

Bem-vindos!

Programação II

Mateus Elias Gündel | mateus8923@gmail.com

Competências

- Conhecer a arquitetura de sistemas de multicamada
- Empregar técnicas de programação
- Arquitetura J2EE
- JSP
- Modelos de Arquitetura
- Tomcat
- Distribuição de aplicativos

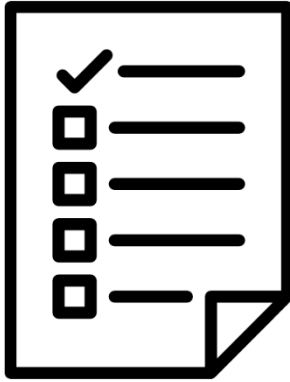
Como serão as aulas

- Dúvidas e questões sobre a última aula/exercícios
- Apresentação e discussão sobre o conteúdo da disciplina
- Desenvolvimento prático do conteúdo apresentado para melhor fixação

Datas importantes

05/abr	EAD - Exercícios valendo nota
19/abr	EAD - Exercícios valendo nota
03/mai	EAD - Exercícios valendo nota
10/mai	Definição do Trabalho
07/jun	EAD - Exercícios valendo nota
14/jun	Revisão Prova
21/jun	Prova
28/jun	Entrega/Apresentação trabalhos
05/jul	Reserva técnica (Revisão exame)
12/jul	EXAME

Avaliação



Prova

+



Trabalho

+



Exercícios

3



Dúvidas ?

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. Some nodes are highlighted with blue circles, and others with blue dots. The lines are thin and gray, creating a mesh-like structure.

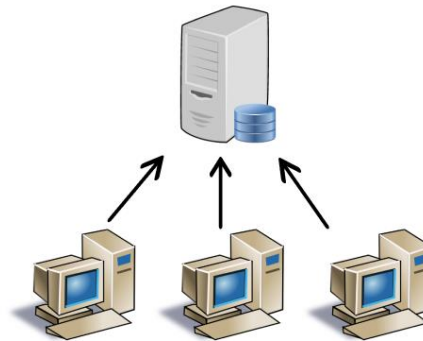
Conhecendo a turma

Quem somos ? De onde viemos ? O que fazemos ?

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It shows a cluster of nodes connected by lines, with several nodes highlighted in blue. The overall style is clean and modern, using a light gray color palette with blue accents.

Arquitetura multicamadas

- Surgiu com adoção de sistemas cliente-servidor
- Sendo o cliente a interface do usuário com a aplicação
- E o servidor possuía o bando de dados para persistência
- Serviu como impulsionador para os modelos atuais



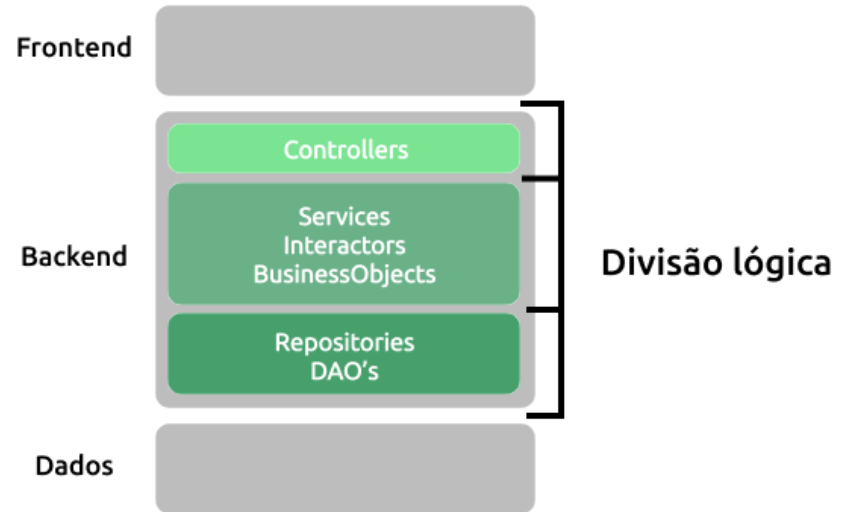
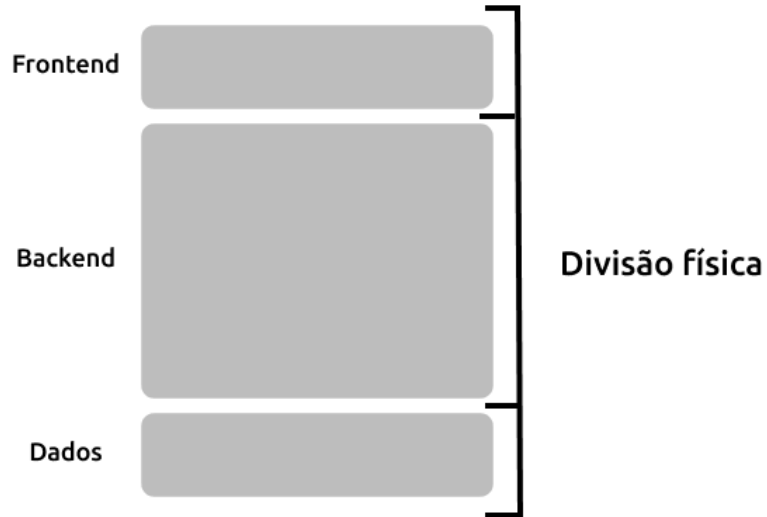
Arquitetura multicamadas

- Com a evolução das interfaces e a computação distribuída, surgiu uma questão: Onde colocar a lógica de negócio?
- Inicialmente era colocado no lado do cliente junto da aplicação, ou no banco de dados, o que normalmente gerava códigos desorganizados e repetidos.
- Assim, foi criada uma nova camada no modelo cliente-servidor, para justamente organizar as regras de negócio.
- Com a chegada da web, e a necessidade de se trabalhar com interfaces no browser, o modelo de camadas se consolidou.

