


Jakarta EE Web Total

Controle Transacional em EJBs




Softblue
cursos online

Tópicos Abordados



- Transações
 - Commit & Rollback
 - Transações no Java EE
- Container-Managed Transactions
 - Transaction Attributes
- Bean-Managed Transactions
 - UserTransaction
- Extended Persistence Context
- EJBs & Exceções

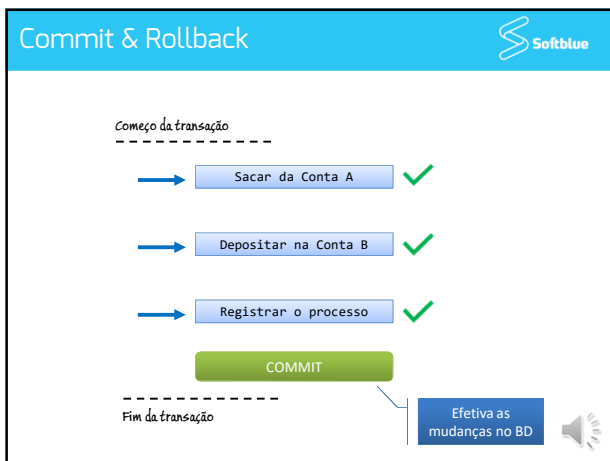
Transações

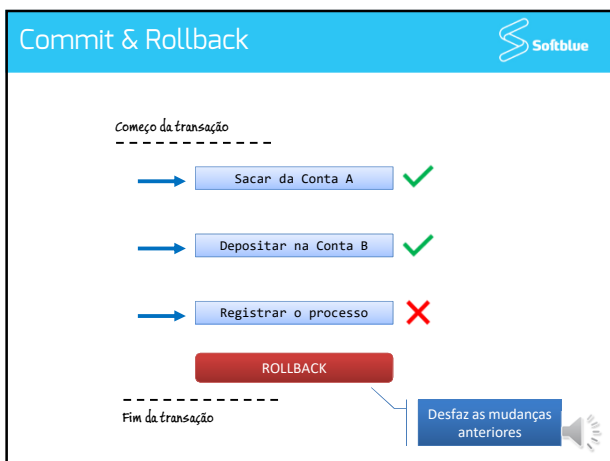


- Uma transação é um conjunto atômico de operações
 - TUDO é executado ou NADA é executado

-> Sacar da Conta A
-> Depositar na Conta B
-> Registrar o processo

} Transação
- O objetivo é manter a consistência das informações





Transações em Java EE

- O Java EE tem um serviço de gerenciamento de transações
- Este gerenciamento pode ser de 2 tipos:
 - CMT (*Container-Managed Transactions*)
 - Gerenciadas pelo EJB container
 - BMT (*Bean-Managed Transactions*)
 - Gerenciadas manualmente, via programação

Softblue logo is in the top right corner.

Container-Managed Transactions



- São gerenciadas pelo container
- Suportadas por EJBs
 - Session Beans e Message-Driven Beans
- A transação começa quando o método inicia e encerra quando o método termina
 - O desenvolvedor não precisa demarcar a transação
- É o tipo padrão de gerenciamento dos EJBs



Exemplo de Uso de CMT



```
@Stateless
public class MyBean {

    @PersistenceContext
    private EntityManager em;

    public void m1() {
        //OP 1
        //OP 2
        //OP 3
    }

    public void m2() {
        //OP 1
        //OP 2
        //OP 3
    }
}
```

Transação

Transação

Se o método terminar normalmente:

COMMIT

Se o método lançar uma *system exception*:

ROLLBACK



Rollback em CMT



- Existem 2 formas de executar um *rollback*
 - Lançar uma *system exception* a partir do método do EJB
 - Chamar o método `setRollbackOnly()`

```
@Stateless
public class MyBean {

    @PersistenceContext
    private EntityManager em;

    @Resource
    private EJBContext ejbContext;

    public void m1() {
        //OP 1
        //OP 2
        ejbContext.setRollbackOnly();
    }
}
```



Transaction Attributes



- Controlam o escopo da transação
 - O que acontece com a transação quando um método chama outro método
- Tipos
 - Required
 - Requires New
 - Mandatory
 - Not Supported
 - Supports
 - Never



