Estácio - Unidade São pedro
RJ 140 Km 2, 512 loja 1, São Pedro da Aldeia - RJ, 28941-182

Desenvolvimento Full Stack
Classe: Missão Prática | Nível 5 | Mundo 3
3º Semestre
Marvin de Almeida Costa

Título da Prática: **2º Procedimento | Servidor Completo e Cliente Assíncrono**

Objetivos da prática:

- Criar servidores Java com base em Sockets.
- Criar clientes síncronos para servidores com base em Sockets.
- Criar clientes assíncronos para servidores com base em Sockets.
- Utilizar Threads para implementação de processos paralelos.
- 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.

Todos os códigos solicitados neste roteiro de aula:

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 marvin
 */

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 pessoaId = reader.readLine();
            output.writeObject(pessoaId);

            // Obter o Id do produto via teclado e enviar para o servidor
            System.out.print("Id do produto: ");
            String produtoId = reader.readLine();
            output.writeObject(produtoId);

            // 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 marvin
 */
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 marvin
 */

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 {
            out = new ObjectOutputStream(s1.getOutputStream());
            in = new ObjectInputStream(s1.getInputStream());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        try {
            saidaFrame.setVisible(true);

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

            Usuario usuario = ctrlUsu.findUsuario(login, senha);
```

```
        if (usuario == null) {
            s1.close();
            return;
        }

        saidaFrame.texto.append((String) "Nova conexão estabelecida" + "\n");

        while (true) {
            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();
                    saidaFrame.texto.append((String) produtoArray[i].getNome() + "::" +
produtoArray[i].getPrecoVenda() + "\n");
                }
                out.writeObject(produtoNames);
            } else if (comando.equals("E") || comando.equals("S")) {
                Movimento movimento = new Movimento();
                movimento.setIdUsuario(usuario);
                char caracterComando = comando.charAt(0);
                movimento.setTipo(caracterComando);

                String idPessoa = (String) in.readObject();
                int idIntPessoa = 0;
                try {
                    idIntPessoa = Integer.parseInt(idPessoa);
                } catch (NumberFormatException e) {
                    System.err.println("Number" + idPessoa);
                }
                movimento.setIdPessoa(ctrlPessoa.findPessoa(idIntPessoa));

                String idProduto = (String) in.readObject();
                int idIntProduto = 0;
                try {
                    idIntProduto = Integer.parseInt(idProduto);
                } catch (NumberFormatException e) {
                    System.err.println("Number" + idProduto);
                }
                movimento.setIdProduto(ctrlProd.findProduto(idIntProduto));

                String quantidade = (String) in.readObject();
```

```
                int intQuantidade = 0;
                try {
                    intQuantidade = Integer.parseInt(quantidade);
                } catch (NumberFormatException e) {
                    System.err.println("Number" + quantidade);
                }
                movimento.setQuantidade(intQuantidade);

                String valorUnitario = (String) in.readObject();
                double valorUnitarioDouble = 0.0;
                try {
                    valorUnitarioDouble = Double.parseDouble(valorUnitario);
                } catch (NumberFormatException e) {
                    System.err.println("Number" + valorUnitario);
                }
                movimento.setValorUnitario(valorUnitarioDouble);

                ctrlMov.create(movimento);

                if (comando.equals("E")) {
                    ctrlProd.incrementQuantidade(idIntProduto, intQuantidade);
                } else {
                    ctrlProd.decrementQuantidade(idIntProduto, intQuantidade);
                }
            }
        }
    } 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 marvin
 */
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 marvin
 */
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 marvin
 */

