

Fortaleza - CE  
Dezembro de 2022

MANUAL DO INICIANTE IV

JAVA SERVLET

Sumário

[ECLIPSE EE 3](#_Toc122034599)

[Servidor 3](#_Toc122034600)

[Tomcat 3](#_Toc122034601)

[CRIANDO UM PROJETO 5](#_Toc122034602)

[PROJETO WEB 5](#_Toc122034603)

[ADICIONADO O PROJETO AO SERVIDOR 7](#_Toc122034604)

# ECLIPSE EE

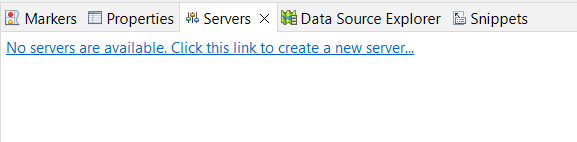
Para desenvolvimento de uma aplicação web é utilizado o eclipse EE, pois ele contém as ferramentas voltadas para esse desenvolvimento.

## SERVIDOR

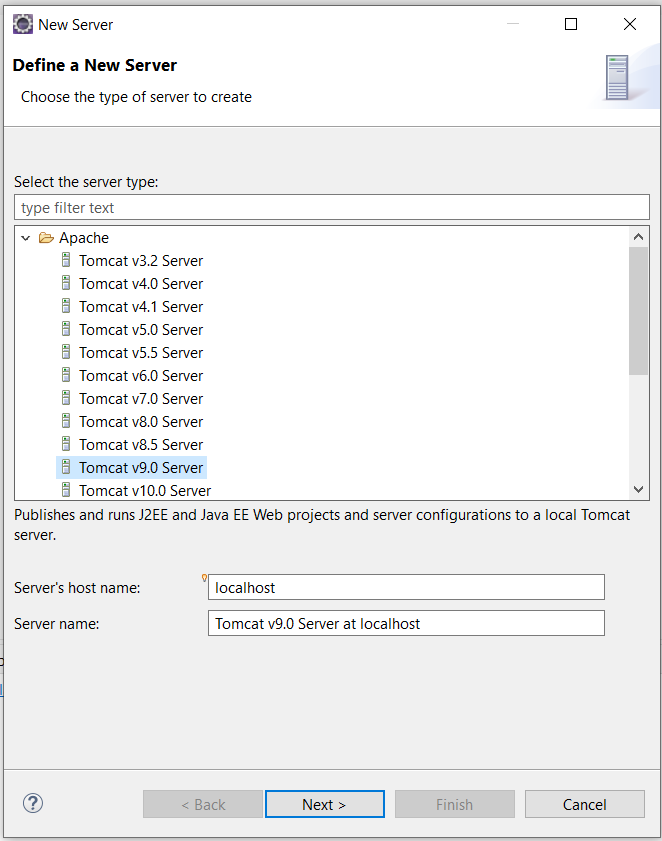
Os servidores é a parte responsável por rodar o código java.

