Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
Diretoria Acadêmica de Gestão e Tecnologia da Informação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Transações em Aplicações Corporativas com Java EE

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

Transações

- Necessárias para a implementação de funcionalidades críticas, para as quais se deseja: confiabilidade e integridade dos dados
- Exemplos:
  - Banco: depósito, saque, transferência, etc.
  - Livraria on-line: compra de livros
  - Sistema médico: gerenciamento de consultas, procedimentos, tratamentos e medicamentos
- Composta por uma ou mais tarefas individuais

Transações

- Exemplo de transação: efetuar transferência
  1. identificação das contas: origem e destino
  2. verificação do saldo
  3. saque na conta de origem
  4. depósito da conta de destino
  5. registrar a movimentação nas duas contas

Transações são ACID

- **A**tômica
  - Executa “tudo ou nada”
- **C**onsistente
  - Integridade no armazenamento de dados
- **I**solada
  - Permissão para executar sem interferências
- **D**urável
  - Alterações nos dados de forma persistente

JTA – *Java Transaction API*

- Define a interface *UserTransaction*
  - Recuperam do JNDI (java:comp/UserTransaction)
  - Aplicações utilizam as funcionalidades: “start”, “commit” e “roll back”; exemplo (pseudocódigo):

```
begin transaction
     debit checking account
     credit savings account
     update history log
commit transaction
```

Transações Gerenciadas pelo Contêiner

- É uma das principais vantagens dos EJBs
  - Não é necessária uma demarcação explícita no código, nem fazer uso de uma API específica
- O gerenciamento declarativo de transação, pode ser definido:
  - Através de anotações

```
@javax.ejb.TransactionAttribute
```

  - (ou) Através do descritor de implantação EJB

Transações Gerenciadas pelo Contêiner

- O comportamento transacional de um EJB pode ser alterado em alterar a lógica do negócio do EJB
- Reduz a complexidade das transações para os desenvolvedores
- Torna mais fácil a criação de aplicativos transacionais robustos

Escopo de Transação

- Refere-se aos EJBs (*sessions* e MDBs) que estão participando de uma transação
- Iniciado o escopo de transação, ele é propagado para os envolvidos (EJBs, serviço de mensagens e gerenciador de entidades)
- Em se tratando de EJBs, as tarefas específicas consistem de cada método de um EJB invocado em uma transação