Arquitetura multicamadas

- Nos dias atuais, temos soluções em rodando em browser, celular e outros diversos dispositivos
- Também, a popularização da cloud, trazendo diversas soluções
- O uso de camadas continua tendo grande valor, mas não mais tendo um número definido de camadas, e sim sendo chamada de arquitetura de N camadas.

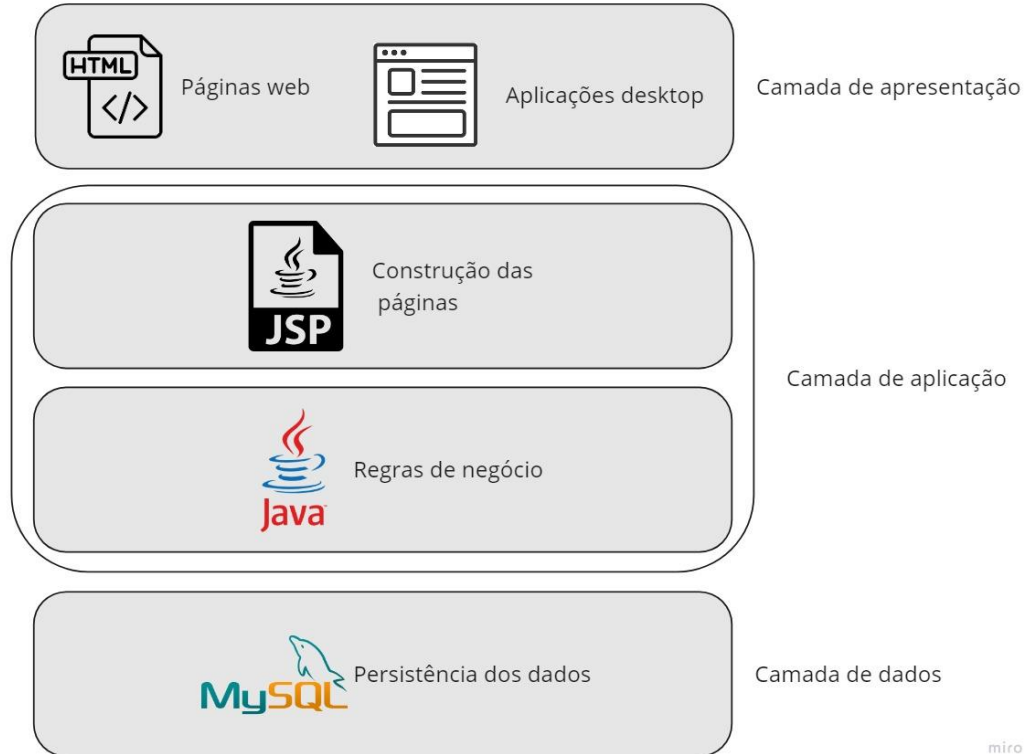
Divisões físicas e lógicas



Vantagens x Desvantagens

- Separação dos códigos, ficando cada um responsável por fazer uma coisa específica
- Permite a mudança de uma camada sem afetar a outra, desde que a forma de comunicação não seja afetada
- Cria uma hierarquia de modos de acesso, protegendo dados sensíveis
- Aumento da complexidade do projeto
- A cada camada criada, precisa-se estabelecer os modos de acesso

Exemplo de arquitetura multicamada que vamos usar



miro

Camada de apresentação

- É responsável pela apresentação, interação com o usuário e recebimento dos dados processados pela aplicação.
- Pode se dar de diferentes formas:
 - Pode ser desenvolvido para web, ou seja, para o acesso via browser, normalmente em java é utilizado o JSP.
 - Pode ser desenvolvido como aplicação desktop, instalando uma aplicação no seu computador, como por exemplo uma interface usando Java Swing ou JavaFX.
 - Pode também, ser um aplicativo de celular ou outro dispositivo.
- O que todos precisam, é uma forma de se comunicar com a camada de aplicação.

Formas de comunicação

- A principal forma de comunicação utilizada para isso, é através do protocolo http ou https.
- A camada de apresentação, toda vez que precisa de algum dado ou página, manda requisições para a camada de aplicação, e recebe de volta, as informações solicitadas.
- O protocolo Http possui alguns métodos para indicar a ação a ser realizada, como por exemplo GET, POST, PUT, DELETE, PATCH, etc.
- Por isso, é importante na hora de desenvolver, verificar como está definida a comunicação.

Camada de aplicação – Construção das páginas

- É onde as páginas são definidas, e os layout de apresentação são montados
- Será utilizado o JSP(JavaServer Pages) para podermos usar código java diretamente no HTML.

