

FIA/P GRADUAÇÃO

# DESENVOLVIMENTO DE SISTEMAS WEB

PROF. RAFAEL MATSUYAMA  
profrafael.matsuyama@fiap.com.br

SERVLETS

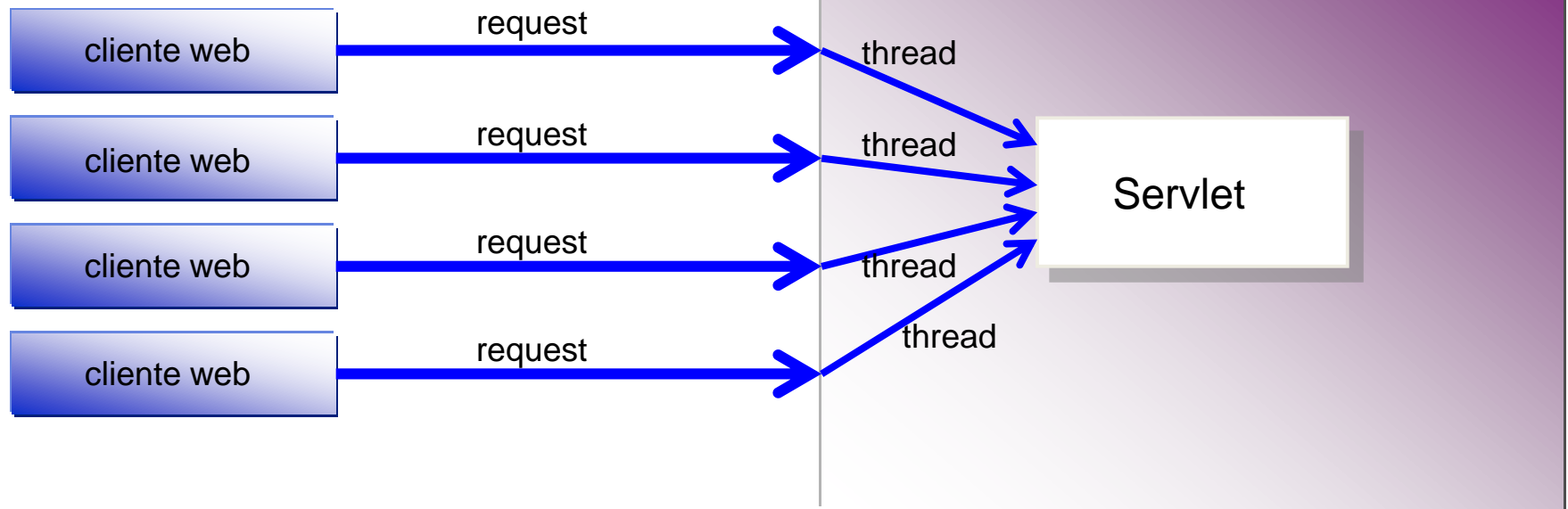
# I Introdução aos Servlets



- É uma Classe Java
- Componente Java EE
- Baseado no modelo Request/Response: seu objetivo é atender requisições
- Deve ser executado dentro de um servidor, dentro do web container
- É controlado pelo servidor



