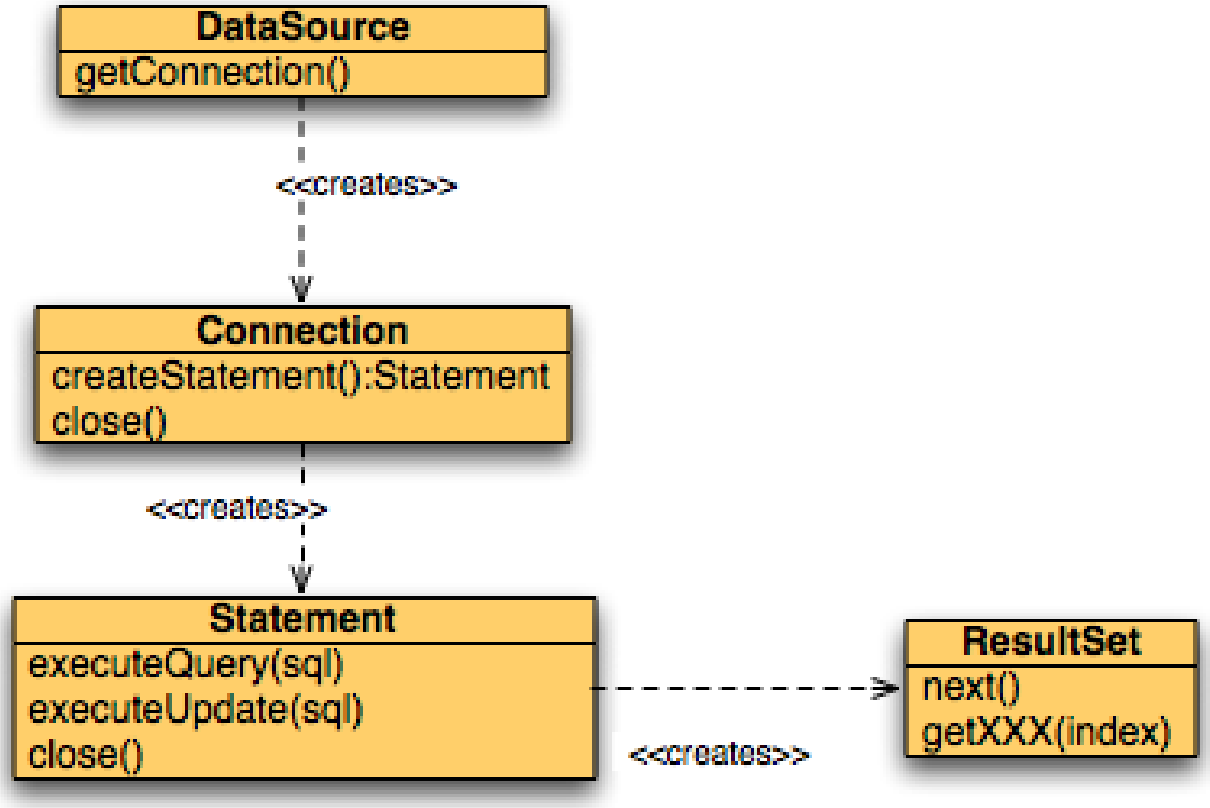
Atomikos JDBC

JDBC API Review



ATOMIKOS

JDBC with JTA/XA

- Statements contain the real SQL action
  - SQL updates are subject to commit or rollback
- With JTA/XA, this coincides with the JTA transaction outcome
  - Commit is subject to 2PC
- However, this requires JTA/XA aware JDBC drivers
  - Such as in ExtremeTransactions