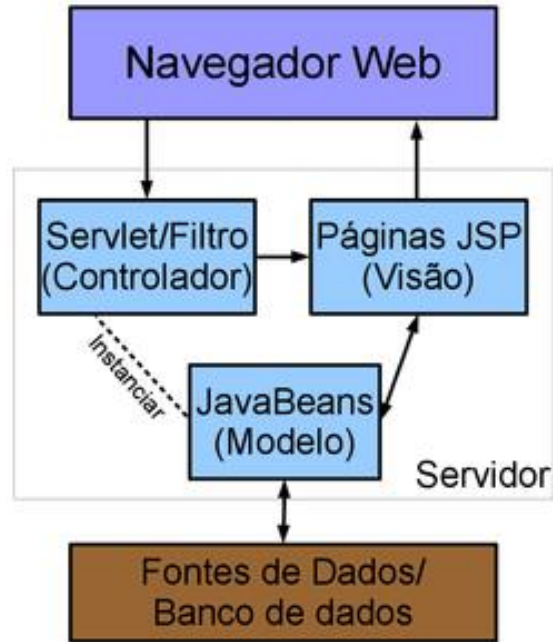
Camada de aplicação – Regras de negócio

- É a camada foco da disciplina, onde é definidas as regras de negócio, os comportamentos da aplicação, a forma como serão salvos os dados.
- Como base para desenvolvimento vamos utilizar o java, juntamente com o servidor Tomcat.
- Assim, desenvolveremos servlets, que irão processar as requisições http enviadas pela camada de apresentação.
- Nos servlets também irão ser definidos as especificações de como serão processadas cada uma destas requisições.

Camada de dados

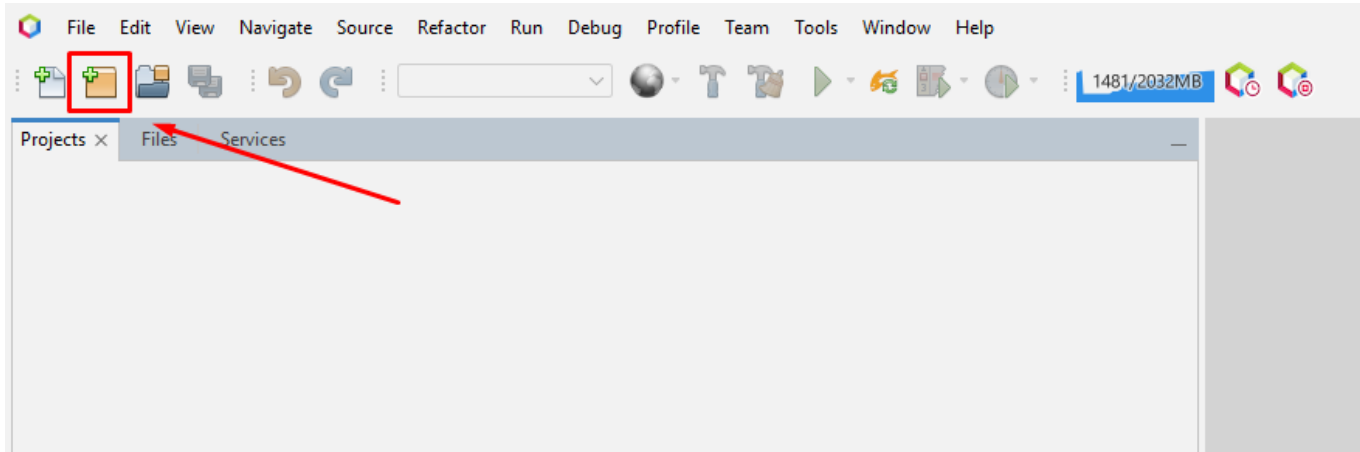
- É a camada onde é realizado o acesso aos dados e os nossos dados da aplicação serão organizados e armazenados.
- Como base para armazenar os dados, iremos utilizar o MySQL.
- Para realizar o acesso, consulta e persistência dos dados iremos utilizar um conector JDBC na camada de aplicação.

Estrutura final desejada



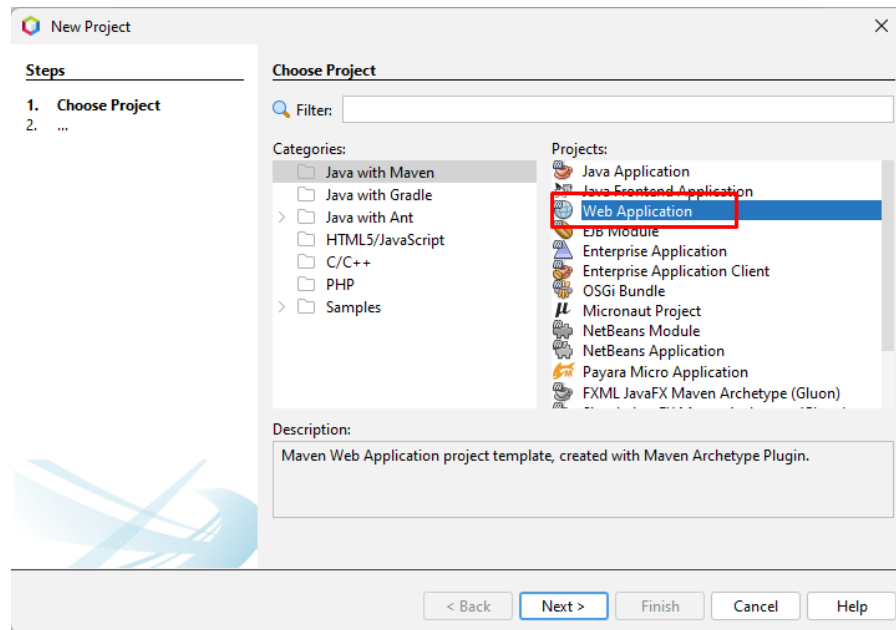
Como criar um Servlet – Configurando ambiente

