Senha Heroku

terça-feira, 8 de novembro de 2022

15:09

Usuario Heroku

|  |  |
| --- | --- |
| Tarcisioms23@hotmail.com |  |
|  |  |

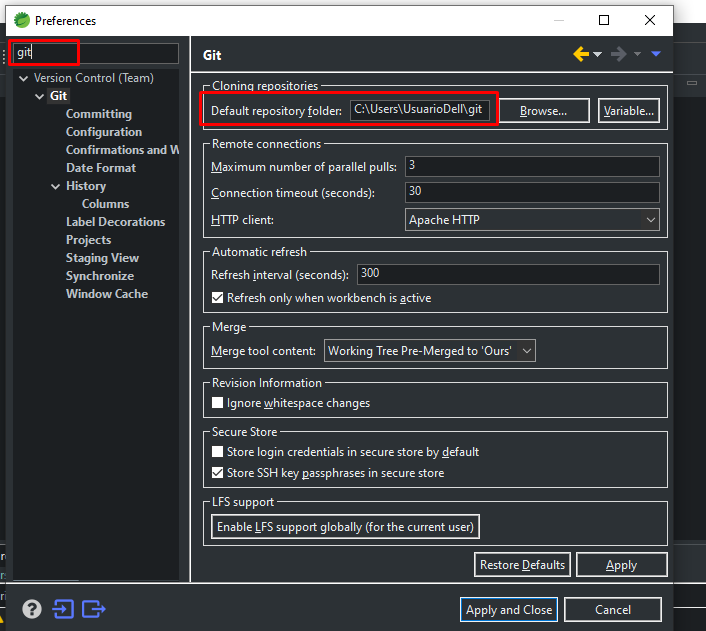
Como utilizar o Heroku

Utilizando Git na IDE Eclipse

quarta-feira, 9 de novembro de 2022

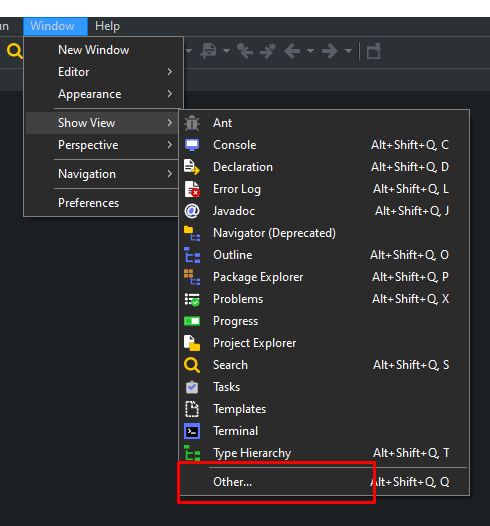
16:40

* 1. Instale o git. Assim que houver a necessidade de versionar e guardar seu codigo
  2. Abrir windows/preferences e digitar git

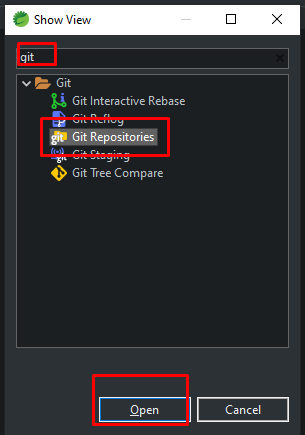


Observe que irá mostrar o local o repositório que seral clonado e versionado.

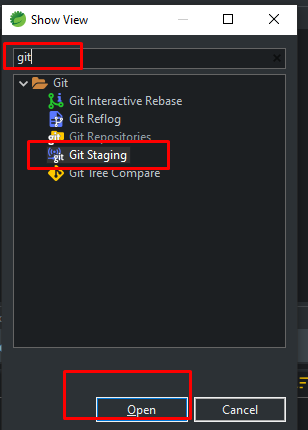
* 1. Para que aparece o git segui o passo abaixo:



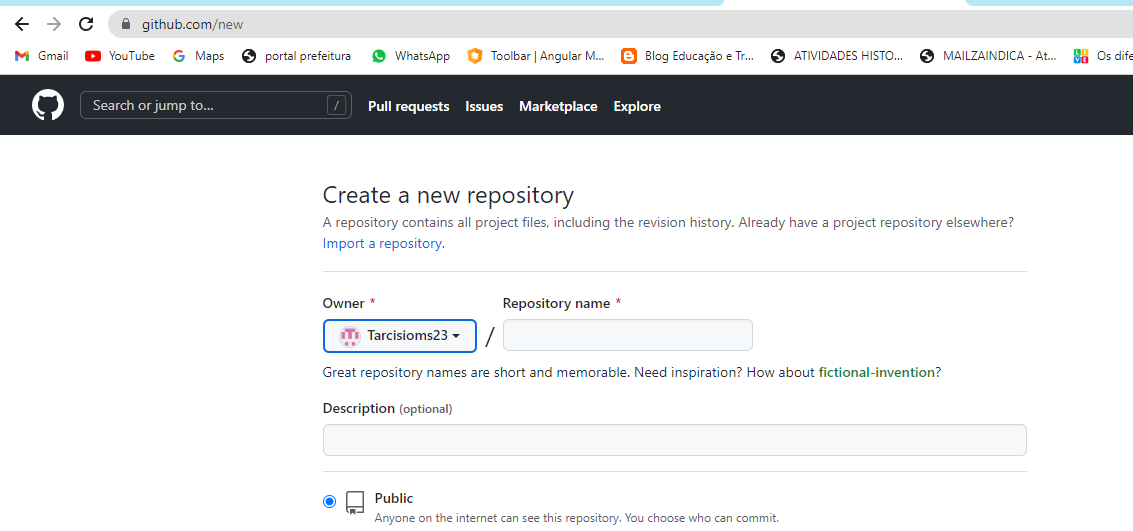
* 1. Abrir essa janela



* 1. E depois Abrir essa janela

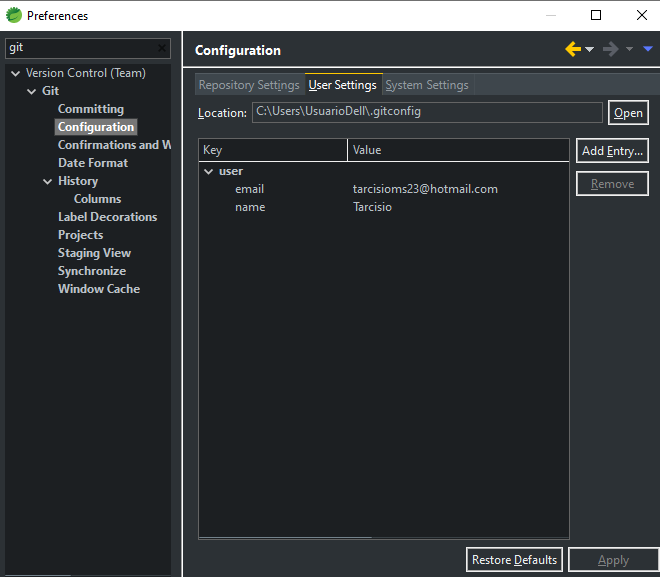


* 1. Crie um repositorio no git

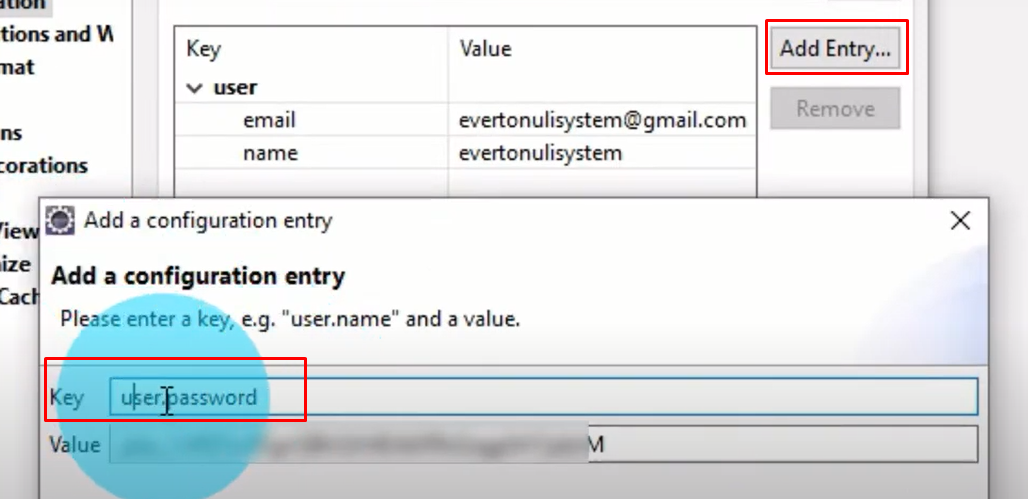


* 1. Abra novamente preference - git e entre na opcção

Voce pode observar suas credenciais ou ate mesmo colocar



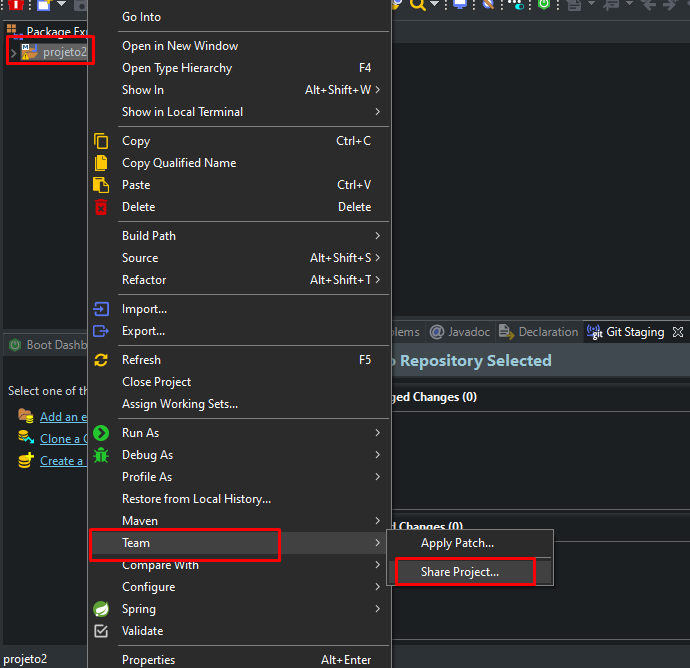
* 1. Pode colocar ate mesmo o token setado conforme abaixo Github



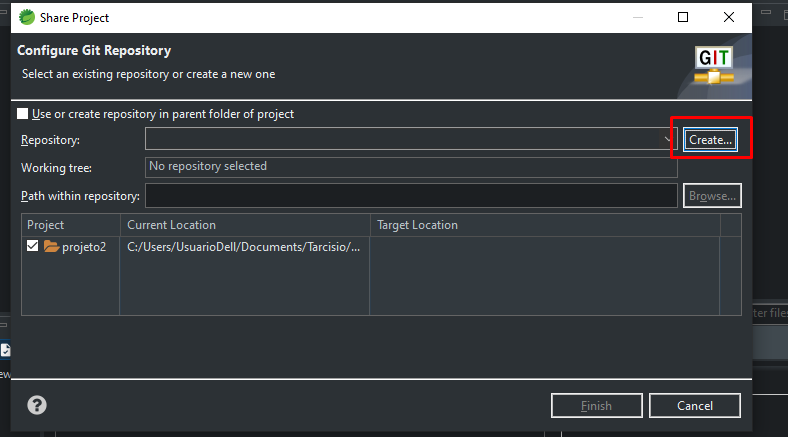
Feito a configuração da senha caso queira.