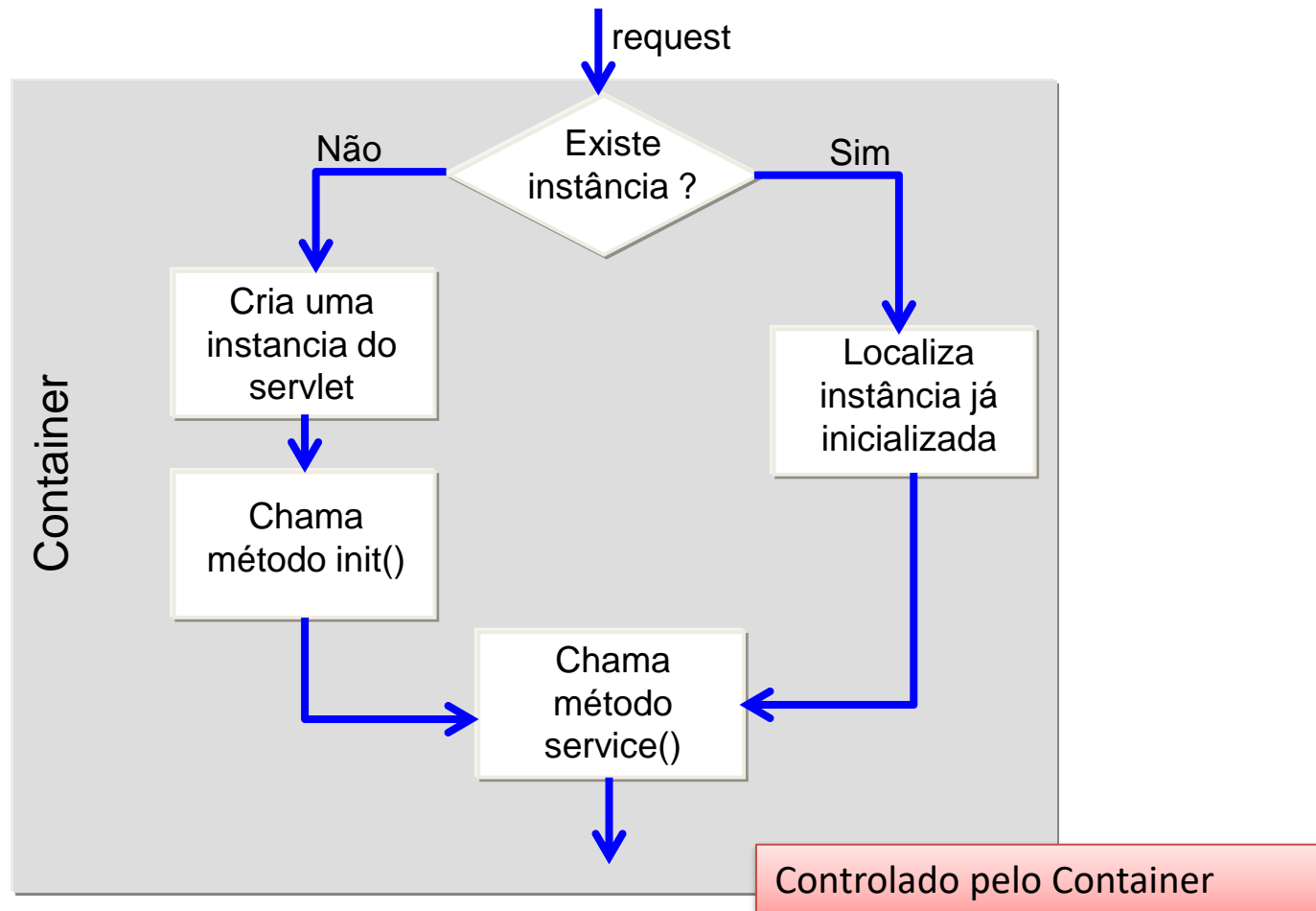
# Requisição a um Servlet



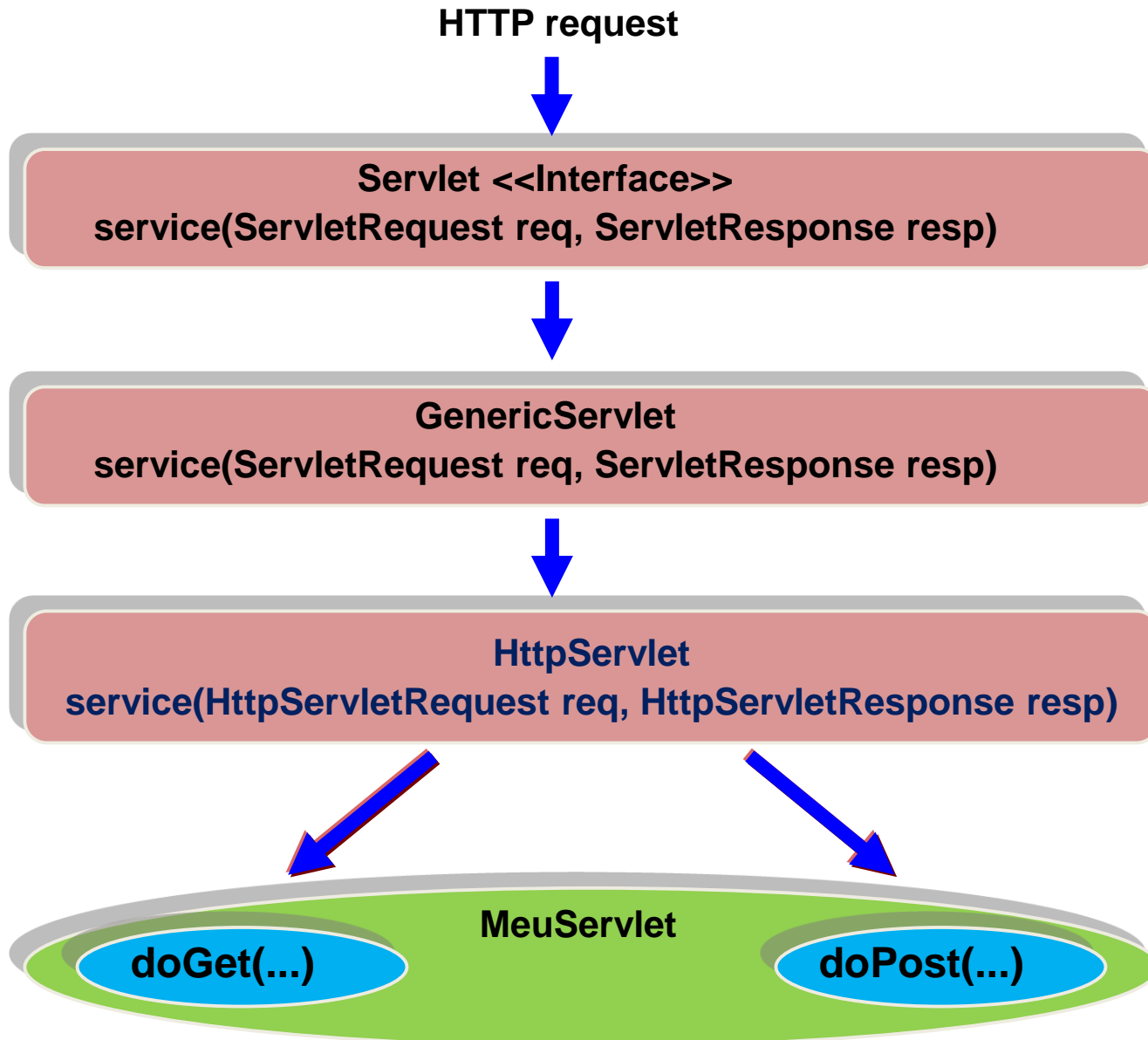
- Cada requisição a uma servlet é executada em uma thread;
- O objeto servlet é único na aplicação;

# Ciclo de vida de um Servlet

- Os servlets são instanciados pelo container, na primeira vez que são acessados;
- Após iniciados, os servlets podem atender a requisições;
- O container decide a hora de destruir os servlets; (chama o método `destroy()`)



