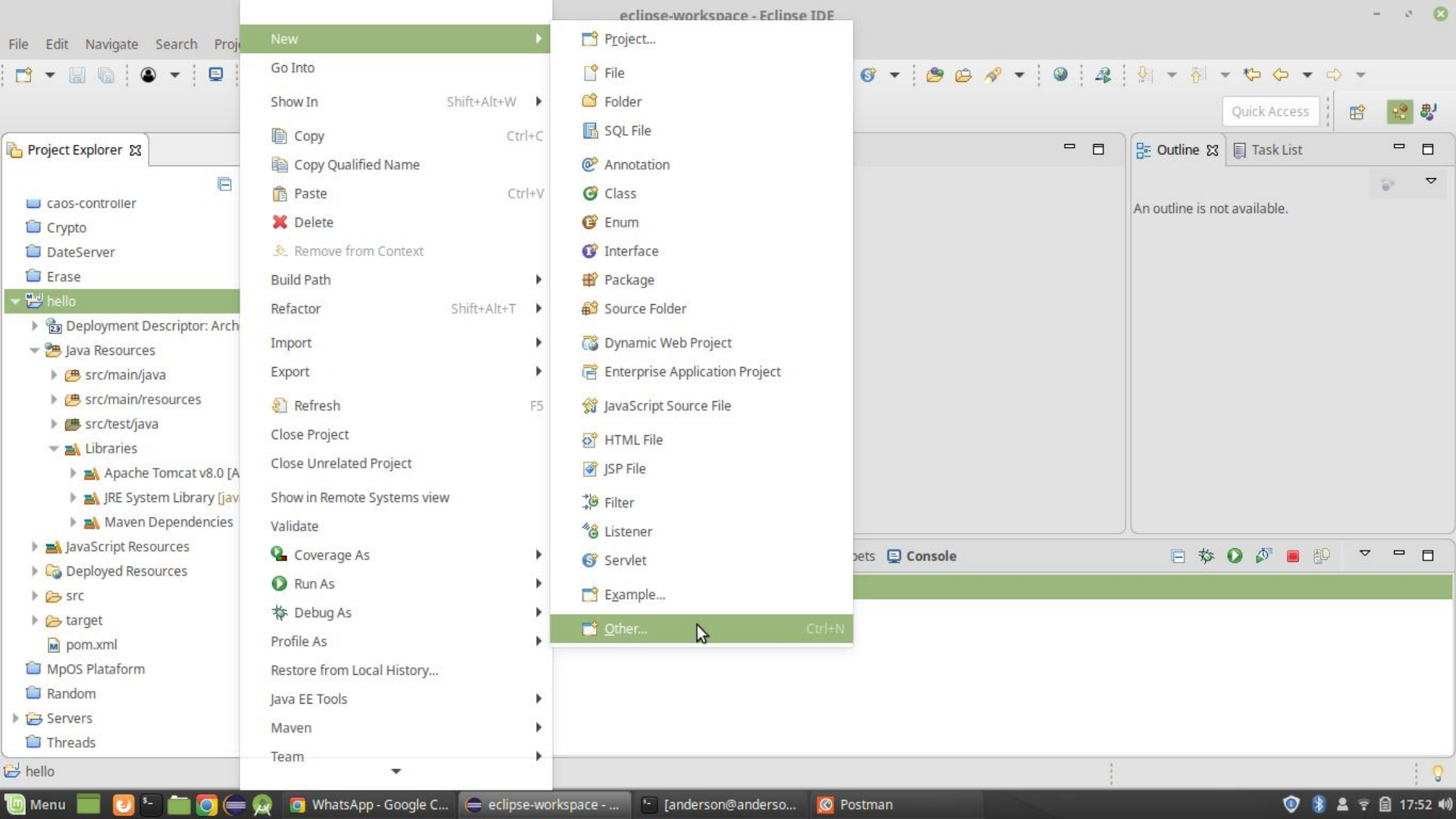
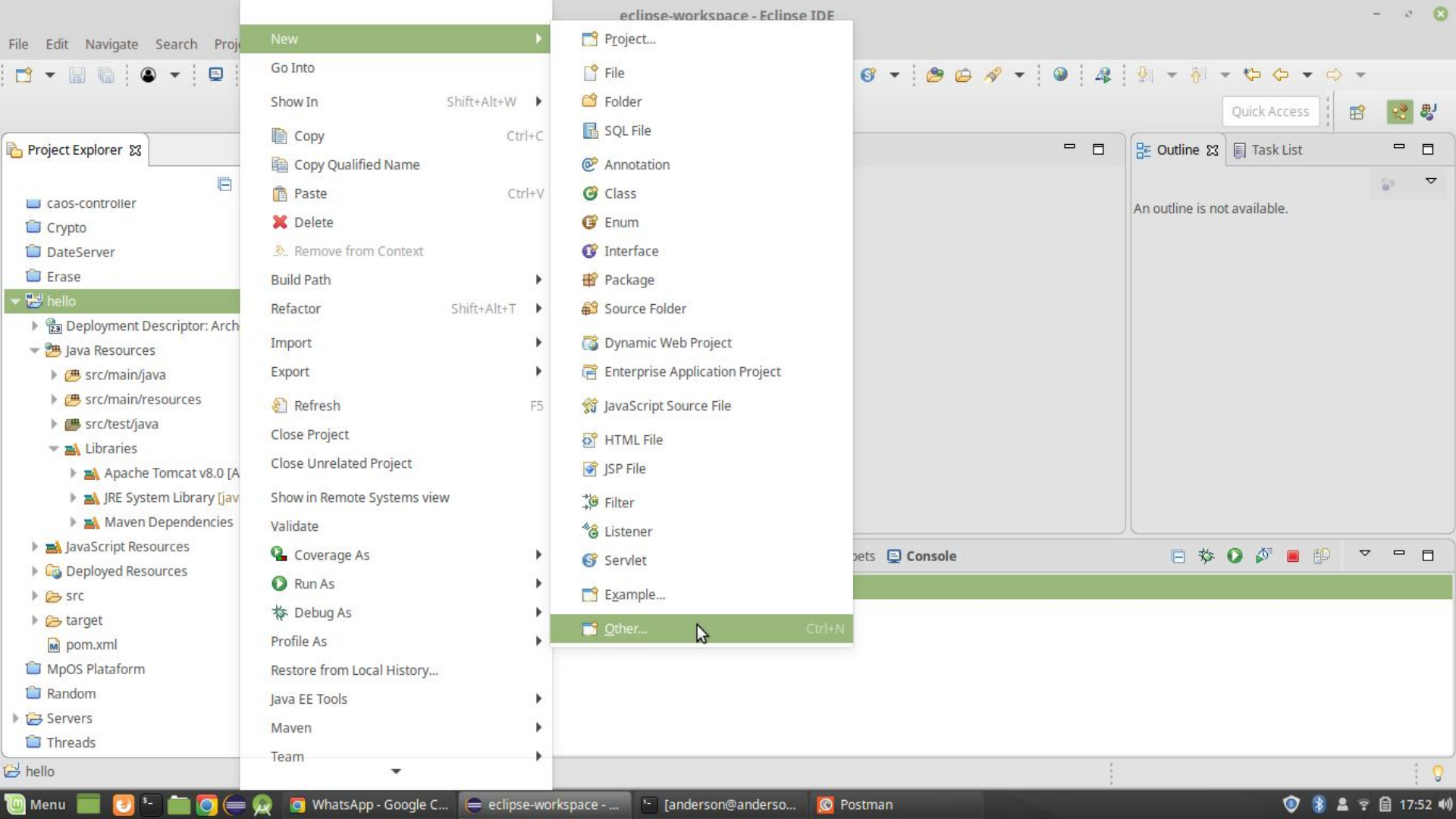
# Servlet