* 1. Iremos enviar nosso projeto para o github

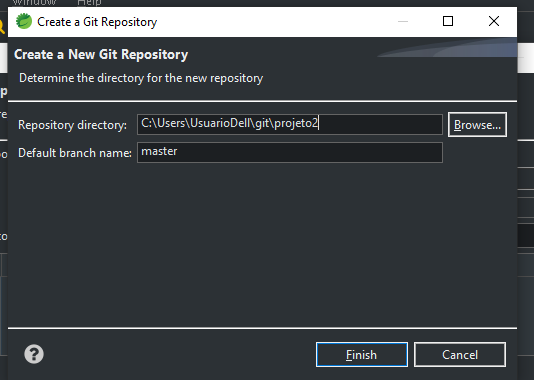
Para isso clliquei botao direto no projeto team - share project(Compartilhar projetos)



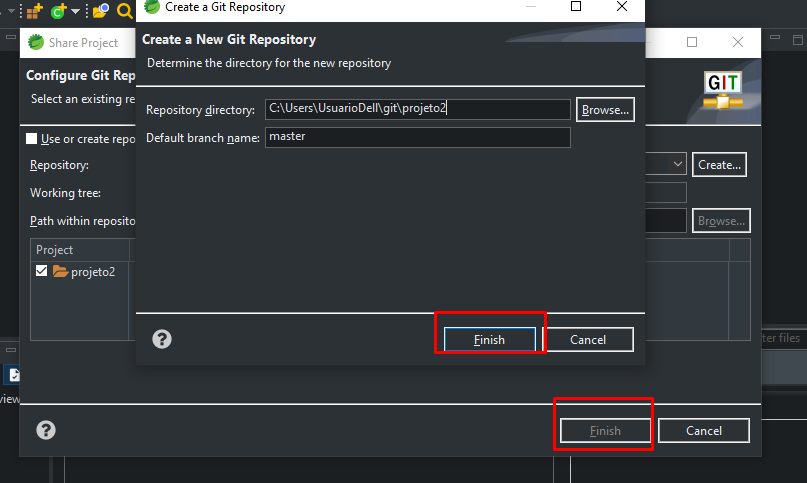
* 1. Click em create



* 1. Pode alterar o nome para o nome do projeto (No caso projeto2)

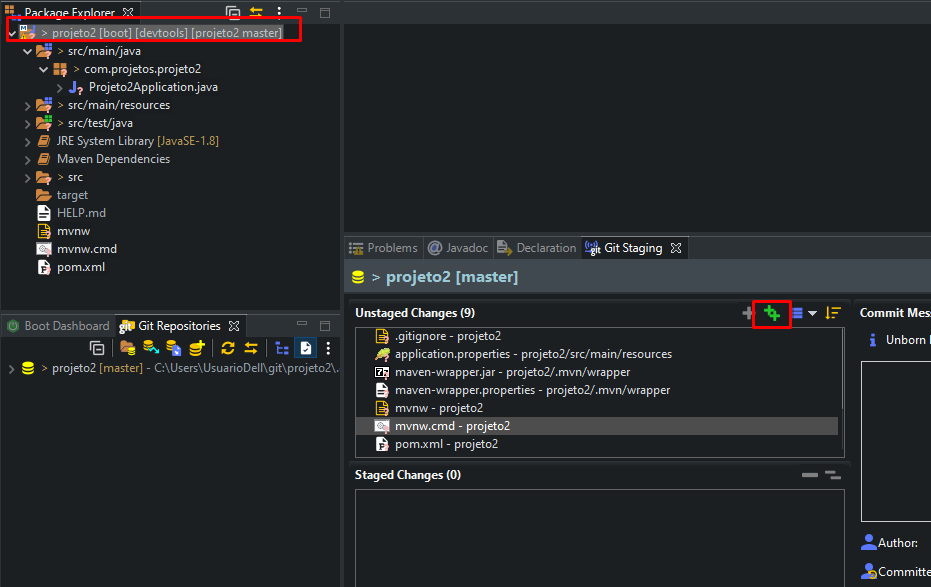


* 1. Finish depois finish

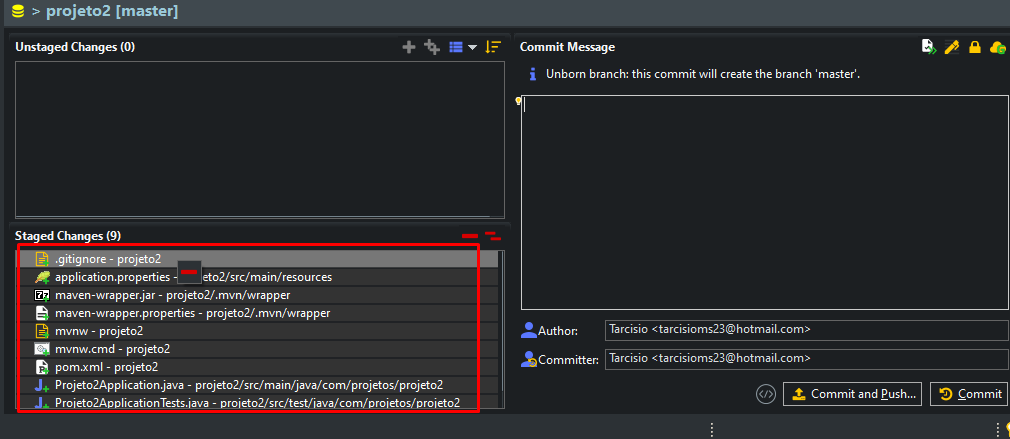


* 1. Para adicionar os seus indices no caso o mesmo de git add .

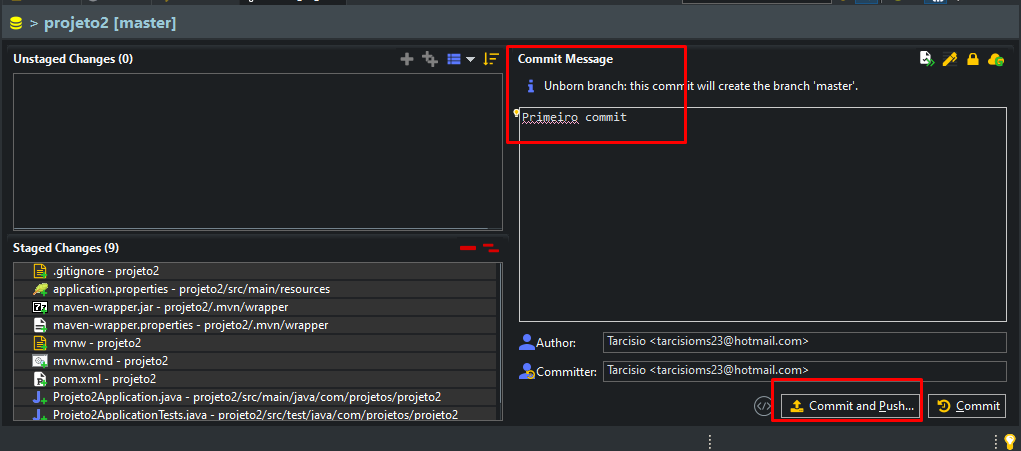
Clique no projeto e depois - add all files ++