Primeiro, abra o netbeans e crie um projeto



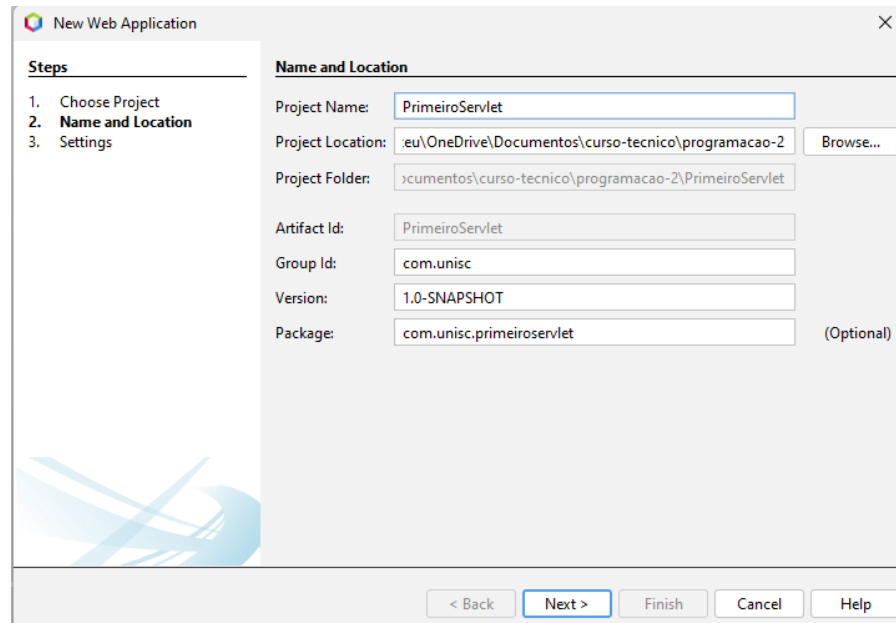
Como criar um Servlet – Configurando ambiente

Selecione Java with Maven, e em projects, Web Application



Como criar um Servlet – Configurando ambiente

Preencha o Project Name e o Group Id se desejar



New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Settings

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

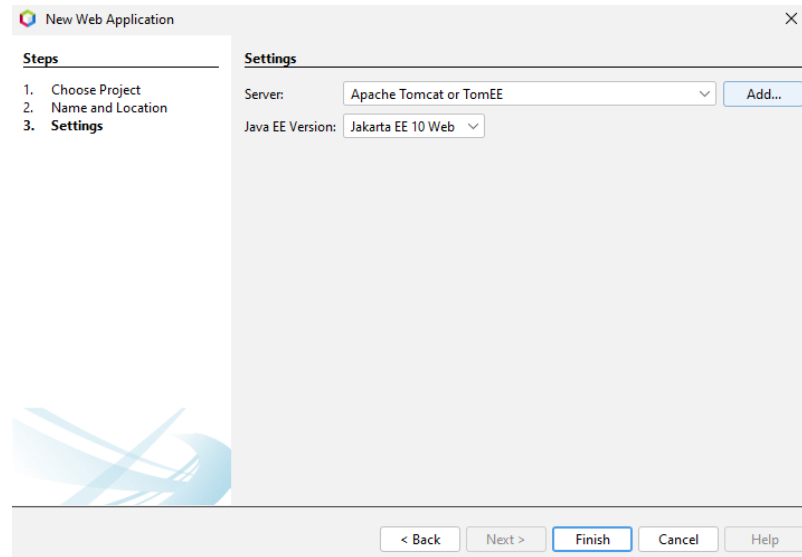
Group Id:

Version:

Package: (Optional)

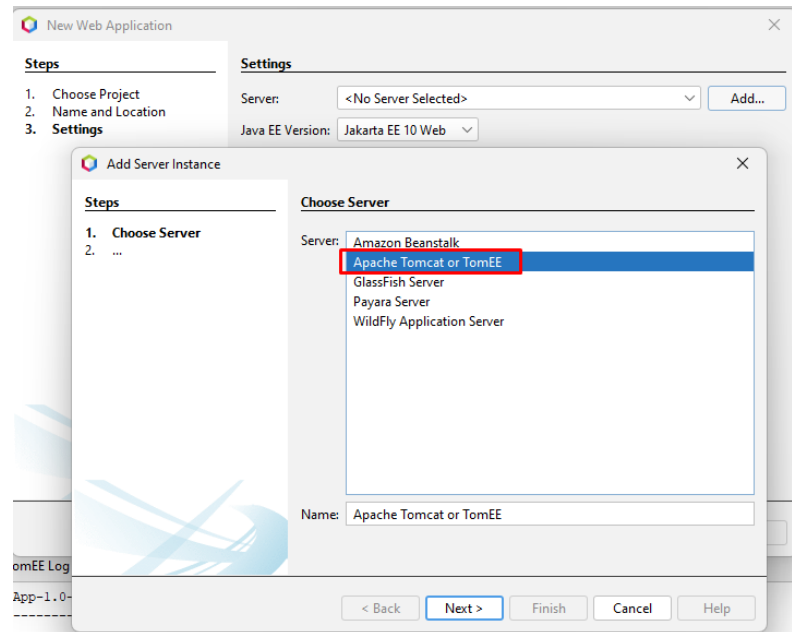
Como criar um Servlet – Configurando ambiente

Selecione o servidor Apache Tomcat or TomEE e Finish – se já aparecer o tomcat, pode pular pro slide 28.