# Sequência de eventos no HttpServlet

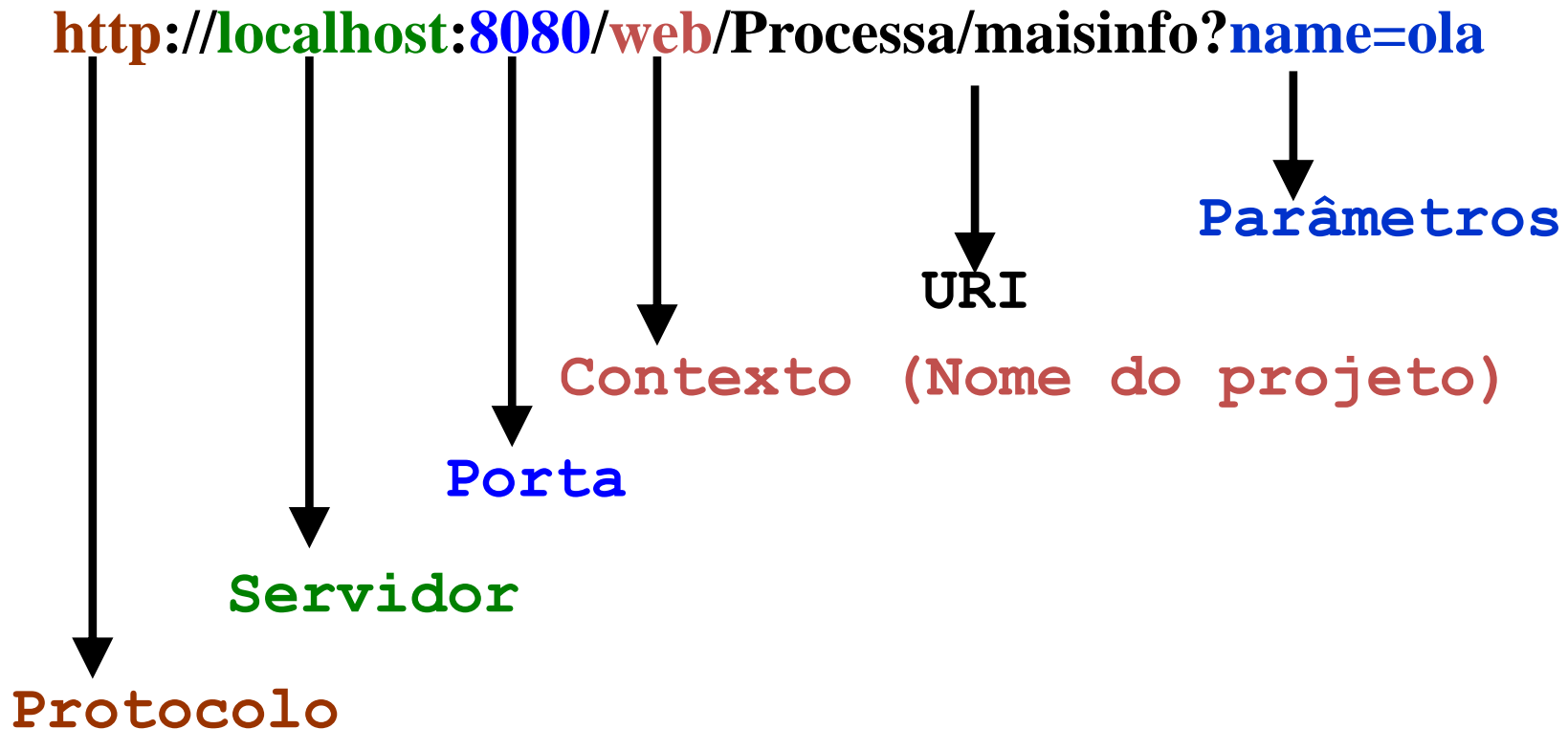


# Exemplo de Servlet

```
public class MeuServlet extends HttpServlet {  
  
    public void init(ServletConfig config) throws  
        ServletException {  
        Inicialização do Servlet  
    }  
  
    public void destroy() {  
        Destruição do Servlet  
    }  
  
    protected void doGet(HttpServletRequest request,  
        HttpServletResponse response) throws  
        ServletException, IOException {  
        Requisição tipo GET  
    }  
  
    public void doPost(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        Requisição tipo POST  
    }  
}
```

# Interface HttpServletRequest

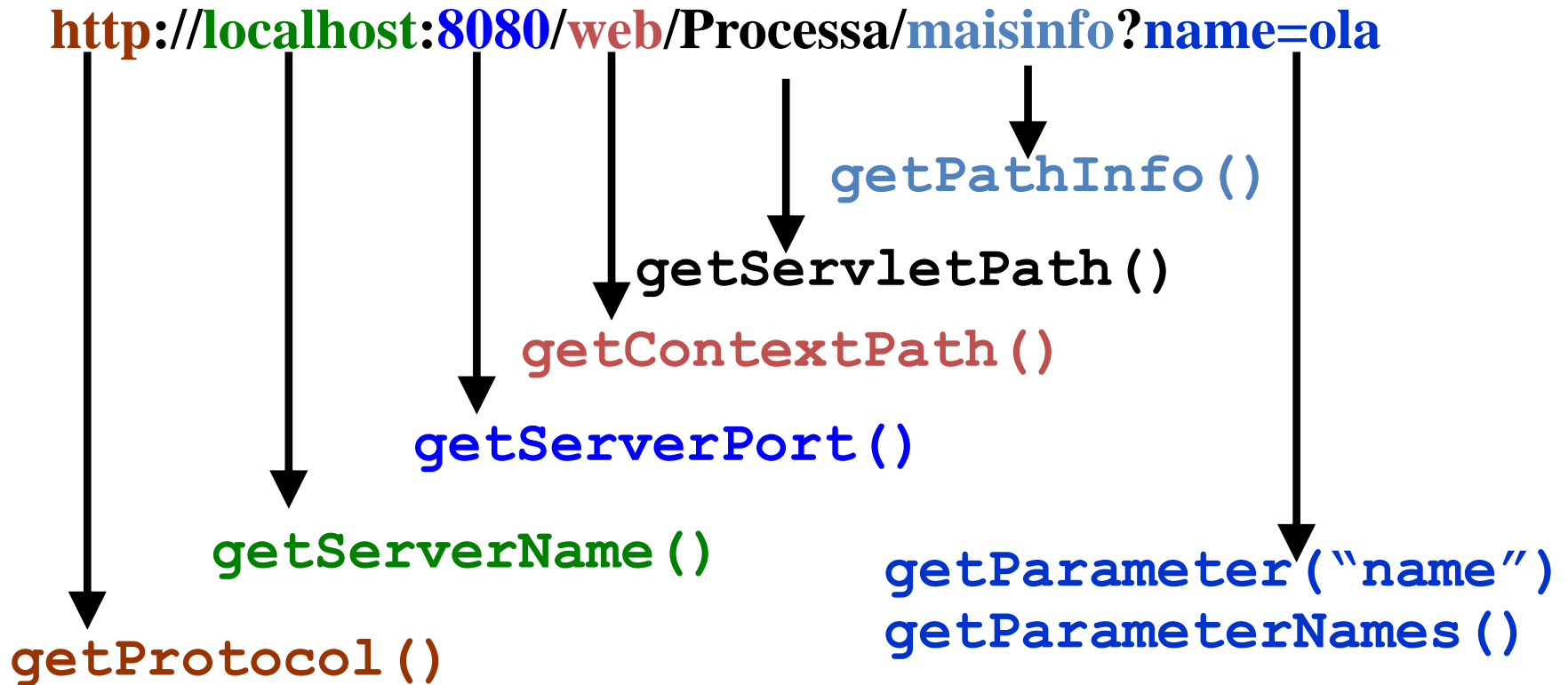
- Representa a requisição feita pelo usuário
- É possível obter dados enviados pelo browser, atributos, informações de endereço de IP, protocolo, etc..
- **Dado o request HTTP:**





