Anna Eliza de Almeida Silva – 202208837093

Polo Santa Cruz – Rio de Janeiro – RJ

Semestre 2022.3

RPG0018 – Por que não paralelizar

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, alem 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.

Análise e conclusão

- **Como funciona as classes Socket e ServerSocket?** O ServerSocket é empregado no servidor para aguardar e aceitar conexões na rede, ao passo que o Socket é utilizado no cliente para estabelecer uma conexão com o servidor. Ambas as classes possibilitam a comunicação bidirecional entre aplicativos por meio de fluxos de dados. Quando a conexão é estabelecida com êxito, essas duas entidades podem se comunicar utilizando o InputStream do objeto Socket para receber dados e enviando informações de volta por meio do OutputStream.
- **Qual a importância das portas para a conexão com servidores?** Portas são fundamentais para as conexões com servidores, pois associam serviços a números específicos, permitindo a operação simultânea de diversos serviços em um mesmo servidor. Essa abordagem simplifica o roteamento preciso do tráfego, otimizando a comunicação entre sistemas. Além disso, as portas desempenham um papel crucial na segurança, possibilitando a aplicação de medidas de controle de acesso com base nas portas designadas para cada serviço, fortalecendo assim a integridade do sistema.
- **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 são ferramentas cruciais no universo Java, proporcionando a capacidade de serialização e desserialização de objetos. Essa funcionalidade é essencial para a transmissão de objetos entre sistemas distribuídos, persistência de dados em arquivos e, em última análise, para garantir a integridade e eficácia da comunicação entre objetos em ambientes diversos.
- **Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?** O uso das classes de entidades JPA no cliente proporciona isolamento do acesso ao banco de dados devido ao mapeamento objeto-relacional, ao gerenciamento de transações, ao uso de caches e à estratégia de lazy loading. Essas características combinadas criam uma camada de abstração que permite ao

cliente interagir com o banco de dados de maneira eficiente, enquanto o JPA lida com os detalhes da comunicação com o banco de dados.

Análise e conclusão

- **Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?** A incorporação de Threads para o tratamento assíncrono de respostas do servidor representa uma estratégia eficaz na gestão de operações de rede em programas clientes. Essa abordagem possibilita que o cliente prossiga com a execução de diferentes tarefas enquanto espera por uma resposta do servidor. A chave para esse cenário reside na execução das operações de rede em Threads independentes, evitando assim o bloqueio da Thread principal do cliente. Ao adotar essa prática, a resposta do servidor pode ser processada ou comunicada ao usuário pela Thread principal no momento em que estiver disponível. Enquanto isso, outras Threads podem continuar a desempenhar suas respectivas tarefas, resultando em uma melhoria significativa na eficiência e na responsividade global do programa. Essa abordagem dinâmica e paralela proporciona uma experiência mais fluida ao usuário, pois reduz o impacto das operações de rede no fluxo principal de execução, tornando o programa mais ágil e capaz de realizar múltiplas atividades simultaneamente.
- **Para que serve o método invokeLater, da classe SwingUtilities?** O método invokeLater da classe SwingUtilities em Java é utilizado para executar uma tarefa Swing de forma assíncrona na Thread de despacho de eventos Swing (EDT). Isso é especialmente útil quando se deseja atualizar a interface gráfica do usuário (GUI) a partir de uma Thread diferente da EDT, evitando problemas de concorrência e garantindo a consistência na interação com componentes Swing. O invokeLater agenda a execução da tarefa na EDT, permitindo uma comunicação segura e eficiente com a GUI em ambientes multithreaded.
- **Como os objetos são enviados e recebidos pelo Socket Java?** Em Java, a comunicação entre objetos por meio de sockets envolve a serialização e desserialização dos objetos. Para enviar um objeto por um Socket, é necessário convertê-lo em uma sequência de bytes por meio do processo de serialização. Isso é feito geralmente usando a interface Serializable. Do lado receptivo, o objeto é reconstruído a partir da sequência de bytes recebida usando o processo de desserialização. Isso permite a transmissão eficiente de objetos entre máquinas distintas por meio de sockets, facilitando a comunicação em redes.
- **Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.** A diferença fundamental entre o comportamento assíncrono e síncrono nos clientes com Socket Java está relacionada ao bloqueio do processamento. Em operações síncronas, o cliente aguarda ativamente a conclusão de uma operação antes de continuar, o que pode resultar em bloqueios, especialmente em redes lentas. Por outro lado, no comportamento assíncrono, o cliente pode continuar executando outras tarefas enquanto aguarda a conclusão da operação de Socket. Isso evita bloqueios desnecessários, melhorando a eficiência e a responsividade do programa, especialmente em situações onde a latência de rede pode ser um fator significativo