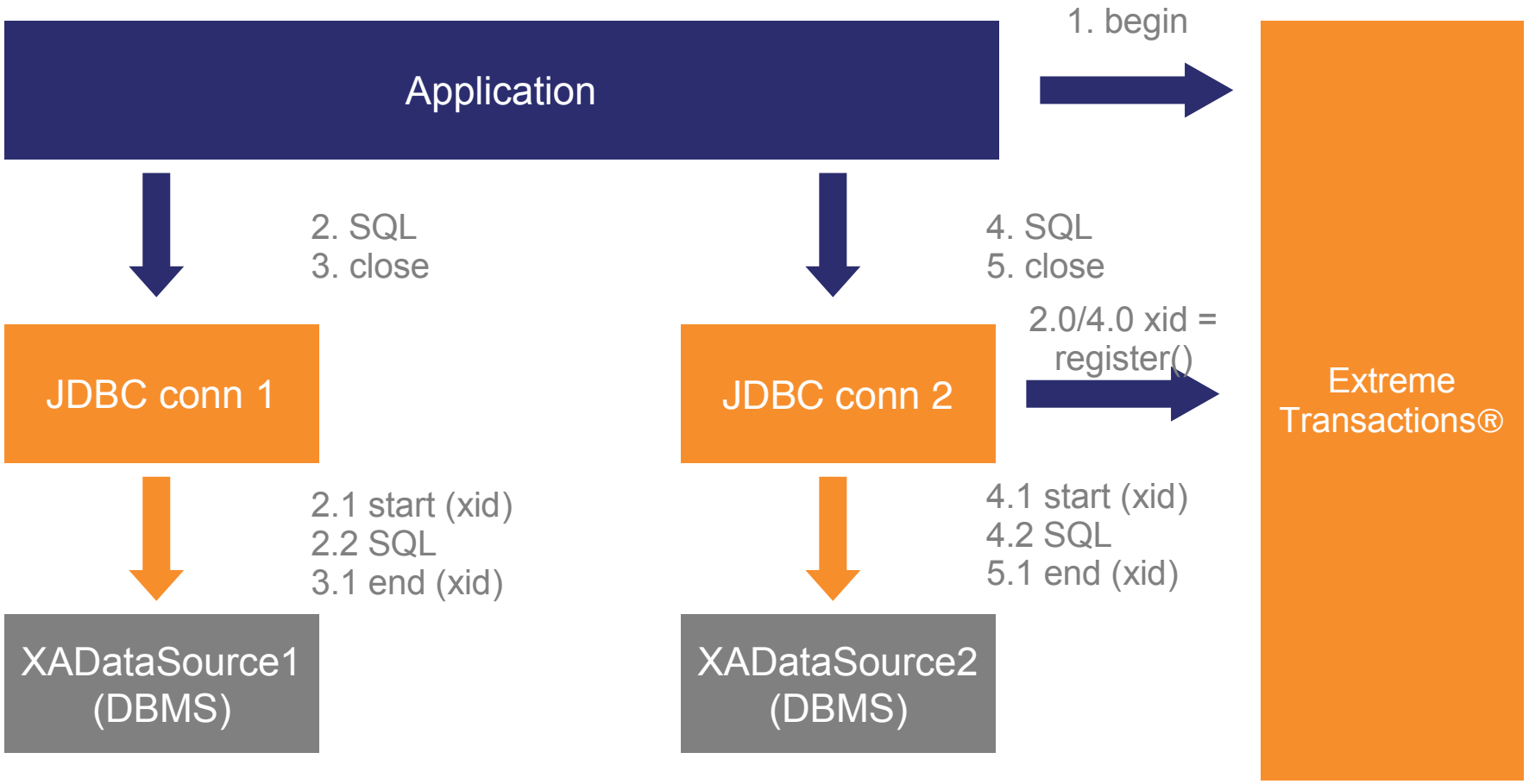
ATOMIKOS

Example Scenario

- A customer is stored in 2 DBMS
- We want to update his address
- This has to be done in both databases to ensure consistency