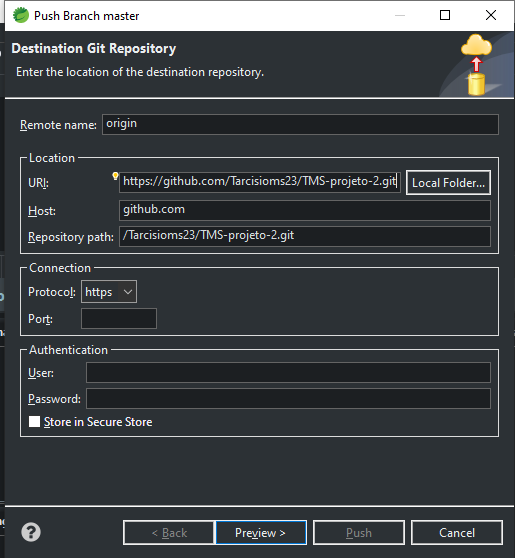
* 1. Aqui mostra que foi para stage



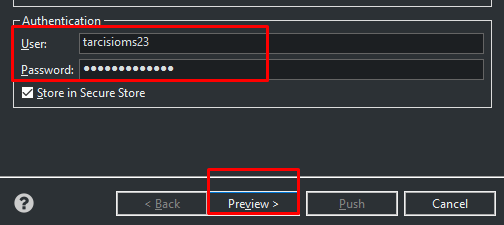
* 1. Após isso prenche o conteudo na commit message e clique no commit na push



* 1. Coloque o link do repositorio que criou do github

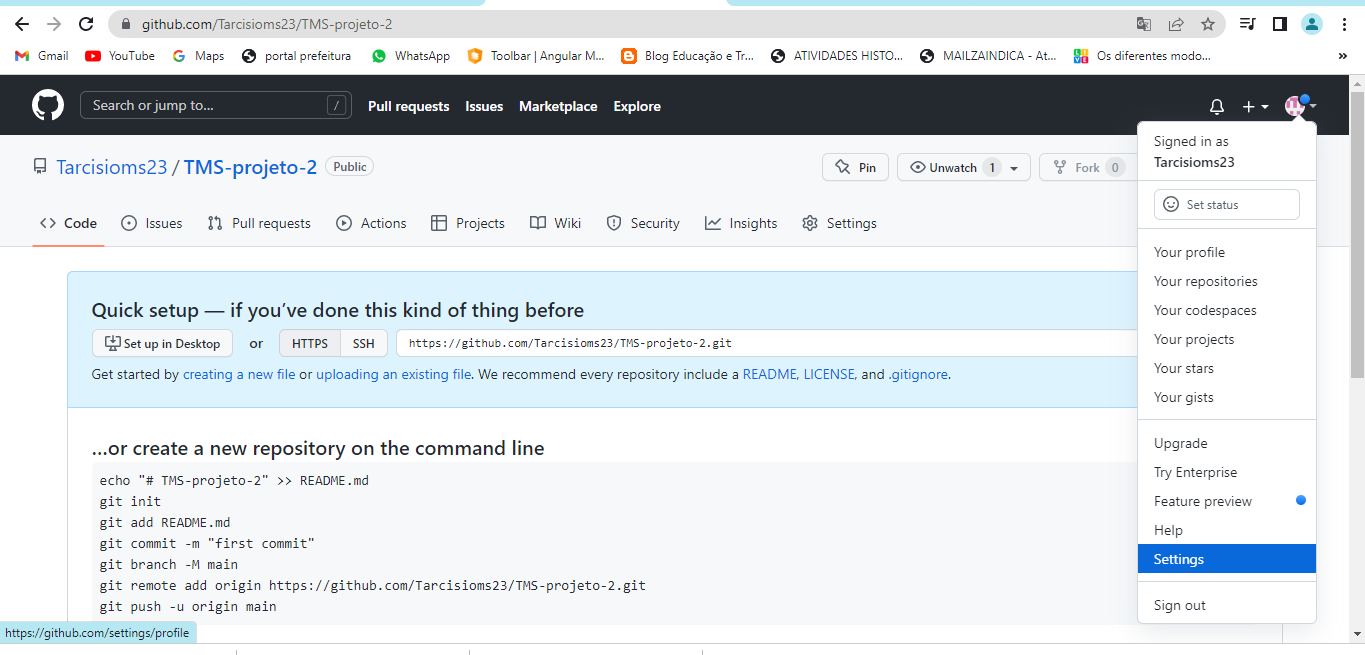


* 1. Prencha usuario

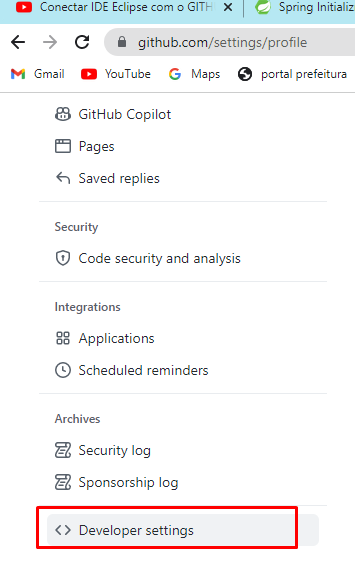


* 1. Caso não tenha colocado o Token, Para gerar o password e necessario GERAR TOKEN NO GITHUB seguir os passos abaixo