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 marvin
 */
public class MovimentoJpaController implements Serializable {

    private EntityManager em;
```

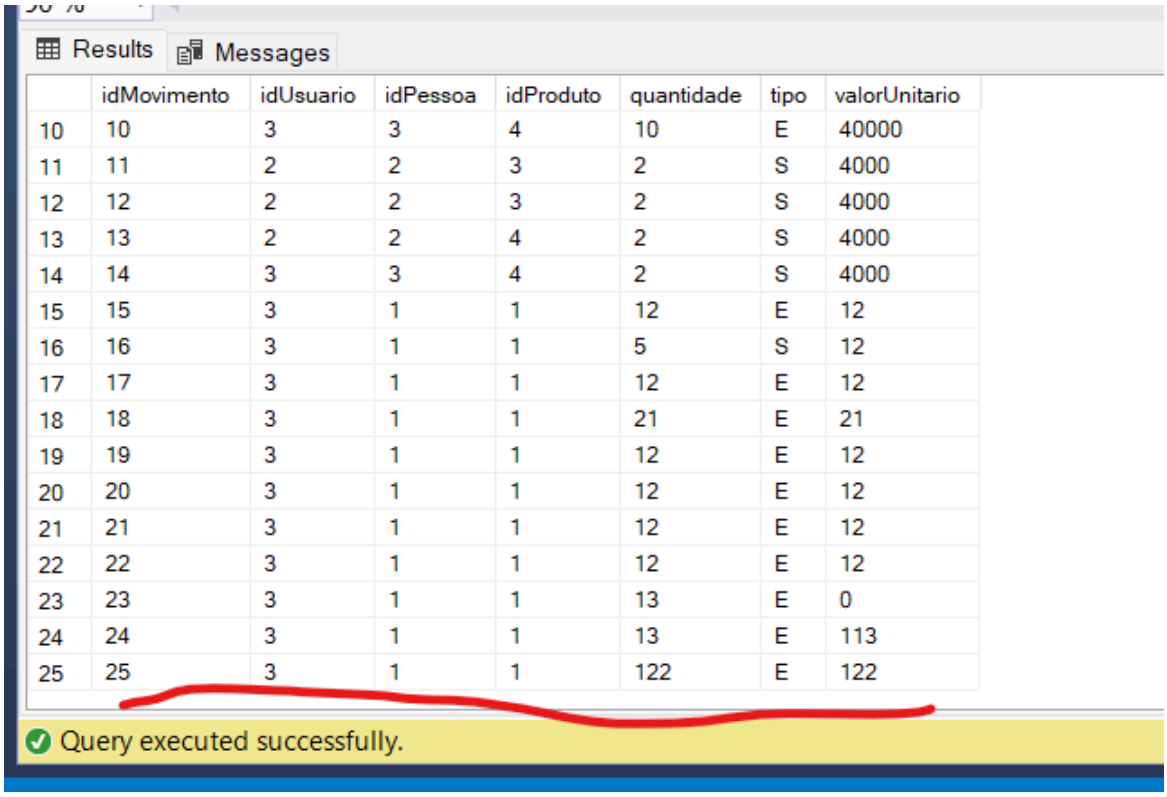
```
    private EntityManagerFactory emf;

