10

Para saber mais: Injeção de dependência e contextos

Durante a aula *Falando sobre escopos,* nós vimos como podemos alterar o comportamento da criação da(s) instância(s) do nosso objeto. Falamos um pouco sobre os escopos e como eles funcionam. Mas uma dúvida recorrente é: Quem cria as instâncias?

O Quarkus implementa parte da especificação *CDI (Contexts and Dependency Injection)*. Pra quem não conhece, o CDI é um mecanismo oferecido pela plataforma Java EE para gerenciamento de dependências entre componentes. No CDI nossas classes são tratadas como beans e um bean pode depender de um a *n* beans. Essa dependência é feita através da Injeção de Dependência, que pode ser realizada através do uso de algumas anotações ou arquivos de configuração. A `@Inject` é um exemplo de anotação para injeção de um bean. Quando o objeto injetado precisa ser usado, o servidor de aplicação é o responsável por fornecer a instância. Este é o conceito de Inversão de Controle ou *inversion of control (IoC)*. Dessa maneira o desenvolvedor pode focar nas regras negociais e não em como deve-se inicializar seus objetos e dependências.

Porém como visto anteriormente, essa forma de criação das instâncias pelo servidor, pode ser alterada a depender do escopo do bean. Em linhas gerais, o escopo de um bean determina o ciclo de vida de suas instâncias, ou seja, quando e onde uma instância deve ser criada e destruída. Os escopos permitidos pelo Quarkus são:

`javax.enterprise.context.ApplicationScoped` : Uma única instância do bean é criada e compartilhada por todos os pontos de injeção. A instância é criada de forma "preguiçosa" *(lazily)*, ou seja, quando um método é invocado através de um *client proxy*.

Client proxy: Um proxy de cliente é basicamente um objeto que delega todas as invocações de método para uma instância de bean de destino. É uma construção de *container* que implementa `io.quarkus.arc.ClientProxy` e estende a classe de bean. Os proxies de cliente apenas delegam invocações de método.

`javax.inject.Singleton` : Mesma ideia do `@ApplicationScoped` , porém aqui o *client proxy* não é utilizado, a instância é criada quando um ponto de injeção que resolve para um bean `@Singleton` está sendo injetado.

`javax.enterprise.context.RequestScope` : A instância é associada à requisição, ou seja, uma instância para cada requisição.

`javax.enterprise.context.Dependent` : O ciclo de vida do `@Dependent` é limitado ao bean que o injeta. Ele será criado e destruído com o bean que o injeta.

Para quem quiser saber mais informações, segue o link da documentação oficial do quarkus: https://quarkus.io/guides/cdi#client_proxies (https://quarkus.io/guides/cdi#client_proxies)