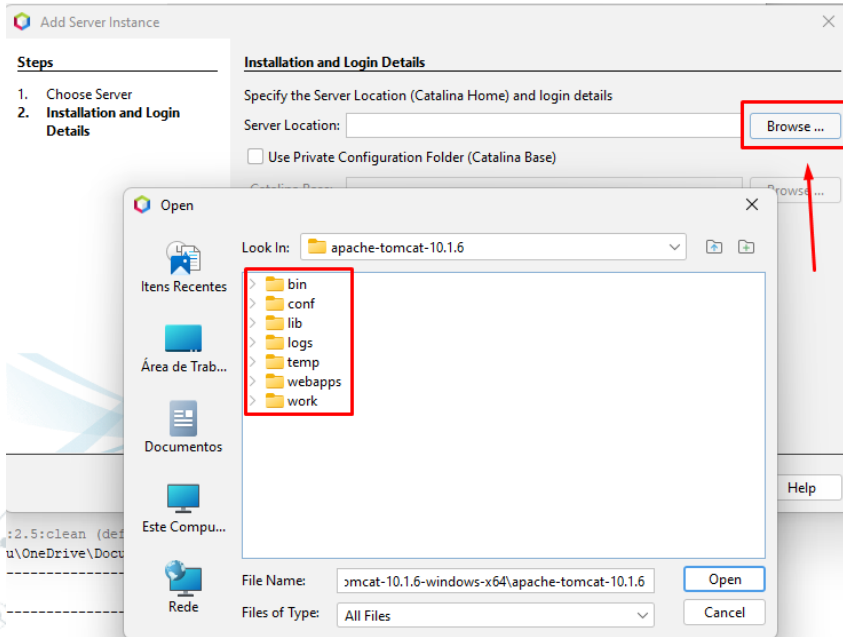
Como criar um Servlet – Configurando ambiente

Se não aparecer nenhum Server para selecionar, será necessário adicionar um, clique em Add e então selecione Apache Tomcat or TomEE e Next.



Como criar um Servlet – Configurando ambiente

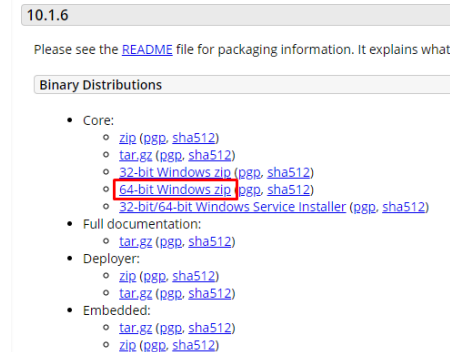
Neste momento, será necessário selecionar onde os arquivos do tomcat estão. Selecione o Browse e então selecione a pasta, como na imagem.



Se não souber onde encontrar o Tomcat ele pode ser baixado em:

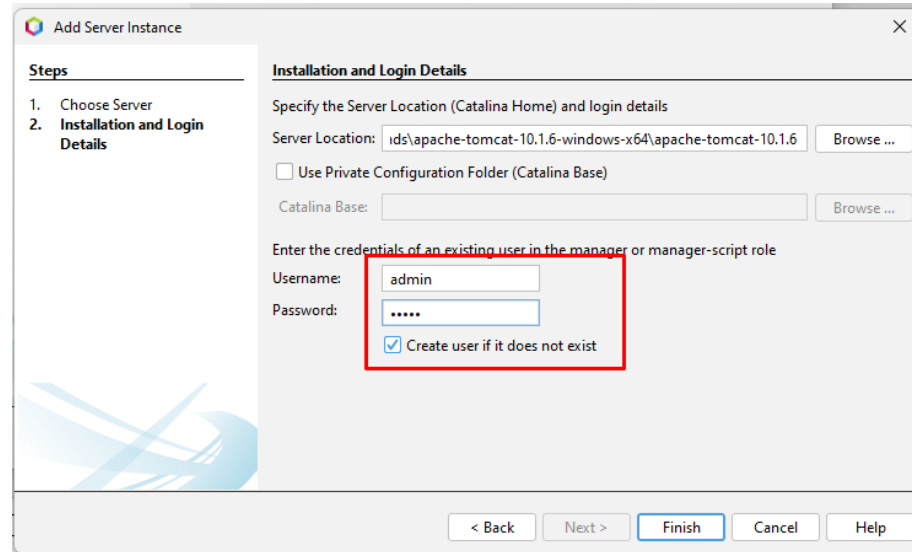
<https://tomcat.apache.org/download-10.cgi>

Após baixar, basta extrair selecionar como no passo anterior



Como criar um Servlet – Configurando ambiente

Depois de selecionar o Tomcat, é necessário colocar alguma credencial, que pode ser criado na hora, por exemplo usei admin:admin



Add Server Instance

Steps

1. Choose Server
2. **Installation and Login Details**

Installation and Login Details

Specify the Server Location (Catalina Home) and login details

Server Location:

☐ Use Private Configuration Folder (Catalina Base)

Catalina Base:

Enter the credentials of an existing user in the manager or manager-script role

Username:

Password:

☒ Create user if it does not exist

< Back Next > **Finish** Cancel Help

Como criar um Servlet – Configurando ambiente

Depois de selecionar o Tomcat, é necessário colocar alguma credencial, que pode ser criado na hora, por exemplo usei admin:admin

Add Server Instance

Steps

1. Choose Server
2. **Installation and Login Details**

Installation and Login Details

Specify the Server Location (Catalina Home) and login details

Server Location:

☐ Use Private Configuration Folder (Catalina Base)

Catalina Base:

Enter the credentials of an existing user in the manager or manager-script role

Username:

Password:

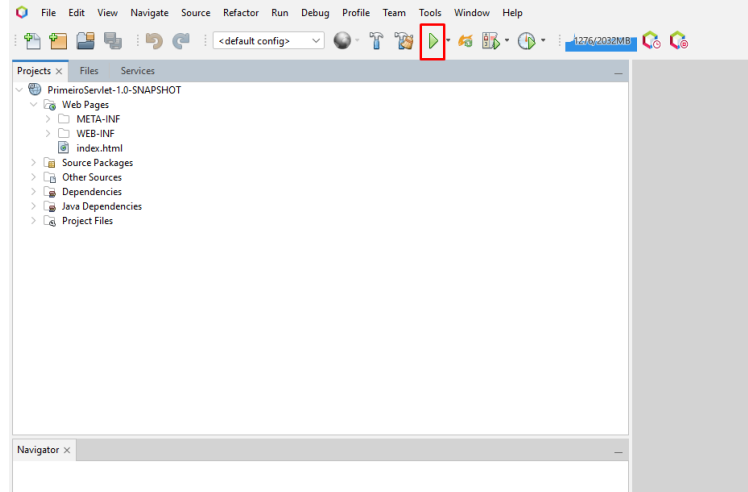
☒ Create user if it does not exist

< Back Next > **Finish** Cancel Help

Como criar um Servlet – Configurando ambiente

Com isso o projeto foi criado e já podemos criar o nosso código.