# Interface HttpServletRequest

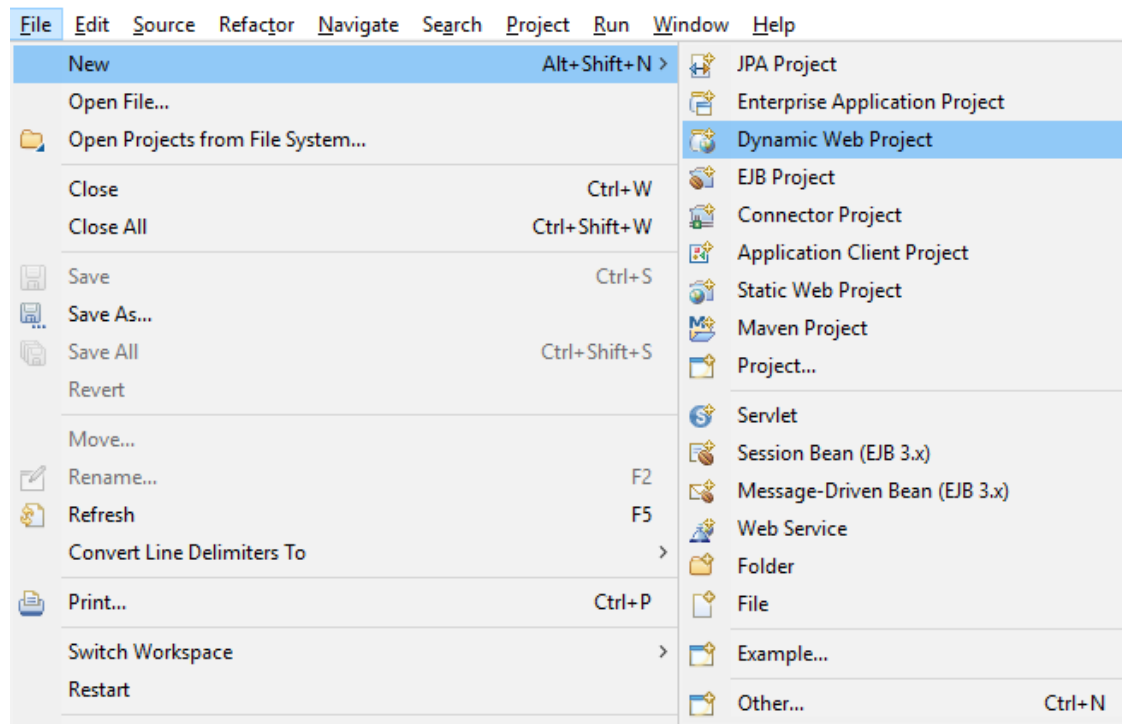
- Obter dados do request com os seguintes métodos:



# Criando um DynamicWebProject

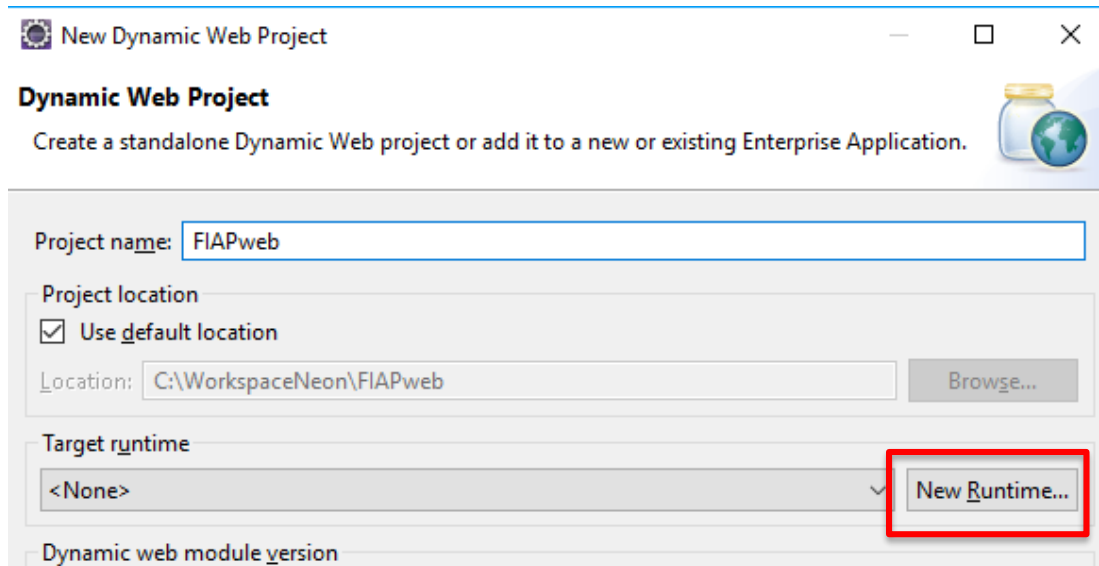
A primeira coisa que temos que fazer é criar nosso projeto, só que desta vez vamos criar um projeto do tipo DynamicWebProject

Clique no menu **File -> New -> Dynamic Web Project**.



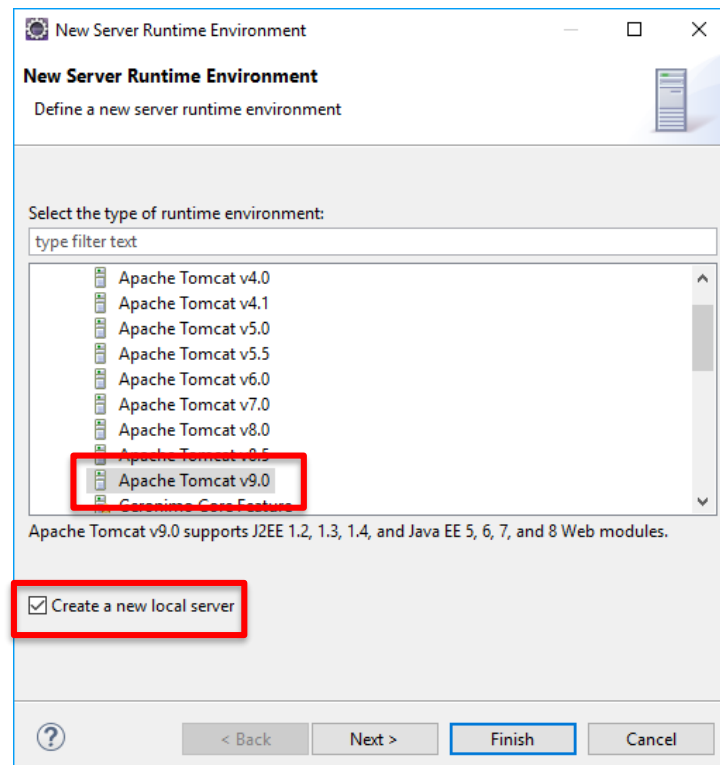
# ❑ Criando um DynamicWebProject

Dê um nome para o projeto, lembre-se de não utilizar espaços ou caracteres especiais. O próximo passo é configurar o Servidor, para isso clique em **“New Runtime..”**