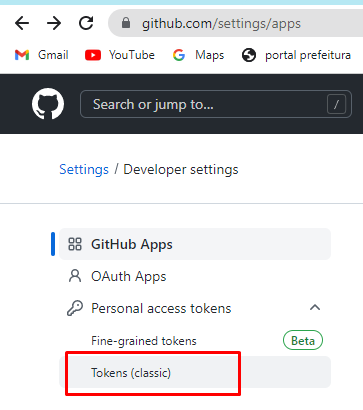
18.1 settings



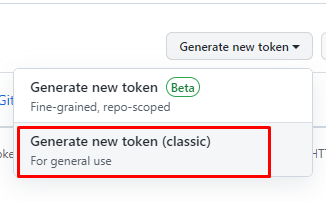
18.2 clique nessa opcao



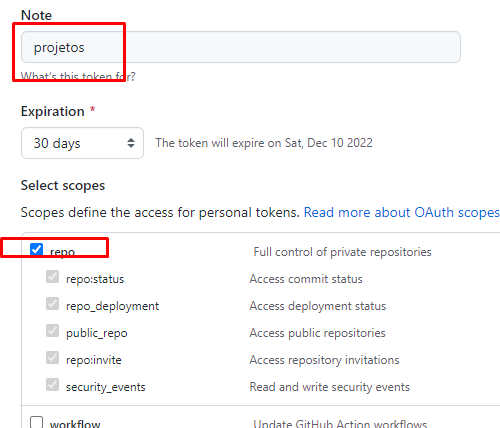
18.3 entre nessa opçao

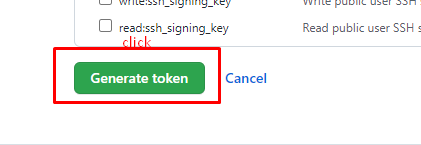


18.4 entre nessa opcao

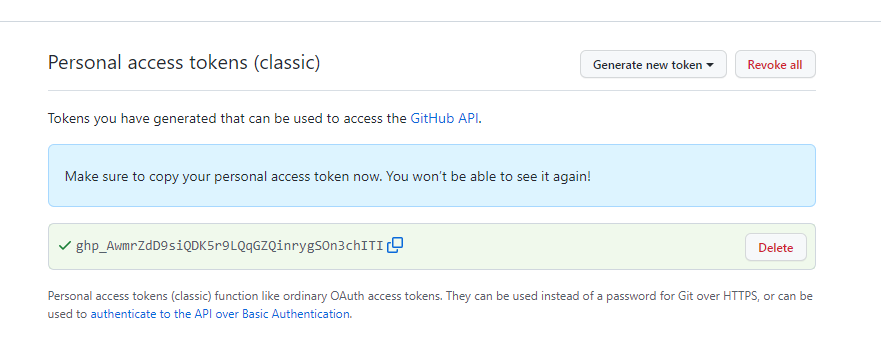


18.5 Preencha os campos abaixo e click no botao

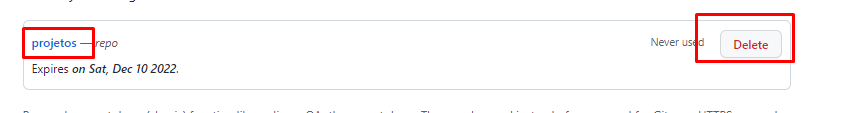




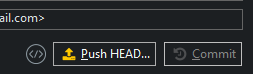
18.6 pronto



Caso não tenho o token pode remover e gerar novamente ou alterar e gerar token novamente. Clicando em qualquer das duas opcões.



Após isso volte no eclipse la e de push commit



Postgress

sexta-feira, 18 de novembro de 2022

14:42

SELECT datname FROM pg\_database;

De <[*https://www.postgresqltutorial.com/postgresql-administration/postgresql-show-databases/*](https://www.postgresqltutorial.com/postgresql-administration/postgresql-show-databases/)>

Listar databases bancos

* + Use \lou \l+in psqlpara mostrar todos os bancos de dados no servidor PostgreSQL atual.
  + Use a SELECTinstrução para consultar dados do pg\_databasepara obter todos os bancos de dados.

De <[*https://www.postgresqltutorial.com/postgresql-administration/postgresql-show-databases/*](https://www.postgresqltutorial.com/postgresql-administration/postgresql-show-databases/)>

Comando para postegresql para acessar o banco equivalente ao "use database"

\c "database"

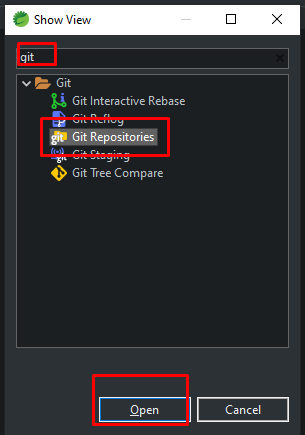
De <[*https://www.guj.com.br/t/postgres-comando-equivalente-use-database/294511*](https://www.guj.com.br/t/postgres-comando-equivalente-use-database/294511)>