Antes de ir ao código, vamos testar a instalação do Tomcat. Vamos executar o projeto sem adicionar nenhum código.



Como criar um Servlet – Configurando ambiente

Se aparecer o seguinte erro:



The screenshot shows an IDE output window with the following text:

```
Output ×
Apache Tomcat or TomEE Log × Run (PrimeiroServlet-1.0-SNAPSHOT) ×

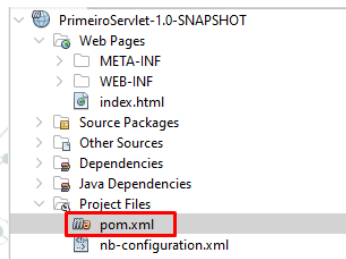
import: Entry[import from realm ClassRealm[maven.api, parent: null]]

-----
: ExceptionInInitializerError: Unable to make field private final java.util.Comparator java.util.TreeMap.comparator accessible: module java.base does not "opens java.util" to unnamed module @477523ba
-> [Help 1]

To see the full stack trace of the errors, re-run Maven with the -e switch.
Re-run Maven using the -X switch to enable full debug logging.

For more information about the errors and possible solutions, please read the following articles:
[Help 1] http://cwiki.apache.org/confluence/display/MAVEN/PluginContainerException
```

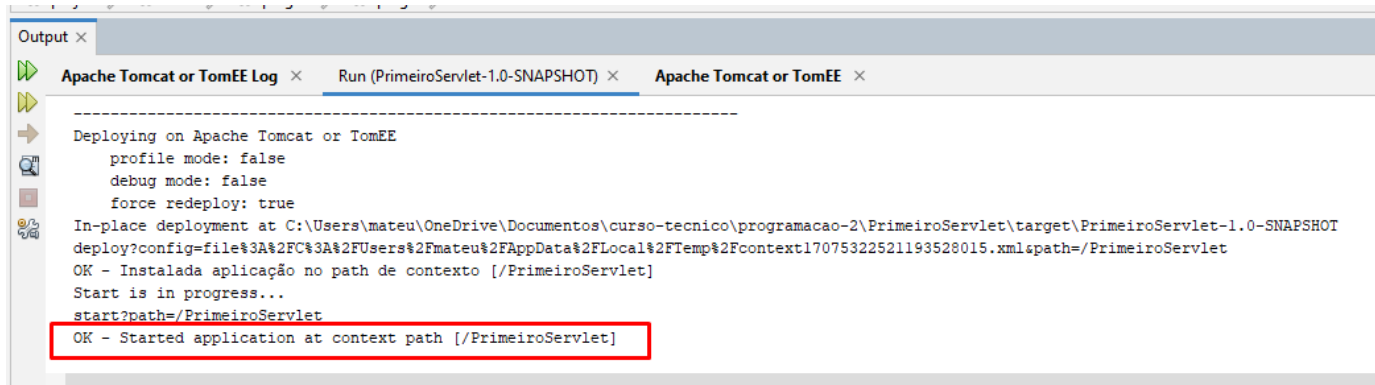
Altere a versão do maven-war-plugin no pom.xml do projeto de `<version>2.3</version>` para `<version>3.3.1</version>` e execute novamente.



```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-war-plugin</artifactId>
  <version>3.3.1</version>
  <configuration>
    <failOnMissingWebXml>>false</failOnMissingWebXml>
  </configuration>
</plugin>
```

Como criar um Servlet – Configurando ambiente

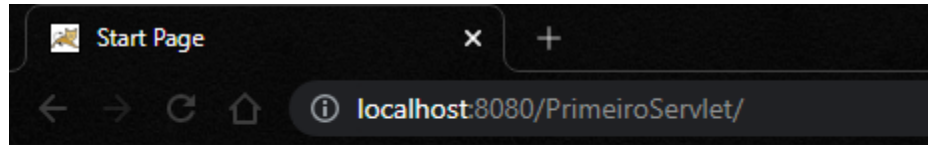
Depois de executar o projeto, o servidor tomcat iniciará e então o projeto irá subir. Após ver a mensagem abaixo, acesse <http://localhost:8080/PrimeiroServlet/> no seu navegador.



```
Output x
Apache Tomcat or TomEE Log x Run (PrimeiroServlet-1.0-SNAPSHOT) x Apache Tomcat or TomEE x
-----
Deploying on Apache Tomcat or TomEE
  profile mode: false
  debug mode: false
  force recompile: true
In-place deployment at C:\Users\mateu\OneDrive\Documents\curso-tecnico\programacao-2\PrimeiroServlet\target\PrimeiroServlet-1.0-SNAPSHOT
deploy?config=file%3A%2FC%3A%2FUsers%2Fmateu%2FAppData%2FLocal%2FTemp%2Fcontext17075322521193528015.xml&path=/PrimeiroServlet
OK - Instalada aplicação no path de contexto [/PrimeiroServlet]
Start is in progress...
start?path=/PrimeiroServlet
OK - Started application at context path [/PrimeiroServlet]
```

Como criar um Servlet – Configurando ambiente

Já estamos rodando o nosso projeto, e o index.html está sendo exibido.