    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;
        }
    }

}
```

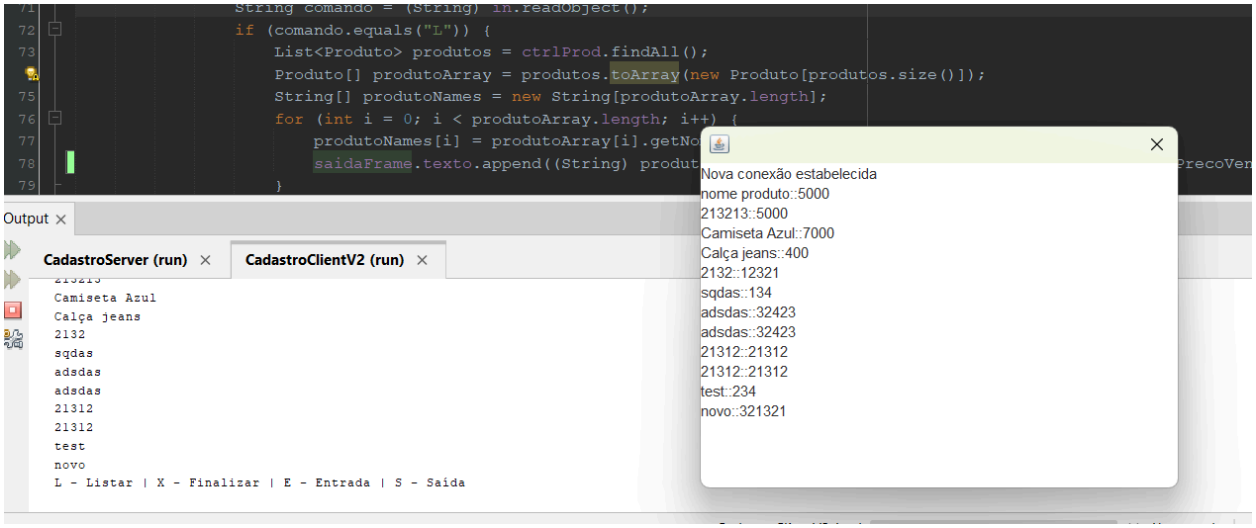
Os resultados da execução dos códigos também devem ser apresentados;

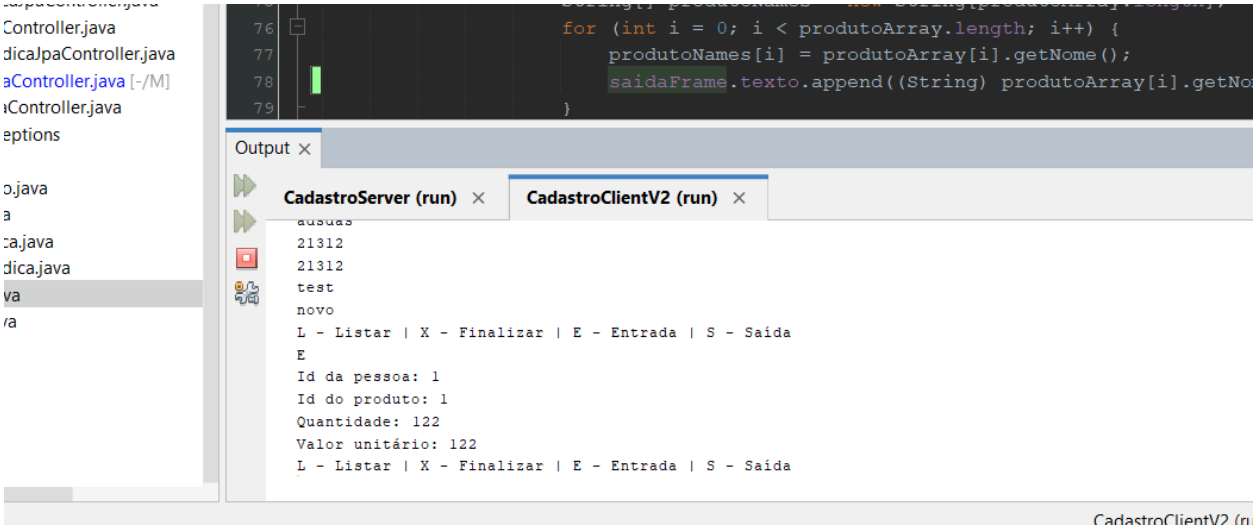


Results | Messages

	idMovimento	idUsuario	idPessoa	idProduto	quantidade	tipo	valorUnitario
10	10	3	3	4	10	E	40000
11	11	2	2	3	2	S	4000
12	12	2	2	3	2	S	4000
13	13	2	2	4	2	S	4000
14	14	3	3	4	2	S	4000
15	15	3	1	1	12	E	12
16	16	3	1	1	5	S	12
17	17	3	1	1	12	E	12
18	18	3	1	1	21	E	21
19	19	3	1	1	12	E	12
20	20	3	1	1	12	E	12
21	21	3	1	1	12	E	12
22	22	3	1	1	12	E	12
23	23	3	1	1	13	E	0
24	24	3	1	1	13	E	113
25	25	3	1	1	122	E	122

Query executed successfully.





Análise e Conclusão:

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

As threads são uma ferramenta poderosa para implementar tratamento assíncrono em aplicações de rede, especialmente quando lidamos com respostas do servidor.

Em programação concorrente, uma thread é uma unidade de execução independente dentro de um processo. Elas permitem que múltiplas tarefas sejam executadas simultaneamente, melhorando a eficiência e a resposta da aplicação.

Para o tratamento assíncrono de respostas do servidor, as threads podem ser usadas para:

- Processar múltiplas requisições simultaneamente
- Reduzir a latência da resposta
- Melhorar a escalabilidade da aplicação

Para que serve o método invokeLater, da classe SwingUtilities?

O método invokeLater da classe SwingUtilities é uma ferramenta importante na biblioteca Java Swing para garantir que operações de interface gráfica sejam executadas de forma segura e eficiente em aplicações multithreaded.

Propósito do invokeLater
O principal propósito do invokeLater é permitir que você execute código em um thread específico da interface gráfica (Event Dispatch Thread - EDT) sem bloquear a execução atual do programa.

- Ele garante que todas as interações com componentes GUI ocorram no thread correto, evitando problemas de concorrência e instabilidade.

- Isso é particularmente útil quando você precisa atualizar a UI ou realizar operações que dependem da resposta do usuário.

Como os objetos são enviados e recebidos pelo Socket Java?

Criação de Sockets