Clonando respositorio no github/git no eclipse

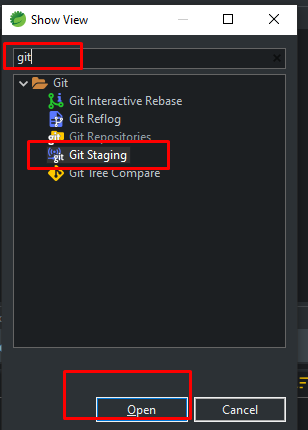
quinta-feira, 17 de novembro de 2022

20:06

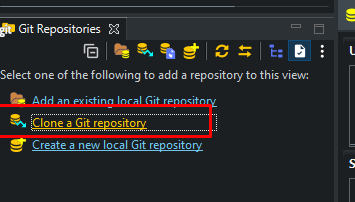
* 1. Abrir essa janela



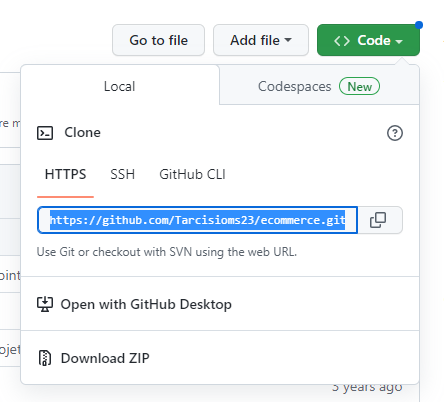
* 1. E depois Abrir essa janela



* 1. Clique aqui

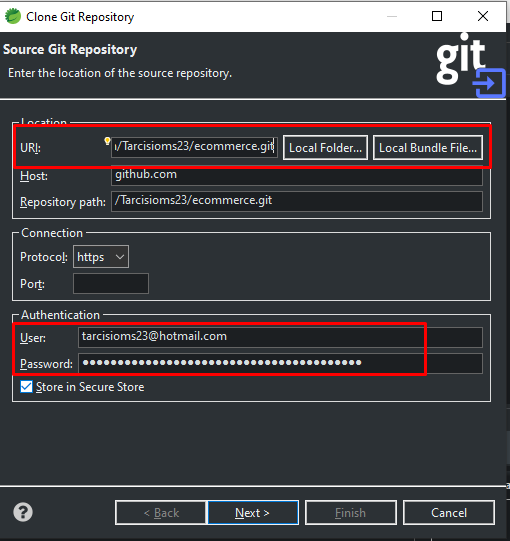


* 1. Pegue o link do repositorio que deseja clonar no github

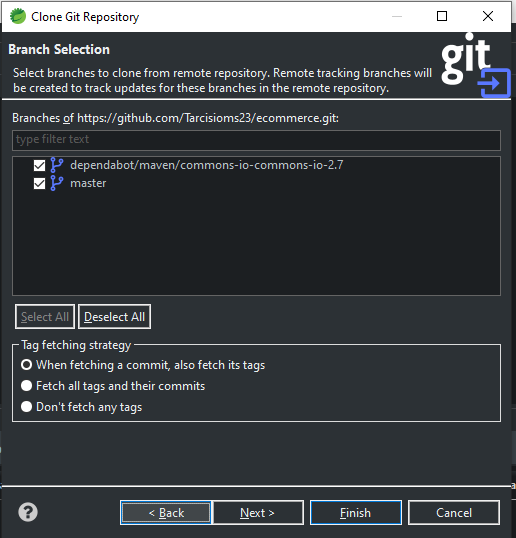


* 1. A princípio cole o URI e após isso se não tiver configurando a senha (Token) necessário colocar o valor do token

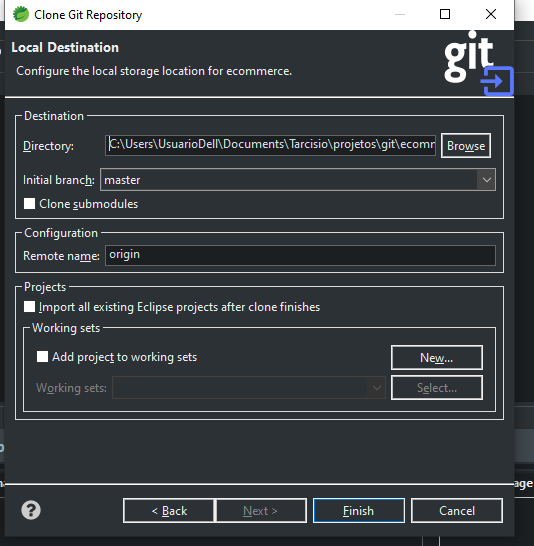
Caso queira deixar preenchido basta ir no passo 11 (next)



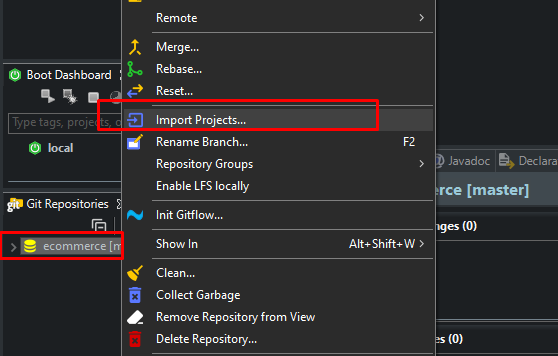
* 1. Escolha a branch - next



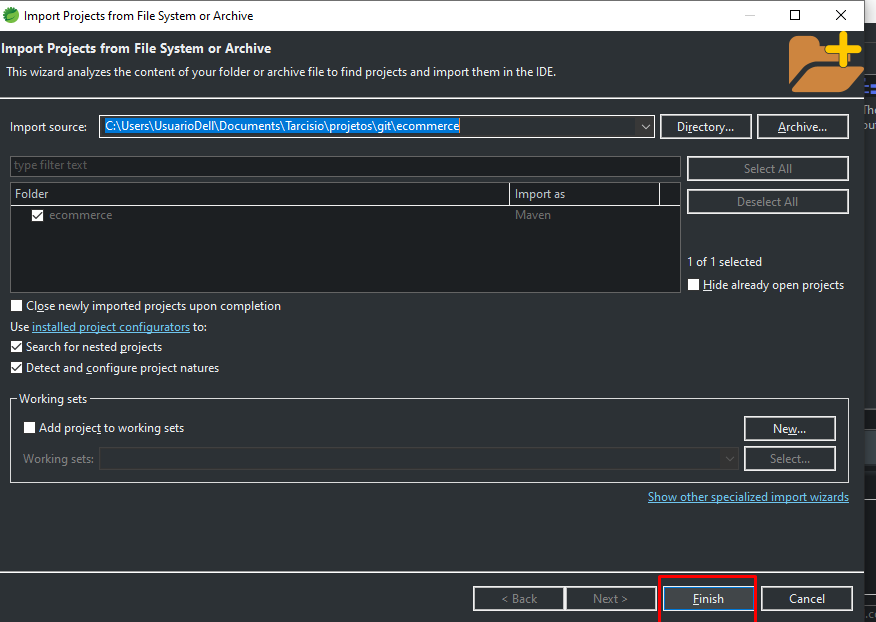
* 1. Aqui mostra onde o repositorio local git sera criado (Finish)