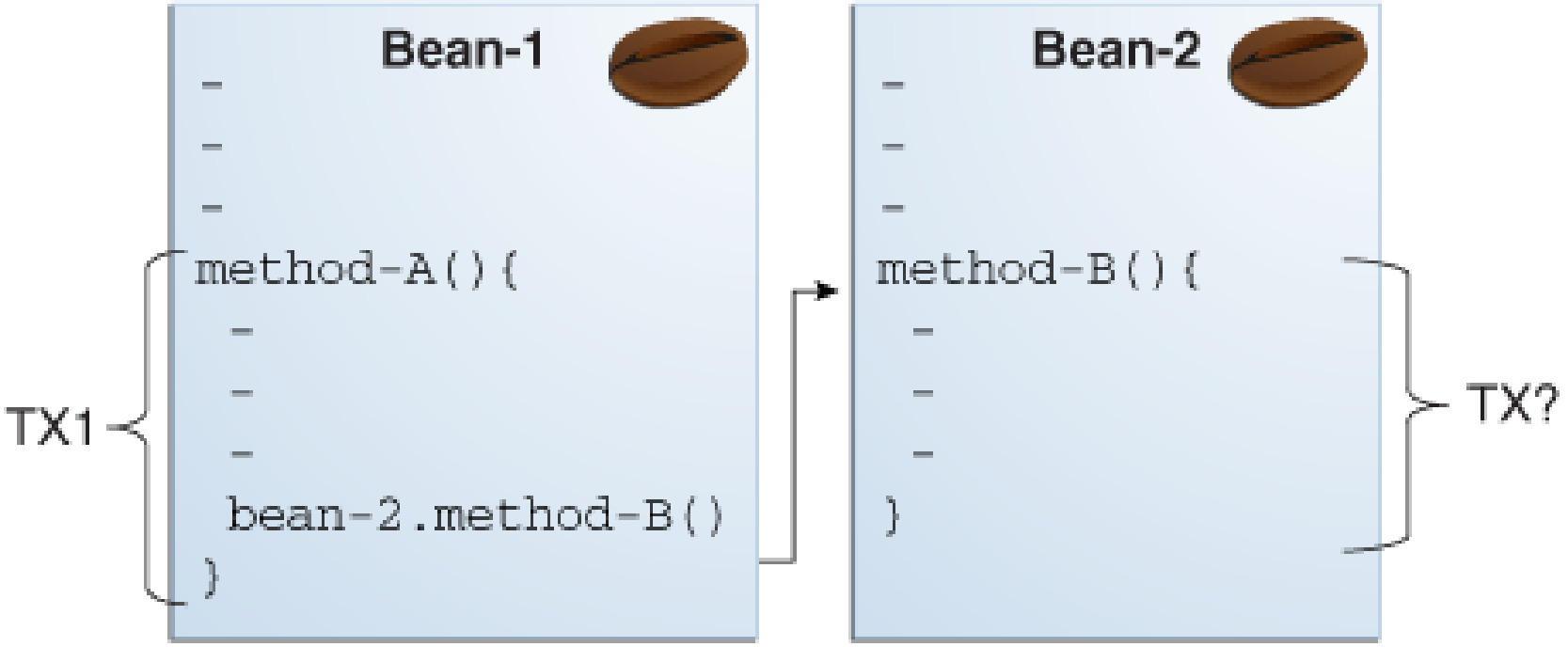
Escopo de Transação

- Participam do escopo de transação todos os EJBs que participam da unidade de trabalho
- Rastreamento do escopo:
  - A execução de um método inicia uma transação
  - A transação é propagada para um EJB se este tem um dos seus métodos invocados
  - O mesmo acontece aos serviços transacionais, incluindo o contexto de persistência

Atributos de Transação

- Utilizando EJBs a transação não precisa ser controlada explicitamente
- A gerencia de transações acontece de forma implícita com base nos atributos de transação
  - Required, RequiresNew, Mandatory, NotSupported, Supports, e Never
- Um atributo pode ser configurado para um EJB inteiro ou para métodos individuais

