JSF Eficaz – Resumo

Uso correto de escopos

@RequestScoped:

A cada requisição, uma nova instância do ManagedBean será criada e usada, dessa maneira, não há o compartilhamento das informações do ManagedBean entre as requisições.

O melhor uso de um ManagedBean no escopo de request é em telas que não necessitam de chamada AJAX, ou em de algum objeto salvo na memória.

Um ManagedBean do tipo RequestScoped é considerado ThreadSafe, ou seja, ele não terá problemas relacionados a vários usuários acessando o mesmo MB ao mesmo tempo. É uma boa utilização na geração de relatório, pois não haveriam duas pessoas usando a mesma instância do ManagedBean e o relatório teria como ter inconsistência de dados. Outra boa utilização seria ao enviar dados para uma outra tela; essa outra poderia exibir todos os dados do objeto presente no request.

@SessionScoped:

Todo atributo de um ManagedBean SessionScoped terá seu valor mantido até o fim da sessão do usuário.

O problema do SessionScoped é que ele é muito fácil de usar. Se um desenvolvedor precisar enviar valor de uma tela para outra, o modo mais fácil de fazer seria salvar esse valor em um ManagedBean do tipo SessionScoped; mas ao mesmo tempo que ele é o mais fácil é também o mais perigoso.

É necessário bastante cuidado ao usá-lo, ainda mais em telas de listagens e cadastro.

@ViewScoped:

Ele tem a característica de existir na memória enquanto o usuário permanecer na página exibida.

Um bom uso para um ManagedBean do tipo ViewScoped é em uma página com diversas chamadas Ajax.

Atente-se a um problema que pode ocorrer com relação a acesso de diversos usuários ao banco de dados. Note uma coisa: o valor que será exibido está armazenado em memória. Imagine que um usuário altere os dados no banco de dados. Nesse caso, se um outro usuário estivesse com essa tela aberta a mais tempo, ele não veria essa alteração.

Para evitar o problema de dados fora de sincronia com o banco de dados, uma rotina de atualização poderia ser executada. A cada 1 minuto uma verificação poderia ser realizada, ou talvez antes de exibir cada registro. Uma outra abordagem seria adotar algum mecanismo ou Framework para tomar conta dessa falta de sincronia. O JPA conta com a opção de verificar se o objeto foi alterado por outra pessoa.

@ConversationScoped:

O ManagedBean do tipo ConversationScoped tem um funcionamento parecido com o ViewScoped. A característica principal do ConversationScoped é que o controle da existência do ManagedBean é feito manualmente.

Para um ManagedBean do tipo ConversationScoped funcionar corretamente ele deve seguir algumas normas:

• Como só pode ser utilizado com CDI, o arquivo beans.xml deve existir dentro da pasta WEB-INF;

• Utilizar a anotação javax.enterprise.context.ConversationScoped na classe;

• Injetar um objeto do tipo javax.enterprise.context.Conversation;

• Chamar os métodos conversation.begin() e conversation.end() para iniciar e finalizar o escopo do ManagedBean.

O ConversationScoped só pode ser utilizado com CDI. Não existe essa opção para os ManagedBeans do pacote javax.faces.bean. Caso a sessão do usuário termine, o ManagedBean do tipo ConversationScoped será eliminado da sessão.

@Dependent:

O escopo Dependent tem como característica não ter um comportamento próprio, mas sim herdar o comportamento de outro ManagedBean em que for injetado.

Esse é o escopo padrão ao se usar JSF com CDI. Não é necessário utilizar a anotação @Dependent na classe.

Quando o @Dependent é utilizado diretamente em uma página, cada EL resultará na criação de um ManagedBean novo. Em geral o escopo Dependent é injetado dentro de outros ManagedBeans, desse modo ele herdará o escopo da classe principal.

Tome bastante cuidado ao injetar um Dependent ManagedBean dentro de um ApplicationScoped.

@ApplicationScoped:

O ApplicationScoped ManagedBean tem o comportamento semelhante ao do padrão de projeto Singleton, mantendo uma única instância de determinado bean na memória.

Não existe individualidade quando se fala de ApplicationScoped. No caso de um ManagedBean ApplicationScoped, todo usuário terá acesso à mesma instância.

O tipo ApplicationScoped está disponível tanto para ManagedBean do pacote javax.faces.bean como para os ManagedBeans que utilizem CDI.

Uma outra observação importante sobre o ApplicationScoped é que as vezes se faz necessário que ele seja iniciado antes de todos ManagedBeans. Isso é possível através da configuração: @ManagedBean(eager = true). Com essa configuração um ApplicationScoped ManagedBean será iniciado antes que qualquer tela da aplicação seja acessada.

@NoneScoped:

O NoneScoped ManagedBean tem por característica servir apenas a uma chamada da EL e depois ser eliminado da memória. Veja que sua utilização difere do RequestScoped quanto a duração das informaçõe., O RequestScoped dura por quantas chamadas de EL forem realizadas durante uma requisição. O NoneScoped será eliminado após uma chamada de EL.

O NoneScoped ManagedBean está disponível através do pacote javax.faces.bean.