Prova escrita

Orientação a objetos

- O que é **injeção de dependência**?

  - A injeção de dependência é a declaração de um objeto tipo classe(Exemplo: nomeclasseA) dento de uma classe(nomeclasseB). Podemos dizer que o "nomeclasseB" tem dependência do objeto tipo classe "nomeclasseA", pois para considerar de fato a injecção de dependência é necessário que o objeto "nomeclasseA" inicie a instância somente fora da classe "nomeclasseA" ou seja instanciado na classe "nomeclasseB".

- O que você entende por **SOLID**?

  - É um padrão de planejamento no desenvolvimento do *script(*programa), para torná-lo mais objetivo e de fácil compreendimento para futuras manutenções.

- Você conhece o design pattern **Strategy**? De um exemplo de quando utilizá-lo.

  - Não conhecia até então, mas entendi nesses dias que a Strategy é um padrão que tira a responsabilidade de um atributo e delega essa para diversa classes para obter o mesmo resultado de formar diferentes. Portanto substituindo uma(s) condição(ões) de vários IFs por atributos tipo classe condicionais para deixar o código mais compreensível para futuras manutenções.

Spring Framework

- Quais **subprojetos** da Spring você conhece? Descreva-os brevemente.

  - Spring MVC é voltado para a estruturação do código do sistema através das divisões Model View Control, onde o Fremawork usa configurações de anotações(Exemplo:@Controller) mapear uma classe ou (Exemplo:@RequestMapping) para chamar um metodo tipo request e além de ser usado para acessar tabelas do banco de dados por mapeamento de anotações @Bean/@Autowires.

  - Spring Scurity é voltado para segurança da aplicação Web com controle de acesso na autenticação e autorização das áreas do sistemas. Esse Framework tem por finalidade de simplificar configuração e torna mais compreensível o entendimento de segurança. Para facilitar a leitura esse também pode ser acessado por tags: authorize , authentication, etc. E/ou anotações: @@PreAuthorize , @PostAuthorize, etc.

- Como podemos criar e injetar **beans** no Spring?

  - Para injetar um beans no Spring pode ser declarado no arquivo **applicationContext.**xml a tags *<bean class="pacote.nomeclasse" id="nomeclasse">*

ou na versão mais atual basta colocar esse linha "*<context:component-scan base-package="pacote.classes />*" no mesmo arquivo e incluir anotações na classe *@Component("nomeclasse")*.

- Como usar **transações** no Spring?

  - Para usar a transações por anotações é preciso incluir a tag: <tx:annotation-driven transaction-manager="transactionManager"/> no arquivo **applicationContext.**xml e após definiremos os metodos que iram usar pela anotação @Transactional com seus parametros de Isolation e Propagation. As transações são realizadas para validações de possíveis excessões durante uma operação.

  Exemplo:

  ```
  @Transactional(porpagation=Propagation=REQUERED, readOnly=false)
  public void save(){
        //código.....
  }
  ```

Rest

- O que você entende sobre o que são serviços web RESTful?

  - O serviço web RESTful do moledo de arquitetura REST é uma implementação de boas práticas de uso dos métodos HTTP/HTTPS bem definidos pelo protocolo de gerencia dos recurso via rede criado pela RFC 2616.

  Por exemplo:

  HTTP GET: Para recuperar um recurso de um servidor usa o método GET;

  HTTP POST: Para criar um recurso no servidor usa o método POST;

  HTTP PUT: Para altarer um determinado recurso no servidor usa o método PUT;

  HTTP DELET: Para excluir permanentimente um recurso no servidor usa o método DELETE;

- Cite algumas boas práticas envolvendo rest?

  -Usar a cache para ter o melhor desempenho;

  - Usar de maneira correta os método HTTP(GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH e etc;

  - Usar de maneira correta as respostas das operações realizadas pelo cliente/servido 100, 200, 400, 500 e etc, Para melhores tratamentos de possíveis erros.

- Cite os métodos HTTP mais usados na arquitetura baseada REST e o que eles fazem?

  -HTTP GET: Para recuperar um informações de um servidor usa o método GET;

  -HTTP POST: Para criar uma informação no servidor usa o método POST;

  -HTTP PUT: Para altarer uma determinada informação no servidor usa o método PUT;

-HTTP DELET: Para excluir permanentimente uma informação no servidor usa o método DELETE;

- Qual a diferença entre os códigos de resposta (status codes) HTTP 200, 400 e 500?

  -HTTP 200 indica que o estado/operação foi totalmente executado com sucesso e retornou com o resultado esperado;

  -HTTP 400 indica que o estado/operação teve um problema na execução por parte do cliente;

  -HTTP 500 indica que o estado/operação teve um problema na execução por parte do servidor;

Prova prática

1.) Acesse https://github.com/leonardohenrique/tokio-test.git e clone o projeto.

2.) Importe o projeto no seu Eclipse

3.) Rode a aplicação Spring Boot e acesse http://localhost:8080/customers (deverá exibir uma lista de clientes).

4.) Abra a classe **com.example.api.web.rest.CustomerController** e adicione **endpoints** para criar um novo cliente, editar um cliente e excluir um cliente.

5.) Valide os dados antes de cadastrar ou editar

6.) Pagine a listagem de clientes.

7.) Possibilite o cadastro de múltiplos endereços para um cliente.

8.) No cadastro de endereço permita inserir apenas o CEP carregando os dados via consumo do serviço: https://viacep.com.br/

9.) Envie a url do seu repositório no github para análise.

Obs.: Será um diferencial implementações como: tratamento de exceções (RestControllerAdvice), testes, validações, frontend, autenticação e documentação.