# Criando um DynamicWebProject

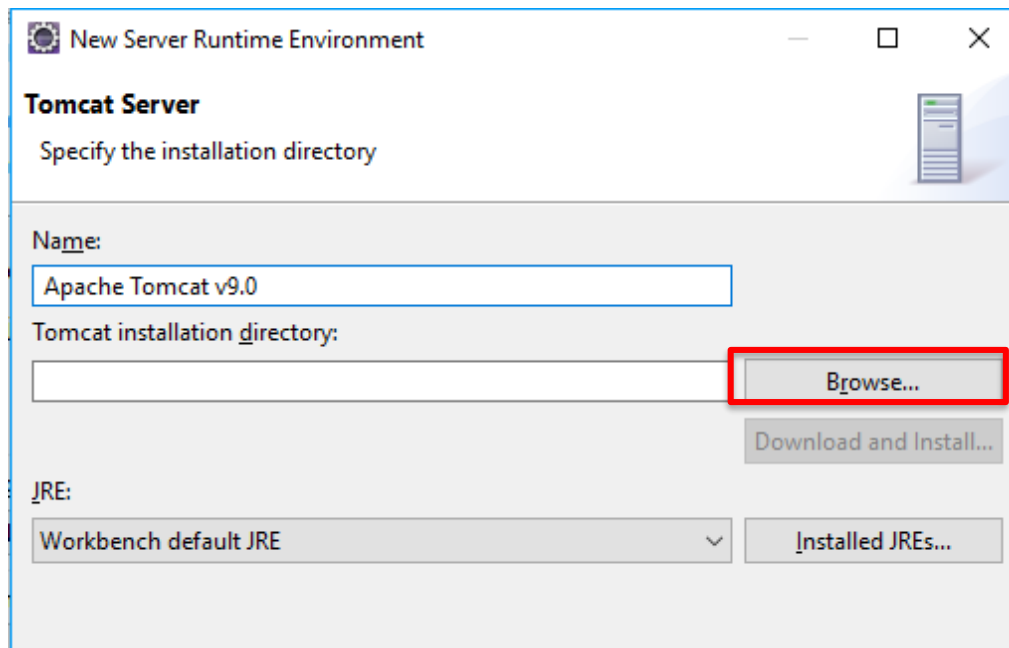
Escolha a opção **Apache Tomcat v9.0**, que está dentro do diretório Tomcat. Next. Se não tiver criado ainda, marque a opção “**Create a new local server**” ele irá criar uma pasta para o servidor no seu Workspace.



# ❑ Criando um DynamicWebProject

Agora precisamos mostrar onde o tomcat está instalado. Clique no botão “Browse..” e navegue até o diretório de instalação do tomcat.

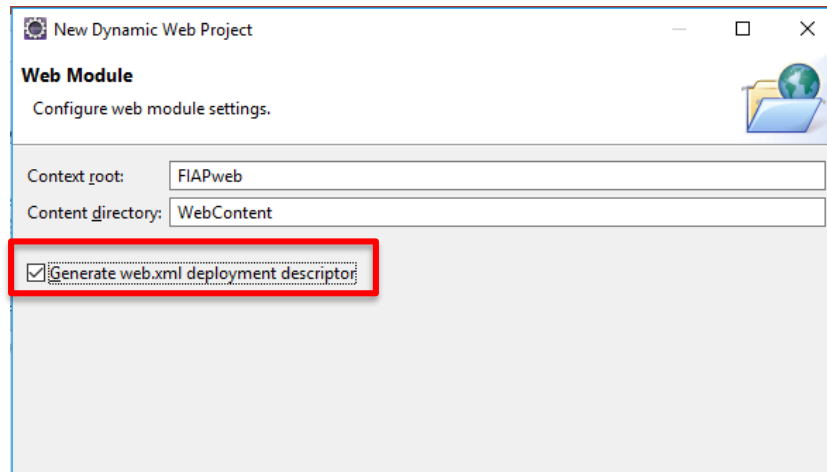
Na FIAP o tomcat está instalado em **C:\opensource\tomcat9**.



# ❖ Criando um DynamicWebProject

Depois é finalizar o procedimento de configuração do Servidor.

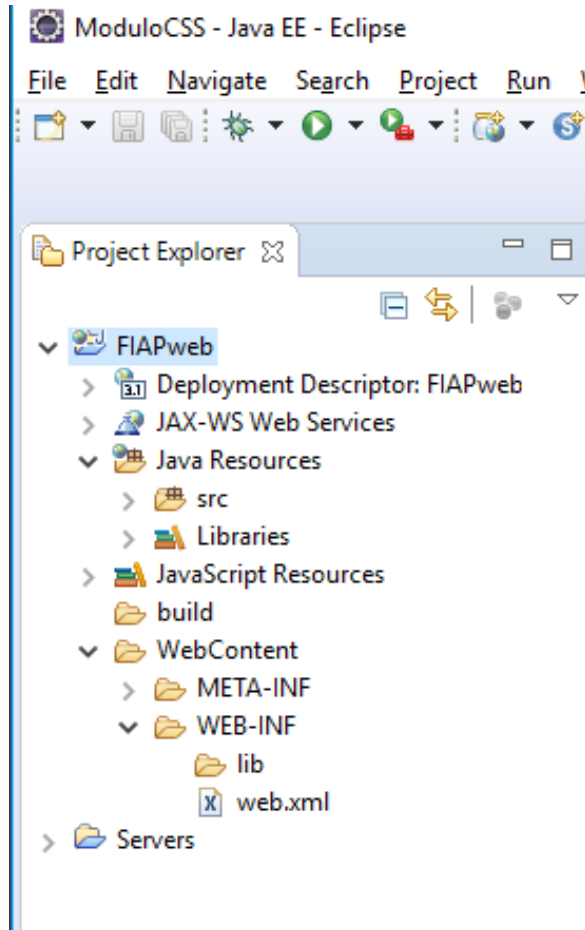
Clique em Next, Next. **Marque o checkbox** para criação do arquivo web.xml (arquivo para configuração da nossa aplicação web) e Finalize!