Atributos de Transação



Atributos de Transação

- Exemplo:

```
@TransactionAttribute(NOT_SUPPORTED)
@Stateful
public class TransactionBean {
...
    @TransactionAttribute(REQUIRES_NEW)
    public void firstMethod() {...}

    @TransactionAttribute(REQUIRED)
    public void secondMethod() {...}

    public void thirdMethod() {...}

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

Atributos de Transação

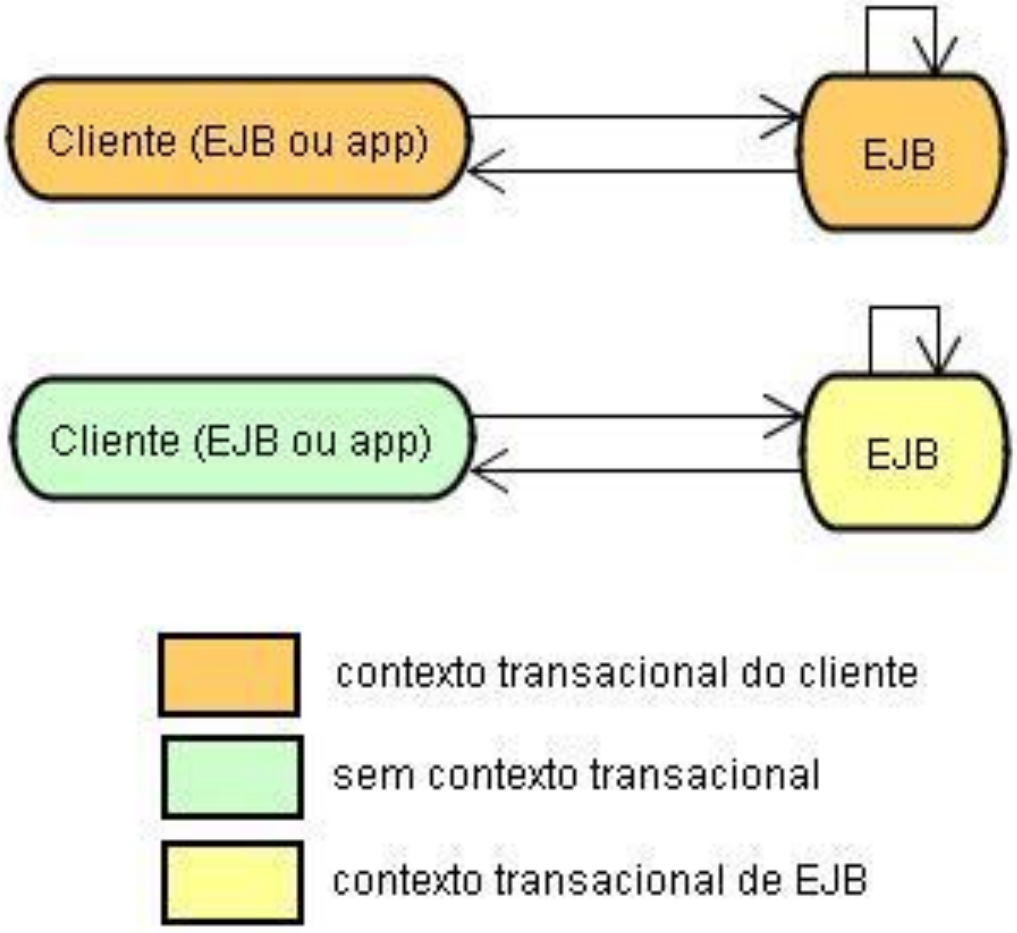
- Nesse EJB exemplo:
  - O atributo de transação padrão será NOT_SUPPORTED
  - O padrão é sobrescrito com a aplicação do atributo REQUIRED e REQUIRES_NEW à métodos específicos
- Caso não seja especificado um atributo de transação e não houver um descritor de implantação o padrão será REQUIRED

Configuração de Transação no Descritor de Implantação