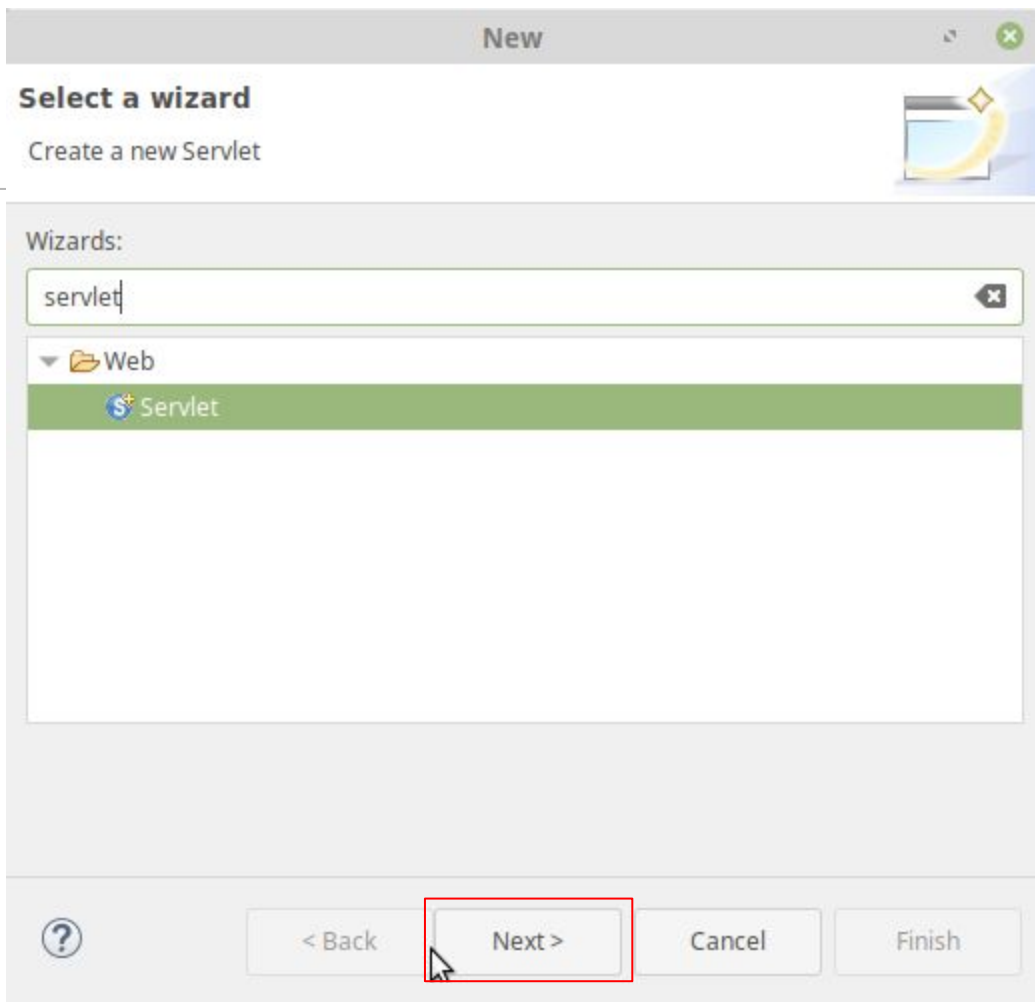
---

- Classe que terá capacidade de gerar conteúdo HTML
- O nome "servlet" vem da ideia de um pequeno servidor cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente.
- Cada servlet é um objeto Java que recebe tais requisições (request) e produz algo (response), como uma página HTML dinamicamente gerada.