# Criando um DynamicWebProject

Resultado final:

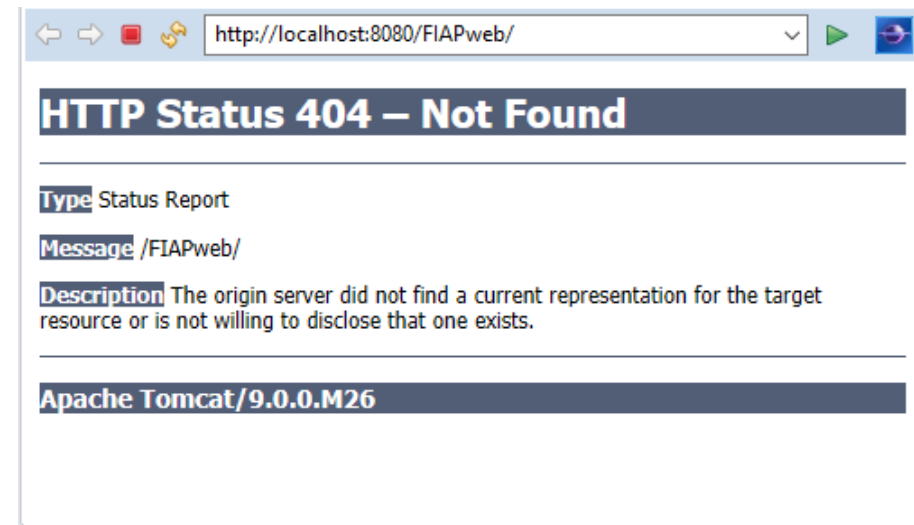
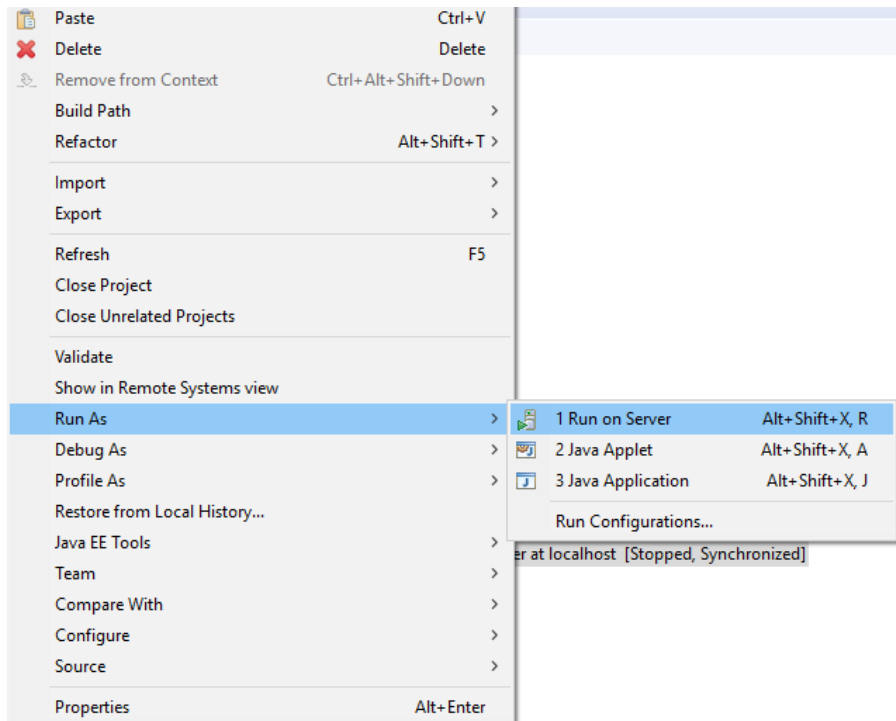
Estrutura do projeto:



- i. **Java Resources/src:** Diretório para as classes java;
- ii. **Build:** classes java compiladas;
- iii. **WebContent:** arquivos web, como html, css, javascript, imagens, etc..
- iv. **WebContent/WEB-INF:** diretório para arquivos de configuração. Não podemos colocar arquivos web (html, css, js, etc..) neste diretório, pois ele não fica visível para o usuário.
- v. **WebContent/WEB-INF/lib:** diretório para as bibliotecas java (.jar)

# Criando um DynamicWebProject

Para executar, basta selecionar o projeto, clicar com o botão direito e escolher: **Run As -> Run on Server**. Você verá que temos o Tomcat ativo, indicando através do erro 404 que não localizou a página inicial da nossa aplicação.





# Criando a primeira Servlet

- Agora no projeto vamos criar nossa primeira Servlet, vamos criá-la manualmente para entendermos como ela funciona.
- Vamos criar um arquivo “**java**” chamado **ExemploServlet** dentro do pacote **br.com.fiap.controle**:

The screenshot shows the 'New Java Class' dialog box. The 'Java Class' section has a green 'C' icon and the text 'Create a new Java class.' Below this, there are three rows for specifying the class location: 'Source folder' with the value 'FIAPweb/src', 'Package' with the value 'br.com.fiap.controle', and 'Enclosing type' which is empty. Each row has a 'Browse...' button. The 'Name' field contains 'ExemploServlet'. The 'Modifiers' section has radio buttons for 'public' (selected), 'package', 'private', and 'protected', and checkboxes for 'abstract', 'final', and 'static'. The 'Superclass' field contains 'java.lang.Object'. The 'Interfaces' section is empty with an 'Add...' button and a 'Remove' button. At the bottom, there is a partially visible question: 'Which method stubs would you like to create?'.