```
<ejb-jar ...>
     <assembly-descriptor>
          <container-transaction>
               <method>
                    <ejb-name>ExemploBean</ejb-name>
                    <method-name>*</method-name>
               </method>
               <trans-attribute>NotSupported</trans-attribute>
          </container-transaction>
          <container-transaction>
               <method>
                    <ejb-name>ExemploBean</ejb-name>
                    <method-name>metodoTransacional</method-
name>
               </method>
               <trans-attribute>Required</trans-attribute>
          </container-transaction>
     </assembly-descriptor>
</ejb-jar>
```

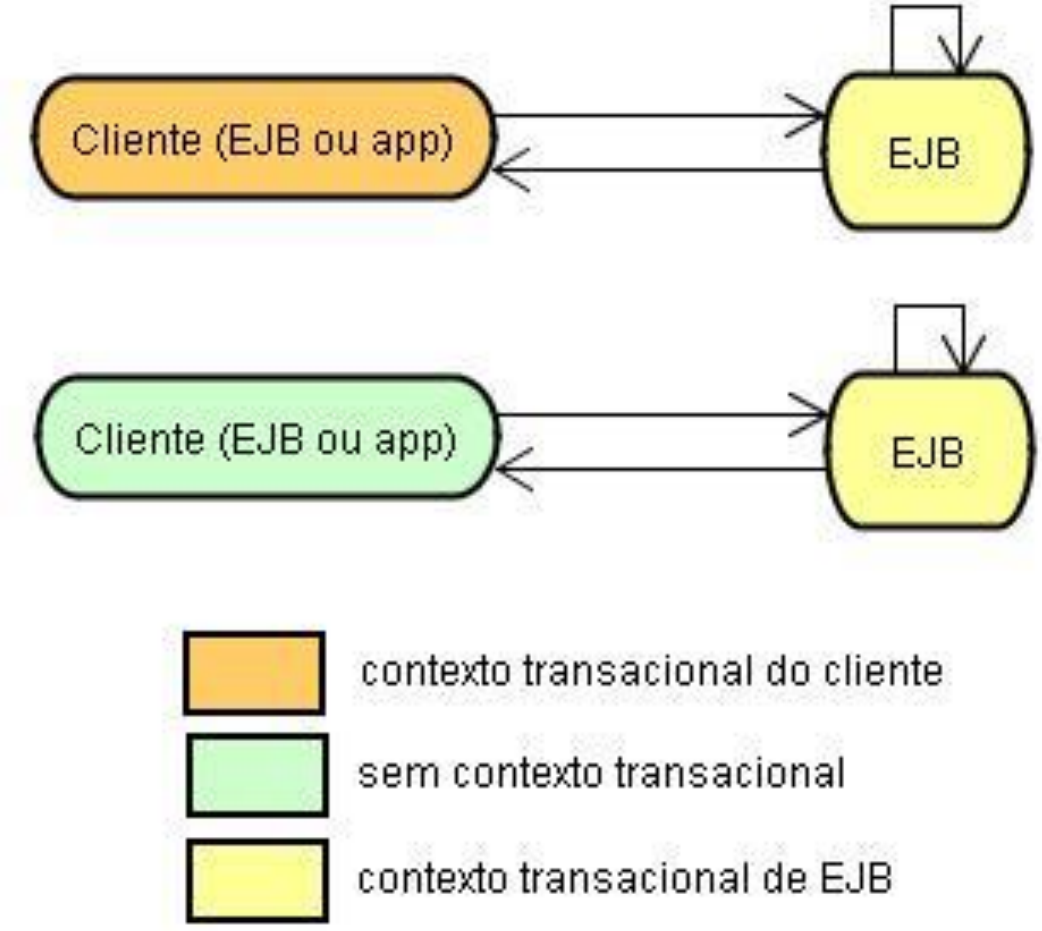
Atributo *Required*

- O método do EJB deve ser invocado de dentro de um contexto de transação
- Se o cliente não estiver em uma transação – é iniciada uma própria



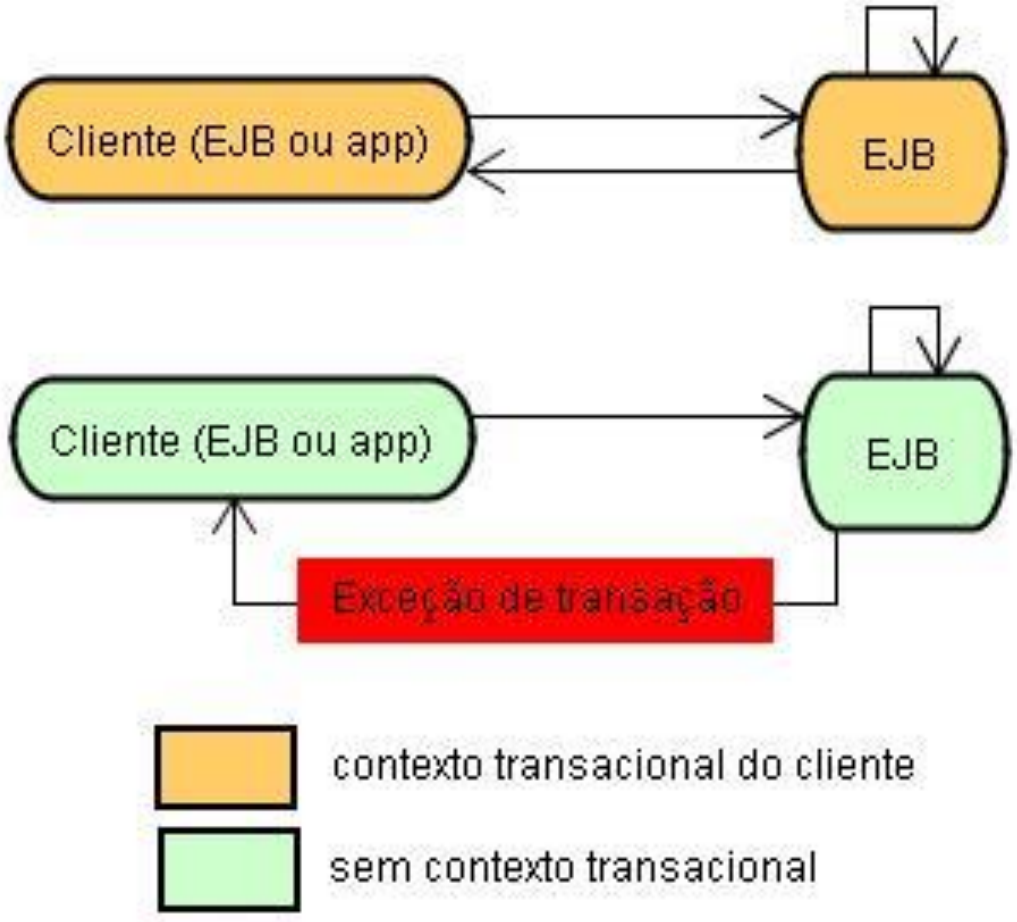
Atributo *RequiresNew*

- Uma nova transação é sempre iniciada quando um método do EJB é invocado
- Se o cliente estiver em uma transação, esta é suspensa até a conclusão do método



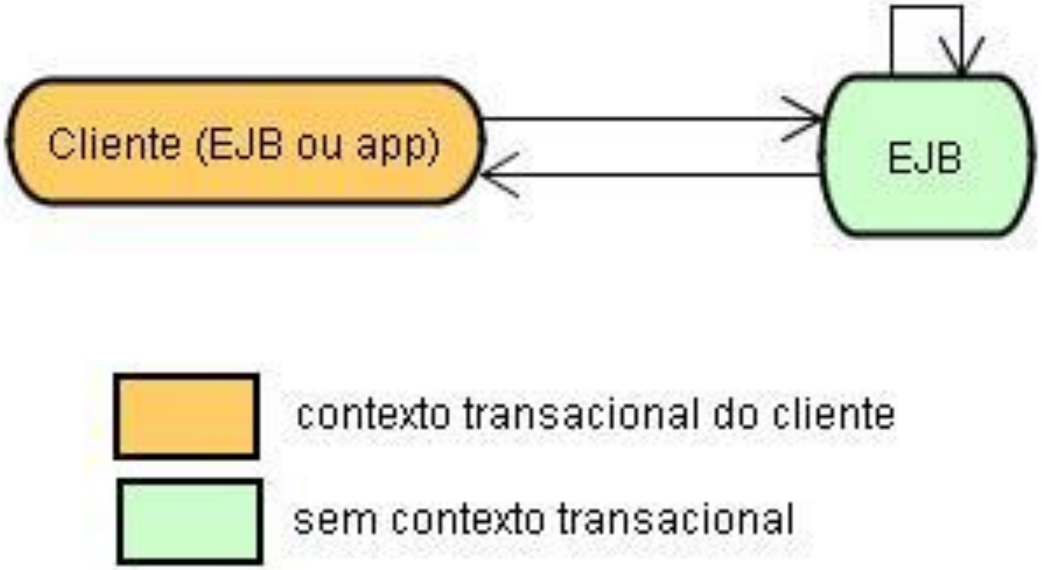
Atributo *Mandatory*

- Indica que o método do EJB sempre deve ser parte do escopo de transação
- Se o cliente não fizer parte de uma transação – lança exceção de transação



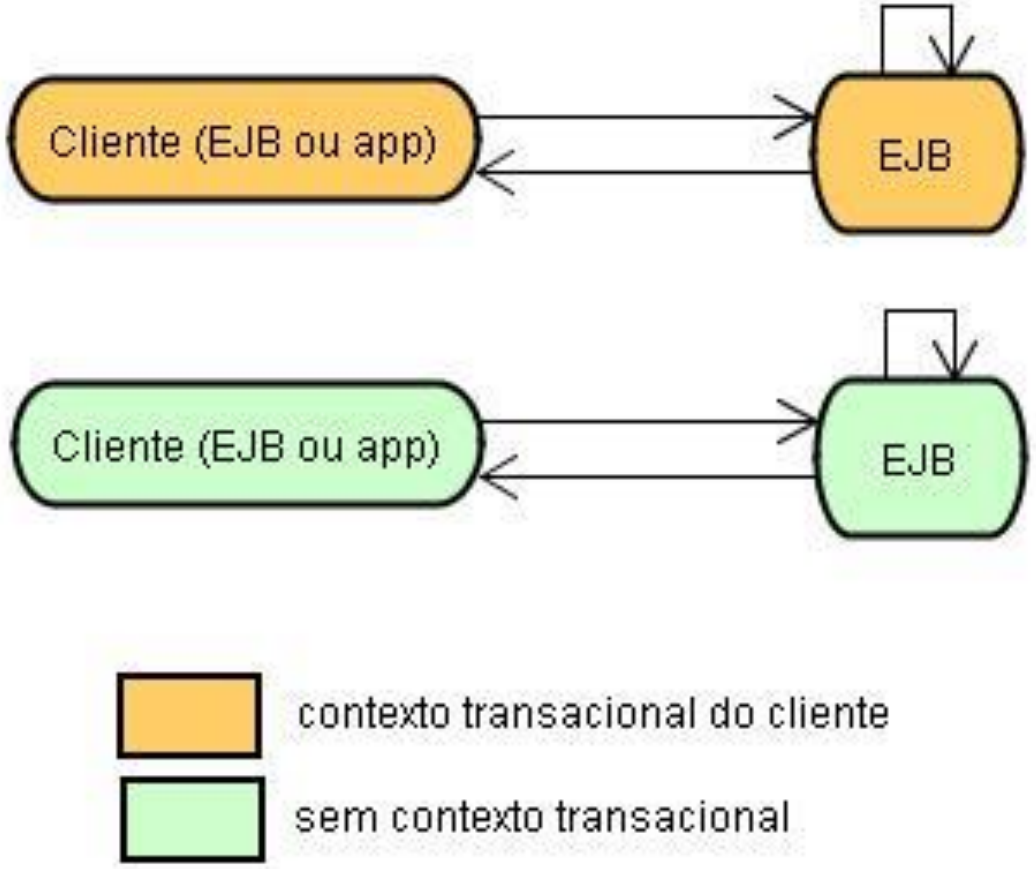
Atributo *NotSupported*

- Ao invocar um método de um EJB com este atributo – a transação é suspensa até o método completar



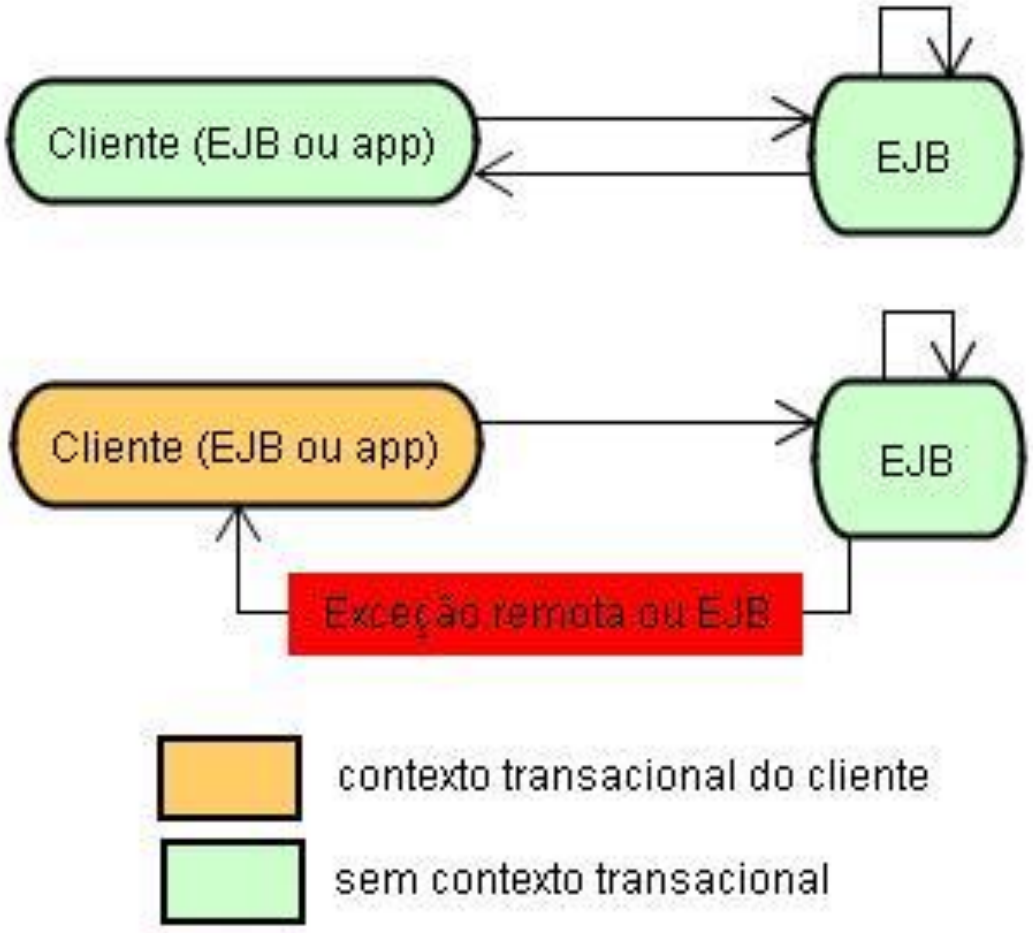
Atributo *Supports*