# Servlet








### Create Servlet

Specify class file destination.



Project: hello

Source folder: /hello/src/main/java Browse...

Java package: Browse...

Class name: HelloAula

Superclass: javax.servlet.http.HttpServlet Browse...

☐ Use an existing Servlet class or JSP

Class name: HelloAula Browse...

? < Back Next > Cancel Finish



Project Explorer

- caos-api-stand-maven
- caos-controller
- Crypto
- DateServer
- Erase
- hello
  - Deployment Descriptor: Hello World
  - Java Resources
  - JavaScript Resources
  - Deployed Resources
  - src
    - main
      - java
        - HelloAula.java**
      - resources
      - webapp
      - test
    - target
    - pom.xml
  - MpOS Platform
  - Random
  - Servers
  - Threads

HelloAula.java web.xml

```
1
2
3 import java.io.IOException;
4
5
6
7
8
9 /**
10  * Servlet implementation class HelloAula
11  */
12 public class HelloAula extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     /**
16      * @see HttpServlet#HttpServlet()
17      */
18     public HelloAula() {
19         super();
20         // TODO Auto-generated constructor stub
21     }
22
23     /**
24      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
25      */
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) {
27         // TODO Auto-generated method stub
28         response.getWriter().append("Served at: ").append(request.getContextPath())
29     }
30 }
```

Outline Task List

Quick Access

- HelloAula
  - serialVersionUID : long
  - HelloAula()
  - doGet(HttpServletRequest, HttpServletResponse)
  - doPost(HttpServletRequest, HttpServletResponse)

Markers Properties Servers Data Source Explorer Snippets Console

Tomcat v8.0 Server at localhost [Started, Synchronized]

- hello [Synchronized]

The screenshot displays the Eclipse IDE interface with the following components:

- Project Explorer (Left):** Shows the project structure. The 'hello' project is expanded, revealing 'src/main/webapp/WEB-INF/web.xml' as the selected file.
- Main Editor:** Displays the content of 'web.xml'. The XML code is as follows:

```
1 <!DOCTYPE web-app PUBLIC
2 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
3 "http://java.sun.com/dtd/web-app_2_3.dtd" >
4
5 <web-app>
6   <display-name>Hello World</display-name>
7   <servlet>
8     <servlet-name>HelloAula</servlet-name>
9     <display-name>HelloAula</display-name>
10    <description></description>
11    <servlet-class>HelloAula</servlet-class>
12  </servlet>
13  <servlet-mapping>
14    <servlet-name>HelloAula</servlet-name>
15    <url-pattern>/HelloAula</url-pattern>
16  </servlet-mapping>
17 </web-app>
18
```
- Outline (Right):** Shows the XML document structure. The 'servlet-name' element under 'servlet-mapping' is selected.
- Servers (Bottom):** Shows the 'Tomcat v8.0 Server at localhost [Started, Synchronized]' and the 'hello [Synchronized]' context.



# Importante

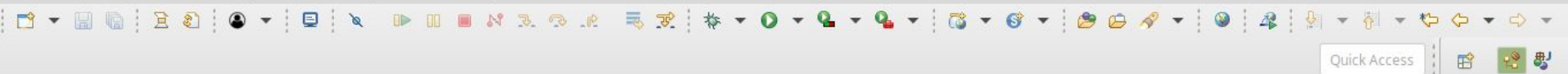
---

**Sempre que não atualizar, restart o tomcat**

The screenshot displays the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure. The 'hello' project is expanded, revealing 'src/main/java' containing 'HelloAula.java', 'resources', 'webapp', and 'WEB-INF'. 'web.xml' and 'index.jsp' are also visible in the 'WEB-INF' directory.
- Code Editor:** Displays the 'HelloAula.java' file. The code is a Java Servlet implementation. The line numbers 2 through 32 are visible. The code includes an import for `java.io.IOException`, a package comment, and a class definition `public class HelloAula extends HttpServlet`. It features a `doGet` method that writes `<h1>Oi</h1>` to the response.
- Outline:** Shows the class hierarchy and methods. The `HelloAula` class is selected, showing its `serialVersionUID` and the `doGet` method.
- Task List:** Currently empty.
- Markers:** Shows the status of the project. The 'Tomcat v8.0 Server at localhost' is started, and the 'hello' project is synchronized.
- Properties:** Shows the properties of the selected project.
- Servers:** Shows the list of servers.
- Data Source Explorer:** Shows the data sources.
- Snippets:** Shows the snippets.
- Console:** Shows the output of the application.

```
2
3 import java.io.IOException;
9
10 /**
11  * Servlet implementation class HelloAula
12  */
13 @WebServlet("/ok")
14 public class HelloAula extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public HelloAula() {
21         super();
22         // TODO Auto-generated constructor stub
23     }
24
25     /**
26      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) {
29         // TODO Auto-generated method stub
30         response.getWriter().append("<h1>Oi</h1>");
31     }
32 }
```



