

# Vorlesung Internetprogrammierung

## Transaktionen

Peter Thiemann

Universität Freiburg

Vorlesung Internetprogrammierung, SS2006

- 1 Einführung
  - ACID Properties
- 2 Transactions and EJB
  - EJB Transaction Attributes
- 3 Explicit Transaction Control

# What is a transaction?

- (general) Agreement, communication, or movement carried out between separate entities
- (database) Sequence of operations that must either succeed altogether or fail
- Example: transfer 200 EUR between bank accounts A and B
  - 1 withdraw 200 EUR from A
  - 2 deposit 200 EUR in B

Both must succeed for a successful transfer

# Transactions must be ACID

**Atomic** either all operations succeed or the whole transaction fails

**Consistent** the database must be in consistent state before and after the transaction (e.g., all integrity constraints must hold)

**Isolated** The effects of incomplete transactions should be invisible to other transactions as much as possible.

**Durable** Changes are permanent when the transaction completes successfully.

# ACID and the Bank Account

**Atomic** withdrawal and deposit must take place

**Consistent** the amount withdrawn must equal the amount deposited

**Isolated** other transactions must not be able to observe the withdrawal from  $A$  before the deposit in  $B$  has taken place

**Durable** if DB crashes . . .

- before the transaction completes, no effect is visible
- after the transaction completes, the transfer is permanent

# Transaction Control in EJBs

## Explicit Transaction Control

- explicit demarcation of transactional events via JTS
- SQL: START TRANSACTION, END TRANSACTION, ROLLBACK, COMMIT
- SAVEPOINT *name*, ROLLBACK TO SAVEPOINT *name*, RELEASE SAVEPOINT *name*
- error prone, mixup with business logic, inflexible

## Implicit Transaction Control

- via EJB deployment descriptor
- separate from business logic, more flexible

# Transactions and EJB

- Transaction
  - begins with client invoking a bean (business) method
  - ends (commits) successful with normal exit of this invocation
  - may fail (rollback) if there is an exception during the invocation
- Operations = bean methods invoked during a transaction
- Scope of Transaction = all EJBs involved in performing the task of the invoked business method
- Transaction may be passed on to subsidiary bean method invocations (depending on *transaction attributes*)

# Setting Transaction Attributes

In the Deployment Descriptor

```
<ejb-jar ...> ...
  <assembly-descriptor> ...
    <container-transaction>
      <method>
        <ejb-name>TravelAgentEJB</ejb-name>
        <method-name>*</method-name>
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
    <container-transaction>
      <method>
        <ejb-name>TravelAgentEJB</ejb-name>
        <method-name>listAvailableCabins</method-name>
      </method>
      <trans-attribute>Supports</trans-attribute>
    </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```



# EJB Transaction Attributes

## Attribute Values

- NotSupported
  - Supports
  - Required
  - RequiresNew
  - Mandatory
  - Never
- 
- Not all of them must be implemented by EJB container
  - Not all make sense for every kind of EJB

# Semantics of Transaction Attributes/1

## NotSupported

- Method runs outside the scope of a transaction
- Invoking the method suspends current transaction (if any)
- Exiting the method resumes the suspended transaction

## Supports

- Method may run outside of transaction
- If there exists a current transaction, then this method becomes part of it.

# Semantics of Transaction Attributes/2

## Required

- Method must run inside a transaction
- If no transaction is current at invocation, then create a new transaction.

## RequiresNew

- Method must run inside a new transaction created afresh for the each invocation (and its descendants)

# Semantics of Transaction Attributes/3

## Mandatory

- Method must run inside a preexisting transaction
- Invocation will **not** create a new transaction
- It is an error to call method outside a transaction

## Never

- Method must not run inside a transaction
- It is an error to call method inside a transaction

# Bean Types and Transaction Attributes

## CMP Entity Beans

- Required, RequiresNew, Mandatory are always supported and should be used
- vendor support for Never, Supports, and NotSupported is optional (but not recommended)

## Message-driven Beans

- NotSupported or Required
- Other attributes are relative to transaction context
- Such context does not exist by invocation through a message queue

## Endpoints (Webservices)

- all except Mandatory

# Guarantees Through Transactions

- Transaction levels configurable on the application server
- Transaction isolation levels as with JDBC
  - Read Uncommitted
  - Read Committed (No dirty reads)
  - Repeatable Read (+ no nonrepeatable read)
  - Serializable (+ no phantom read)
- The higher the transaction level
  - the more guarantees provided
  - the slower
- Different isolation levels for different methods possible

# Explicit Transaction Control

- Don't use unless forced to!
- In deployment descriptor:

```
<session> ...  
    <transaction-type>Bean</transaction-type> ...  
</session>
```

- Obtain JTS UserTransaction object

```
Context jndiCtx = new InitialContext();  
UserTransaction ut = (UserTransaction)  
    jndiCtx.lookup ("java:comp/UserTransaction");  
ut.begin();  
// transactional stuff  
ut.commit();
```

- Or through EJB context

```
ejbContext.getUserTransaction()
```

# The UserTransaction Interface

```
public interface UserTransaction {
    void begin() throws
        NotSupportedException, SystemException;
    void commit() throws
        RollbackException, HeuristicMixedException,
        HeuristicRollbackException, SecurityException,
        IllegalStateException, SystemException;
    void rollback() throws IllegalStateException,
        SecurityException, SystemException;
    void setRollbackOnly() throws
        IllegalStateException, SystemException;
    int getStatus() throws
        SystemException;
    void setTransactionTimeout(int seconds) throws
        SystemException;
}
```