ATOMIKOS

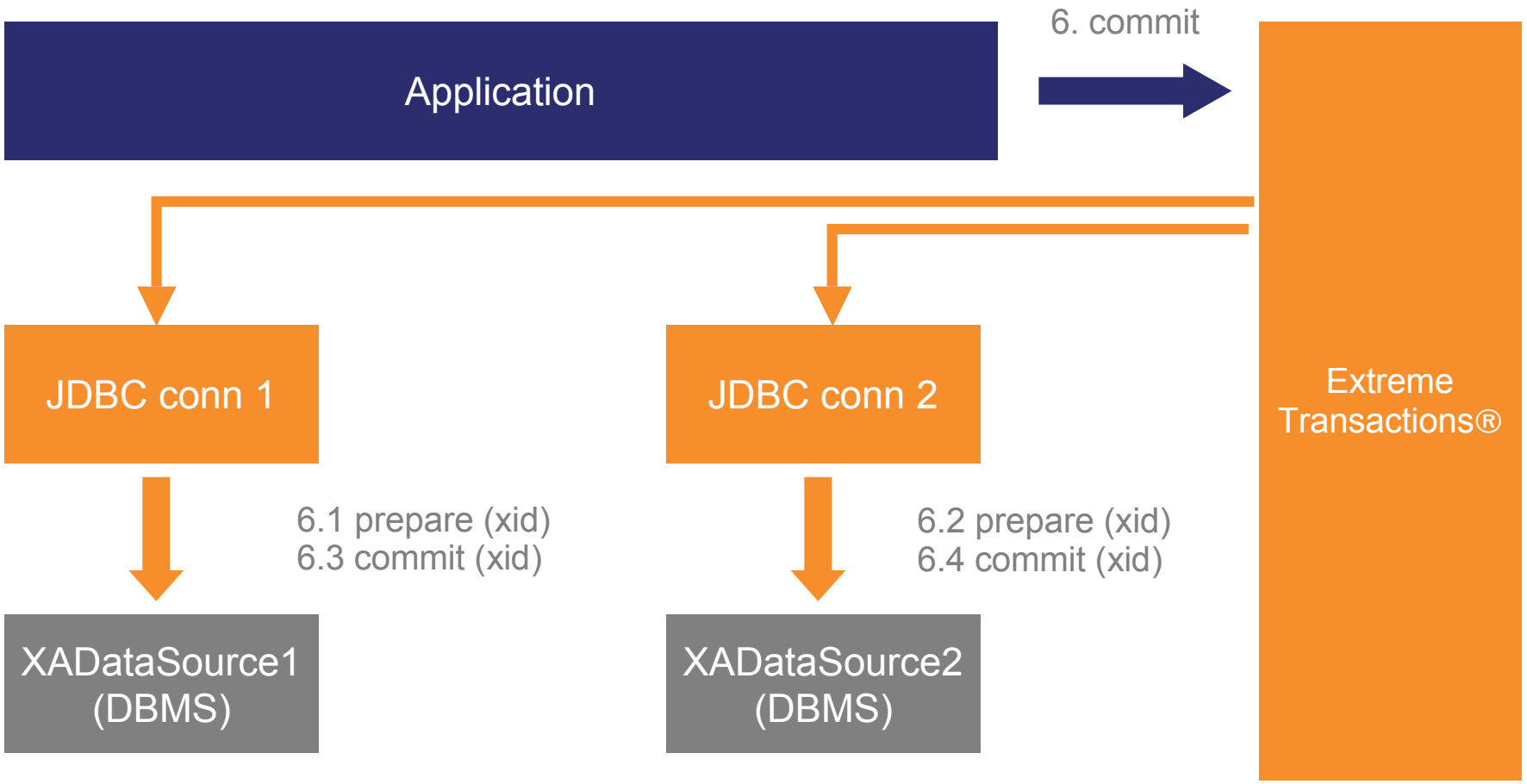
XA Enlist in JDBC



Note: Atomikos drivers delegate to underlying vendor XA drivers!

ATOMIKOS

2PC in JDBC



ATOMIKOS

Observations

- Atomikos data source needed
  - To obtain the Atomikos connection
  - Uses vendor drivers underneath
- Connection.close is not really destroy
  - Connection has to stay open for 2PC!
  - Connection is pooled again afterwards
- No connection-level commits with JTA!

ATOMIKOS

Q&A

- What are the 2 main purposes of the Atomikos data source classes?
- How does this increase performance?
- What is connection failover? Is this supported?