### Servidor Apache Tomcat

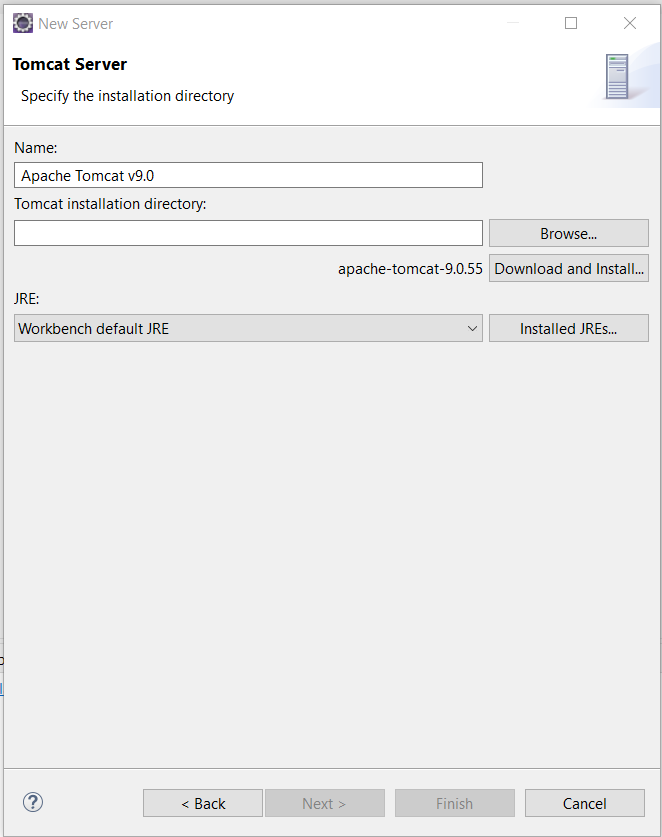
Adicionado o Tomcat no Eclipse EE



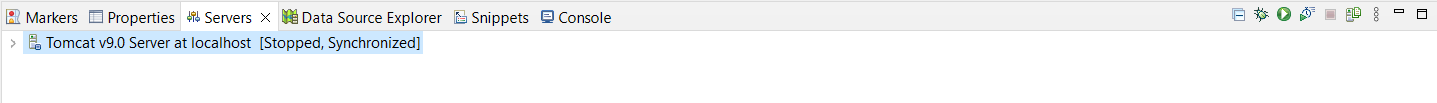
Essa ação vai abri um caixa para que seja selecionado o servidor e sua versão, no **Server’s host name** é definido o nome que vai chamar o servidor no navegador, por padrão é utilizado localhost.



Próxima etapa é necessário selecionar o arquivo do servidor e o java que será vinculado a ele. O servidor Tomcat pode ser baixado no site https://tomcat.apache.org/, ele vem em formato zip e após extrair basta adicionar o caminho da pasta no **Tomcat installation directory** e na opção **JRE** é definido o java que vai ser rodado no servidor.



A última opção antes de instalar o servidor é para caso exista um projeto já na workspace pode ser adicionado antes de configurar o servidor, caso contrário só finalizar, após vincular, o Tomcat vai passar a ser exibido na aba **Servers**, e os botões a direita vai rodar o servidor.



Após o servidor iniciar basta acessa através do host definido e a porta que foi iniciada, exemplo <http://localhost:8080>.

### Portas

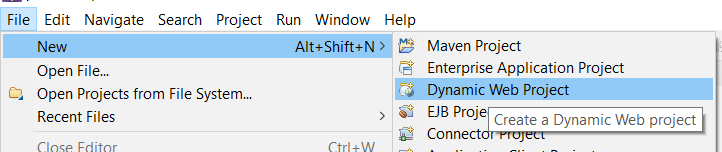
Dentro do servidor na pasta **conf** arquivo **server.xml**, alterar a porta para 80, porta padrão, assim não sendo necessário mencionar porta no navegador, já que a porta 80 é a porta padrão.

A parti do Tomcat 10, não é utilizado javax.