Project Explorer

- caos-api-stand-maven
- caos-controller
- Crypto
- DateServer
- Erase
- hello
  - Deployment Descriptor: Hello World
  - Java Resources
  - JavaScript Resources
  - Deployed Resources
  - src
    - main
      - java
        - HelloAula.java
      - resources
      - webapp
        - WEB-INF
          - web.xml
          - index.jsp
    - test
    - target
    - pom.xml
  - MoOS Platform

web.xml

```
1 <!DOCTYPE web-app PUBLIC
2 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
3 "http://java.sun.com/dtd/web-app_2_3.dtd" >
4
5 <web-app>
6   <display-name>Hello World</display-name>
7 </web-app>
8
```

Design Source

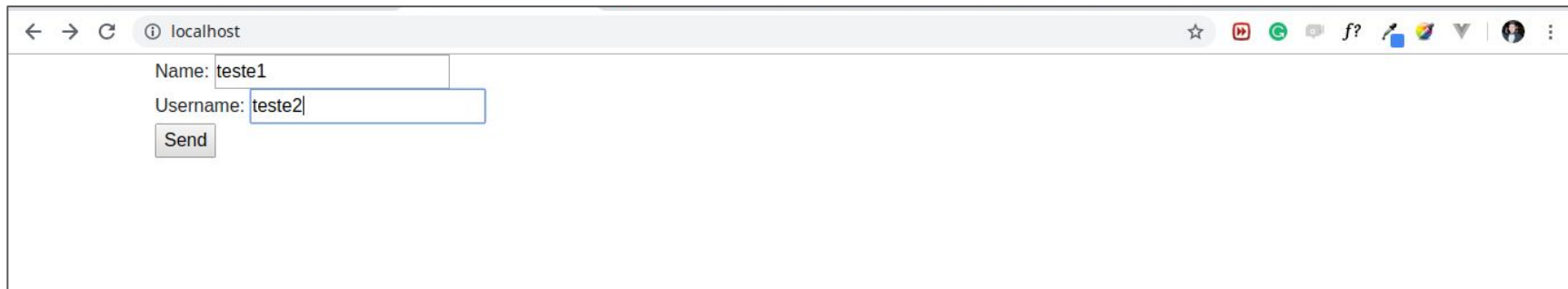
Markers Properties Servers Data Source Explorer Snippets Console

Tomcat v8.0 Server at localhost [Started, Synchronized]

- hello [Synchronized]

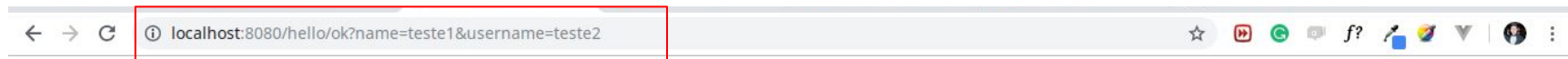
# Parâmetros - Formulários

```
<form action="http://localhost:8080/hello/ok" method="get">  
  Name: <input type="text" name="name" id=""><br>  
  Username: <input type="text" name="username" id=""><br>  
  <button type="submit">Send</button>  
</form>
```



A screenshot of a web browser window displaying a form. The browser's address bar shows 'localhost'. The form contains two text input fields: 'Name' with the value 'teste1' and 'Username' with the value 'teste2'. Below these fields is a 'Send' button. The browser's toolbar includes standard navigation icons (back, forward, refresh) and a search bar.