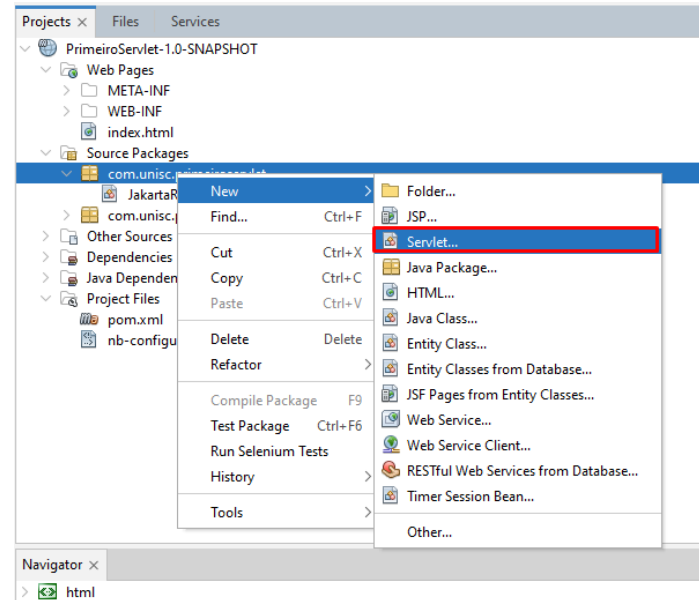


Hello World!

Como criar um Servlet

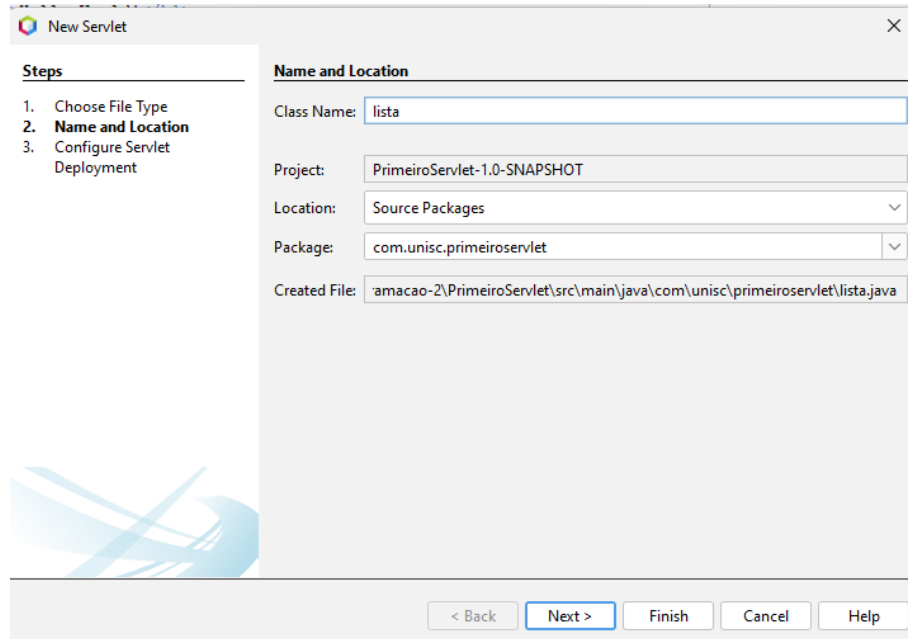
Agora vamos criar o nosso Servlet para renderizar uma página Jsp e exibir uma lista.

Em Source Packages, clique com o botão direito no pacote primeiroervlet, vá em new e então em Servlet



Como criar um Servlet

Dê um nome ao servlet, eu nomeei de lista e então clique em finish



New Servlet

Steps

1. Choose File Type
2. **Name and Location**
3. Configure Servlet Deployment

Name and Location

Class Name:

Project:

Location:

Package:

Created File: amacao-2\PrimeiroServlet\src\main\java\com\unisc\primeiroservlet\lista.java

< Back **Next >** Finish Cancel Help

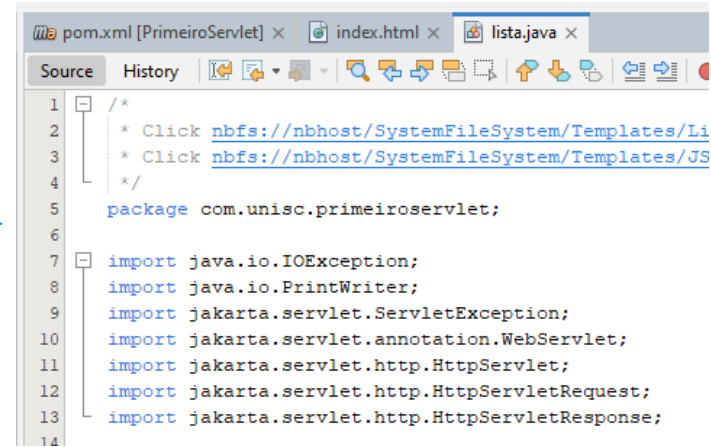
Como criar um Servlet

O netbeans, cria o servlet com as bibliotecas antigas que eram usadas no servlet, então o arquivo criado, terá alguns erros.