ATOMIKOS

Atomikos JDBC data source adapters

- AtomikosDataSourceBean
  - For ‘real’ XA two-phase commit
  - Requires XA drivers from JDBC vendor
- AtomikosNonXADataSourceBean
  - Workaround for when XA not available
  - Uses ‘regular’ drivers from JDBC vendor
  - Not really safe for updates
  - OK for read-only use
- Shipped as transactions-jdbc.jar

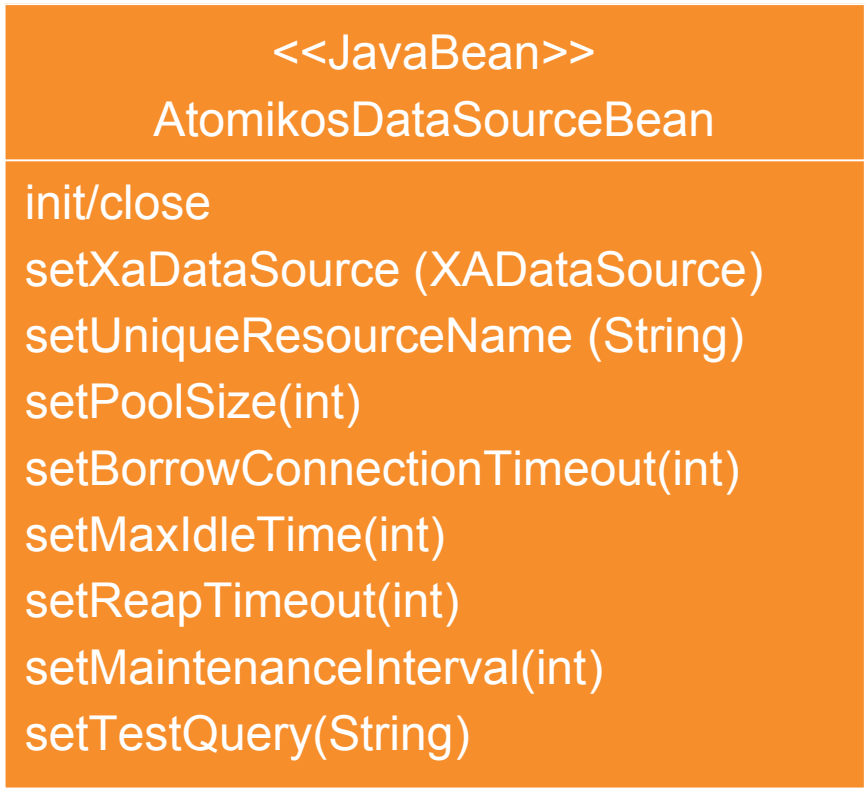
ATOMIKOS

Older Adapters

- SimpleDataSourceBean
- NonXADataSourceBean
- Shipped in transactions-jdbc-deprecated.jar