Definindo Transaction Attributes



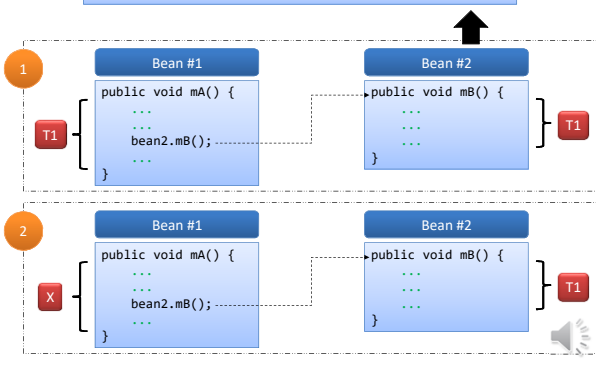
- Uso da anotação `@TransactionAttribute`
 - Na classe
 - Todos os métodos do EJB vão usar este *transaction attribute*
 - No método
 - Apenas o método anotado do EJB vai usar este *transaction attribute*
- Se a anotação existir na classe e no método, o que foi definido no método tem precedência

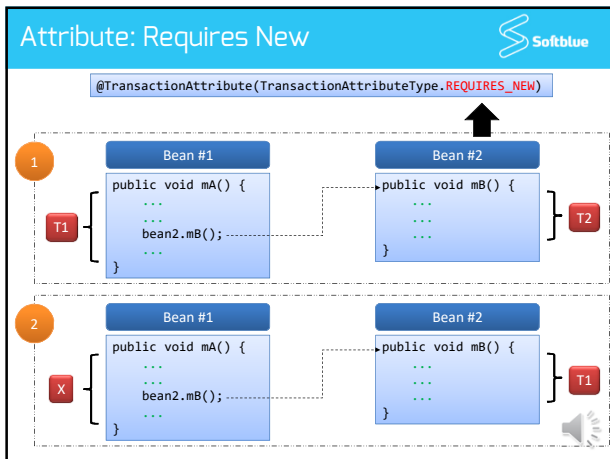


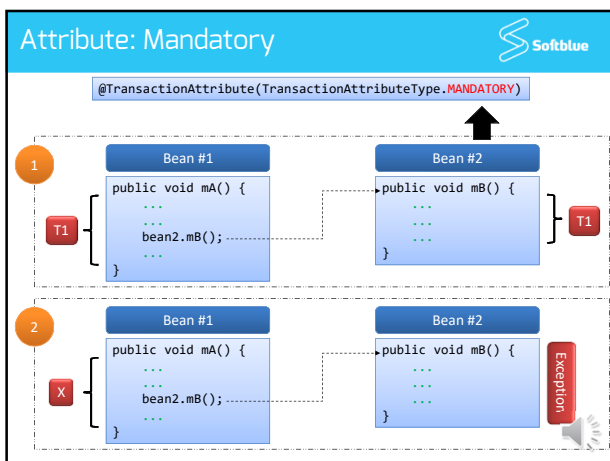
Attribute: Required

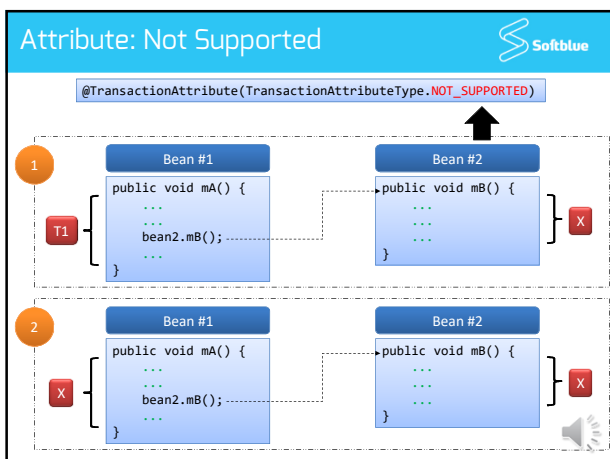


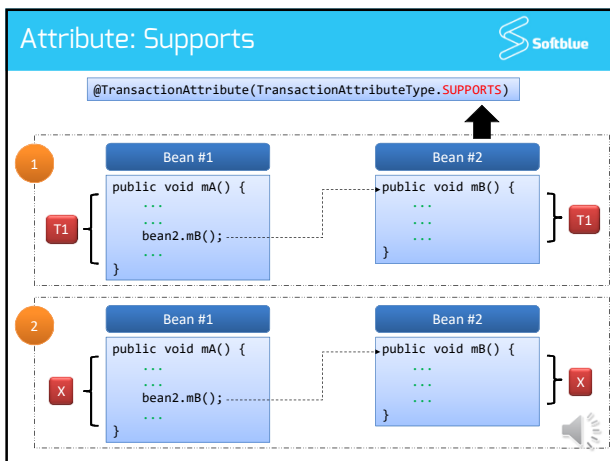
`@TransactionAttribute(TransactionAttributeType.REQUIRED)`

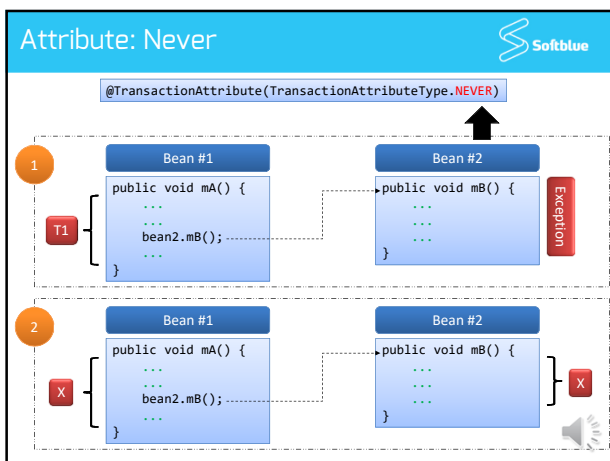













Bean-Managed Transactions

- São gerenciadas pelo desenvolvedor, via programação
- Permitem um controle mais fino a respeito do escopo da transação
 - Métodos que usam CMT são limitados: ou o método inteiro executa em uma transação ou em nenhuma

Exemplo de Uso de BMT



```

@Stateless
@TransactionManagement(TransactionManagementType.BEAN)
public class MyBean {

    @Resource
    private UserTransaction ut;

    public void m1() {
        ut.begin();
        try {
            ...
            ut.commit();
        } catch (Exception e) {
            ut.rollback();
        }
    }
}
    
```

Define o bean como BMT


Gerenciamento da transação

Inicia a transação

Transação

commit() e rollback() encerram a transação

Extended Persistence Context



- O manuseio de entidades na JPA é feito de um contexto de persistência
- Por padrão, esse contexto tem o mesmo tempo de duração da transação

```

@Stateless
public class MyBean {

    @PersistenceContext
    private EntityManager em;


