



Desenvolvimento Full Stack

Aluno: Victor de A. Costa

Missão Prática | Nível 5 | Mundo 3 - 2024.3
POR QUE NÃO PARALELIZAR!

Servidores e clientes baseados em Socket, com uso de Threads tanto no lado cliente quanto no lado servidor, acessando o banco de dados via JPA.

2º Procedimento | Servidor Completo e Cliente Assíncrono

---> CÓDIGOS UTILIZADOS

```
CadastroClientV2.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 cadastroclientv2;
/**
 *
 * @author victorcosta
 */
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
public class CadastroClientV2 {
public static void main(String[] args) {
try {
// Instanciar um Socket apontando para localhost, na porta 4321
Socket socket = new Socket("localhost", 4321);
// Encapsular os canais de entrada e saída do Socket em objetos dos tipos
// ObjectOutputStream (saída) e ObjectInputStream (entrada)
ObjectOutputStream output = new ObjectOutputStream(socket.getOutputStream());
ObjectInputStream input = new ObjectInputStream(socket.getInputStream());
// Escrever o login e a senha na saída, utilizando os dados de algum dos registros
// da tabela de usuários (op1/op1)
String login = "op1";
String senha = "op1";
output.writeObject(login);
output.writeObject(senha);
// Encapsular a leitura do teclado em um BufferedReader
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
// Instanciar a janela para apresentação de mensagens
System.out.println("Bem-vindo ao sistema de cadastro!");
// Instanciar a Thread para preenchimento assíncrono
// (not implemented in this example, as it's not clear what this thread should do)
while (true) {
// Apresentar um menu com as opções: L– Listar, X– Finalizar, E– Entrada, S
Saída
System.out.println("L- Listar | X- Finalizar | E- Entrada | S- Saída");
// Receber o comando a partir do teclado
String command = reader.readLine();
// Tratar os comandos
if (command.equals("L")) {
output.writeObject("L");
try {
System.out.println("Produtos");
String[] produtoNames = (String[]) input.readObject();
for (String nome : produtoNames) {
System.out.println(nome);
}
} catch (ClassNotFoundException e) {
System.err.println("Error ClassNotFoundException: " + e.getMessage());
}
}
```

```

} else if (command.equals("X")) {
break;
} else if (command.equals("E") || command.equals("S")) {
output.writeObject(command);
// Obter o Id da pessoa via teclado e enviar para o servidor
System.out.print("Id da pessoa: ");
String pessoalId = reader.readLine();
output.writeObject(pessoalId);
// Obter o Id do produto via teclado e enviar para o servidor
System.out.print("Id do produto: ");
String produtolId = reader.readLine();
output.writeObject(produtolId);
// Obter a quantidade via teclado e enviar para o servidor
System.out.print("Quantidade: ");
String quantidade = reader.readLine();
output.writeObject(quantidade);
// Obter o valor unitário via teclado e enviar para o servidor
System.out.print("Valor unitário: ");
String valorUnitario = reader.readLine();
output.writeObject(valorUnitario);
}
}

socket.close();
} catch (IOException e) {
System.err.println("Erro ao conectar ao servidor: " + e.getMessage());
}
}
}

CadastroServer.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 cadastroserver;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import controller.PessoaJpaController;
import controller.MovimentoJpaController;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.IOException;
public class CadastroServer {
public static void main(String[] args) {
// Instanciar um objeto do tipo EntityManagerFactory a partir da unidade de persistência
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("CadastroServerPU");
// Instanciar o objeto ctrl, do tipo ProdutoJpaController
ProdutoJpaController ctrl = new ProdutoJpaController(emf);
// Instanciar o objeto ctrlUsu do tipo UsuarioJpaController
UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
// Instanciar o objeto ctrlUsu do tipo UsuarioJpaController
MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
// Instanciar o objeto ctrlUsu do tipo UsuarioJpaController
PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);
ServerSocket serverSocket = null;
// Instanciar um objeto do tipo ServerSocket, escutando a porta 4321
try {
serverSocket = new ServerSocket(4321);
} catch (IOException e) {
System.err.println("Error Server: " + e.getMessage());
}
System.out.println("Servidor iniciado. Aguardando conexões...");
while (true) {
// Obter a requisição de conexão do cliente
Socket socket = null;
try {
socket = serverSocket.accept();
} catch (IOException e) {
System.err.println("Error Socket: " + e.getMessage());
}
System.out.println("Nova conexão estabelecida.");
}
}