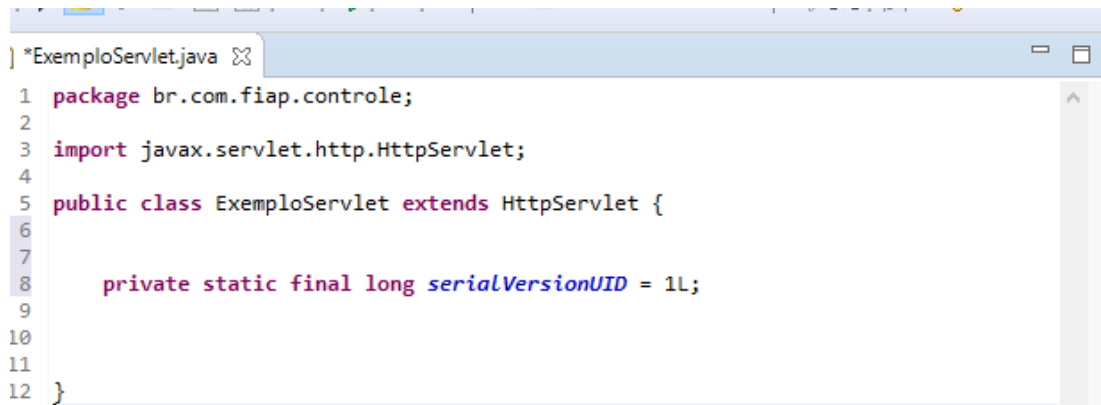
# Criando a primeira Servlet

- Essa classe java para se tornar uma servlet deve estender de **HTTPServlet**



```
ExemploServlet.java
1 package br.com.fiap.controle;
2
3 import javax.servlet.http.HttpServlet;
4
5 public class ExemploServlet extends HttpServlet {
6
7
8
9 }
10
```

- Repare que no nome da classe tem um sublinhado de alerta, o eclipse está pedindo que você implemente o serializable, com ele você declara que esta classe trabalha com tramite de dados.



```
*ExemploServlet.java
1 package br.com.fiap.controle;
2
3 import javax.servlet.http.HttpServlet;
4
5 public class ExemploServlet extends HttpServlet {
6
7
8     private static final long serialVersionUID = 1L;
9
10
11
12 }
```

# Criando a primeira Servlet

- Nas nossas servlets vamos trabalhar com dois métodos de envio o GET e o POST. Neste primeiro exemplo vamos trabalhar com o get.
- Digite “doGet” e pressione **ctrl+barra**, ele irá criar o método e importar todas as bibliotecas necessárias.

```
8     private static final long serialVersionUID = 1L;
9
10    doGet
11
12    }
13
```

```
1 package br.com.fiap.controle;
2
3 import java.io.IOException;
4
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 ExemploServlet extends HttpServlet {
11
12     private static final long serialVersionUID = 1L;
13
14     @Override
15     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
16         throws ServletException, IOException {
17         // TODO Auto-generated method stub
18         super.doGet(req, resp);
19     }
20 }
21
22 }
```

# Criando a primeira Servlet

- Agora vamos configurar nosso arquivo **web.xml**, ele irá mapear e direcionar para nossa servlet sempre que ela for chamada. Abra o arquivo web.xml que está dentro da pasta WEB-INF.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" >
3   <display-name>FIAPweb</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12 </web-app>
```

# Criando a primeira Servlet

- Logo após o fechamento da TAG **welcome-file-list**, vamos configurar nossa servlet da seguinte forma:

Servlet que irá processar o request

```
<servlet>
  <servlet-name>ExemploServlet</servlet-name>
  <servlet-class>br.com.fiap.controle.ExemploServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name> ExemploServlet </servlet-name>
  <url-pattern>/ExemploServlet</url-pattern>
</servlet-mapping>
```

<http://localhost:8080/FIAPWeb/ExemploServlet>

# Criando a primeira Servlet

- Vai ficar assim:

```
9      <welcome-file>default.htm</welcome-file>
10     <welcome-file>default.jsp</welcome-file>
11 </welcome-file-list>
12 <servlet>
13     <servlet-name>ExemploServlet</servlet-name>
14     <servlet-class>br.com.fiap.controle.ExemploServlet</servlet-class>
15 </servlet>
16 <servlet-mapping>
17     <servlet-name>ExemploServlet</servlet-name>
18     <url-pattern>/ExemploServlet</url-pattern>
19 </servlet-mapping>
20 </web-app>
21
```

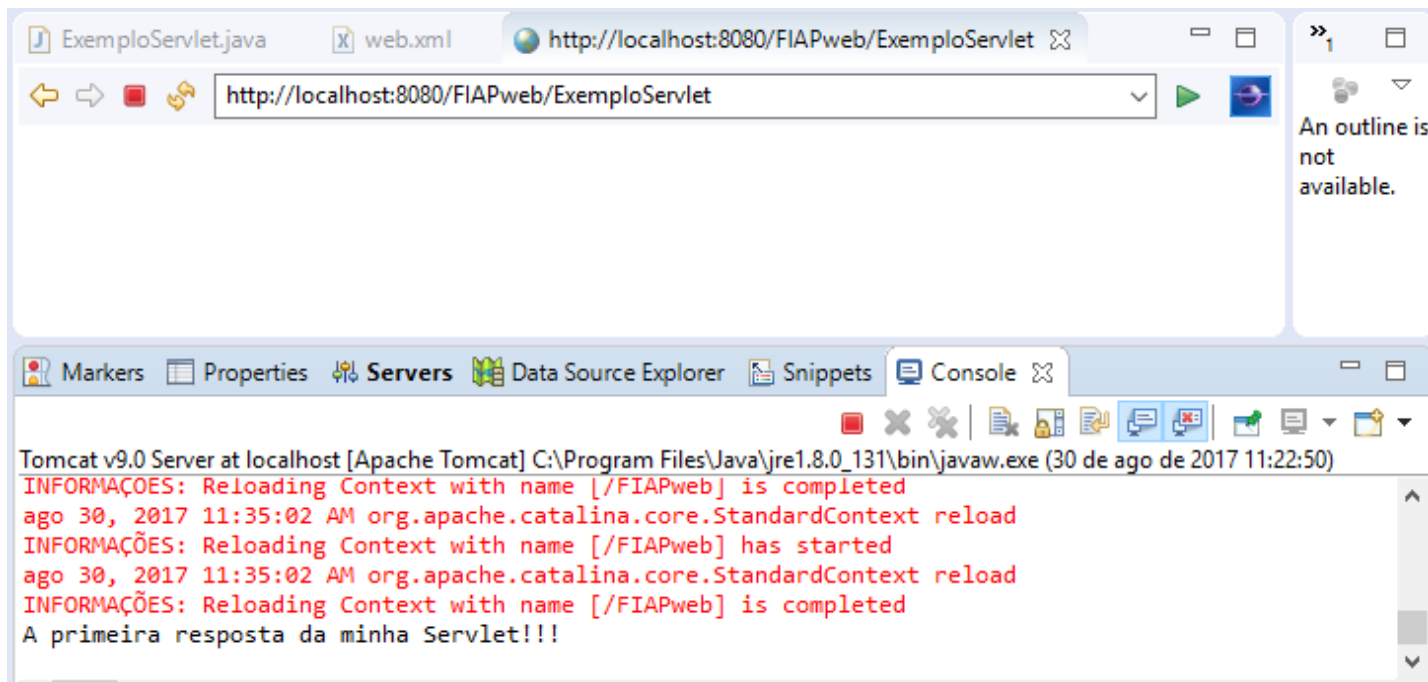
# Criando a primeira Servlet

- Voltando para a nossa Servlet, vamos criar uma saída no console para vermos o seu funcionamento.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

    System.out.println("A primeira resposta da minha Servlet!!!");
}
```