* 1. Após importe o projeto do repositorio informado. Clique no repostirio e depois importe o projeto

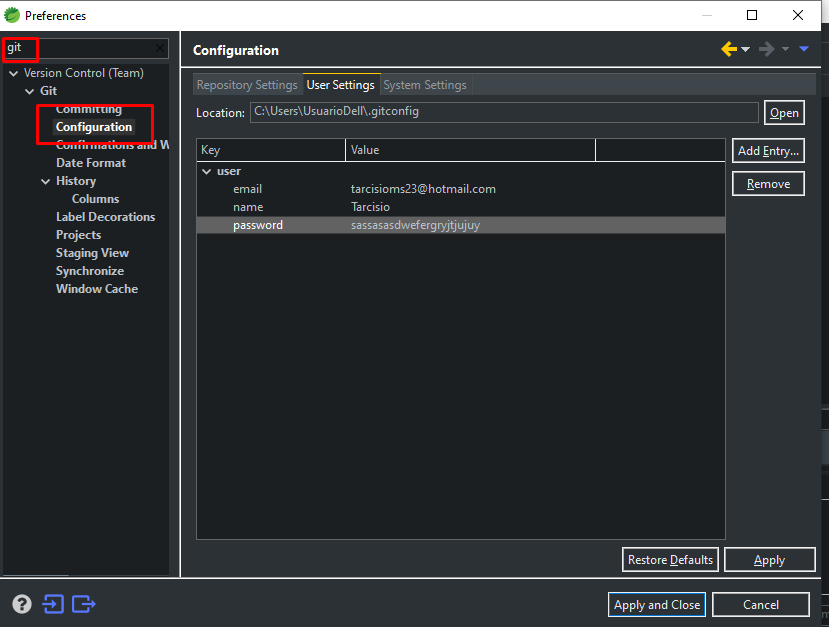


* 1. Após isso aperte finish



* 1. Configure a senha entrando em window; preferencias ; GIT ; Configuration

Basta colocar user.password : e colocar o token em Value



Spring 1ª etapa - Package models - classe models - java

sexta-feira, 18 de novembro de 2022

10:12

Classes que possui os objetos e seus getters e setters

Tera essas anotações na classe

@**Entity** //MOSTRA que é uma entidade (JPA HIBERNET)

@**Table**(name= "TB\_PRODUTO") // nomeia a tabela

public class Produto implements Serializable{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

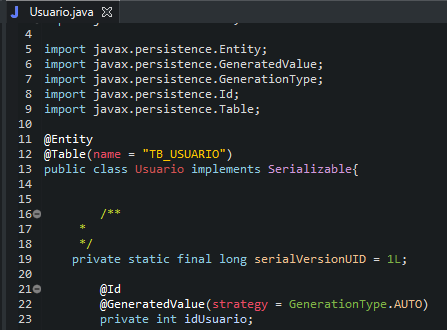
@**Id** // identificar que é um id

@**GeneratedValue**(strategy = GenerationType.AUTO) // Gera id único automaticamente

private Long Id;

private String nome;

OBS crie os getters e setters.



Bibliotecas usadas

import java.io.Serializable;

import java.math.BigDecimal;

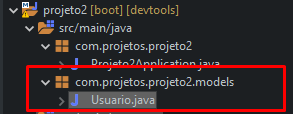
import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;



Spring 2ª etapa - application.properties

sexta-feira, 18 de novembro de 2022

10:36

Aqui vc ira criar a configuração do seu banco de dados - local para configurar

Src/main/resources/aplication.properties

<<application.properties>>

No exemplo abaixo configuro a porta que quero utilizar tbm

server.port = 8091

# Coloquei devido a uma falha ao importar a dependencia swagger 2

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

#Importante criar.

spring.jpa.properties.hibernate.jdbc.non\_contextual\_creation= true

#Banco local Tarcisio - Lembrando que se dever cria o database antes no banco

# CREATE DATABASE projeto\_1;

spring.datasource.url=jdbc:postgresql://localhost:5432/projeto\_1

spring.datasource.username=postgres

spring.datasource.password=admin

spring.jpa.hibernate.ddl-auto=update

Lembrando que no arquivo pom precisa ter a dependencia do banco que ira utilzar

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<scope>runtime</scope>

</dependency>

Spring 3ª etapa - criar repository

sexta-feira, 18 de novembro de 2022

14:57

Iremos criar um repository dessa Da entidade (models)

Ele sempre vai extends Jpa Repository (PASSAR ENTIDADE E O ID)

Jpa Repository (Por isso criamos uma instancia dessa interface para facilitar pois já contem varios metodos como (salvar deletar buscar find all find by all já contem nessa interface.)

Caminho a ser criado



package com.projetos.projeto2.repository;

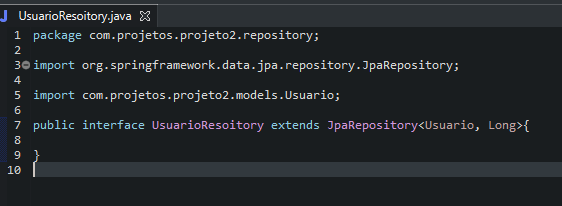
import org.springframework.data.jpa.repository.JpaRepository;

import com.projetos.projeto2.models.Usuario;

public interface UsuarioResoitory extends JpaRepository<Usuario, Long>{

Usuario findById (long id); // metodo findby id precisar declarar

}

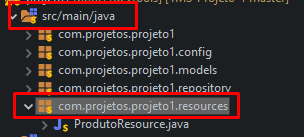


Spring 4ª etapa - criar resources (Nosso Controller)

sexta-feira, 18 de novembro de 2022

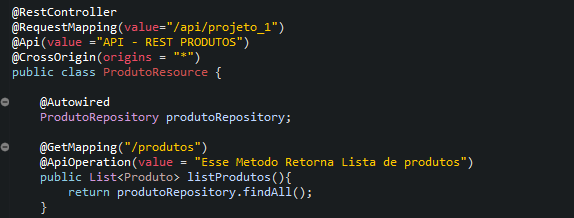
14:48

Criar package resources com o padrao determinado



Criar classe resource

Anotações importantes



package com.projetos.projeto1.resources;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.projetos.projeto1.repository.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import com.projetos.projeto1.models.Produto;