- Esse atributo significa que o método do EJB será incluído no escopo de transação se for invocado dentro de uma transação
  - Propaga a transação



Atributo *Never*

- Significa que o método do EJB nunca deve ser invocado dentro do escopo de transação
- Se for invocado de dentro de uma transação – lança uma exceção EJB



Atributos de Persistência e de Transação

- A especificação EJB recomenda que o *EntityManager* seja acessado dentro do escopo de uma transação
- Em caso de encapsular o acesso às entidades persistentes com EJB, utilize apenas:
  - Required, RequiresNew e Mandatory

Beans Orientados a Mensagens e Atributos de Transação

- *Beans* orientados à mensagens só podem declarar os atributos *NotSupported* ou *RequiresNew*
- Os demais atributos não fazem sentido para um *bean* orientado à mensagem pois este é desacoplado do cliente

Papel do Servidor EJB

- Como um monitor de transação, um servidor EJB observa cada chamada de método na transação
  - Se uma das atualizações falhar, todas as atualizações para todos EJBs e entidades são revertidas (***rollback***)
  - Sem falhas, as atualizações são confirmadas – atualização permanente (***commit***)

Roll Back em uma Transação Gerenciada pelo

- Duas formas de ser realizado
  - Se uma exceção for lançada
  - Invocando o método *setRollbackOnly* da interface *EJBContext*

Sincronizando as Variáveis de Instância de um *Bean* de Sessão

- A interface SessionSynchronization é opcional
  - Permite a instâncias um *bean* de sessão receber notificações de transações
  - Métodos: *afterBegin*, *beforeCompletion* e *afterCompletion*

Não Permitidos nas Transações Gerenciadas pelo Contêiner