- Agora vamos atualizar nossa aplicação e incluir o endereço da nossa servlet no navegador



# Criando a primeira Servlet

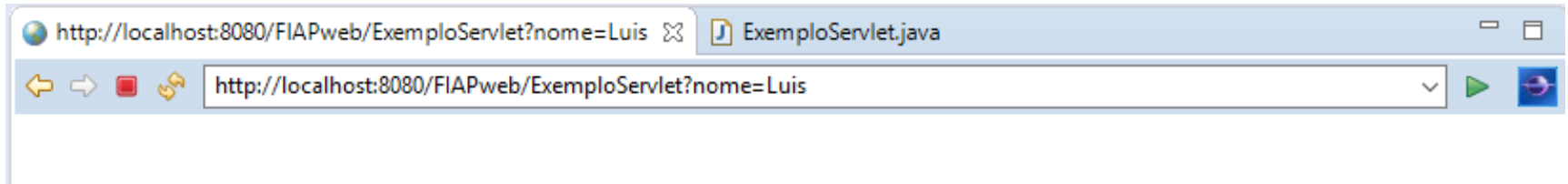
- A servlet recebe informações do usuário através de parâmetros, com o nome do parâmetro podemos resgatar as informações dentro da servlet e utilizar em nossos processos. O método `getParameter("name")` é usado para pegarmos o valor na requisição (request).

```
15 @Override
16 protected void doGet(HttpServletRequest req, HttpServletResponse resp)
17     throws ServletException, IOException {
18
19     System.out.println("A primeira resposta da minha Servlet!!!");
20
21
22     String nome = req.getParameter("nome");
23
24     System.out.println(nome);
25 }
26 }
```

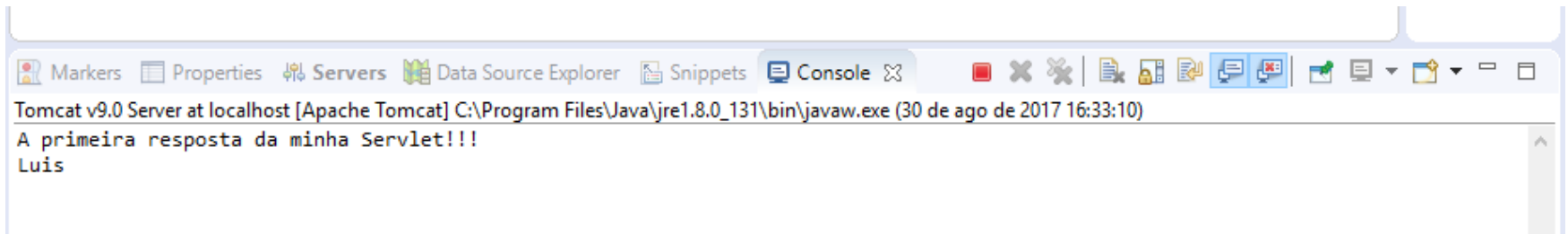


# Criando a primeira Servlet

- Como estamos trabalhando no método GET, podemos testar passando o valor do campo nome direto pelo Browser, assim:



- Quando atualizamos a página, teremos o resultado no nosso console:



# Criando a primeira Servlet

- Agora vamos receber estes valores de um formulário e mostrar o resultado na tela ou invés do console. Crie um arquivo html chamado index e adicione o formulário abaixo:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Página de Cadastro</title>
6 </head>
7 <body>
8   <form action="ExemploServlet" method="get">
9     <label>Nome:</label>
10    <input type="text" name="nome"><br>
11    <label>Email:</label>
12    <input type="email" name="email"><br>
13    <label>Idade:</label>
14    <input type="number" name="idade"><br>
15    <input type="submit" value="Cadastrar">
16  </form>
17 </body>
18 </html>
```

- É muito importante colocar corretamente os valores da action **"URL da Servlet"** e o método de envio, no nosso caso **GET**.

# Criando a primeira Servlet

- Na nossa servlet vamos usar o PrintWriter para mandarmos nossa resposta na janela do navegador:

```
9 import javax.servlet.http.HttpServletResponse;
10
11 public class ExemploServlet extends HttpServlet {
12
13
14     private static final long serialVersionUID = 1L;
15
16     @Override
17     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
18         throws ServletException, IOException {
19
20         String nome = req.getParameter("nome");
21         String email = req.getParameter("email");
22         int idade = Integer.parseInt(req.getParameter("idade"));
23
24         PrintWriter saida = resp.getWriter();
25
26         saida.print("<html><body>");
27         saida.print("<h1>Aluno Cadastrado</h1>");
28         saida.print("<p>Nome: "+nome+"</p>");
29         saida.print("<p>Email: "+email+"</p>");
30         saida.print("<p>Idade: "+idade+"</p>");
31         saida.print("</body></html>");
32     }
33 }
```

- Crie um projeto dinâmico que chamado Pizzaria, onde o cliente informando os dados em formulário (html):
- **Cliente:**
  - - Nome;
  - -Endereço;
  - - Telefone;
- **Pedido:**
  - - Tipo de Pizza;
  - - Valor
  - - Quantidade;
- A servlet deve gerar uma resposta na tela de forma organizada com o valor total do pedido.
-

- Crie um projeto dinâmico que chamado Clinica, onde o cliente vai solicitar um agendamento de consulta (html) com os dados:
- **Paciente:**
  - - Nome;
  - -Endereço;
  - - Telefone;
  - - Plano;
- **Consulta:**
  - - Especialidade;
  - - Médico;
  - - Dia;
  - - Horário.
- A servlet deve gerar uma resposta na tela de forma organizada.
- **Obs. Na servlet manipule as datas usando Strings.**

Copyright © 2018 Prof. Rafael Matsuyama / Prof. Luís Carlos de S Silva

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).