ATOMIKOS

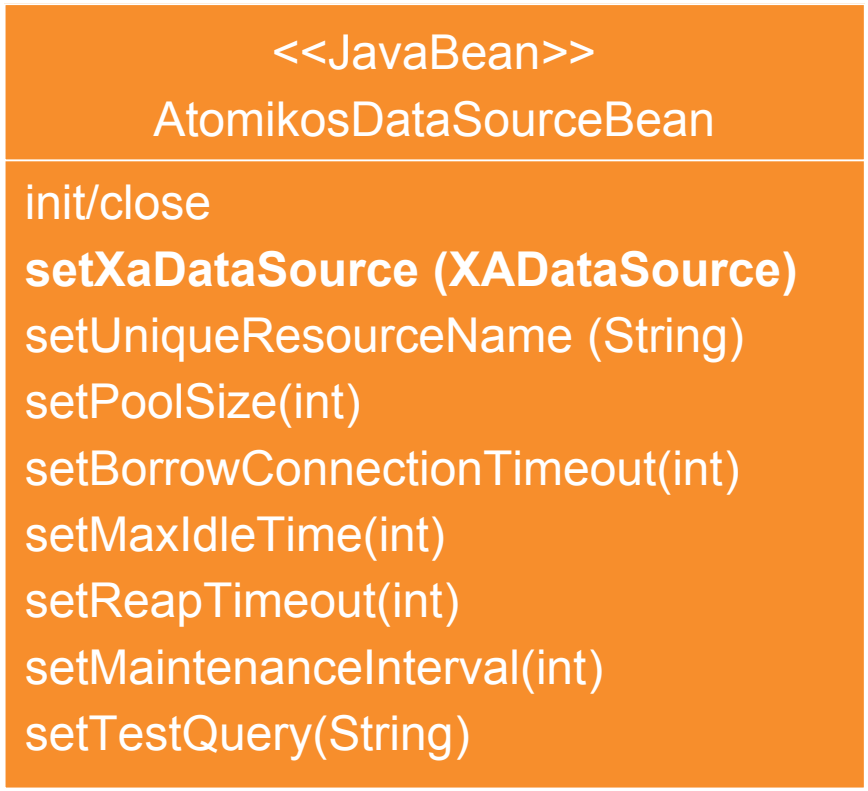
AtomikosDataSourceBean



- Configurable data source
- Wraps XADataSource
  - presents javax.sql.DataSource to a programmer
- Takes care of
  - connection pooling
  - resource enlistment and delistment
  - transparent recovery
- Supports both JTA and non-JTA use
  - (if underlying XADataSource supports non-JTA)

ATOMIKOS

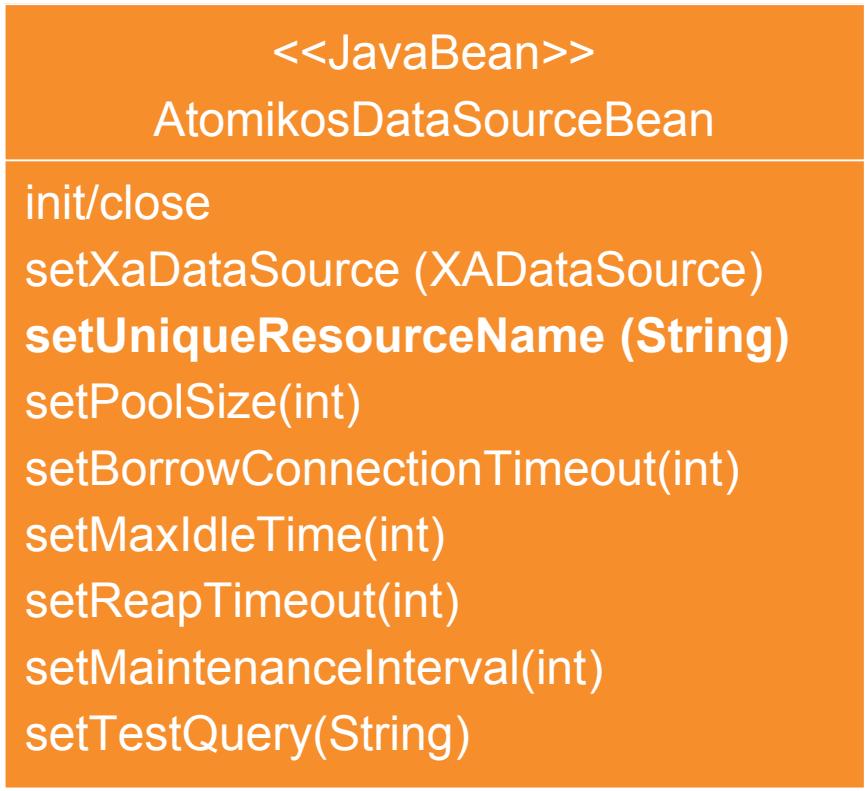
setXaDataSource



- Sets vendor-specific XA JDBC driver
- Needed to connect to the back-end
- Required