    public void m1() {
        ...
    }
}
    
```

Transação

Assim que a transação termina, o contexto de persistência é encerrado

Todas as entidades gerenciadas pela JPA são desatachadas

Extended Persistence Context



- Esse comportamento pode ser alterado

```

@Stateful
public class MyBean {

    @PersistenceContext(type = PersistenceContextType.EXTENDED)
    private EntityManager em;

    public void m1() {
        ...
    }
}
    
```

Transação

O ciclo de vida do contexto de persistência é o mesmo do EJB

Chamadas subsequentes de métodos usam o mesmo contexto de persistência

Só tem sentido em Stateful Session Beans

EJBs & Exceções



- Existem 2 tipos de exceções
 - System Exception
 - RuntimeException (ou subclasse)
 - RemoteException (ou subclasse)
 - Application Exception
 - As outras que não são *system exceptions*



EJBs & Exceções



- Quando uma exceção é lançada, o que o EJB container faz?
 - System Exception
 - Se houver uma transação ativa
 - Rollback da transação
 - Empacota a exceção em uma exceção do tipo `EJBTransactionRolledBackException` e lança
 - Se não houver uma transação ativa
 - Empacota a exceção em uma exceção do tipo `EJBException` ou `RemoteException` e lança
 - Application Exception
 - Lança a exceção (sem empacotar)



EJBs & Exceções



- É possível fazer com que uma *system exception* se comporte como uma *application exception*

```
@ApplicationException(rollback = true)
public class MyException extends RuntimeException {
    //...
}
```



- Quando um bean lança uma *system exception*, a instância dele é destruída
 - Session Beans
 - Stateful
 - Stateless
 - Message-Driven Beans