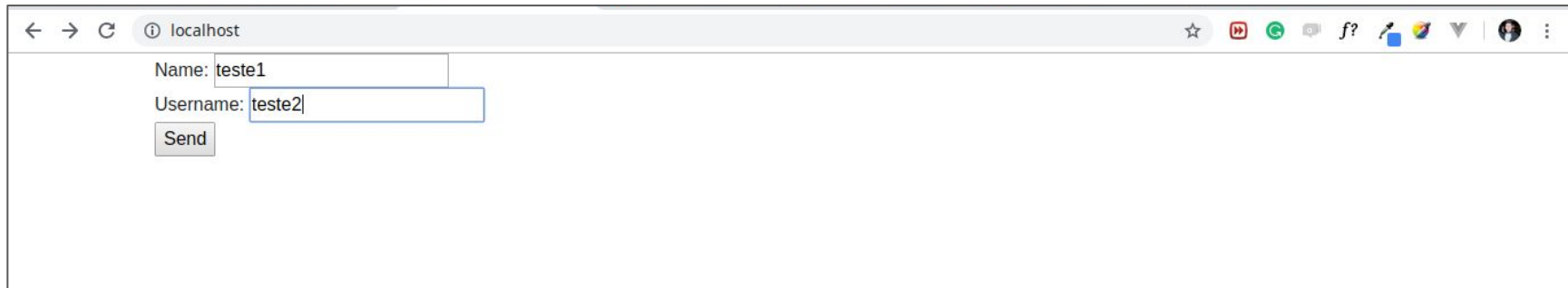
## Resposta no servidor



Oi

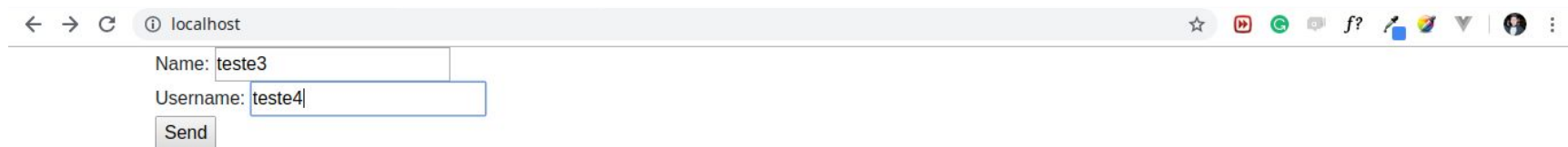
# Parâmetros - Formulários

```
<form action="http://localhost:8080/hello/ok" method="post">  
  Name: <input type="text" name="name" id=""><br>  
  Username: <input type="text" name="username" id=""><br>  
  <button type="submit">Send</button>  
</form>
```



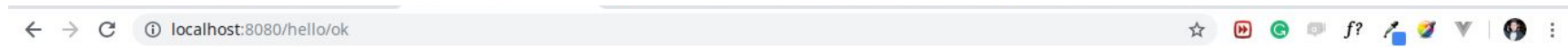
A screenshot of a web browser window displaying the rendered HTML form. The browser's address bar shows 'localhost'. The form contains two text input fields: 'Name: teste1' and 'Username: teste2'. Below these fields is a 'Send' button. The browser's toolbar includes standard navigation icons (back, forward, refresh) and a search bar.

## Requisição POST no frond-end



A screenshot of a web browser window. The address bar shows 'localhost'. The page contains a form with two input fields: 'Name:' with the value 'teste3' and 'Username:' with the value 'teste4'. Below the fields is a 'Send' button. The browser's toolbar includes back, forward, and refresh buttons, as well as various extension icons.

## Resposta no servidor



A screenshot of a web browser window. The address bar shows 'localhost:8080/hello/ok'. The browser's toolbar includes back, forward, and refresh buttons, as well as various extension icons.

Oi

# Parâmetros - Servidor - 1

---

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    String name = request.getParameter("name");
    String username = request.getParameter("username");
    response.getWriter().append("Name: " + name + "\nUsername: " +
username);
}
```



## Parâmetros - Servidor - 2

---

**getParameter**(String nomeParametro):

- Retorna o valor relativo a um determinado parâmetro passado;
- Esse parâmetro pode ser um dado de um formulário submetido ou um dado passado na URL;
- O retorno de `getParameter` sempre é `String`.
- O nome do parâmetro é case sensitive.
- Se o parâmetro não existir na requisição, `getParameter` retorna `null`.

## Parâmetros - Servidor - 3

---

**getParameterValues(String nomeParametro):**

- Retorna os valores relativos a um determinado parâmetro passado;
- Use esse método quando um dado de um formulário tiver mais de um valor (um “select”, por exemplo)
- O retorno de `getParameterValues` é um array de Strings

# Sessões - 1

---

```
HttpSession session = request.getSession();
```

**getSession()**, sem parâmetros, sempre cria uma sessão, caso ela não exista.

Com o parâmetro **boolean**, indica se deve criar ou não sessão caso ela não exista

## Sessões - 2

---

**setAttribute**(String name, Object o): Guarda um objeto na sessão.

Object **getAttribute**(String name): Obtém um determinado objeto armazenado na sessão. Caso ela não exista, o método retorna null.

**removeAttribute**(String name): Remove o atributo da sessão.

**getId**(): Retorna o número da sessão. Toda sessão tem um número único



Project Explorer

- DateServer
- Erase
- hello
  - Deployment Descriptor: Hello World
  - Java Resources
    - src/main/java
      - (default package)
        - HelloAula.java
        - Pessoa.java
      - src/main/resources
      - src/test/java
      - Libraries
    - JavaScript Resources
    - Deployed Resources
  - src
    - main
      - java
        - HelloAula.java
        - Pessoa.java
      - resources
      - webapp
      - test
      - target

HelloAula.java web.xml index.jsp Tomcat v8.0 Server at loca Pessoa.java