- Métodos proibidos:
  - Os métodos *commit*, *setAutoCommit* e *rollback* de *java.sql.Connection*
  - O método *getUserTransaction* de *javax.ejb.EJBContext*
  - Qualquer método de *javax.transaction.UserTransaction*

Garantias de

ISOLAMENTO

Isolamento e Bloqueio de Banco de Dados

- O “I”solamento de transação é uma das partes fundamentais de um sistema de controle de transações
  - Acesso simultâneo aos mesmos dados
- Condições a prevenir com o isolamento:
  - Leitura suja
  - Leituras repetíveis
  - Leituras fantasmas

Leituras Sujas

- Ocorre quando uma transação lê as alterações não-confirmadas feitas por uma transação anterior
  - Se a primeira transação for revertida, a leitura dos dados pela segunda transação fica inválida – não há como a segunda transação ficar sabendo

Leituras Repetíveis

- Ocorre quando há garantias de que a leitura dos dados seja a mesma se for feita uma nova leitura durante a mesma transação
  - Ou os dados lidos estão bloqueados contra alterações
  - Ou se forem um “instantâneo” que não reflete as alterações

Leituras Fantasmas

- Ocorre quando novos registros adicionados ao banco de dados podem detectados por transações que iniciaram antes da inserção
  - As consultas incluirão registros adicionados por outras transações posteriores que a referida transação iniciou

Bloqueios de Banco de Dados

- Tipos de bloqueio:
  - De leitura
  - De gravação
  - De gravação exclusivos
  - Instantâneos

Bloqueio de Leitura

- Evitam que outras transações de dados os modifiquem durante uma transação até que essa terminar