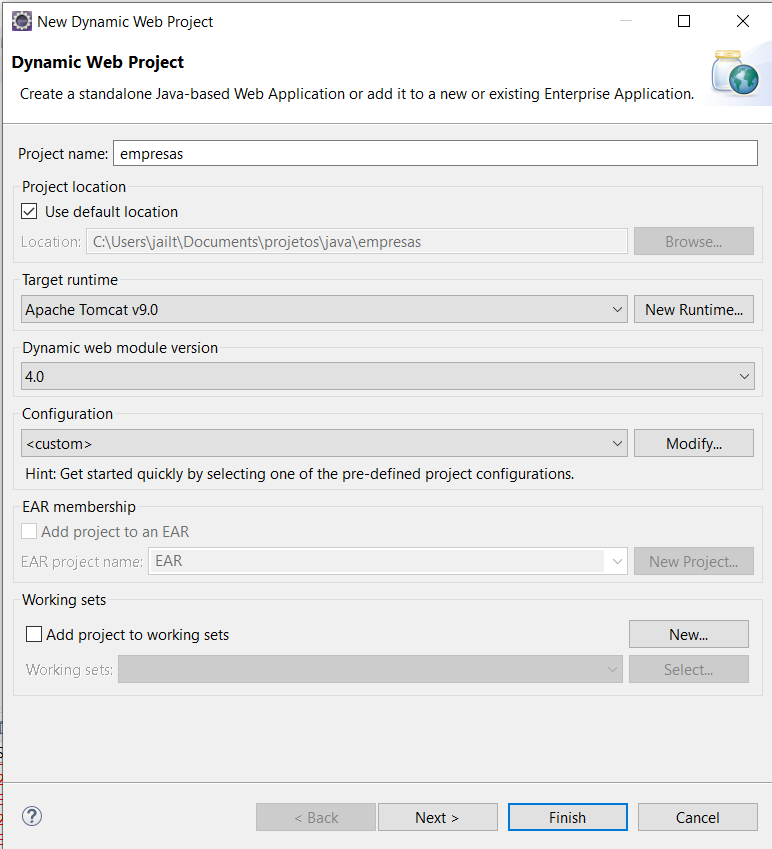
# CRIANDO UM PROJETO

## PROJETO WEB

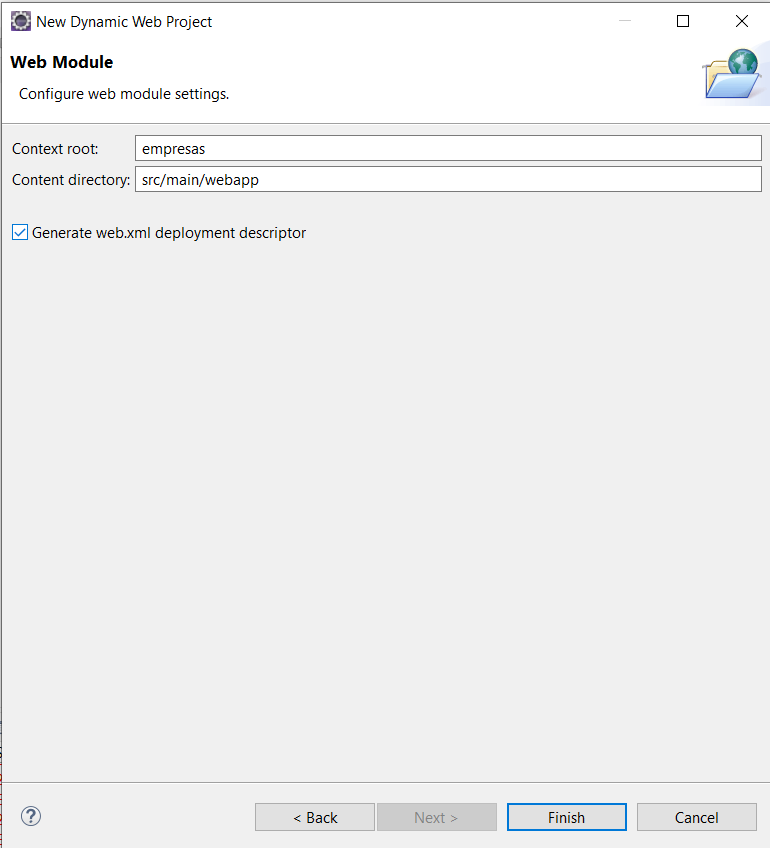
A opção abaixo é utilizada para criar um projeto web.