```
1
2 public class Pessoa {
3     String name;
4     String username;
5
6     public Pessoa(String name, String username) {
7         this.name = name;
8         this.username = username;
9     }
10 }
11
```

Outline Task List

Pessoa

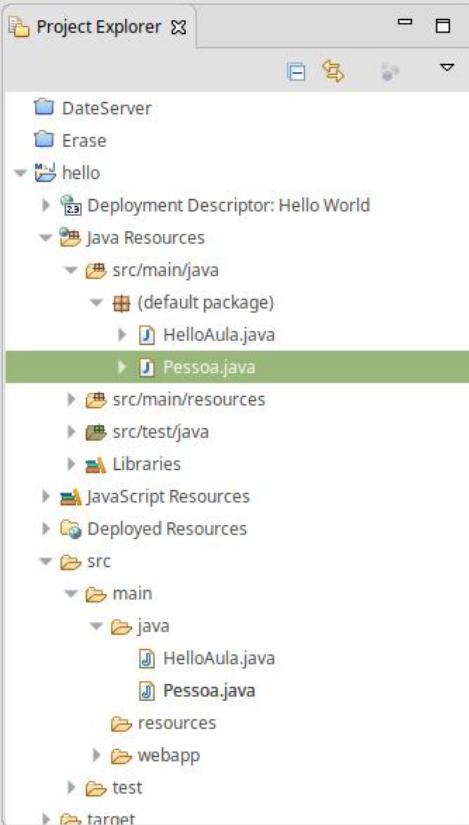
- name : String
- username : String
- Pessoa(String, String)

Tomcat v8.0 Server at localhost [Started, Restart]

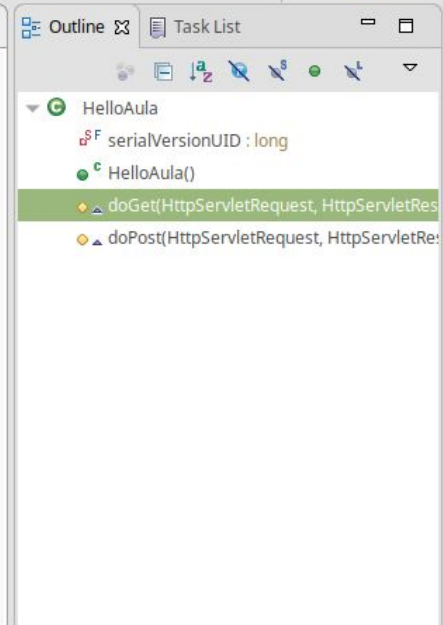
File Edit Source Refactor Navigate Search Project Run Window Help



Quick Access