Primeiramente, é necessário criar um socket no lado do cliente e do servidor. No lado do cliente, você usa Socket para conectar-se ao servidor, enquanto no servidor, você usa ServerSocket para aceitar conexões.

Serialização de Objetos

Para enviar objetos através de um socket, eles precisam ser serializados. A serialização converte um objeto em uma sequência de bytes que pode ser facilmente transmitida via rede.

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

A escolha entre o uso de comportamentos assíncronos ou síncronos em clientes que interagem com sockets em Java depende de vários fatores, incluindo o tipo de aplicação, os requisitos de desempenho e a necessidade de resposta rápida. Vamos comparar esses dois abordagens, focando nas características relacionadas ao bloqueio do processamento.

Comportamento Síncrono

No comportamento síncrono, o cliente espera por uma resposta antes de prosseguir com outras operações. Este é o método mais simples e direto para interagir com sockets em Java.

Características relacionadas ao bloqueio do processamento:

- Bloqueio total: O thread principal do cliente fica bloqueado até que a operação seja concluída.
- Menor sobrecarga de memória: Não há necessidade de gerenciar múltiplos threads ou estados assíncronos.
- Fácil implementação: Requer menos código e lógica para lidar com as respostas.
- Potencial para problemas de desempenho: Se a operação demorar muito, pode causar um tempo de resposta longo para o usuário.

Comportamento Assíncrono

O comportamento assíncrono permite que o cliente continue executando outras operações enquanto aguarda uma resposta do servidor.

Características relacionadas ao bloqueio do processamento:

- Minimização de bloqueios: O thread principal não fica bloqueado, permitindo que outras operações sejam realizadas simultaneamente.
- Melhor uso de recursos: Reduz a sobrecarga de CPU e memória, especialmente em aplicações com muitas requisições concorrentes.
- Maior flexibilidade: Permite que o cliente execute outras tarefas enquanto espera por uma resposta.
- Complexidade aumentada: Requer gerenciamento de estado assíncrono e tratamento de callbacks.
- Possível sobreposição de operações: Pode levar a situações onde várias operações são iniciadas sem serem concluídas.