Mas basta atualizar os import de javax para jakarta, como abaixo.



```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/lic
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/
4   */
5  package com.unisc.primeiroervlet;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import javax.servlet.ServletException;
10 import javax.servlet.annotation.WebServlet;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14
15
```



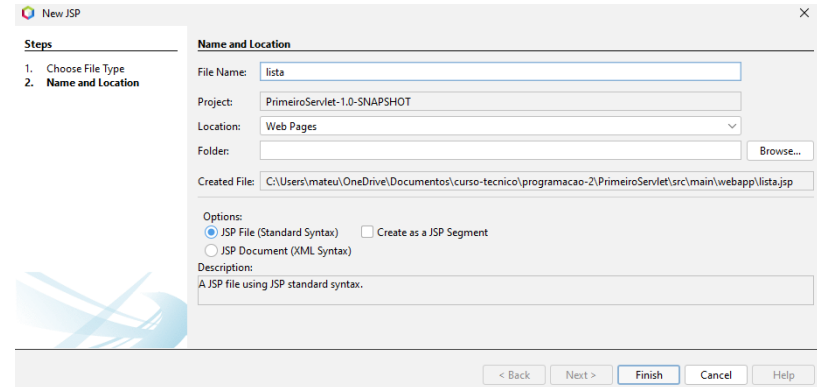
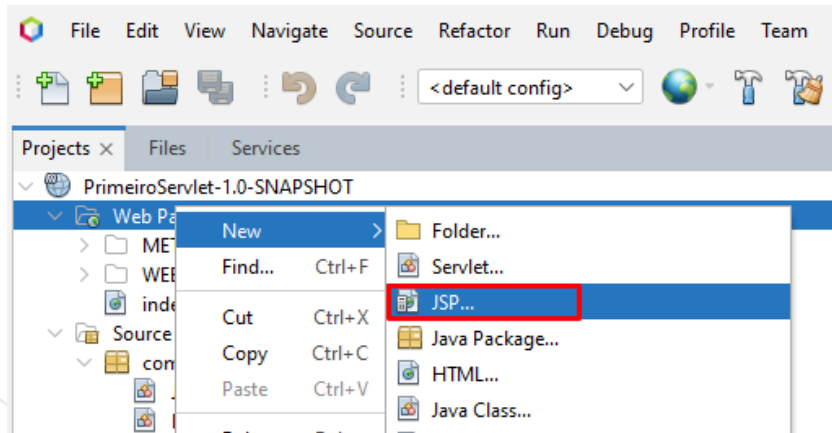
```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Li
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JS
4   */
5  package com.unisc.primeiroervlet;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import jakarta.servlet.ServletException;
10 import jakarta.servlet.annotation.WebServlet;
11 import jakarta.servlet.http.HttpServlet;
12 import jakarta.servlet.http.HttpServletRequest;
13 import jakarta.servlet.http.HttpServletResponse;
14
15
```

Retornando um JSP

Se acessarmos agora, <http://localhost:8080/PrimeiroServlet/lista> Já podemos ver a resposta do nosso servlet criado. Mas no momento estamos retornando o código html através de prints. Vamos criar um jsp e retorná-lo.

Clique com o botão direito em Web Pages, e então em Jsp.

De um nome para ele, e clique em finish.



Retornando um JSP

Com o JSP criado, podemos retornar ele para o usuário alterando o nossa função `processRequest` para ficar como abaixo.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    request.getRequestDispatcher("lista.jsp").forward(request, response);
}
```

Ao salvar, basta acessar novamente a página <http://localhost:8080/PrimeiroServlet/lista> e o conteúdo deve ter sido atualizado.

Mas o que fizemos ?

Atualizamos o método `PrintWriter`, que escreve somente texto, e agora utilizamos o `dispatcher` para retornar um `jsp`.

O que podemos fazer com JSP

Ao utilizar o JSP, podemos utilizar código java nele, para facilitar o desenvolvimento, este código java, será traduzido para html e então enviado para quem mandou a requisição.

Por exemplo, podemos printar um olá mundo e também podemos mostrar a data atual no nosso html, fazendo o import da biblioteca Date na tag Page.

```
6
7  <%@page contentType="text/html" import="java.util.Date, java.text.*" pageEncoding="ISO-8859-1"%>
8
9
10 <!DOCTYPE html>
11 <html>
12 <head>
13   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14   <title>JSP Page</title>
15 </head>
16 <body>
17   <h1>
18     <%
19       out.println("Ola Mundo");
20     %>
21   </h1>
22   <br>
23   <p><%=new Date() %></p>
24 </body>
25
26 </html>
```

O que podemos fazer com JSP

Ou seja, podemos fazer muita coisa com o java no JSP, tornando assim mais dinâmica a programação da interface.

Vamos mais além, passar um parâmetro do nosso servlet para ser exibido no Jsp. Vamos criar uma string que queremos passar, e passar para a nossa request, como um atributo.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType(string: "text/html;charset=UTF-8");
    String str = "Quero mostrar isso no HTML";
    request.setAttribute(string: "valor", o: str);
    request.getRequestDispatcher(string: "lista.jsp").forward(sr: request, srl: response);
}
```

O que podemos fazer com JSP

Após passar no meu Servlet o atributo, posso busca-lo no jsp usando a request também, como abaixo.

```
~~~~~  
<p><%=new Date () %></p>  
<p><%=request.getAttribute ("valor") %></p>  
>ody>
```

Com essa funcionalidade, podemos criar muitas coisas diferentes.

O que podemos fazer com JSP

- Podemos usar o for e o if, como se estivéssemos no java

```
<ol>  
<%  
  for(int i=0; i<5; i++) {  
    if (i > 2) {  
      %>  
      <li>  
        <%  
          out.println(i);  
        %>  
      </li>  
    }  
  }  
  %>  
</ol>
```


Exercício

Agora que já viu como fazer algumas operações no JSP, crie uma lista no servlet, passe ela para o Jsp e mostre ela no html do usuário.