```
15 public HelloAula() {
16     super();
17 }
18
19 protected void doGet(HttpServletRequest request, HttpServletResponse response)
20     throws ServletException, IOException {
21     HttpSession session = request.getSession();
22
23     if(session.getAttribute("pessoa") == null) {
24         Pessoa pessoa = new Pessoa(request.getParameter("name"), request.getPa
25         session.setAttribute("pessoa", pessoa);
26         response.getWriter().append("Name: " + pessoa.name + "\nUsername: " +
27     } else {
28         Pessoa pessoa = (Pessoa) session.getAttribute("pessoa");
29         response.getWriter().append("Logado !!\nName: " + pessoa.name + "\nUse
30     }
31
32
33 protected void doPost(HttpServletRequest request, HttpServletResponse response)
34     throws ServletException, IOException {
35     doGet(request, response);
36 }
37 }
38
```



Markers Properties Servers Data Source Explorer Snippets Problems Console

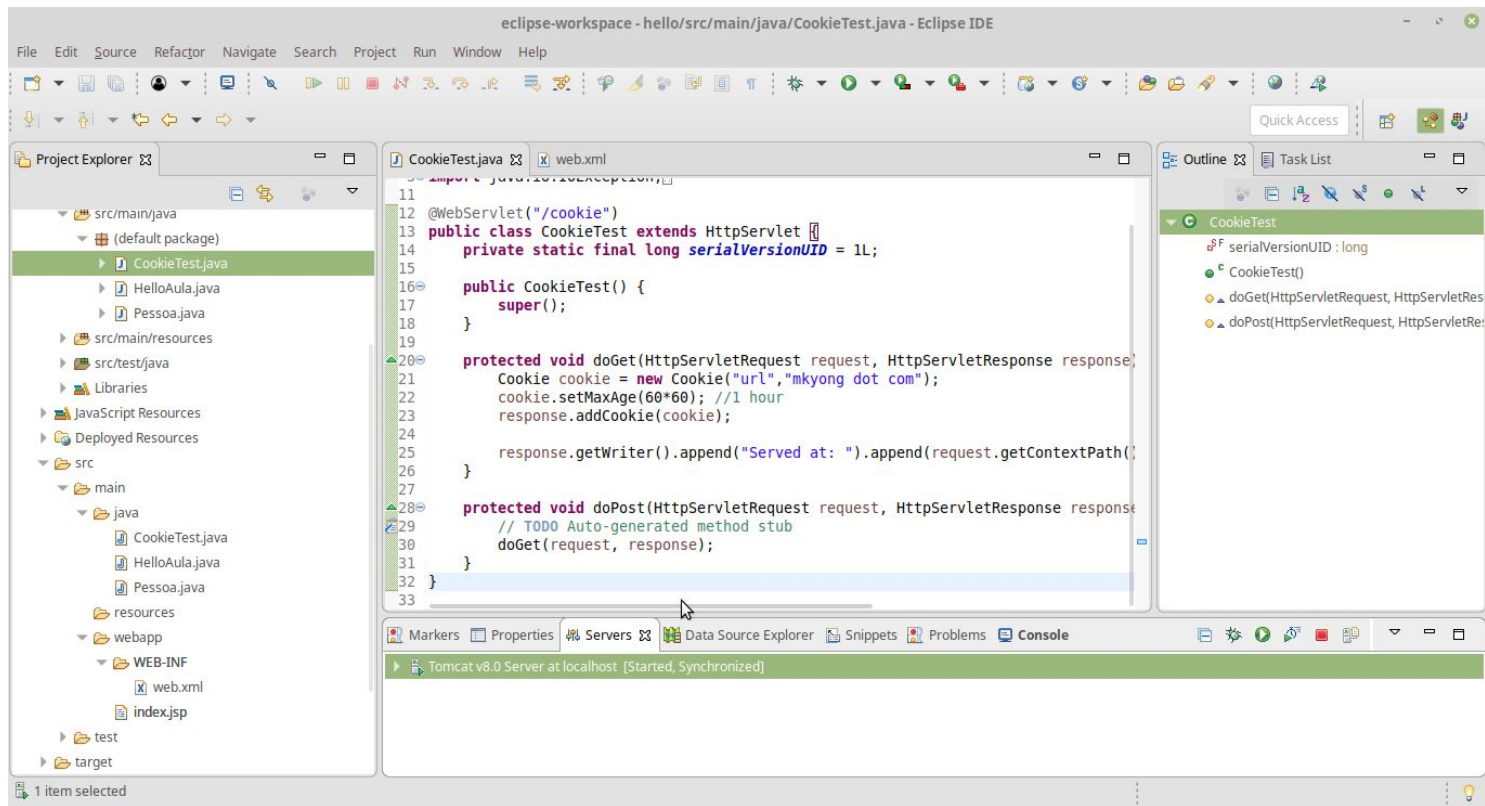
Tomcat v8.0 Server at localhost [Started, Restart]

Writable

Smart Insert

20:51

# Cookies - 1



Served at: /hello

Application

Manifest  
Service Workers  
Clear storage

Storage

- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies
  - http://localhost:8080

Cache

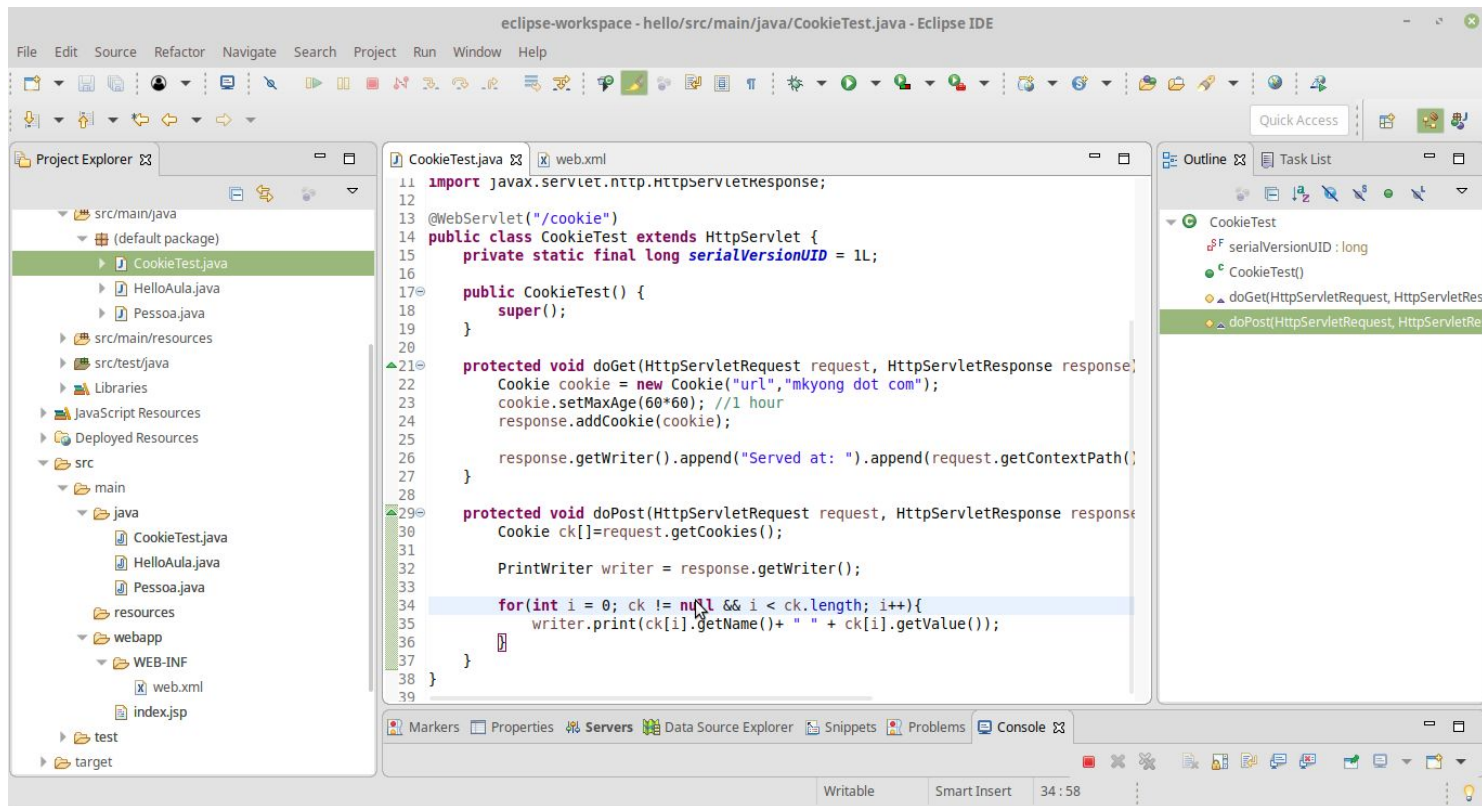
- Cache Storage
- Application Cache

Filter

Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure	SameSite
JSESSIONID	C306CAABFF367C2FA801E46D41A611F9	localhost	/hello/	N/A	42	✓		
url	"mkyong dot com"	localhost	/hello	2019-04-18T02:...	19			



# Cookies - 2



## Redirecionamiento

---