- Outras transações podem ler os dados, mas não gravá-los
- A transação atual também não pode fazer alterações
- Boqueio de registro ou tabela depende do BD

Bloqueio de Gravação

- São utilizados para atualizações
- Impede que outras transações alterem os dados até a transação atual estar completa
- Permite leituras sujas

Bloqueio de Gravação Exclusivo

- Também são utilizados para atualizações
- Impede que outras transações leiam ou alterem os dados até que a transação atual estar completa
- Evita leituras sujas
- Alguns bancos não permitem que transações leiam os seus próprios dados enquanto estão bloqueados exclusivamente

Bloqueio Instantâneo

- Gera uma visualização congelada dos dados que é tirada quando uma transação é iniciada
- Alguns bancos evitam o bloqueio fornecendo a cada transação seu próprio instatâneo
- O problema é que os dados não são em tempo real, são antigos

Níveis de Isolamento de Transação

- Níveis de isolamento são utilizados para descrever como o bloqueio é aplicado aos dados dentro de uma transação
  - Leitura não-confirmada
  - Leitura confirmada
  - Leitura repetível
  - Serializável

Leitura Não-Confirmada

- A transação pode ler dados não-confirmados (dados modificados por uma transação diferente que ainda não acabou)
- Métodos de beans com esse nível de isolamento podem ler dados não-confirmados

Leitura Confirmada

- A transação não pode ler dados não-confirmados
- Não evita leituras não-repetíveis e leituras fantasmas
- Métodos de beans com esse nível de isolamento não podem ler dados não-confirmados

Leitura Repetível