```

```

// Instanciar uma Thread, com a passagem de ctrl, ctrlUsu e do Socket da conexão
CadastroThreadV2 thread = new CadastroThreadV2(ctrl, ctrlUsu, ctrlMov, ctrlPessoa,
socket);
// Iniciar a Thread
thread.start();
}
}
}
}

SaidaFrame.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;
/***
*
* @author victorcosta
*/
import javax.swing.JDialog;
import javax.swing.JTextArea;
public class SaidaFrame extends JDialog {
public JTextArea texto;
public SaidaFrame() {
// Define the dimensions of the window
setBounds(100, 100, 400, 300);
// Set the modal status to true
setModal(false);
// Create a JTextArea and add it to the window
texto = new JTextArea();
add(texto);
}
}

CadastroThreadV2.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;
/***
*
* @author victorcosta
*/
import model.Usuario;
import model.Movimento;
import model.Produto;
import java.util.*;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
public class CadastroThreadV2 extends Thread {
private ProdutoJpaController ctrlProd;
private UsuarioJpaController ctrlUsu;
private MovimentoJpaController ctrlMov;
private PessoaJpaController ctrlPessoa;
private Socket s1;
private ObjectOutputStream out;
private ObjectInputStream in;
private SaidaFrame saidaFrame;
public CadastroThreadV2(ProdutoJpaController ctrlProd, UsuarioJpaController ctrlUsu,
MovimentoJpaController ctrlMov, PessoaJpaController ctrlPessoa,
Socket s1) {
this.ctrlProd = ctrlProd;
this.ctrlUsu = ctrlUsu;
this.ctrlMov = ctrlMov;
this.ctrlPessoa = ctrlPessoa;
this.s1 = s1;
this.saidaFrame = new SaidaFrame();
try {

```



```
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Fim da conexão" + e);
    } finally {
        try {
            s1.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

UsuarioJpaController.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 controller;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import javax.persistence.NoResultException;
import model.Usuario;
/**
 *
 * @author victorcosta
 */
public class UsuarioJpaController {
    private EntityManager em;
    public UsuarioJpaController(EntityManagerFactory emf) {
        this.em = emf.createEntityManager();
    }
    public Usuario findUsuario(String login, String senha) {
        try {
            Query query = em.createNamedQuery("Usuario.findByLoginSenha");
            query.setParameter("login", login);
            query.setParameter("senha", senha);
            return (Usuario) query.getSingleResult();
        } catch (NoResultException e) {
            return null;
        }
    }
}

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 controller;
/**
 *
 * @author victorcosta
 */
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.NoResultException;
import javax.persistence.Query;
import model.Produto;
public class ProdutoJpaController {
    private EntityManager em;
    private EntityManagerFactory emf;
    public ProdutoJpaController(EntityManagerFactory emf) {
        this.emf = emf;
        this.em = emf.createEntityManager();
    }
    public List<Produto> findAll() {
        EntityManager em = emf.createEntityManager();
        try {
            return em.createNamedQuery("Produto.findAll").getResultList();
        } finally {
            em.close();
        }
    }
}
```

```
public Produto findProduto(int idProduto) {
try {
Query query = em.createNamedQuery("Produto.findByIdProduto");
query.setParameter("idProduto", idProduto);
return (Produto) query.getSingleResult();
} catch (NoResultException e) {
return null;
}
}
public void incrementQuantidade(int idProduto, int quantidade) {
em=emf.createEntityManager();
try {
Produto produto = findProduto(idProduto);
if (produto != null) {
produto.setQuantidade(produto.getQuantidade() + quantidade);
em.getTransaction().begin();
em.merge(produto);
em.getTransaction().commit();
}
} catch (Exception e) {
em.getTransaction().rollback();
throw e;
}
}
public void decrementQuantidade(int idProduto, int quantidade) {
em=emf.createEntityManager();
try {
Produto produto = findProduto(idProduto);
if (produto != null) {
produto.setQuantidade(produto.getQuantidade()- quantidade);
em.getTransaction().begin();
em.merge(produto);
em.getTransaction().commit();
}
} catch (Exception e) {
em.getTransaction().rollback();
throw e;
}
}
}
PessoaJpaController.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 controller;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.NoResultException;
import javax.persistence.Query;
import model.Pessoa;
/**
 *
 * @author victorcosta
 */
public class PessoaJpaController {
private EntityManager em;
public PessoaJpaController(EntityManagerFactory emf) {
this.em = emf.createEntityManager();
}
public Pessoa findPessoa(int idPessoa) {
try {
Query query = em.createNamedQuery("Pessoa.findByIdPessoa");
query.setParameter("idPessoa", idPessoa);
return (Pessoa) query.getSingleResult();
} catch (NoResultException e) {
return null;
}
}
}
```

```

MovimentoJpaController.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 controller;
import java.io.Serializable;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import model.Movimento;
/**
 *
 *
 * @author victorcosta
 */
public class MovimentoJpaController implements Serializable {
private EntityManager em;
private EntityManagerFactory emf;
public MovimentoJpaController(EntityManagerFactory emf) {
this.emf = emf;
}
public void create(Movimento movimento) {
em=emf.createEntityManager();
try {
em.getTransaction().begin();
em.persist(movimento);
em.getTransaction().commit();
} catch (Exception e) {
System.err.println("movimento" + e);
em.getTransaction().rollback();
throw e;
}
}
}