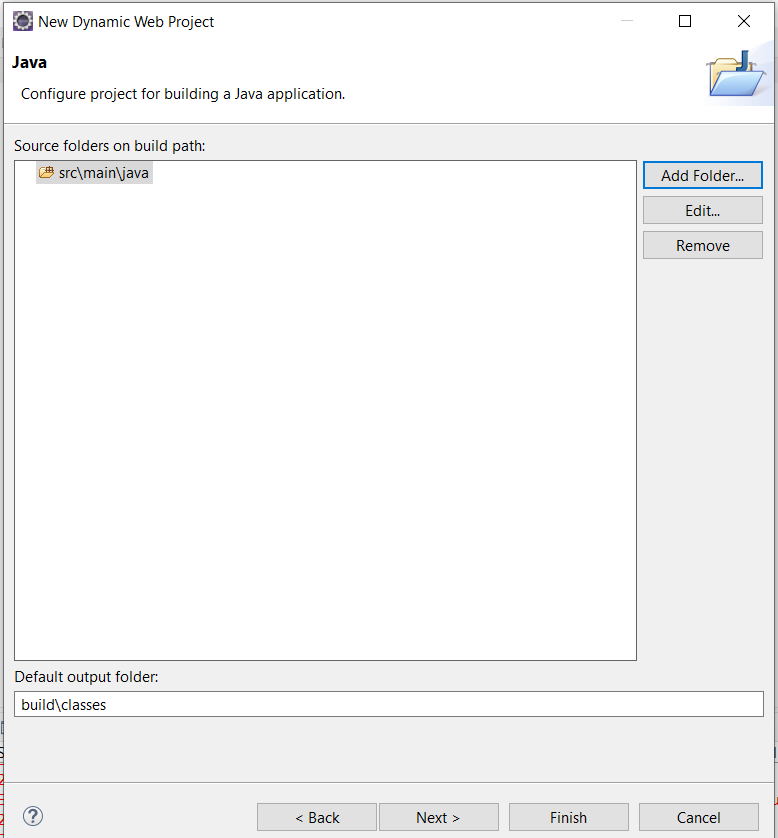
Na opção **Target runtime** selecionar a versão e o servidor que vai rodar o projeto. Em casos de outras configurações para o projeto, como JSF, é necessário utilizar a opção **Configuration**.



O **Context root** é o nome no qual o servidor vai chamar a aplicação, exemplo com nome utilizado “empresas” e servidor <http://localhost:8080>, para chamar esse projeto a URL vai ser <http://localhost:8080/empresas/>. **Content directory** é onde as páginas web vão está, a pasta padrão pode variar de acordo com a versão nesse caso ficaria dentro da **webapp**, porém em outras versões pode ser **webContent**.

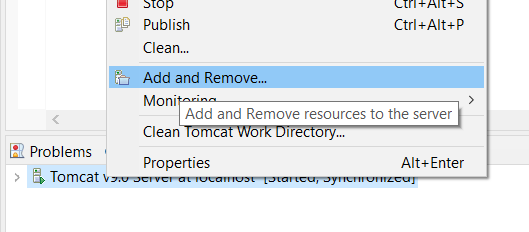


**Default output folder** é pasta que vai ser feito o build da aplicação.

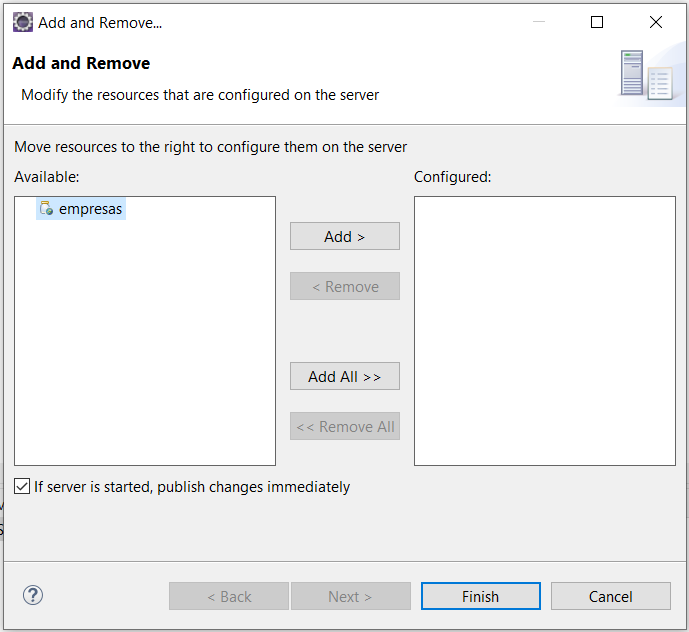


## ADICIONADO O PROJETO AO SERVIDOR

Para rodar um projeto é necessário vincular ele ao servidor, clicando com botão direito em cima do servidor selecionar a opção **Add and Remove** ou arrastando a pasta raiz do projeto até o servidor.



