Missão Prática | Nível 5 | Mundo 3

João Rainier de Castro Carvalho (202208942661)

Estácio Distrito Federal (Polo Asa Sul)/DF – Desenvolvimento Full Stack – 2022.3 – Mundo 3
Link GitHub: *Faculdade-Estacio-3-semestre/Nível 5 - Por Que Não Paralelizar at main · joaorainier/Faculdade-Estacio-3-semestre (github.com)*

RPG0018 - 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.

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.

Materiais necessários para a prática:
SQL Server, com o banco de dados gerado em prática anterior (loja).
JDK e IDE NetBeans.
Navegador para Internet, como o Chrome.
Banco de dados SQL Server com o Management Studio.
Equipamentos:
- **Computador com acesso à Internet.**
- **JDK e IDE NetBeans.**
- **Banco de dados SQL Server.**
- **Navegador de Internet instalado no computador.**

1 – Como funcionam as classes Socket e ServerSocket?

Essas classes são fundamentais para a construção de aplicações cliente-servidor em Java e fornecem a base para a implementação de protocolos de comunicação mais complexos. O Socket é usado para a comunicação efetiva, enquanto o ServerSocket é usado para aguardar e aceitar conexões entrantes.

2 - Qual a importância das portas para a conexão com servidores?

As portas desempenham um papel fundamental na comunicação entre dispositivos em redes, especialmente na internet. Elas são essenciais para direcionar o tráfego de dados para os serviços corretos em um servidor.
as portas são componentes cruciais para a organização e eficiência da comunicação em redes. Elas permitem a identificação e roteamento eficaz do tráfego, garantindo que os dados sejam entregues aos serviços corretos nos servidores. O entendimento e o gerenciamento adequado das portas são essenciais para manter a segurança e o funcionamento suave de sistemas e redes.

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

As classes ObjectInputStream e ObjectOutputStream em Java são utilizadas para realizar a serialização e desserialização de objetos. A serialização é o processo de converter objetos em uma sequência de bytes, enquanto a desserialização é o processo inverso, convertendo essa sequência de bytes de volta para objetos. Essas classes são frequentemente usadas para transmitir objetos através de uma rede, armazenar objetos em arquivos, ou para comunicação entre processos.
É importante notar que a interface Serializable em Java é uma marcação e não contém métodos a serem implementados. Quando uma classe implementa Serializable, ela indica que está pronta para ser serializada e desserializada pelo mecanismo padrão de serialização Java.

4 - Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?

O isolamento do acesso ao banco de dados ao utilizar classes de entidades JPA (Java Persistence API) no lado do cliente pode ser alcançado principalmente por meio do conceito de "Entidade Gerenciada" e da "Unidade de Persistência". A JPA oferece um mecanismo de persistência que permite às aplicações Java interagir com bancos de dados relacionais de maneira mais eficiente, abstrai o acesso ao banco de dados e fornece um gerenciamento efetivo do ciclo de vida das entidades.
Ao utilizar entidades JPA dessa maneira, você aproveita o gerenciamento de ciclo de vida, a persistência transparente, a isolamento transacional e outros recursos fornecidos pela JPA. O uso adequado desses conceitos permite que o acesso ao banco de dados seja efetuado de maneira isolada e eficiente, enquanto a JPA lida com muitos detalhes complexos relacionados à persistência de dados. Isso facilita o desenvolvimento de aplicações robustas e escaláveis.

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

As threads podem ser utilizadas para tratar respostas assíncronas enviadas pelo servidor em aplicações Java. A ideia é executar determinadas tarefas em threads separadas para não bloquear a thread principal, permitindo que a aplicação continue a responder a eventos, aceitar entradas do usuário, e realizar outras operações enquanto aguarda respostas assíncronas do servidor. Essas abordagens permitem que tarefas assíncronas sejam realizadas em paralelo com a thread principal, proporcionando uma experiência mais responsiva para o usuário enquanto aguarda respostas do servidor. A escolha entre essas abordagens depende do contexto da aplicação (console, aplicação desktop, aplicação web, etc.) e dos requisitos específicos.

6 - Para que serve o método invokeLater, da classe SwingUtilities?

O método invokeLater da classe SwingUtilities em Java é utilizado para agendar a execução de uma tarefa (Runnable) na Event Dispatch Thread (EDT) do Swing. A EDT é uma thread especial dedicada à manipulação de eventos relacionados à interface gráfica do usuário (GUI) em aplicações Swing. É importante realizar operações relacionadas à interface gráfica na EDT para garantir a consistência e a segurança na manipulação de componentes Swing. Em resumo, invokeLater é uma ferramenta essencial para garantir a execução de tarefas relacionadas à GUI na EDT, proporcionando um ambiente seguro e consistente para o desenvolvimento de aplicações Swing.

7 - Como os objetos são enviados e recebidos pelo Socket Java?

Em Java, a comunicação entre sistemas distribuídos via sockets pode ser estabelecida usando as classes Socket e ServerSocket para implementar o lado do cliente e do servidor, respectivamente. A comunicação pode envolver a troca de objetos entre o cliente e o servidor, e isso geralmente é feito por meio de ObjectInputStream e ObjectOutputStream para serializar e desserializar objetos.

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

A utilização de comportamento assíncrono ou síncrono em clientes Java que usam sockets tem implicações significativas em termos de bloqueio de processamento.
A escolha entre comportamento síncrono e assíncrono depende dos requisitos específicos da aplicação e das considerações de desempenho. Em resumo:

Síncrono: Mais fácil de entender, mas pode resultar em bloqueios que afetam a responsividade da aplicação.
Assíncrono: Mais eficiente em termos de utilização de recursos, permite maior concorrência, mas pode aumentar a complexidade do código.
Frameworks modernos, como Netty ou CompletableFuture em Java, oferecem opções para operações assíncronas e facilitam a implementação de lógica assíncrona de maneira mais gerenciável. A escolha dependerá dos requisitos específicos do projeto e das preferências do desenvolvedor.