Wellington Cesar Fonseca

DESENVOLVIMENTO FULL STACK

MISSÃO PRÁTICA | NÍVEL 3 | MUNDO 5

RPG0018 - POR QUE NÃO PARALELIZAR

OBJETIVO

1. Criar servidores Java com base em Sockets.
2. Criar clientes síncronos para servidores com base em Sockets.
3. Criar clientes assíncronos para servidores com base em Sockets.
4. Utilizar Threads para implementação de processos paralelos.
5. No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

1º PROCEDIMENTO | CRIANDO O SERVIDOR E CLIENTE DE TESTE

Como funcionam as classes Socket e ServerSocket?
R:

- **Socket:** É utilizado para estabelecer uma conexão entre o cliente e o servidor. Ele representa um único endpoint em uma comunicação de rede bidirecional.
- **ServerSocket:** É utilizado no servidor para aguardar solicitações de conexão de clientes. Ele escuta por conexões em uma porta específica e cria um novo Socket para cada conexão aceita.

Qual a importância das portas para a conexão com servidores?
R:
As portas são usadas para identificar aplicações específicas em um servidor. Elas permitem que múltiplos serviços (como HTTP, FTP) operem simultaneamente no mesmo servidor, distinguindo o tráfego de rede por número de porta.

Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?
R:

- **ObjectInputStream:** Lê objetos Java a partir de uma fonte de entrada, como um arquivo ou socket.
- **ObjectOutputStream:** Escreve objetos Java para um destino de saída, permitindo a transmissão de dados pela rede.
- **Serializáveis:** Os objetos devem ser serializáveis para serem convertidos em um formato de byte-stream, necessário para armazenar ou transmitir objetos em rede ou entre diferentes componentes do sistema.

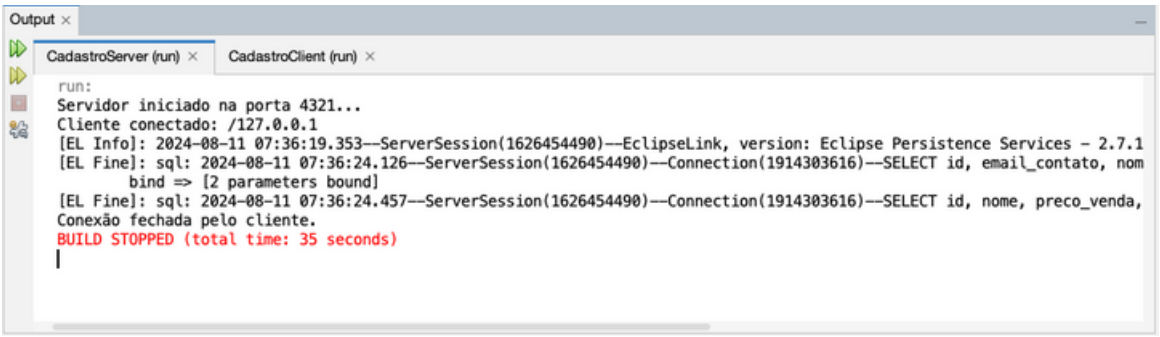
Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?
R:
Pois o JPA (Java Persistence API) abstrai a lógica de acesso a dados, permitindo que a aplicação cliente interaja com entidades sem expor detalhes do banco de dados subjacente. O isolamento é garantido pelo uso de uma camada de persistência que gerencia transações e conexões, assegurando que as operações de banco de dados sejam feitas de forma segura e consistente, mesmo quando acessadas remotamente.

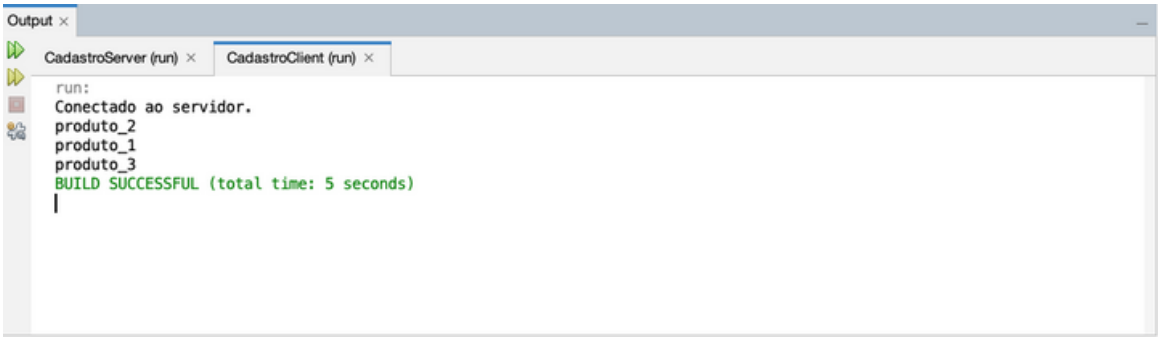
IMPORTANTE!
Tive que remover parte do persistence, pois o github está rejeitando o push por:

```
remote: - GITHUB PUSH PROTECTION
remote:   ——————————————————————————————————————————————————
remote:     Resolve the following violations before pushing again
remote:
remote:     - Push cannot contain secrets
```