Seleciona o projeto em **Available** e clicar em adicionar para adicionar o projeto, para remover selecionar o projeto em **Configured** e clicar em **Remove**.



# SERVLET

## MAPEANDO CLASSE

A classe que recebe as anotações servlet elas podem ser chamadas via HTTP, além disse tem disponível diversos recursos para gerenciar a requisição e resposta.

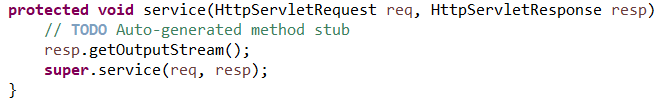
### Definido uma classe Servlet

É necessário estender de HttpServlet.   


É preciso mapear a classe com anotação para que seja identificada na requisição HTTP.



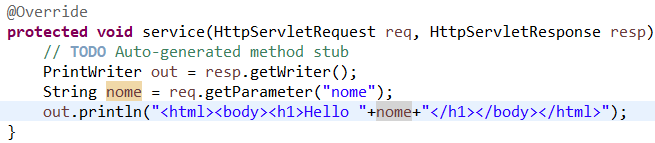
Uma classe Servlet precisa ter pelo menos um método que gerencie a requisição HTTP, a método “comum” é **service** que não tem restrição de tipo de requisição.



### Métodos de requisição

Uma classe Servlet pode ter métodos especifico para atender o tipo de requisição, POST, GET, DELETE e etc. esses métodos são equivalentes ao método service porém só aceita o tipo especifico de requisição. Exemplo o método **doPost** só aceita requisição post, método **doGet** só aceita requisição get e assim por diante.

Para capturar os parâmetros que vem via request utilizar o método “.getParameter(“nomeDoParametro”)”.



# JSP

JSP (Java Servlet Face) é uma página que é capaz de interpretar código JAVA e código HTML, funcionando de maneira similar ao PHP. Na página JSP para escrever código JAVA é necessário ser escrito entre **<% %>**.

## IMPRIMINDO CÓDIGO NA PÁGINA

Para imprimi uma variável ou resultado de um código na tela é necessário dentro das tags <% %> utilizar a função **out.println(var)**;



Essa função ela pode ser substituída por <%**= %>** onde o resultado interno do código vai ser impresso.



## EXPRESSION LANGUAGE

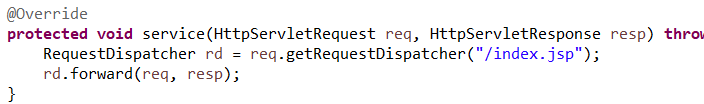
É um interpretador de código utilizado dentro do JSP que faz basicamente a mesma coisa do **<%= %>**, para utilizar ele é utilizado o código **${}** onde o código vai ficando na parte interna das chaves.

## INTEGRANDO O JSP COM O SERVLET

No método de um classe Servlet um dos parâmetro de entrada é HttpServletRequest, ele é responsável por integrar a página JSP na chamada do Servlet, faz isso através da funções (getRequestDispatcher e forward).

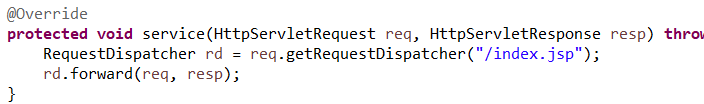
### getRequestDispatcher

Essa função é responsável por pegar a página JSP para integrar com o Servlet.