- A transação não pode alterar os dados que estão sendo lidos por uma transação diferente
- Podem ocorrer leituras fantasmas
- Métodos de beans com esse nível de isolamento tem as mesmas restrições daqueles no nível “leitura confirmada”

Serializável

- A transação tem privilégios de leitura exclusiva e de atualização

Controlando os Níveis de Isolamento

- Pode ser definido pelo desenvolvedor e aplicado pelo servidor EJB
- Exemplo:

```
@Resource(lookup="java:comp/env/oBd")
DataSource ds;

Connection conn = ds.getConnection();

conn.setTransactionIsolation(
                Connection.TRANSACTION_SERIALIZABLE);
```

Bloqueio Otimista

- Não bloqueia no sentido transacional
- Assume-se que não vai haver acessos a dados de forma simultânea
- Recurso de versionamento do JPA

```
@Entity
public class Exemplo {
    private long version;
    ...
    @Version
    protected long getVersion() { return version;}
}
```

Bloqueio Otimista

- A alteração dos dados da entidade só poderá acontecer se a versão da entidade que está sendo trabalhada na memória for igual a do armazenamento persistente

Bloqueio Programático

- O gerenciador de entidades – ***EntityManager*** – possui o método ***lock*** para realizar o bloqueio de entidades
  - void lock(Object entity, LockModeType type);
    - LockModeType.READ
    - LockModeType.WRITE

Gerenciamento Explícito de Transação

- O gerenciamento explícito de transações é realizado utilizando o Object Transaction Service (OTS) da OMG
- Ou a sua implementação Java o Java Transaction Service (JTS)
- Os EJBs suportam uma API mais simples, a Java Transaction API (JTA)