```
response.sendRedirect("http://...");
```

# Model-View-Controller

---

- Padrão arquitetural
- Separar o projeto em três camadas independentes, que são o modelo, a visão e o controlador.
- Reduzir o acoplamento, facilitar a manutenção e reutilização em outros projetos

# Model-View-Controller

---

- **Controller**

- A comunicação entre interfaces e regras de negócios é definida através de um controlador, e é a existência deste controlador que torna possível a separação entre as camadas.
- Quando um evento é executado na interface gráfica, como um clique em um botão, a interface irá se comunicar com o controlador que por sua vez se comunica com as regras de negócios.

# Model-View-Controller

---

- **Model**

- O Model realiza a operação matemática e retorna o valor calculado para o Controller, que também é o único que possui conhecimento da existência da camada de visualização.
- Tendo o valor em “mãos”, o intermediador o repassa para a interface gráfica que exibirá para o usuário. Caso esta operação deva ser registrada em uma base de dados, o Model se encarrega também desta tarefa.

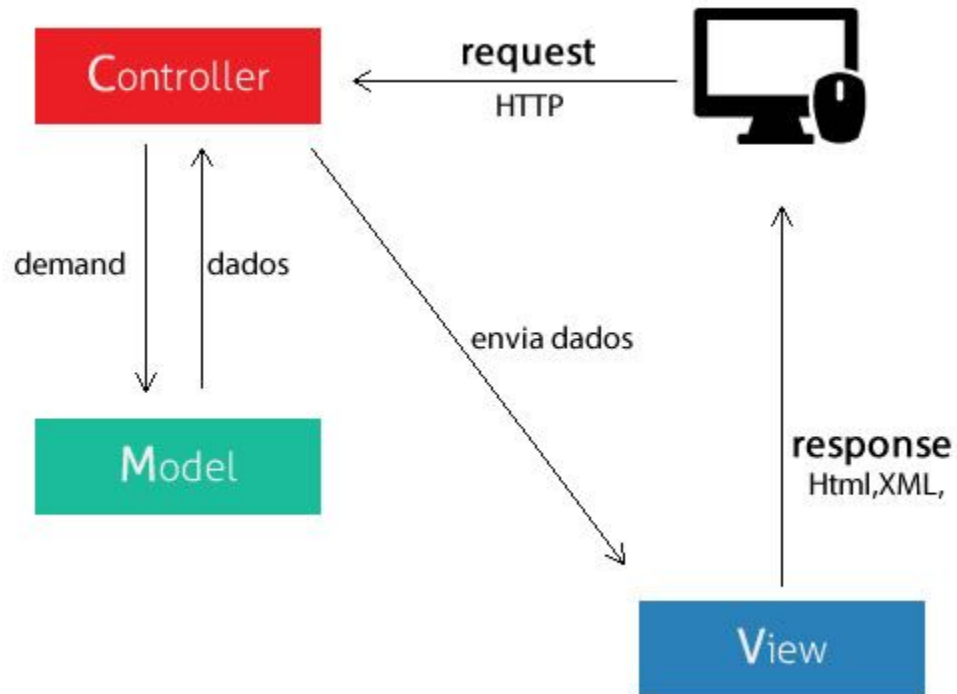
# Model-View-Controller

---

Input ➤ Processing ➤ Output

Controller ➤ Model ➤ View

# Model-View-Controller



## O diálogo das camadas

---

- **View:** Fala Controller ! O usuário acabou de pedir para acessar o Facebook !  
Pega os dados de login dele aí.
- **Controller:** Blz. Já te mando a resposta. Ai model, meu parceiro, toma esses dados de login e verifica se ele loga.
- **Model:** Os dados são válidos. Mandando a resposta de login.
- **Controller:** Blz. View, o usuário informou os dados corretos. Vou mandar pra vc os dados dele e você carrega a página de perfil.
- **View:** Vlw. Mostrando ao usuário...



# Data Access Object

---

- O padrão de projeto DAO surgiu com a necessidade de separarmos a lógica de negócios da lógica de persistência de dados.
- Este padrão permite que possamos mudar a forma de persistência sem que isso influencie em nada na lógica de negócio, além de tornar nossas classes mais legíveis.
- Classes DAO são responsáveis por trocar informações com o SGBD e fornecer operações CRUD e de pesquisas

# Data Access Object

---

- Elas devem ser capazes de buscar dados no banco e transformar esses em objetos ou lista de objetos,
- Deverão receber os objetos, converter em instruções SQL e mandar para o banco de dados.

## Exercício

---

- Faça a pagina inicial de **login.html**
- Faça um Servlet de login onde ele irá autenticar o usuário pelo nome e senha.
- Caso o nome e a senha não esteja correto redirecione para a página **erro.html** de erro informando qual foi o erro.
- Caso esteja tudo correto, crie a sessão e redirecione para a página **index.html** informando o login do usuário.

# Links importantes

---

- <https://www.caelum.com.br/apostila-java-web/servlets/>
- <https://www.caelum.com.br/apostila-java-web/mvc-model-view-controller/>
- <https://www.oracle.com/technetwork/java/dataaccessobject-138824.html>



# Dúvidas??

E-mail: [almada@crateus.ufc.br](mailto:almada@crateus.ufc.br)