@RestController // Definiir nosso controler

@RequestMapping(value="/api/projeto\_1") // Anotação para determinar o caminho da nossa api

public class ProdutoResource {

@Autowired

ProdutoRepository produtoRepository;

@GetMapping("/produtos")

public List<Produto> listProdutos(){

return produtoRepository.findAll();

}

@GetMapping("/produto/{id}")

public Produto idProduto(@PathVariable(value ="id") long id){

return produtoRepository.findById(id);

}

@PostMapping("/produto")

public Produto salvaProduto(@RequestBody Produto produto) {

return produtoRepository.save(produto);

}

@PutMapping("/produto")

public Produto alteraProduto(@RequestBody Produto produto) {

return produtoRepository.save(produto);

}

@DeleteMapping("/produto")

public String deleteProduto(@RequestBody Produto produto) {

produtoRepository.delete(produto);

return "Produto deletado com sucesso " + produto.getNome();

}

}

Spring 5ª etapa - criando Swagger

sexta-feira, 18 de novembro de 2022

16:02

No arquivo pom colocar as duas dependencias

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

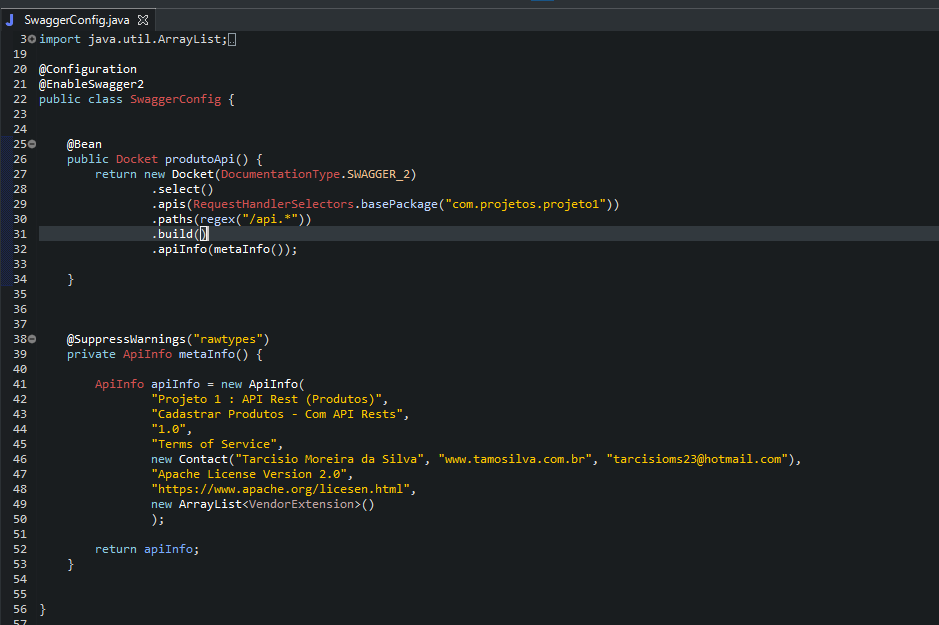
<version>2.7.0</version>

</dependency>

<https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>

Alem disso criar package config e classe SwaggerConfig





package com.projetos.projeto1.config;

import java.util.ArrayList;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.RequestHandlerSelectors;

import springfox.documentation.service.ApiInfo;

import springfox.documentation.service.Contact;

import springfox.documentation.service.VendorExtension;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import static springfox.documentation.builders.PathSelectors.regex;

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket produtoApi() {

return new Docket(DocumentationType.SWAGGER\_2)

.select()

.apis(RequestHandlerSelectors.basePackage("com.projetos.projeto1"))

.paths(regex("/api.\*"))

.build()

.apiInfo(metaInfo());

}

@SuppressWarnings("rawtypes")

private ApiInfo metaInfo() {

ApiInfo apiInfo = new ApiInfo(

"Projeto 1 : API Rest (Produtos)",

"Cadastrar Produtos - Com API Rests",

"1.0",

"Terms of Service",

new Contact("Tarcisio Moreira da Silva", "[www.tamosilva.com.br](http://www.tamosilva.com.br)", "tarcisioms23@hotmail.com"),

"Apache License Version 2.0",

"<https://www.apache.org/licesen.html>",

new ArrayList<VendorExtension>()

);

return apiInfo;

}

}

Existe alguma anotações que são feitas no resource para configurar o Swagger para configurações do swagger

Configurações do swagger na classe resource em amarelo

@RestController // Definiir nosso controler

@RequestMapping(value="/api/projeto\_1") // Anotação para determinar o caminho da nossa api

@Api(value ="API - REST PRODUTOS") // Anotação para configurar o swagger

@CrossOrigin(origins = "\*") // Anotação para configurar o swagger

public class ProdutoResource {

@Autowired

ProdutoRepository produtoRepository;

@GetMapping("/produtos")

@ApiOperation(value = "Esse Metodo Retorna Lista de produtos") ///Swagger configuração

public List<Produto> listProdutos(){

return produtoRepository.findAll();

}

@GetMapping("/produto/{id}")

@ApiOperation(value = "Esse Metodo Retorna produto por id") ///Swagger configuração

public Produto idProduto(@PathVariable(value ="id") long id){

return produtoRepository.findById(id);

}

@PostMapping("/produto")

@ApiOperation(value = "Esse Metodo que salva um produto")//Swagger configuração

public Produto salvaProduto(@RequestBody Produto produto) {

return produtoRepository.save(produto);

}

@PutMapping("/produto")

@ApiOperation(value = "Esse Metodo atualiza um produto")///Swagger configuração

public Produto alteraProduto(@RequestBody Produto produto) {

return produtoRepository.save(produto);

}

@DeleteMapping("/produto")

@ApiOperation(value = "Esse Metodo deleta um produto")///Swagger configuração

public String deleteProduto(@RequestBody Produto produto) {

produtoRepository.delete(produto);

return "Produto deletado com sucesso " + produto.getNome();

}

}