### forward

Essa função é responsável por mostrar o conteúdo da página quando foi requisitado o Servlet e o método.



### setAttribute

Quando é necessário passar parâmetro para dentro da página JSP é utilizado a essa função, com o **nome** do atributo e seu **valor**.



## DEPENDÊNCIAS JSP

Para utilizar classes e outras dependências dentro de uma página JSP é necessário declarar ela na página JSP.

**<%@ page import=”dependência,dependencia” %>**

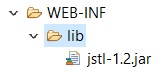
As declarações das dependências têm que ser declaradas assim como é feita dentro das classes, exemplo: br.com.empresas.operadores, e devem ser paradas por virgulas quando existir mais de uma.

## ESCONDENDO JSP

Para que os JSP não sejam mais possíveis o acesso direto é necessário colocar os arquivos dentro da pasta WEB-INF por conversão é necessário criar uma pasta view onde vai conter os arquivos JSP.

# JSTL

O JSTL é biblioteca que abstrai o código JAVA para ficar similar ao HTML. Para instalar em um projeto web através de um arquivo .jar basta colocar na pasta **WEB-INF** > **LIB** o arquivo .jar do JSTL.



Dentro da página JSP é necessário mencionar a biblioteca e definir um **apelido** para que seja chamado o JSTL.



Nessa biblioteca temos quatro tipos de sub bibliotecas: core, fmt, ~~sql~~ e ~~xml~~

## FOR EACH

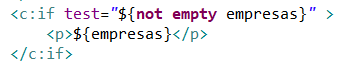
Para definir um loop utilizando JSTL.



O prefixo **c:** no início é o apelido dado na chamada da biblioteca JSTL e sinaliza que essa tag é uma JSTL. Durante teste foi visto que o loop só acontece em strings separadas por virgulas, ou objetos, ListArray com a utilização da Expression Languagem não funcionou.

## CONDICIONAL

Para cada abstração existi uma tag diferente, com condicionais e outras, consultar documentação.



## IMPORT

Para importar um componente dentro de uma página JSP utilizando JSTL.  


## FMT

Sub biblioteca fmt serve para formatar números, data e etc.



**Formatando Data**

**SimpleDateFormat** sdf = new **SimpleDateFormat(“**dd/MM/yyyy**”);  
Date** data = sdf.parse**(“**stringDate**”);**

Pega uma data em string yyyy-MM-dd e retorna uma data, necessário o uso do try e catch.

Via **Servlet** o redirecionamento pode ser feito pelo parâmetro **HttpServletResponse** de uma classe **Servlet**.

response.sendRedirect**(“**login**”)**;

<http://localhost:8080/empresas/login>

# WEB.XML

## MAPEAMENTO SERVLET

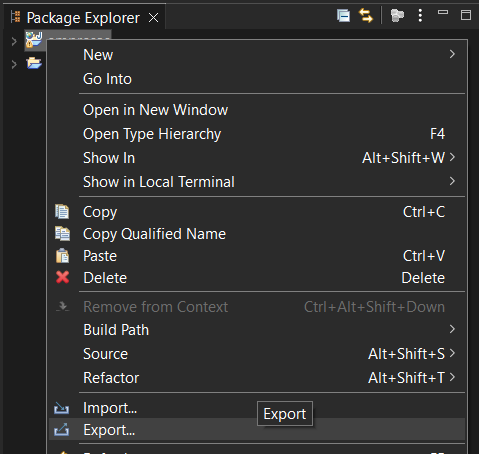
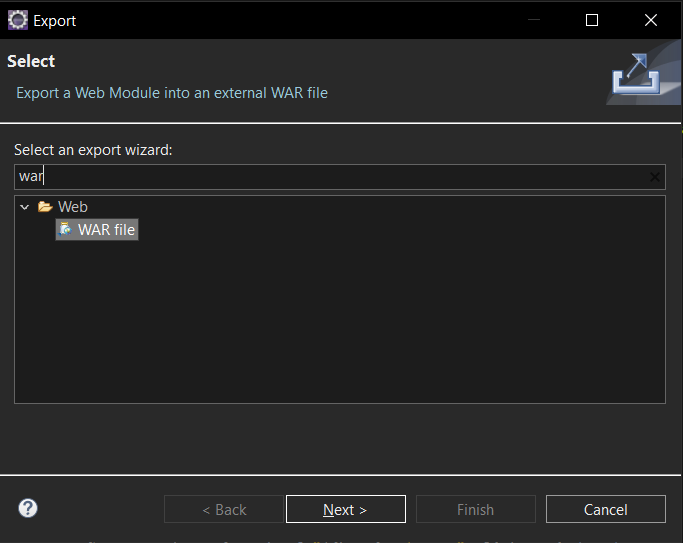
Mapeamento de classes Servlet manual via arquivo XML.