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
 <persistence-unit name="CadastroServerPU" transaction-type="RESOURCE_LOCAL">
 <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
 <class>cadastroserver.model.Movimentos</class>
 <class>cadastroserver.model.Produto</class>
 <class>cadastroserver.model.Usuarios</class>
 <class>cadastroserver.model.Pessoa</class>
 <properties>
 <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://url_do_meu_server_remoto"/>
 <property name="javax.persistence.jdbc.user" value="faculdade"/>
 <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>
 <property name="javax.persistence.jdbc.password" value="minha_senha"/>
 </properties>
 </persistence-unit>
</persistence>
```





CadastroServer.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroserver;

import cadastroserver.controller.ProdutoJpaController;
import cadastroserver.controller.UsuariosJpaController;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author wellingtonfonseca
 */
public class CadastroServer {

    public static void main(String[] args) {
        // Criação do EntityManagerFactory
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("CadastroServerPU");

        // Criação dos controladores
        ProdutoJpaController ctrl = new ProdutoJpaController(emf);
        UsuariosJpaController ctrlUsu = new UsuariosJpaController(emf);

        // Criação do ServerSocket
        try (ServerSocket serverSocket = new ServerSocket(4321)) {
            System.out.println("Servidor iniciado na porta 4321...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Cliente conectado: " + clientSocket.getInetAddress());

                CadastroThread thread = new CadastroThread(ctrl, ctrlUsu, clientSocket);
                thread.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

CadastroThread.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroserver;

import cadastroserver.controller.ProdutoJpaController;
import cadastroserver.controller.UsuariosJpaController;
import cadastroserver.model.Usuarios;
import cadastroserver.model.Produto;
import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;

/**
 *
 * @author wellingtonfonseca
 */
public class CadastroThread extends Thread {

    private final ProdutoJpaController ctrl;
    private final UsuariosJpaController ctrlUsu;
    private final Socket s1;

    public CadastroThread(ProdutoJpaController ctrl, UsuariosJpaController ctrlUsu, Socket s1) {
        this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
        this.s1 = s1;
    }

    @Override
    public void run() {
        try (ObjectOutputStream out = new ObjectOutputStream(s1.getOutputStream()); ObjectInputStream in = new
ObjectInputStream(s1.getInputStream())) {

            String login = (String) in.readObject();
            String senha = (String) in.readObject();

            Usuarios usuario = ctrlUsu.findUsuario(login, senha);
            if (usuario == null) {
                s1.close();
                return;
            }

            while (true) {
                String command = (String) in.readObject();
                if ("L".equals(command)) {
                    List<Produto> produtos = ctrl.findProdutoEntities();
                    out.writeObject(produtos);
                } else {
                    // Se comando desconhecido, talvez encerrar a conexão
                    break;
                }
            }
        } catch (EOFException e) {
            // Handle EOF, normalmente quando o cliente fecha o socket
            System.out.println("Conexão fechada pelo cliente.");
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
            try {
                s1.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
}
```

CadastroClient.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastroclient;

import cadastroserver.model.Produto;
import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;

public class CadastroClient {

    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 4321); ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream()); ObjectInputStream in = new ObjectInputStream(socket.getInputStream())) {
            System.out.println("Conectado ao servidor.");

            out.writeObject("123"); // Login
            out.writeObject("U3VhU3RyaW5nQXF1aQ=="); // Senha

            out.writeObject("L");

            List<Produto> produtos = (List<Produto>) in.readObject();
            for (Produto produto : produtos) {
                System.out.println(produto.getNome());
            }
        } catch (EOFException e) {
            // Handle EOFException
            System.out.println("Conexão com o servidor foi encerrada.");
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

UsuariosJpaController.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroserver.controller;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import cadastroserver.model.Usuarios;
import java.util.List;
import javax.persistence.TypedQuery;

/**
 *
 * @author wellingtonfonseca
 */
public class UsuariosJpaController {

 private EntityManagerFactory emf = null;

 public UsuariosJpaController(EntityManagerFactory emf) {
 this.emf = emf;
 }

 public EntityManager getEntityManager() {
 return emf.createEntityManager();
 }

 public Usuarios findUsuario(String login, String senha) {
 EntityManager em = getEntityManager();
 try {
 TypedQuery<Usuarios> query = em.createQuery(
 "SELECT u FROM Usuarios u WHERE u.usuario = :usuario AND u.senha = :senha", Usuarios.class);
 query.setParameter("usuario", login);
 query.setParameter("senha", senha);
 List<Usuarios> result = query.getResultList();
 return result.isEmpty() ? null : result.get(0);
 } finally {
 em.close();
 }
 }
}
```

ProdutoJpaController,java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroserver.controller;

import cadastroserver.model.Produto;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;

/**
 *
 * @author wellingtonfonseca
 */
public class ProdutoJpaController {

 private EntityManagerFactory emf;

 public ProdutoJpaController(EntityManagerFactory emf) {
 this.emf = emf;
 }

 public EntityManager getEntityManager() {
 return emf.createEntityManager();
 }

 public List<Produto> findProdutoEntities() {
 EntityManager em = getEntityManager();
 try {
 return em.createQuery("SELECT p FROM Produto p", Produto.class).getResultList();
 } finally {
 em.close();
 }
 }
}
```