```

---> RESULTADO

String comando = (String) in.readObject();
if (comando.equals("L")) {
List<Produto> produtos = ctrlProd.findAll();
Produto[] produtoArray = produtos.toArray(new Produto[produtos.size()]);
String[] produtoNames = new String[produtoArray.length];
for (int i = 0; i < produtoArray.length; i++) {
produtoNames[i] = produtoArray[i].getNome();
saídaFrame.texto.append((String) produtoNames[i]);
}
}

String comando = (String) in.readObject();
if (comando.equals("C")) {
ctrlProd.create(new Produto(Integer.parseInt(pessoa), Integer.parseInt(produto), Integer.parseInt(quantidade), Double.parseDouble(valor)));
}
}

for (int i = 0; i < produtoArray.length; i++) {
produtoNames[i] = produtoArray[i].getNome();
saídaFrame.texto.append((String) produtoNames[i]);
}
}

Output X

CadastroServer (run) × CadastroClientV2 (run) ×

L - Listar | X - Finalizar | E - Entrada | S - Saída

Id da pessoa: 1
Id do produto: 1
Quantidade: 122
Valor unitário: 122

L - Listar | X - Finalizar | E - Entrada | S - Saída

---> ANÁLISE E CONCLUSÃO

A. Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?

O uso de threads em aplicações de servidor permite o tratamento assíncrono das respostas, melhorando a eficiência e a escalabilidade do sistema. Através da Criação de Threads no Servidor, como Threads para cada requisição, ou utilizar as Thread Pools, para reutilizar threads existentes e gerenciar eficientemente os recursos do sistema. Algumas das vantagens do Tratamento Assíncrono com Threads, são a melhor utilização de recursos, escalabilidade e responsividade.

B. Para que serve o método invokeLater, da classe SwingUtilities?

O método invokeLater, da classe SwingUtilities, é utilizado para garantir que um trecho de código seja executado no thread de despacho de eventos do Swing. Isso é essencial em aplicações Swing para garantir que atualizações na interface gráfica do usuário (GUI) sejam feitas de maneira segura e responsiva.

C. Como os objetos são enviados e recebidos pelo Socket Java?

Para enviar e receber objetos através de um Socket em Java, utiliza-se as classes ObjectOutputStream e ObjectInputStream. Esses streams permitem a serialização e desserialização de objetos, convertendo-os em um fluxo de bytes para transmissão e reconvertendo-os no lado do receptor.

D. Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento

Característica	Comportamento Síncrono	Comportamento Assíncrono
Bloqueio	Bloqueia o thread até a conclusão da operação	Não bloqueia, permite continuar a execução
Simplicidade	Mais simples de implementar e entender	Mais complexo, exige manejo de callbacks e eventos
Eficiência	Menos eficiente, thread ocioso enquanto aguarda	Mais eficiente, melhor uso de recursos
Uso	Adequado para operações rápidas ou bloqueio tolerável	Ideal para operações lentas e tarefas paralelas