<servlet>  
<servlet-name>apelido</servlet-name>  
<servlet-class>br.com.empresas.class</servlet-class>  
<servlet>  
<servlet-mapping>  
<servlet-name>apelido</servlet-name>  
<url-pattern>/login</url-pattern>  
</servlet-mapping>

# DEPLOY

## EXPORTANDO

Os deploy dos projetos é feito no formato (**.war**) clicando com botão direito em cima do projeto e selecionando.

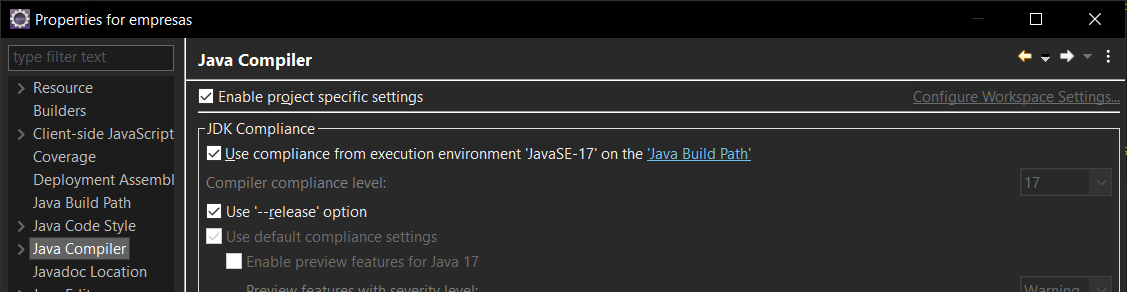
 

## EXECUTANDO .WAR

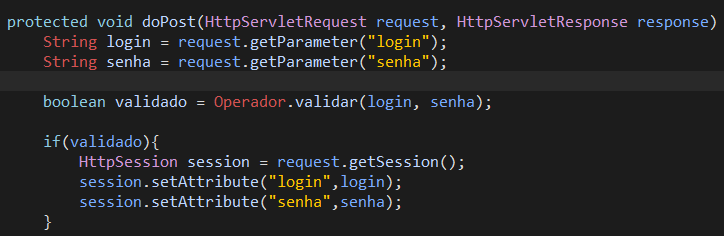
Para executar um arquivo **(.war**) dentro do Tomcat é necessário colocar o arquivo dentro da pasta **webapps**, todo servidor assim como o Tomcat existi um pasta similar para executar os deploy.

Para rodar o Tomcat manualmente, executar a bat **startup.bat** dentro da pasta do **bin** do servidor.

Antes compilar o projeto verifique se o compilador do projeto é o mesmo do servidor, clicando com botão direito em cima do projeto, **Java Compiler** > **Compiler compliace level**.



# HTTP SESSION

A classe responsável por controlar a sessão é a **HttpSession** que provem dos métodos da classe Servlet.

## LOGIN E VALIDAÇÃO

Para inserir um atributo na sessão utilizar o método **setParameter()** e para consultar atributo utiliza **getParameter()**.

## LOGOUT

Para destruir um atributo da sessão utilizar o método **removeAttribute()** para destruí a sessão inteira utilizar **invalidate()**.