ATOMIKOS

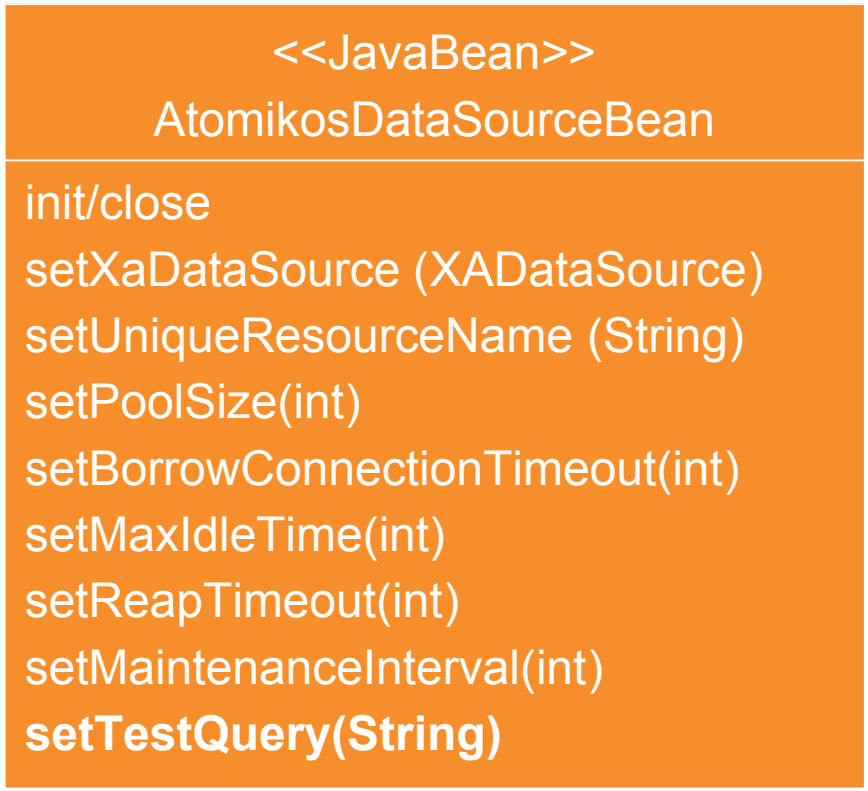
setUniqueResourceName



- Sets the unique resource name for the resource
- Needed for recovery and JNDI binding (internally)
- Required

ATOMIKOS

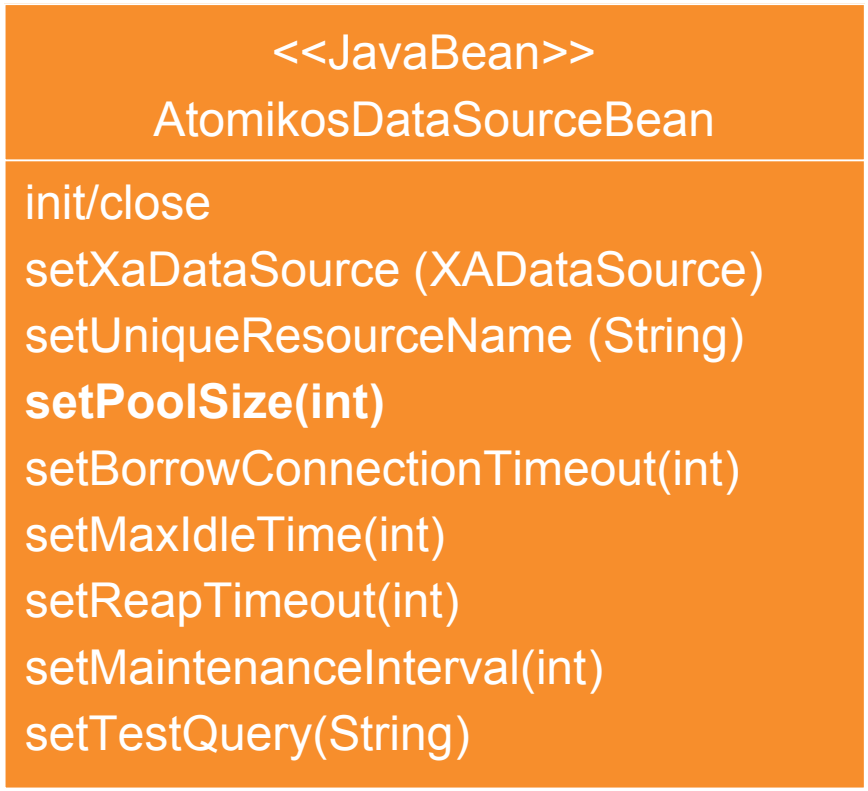
setTestQuery



- Sets a test query to validate connections before being used
- Without it: no automatic connection failover!!!
- Optional
  - default is null (disabled)

ATOMIKOS

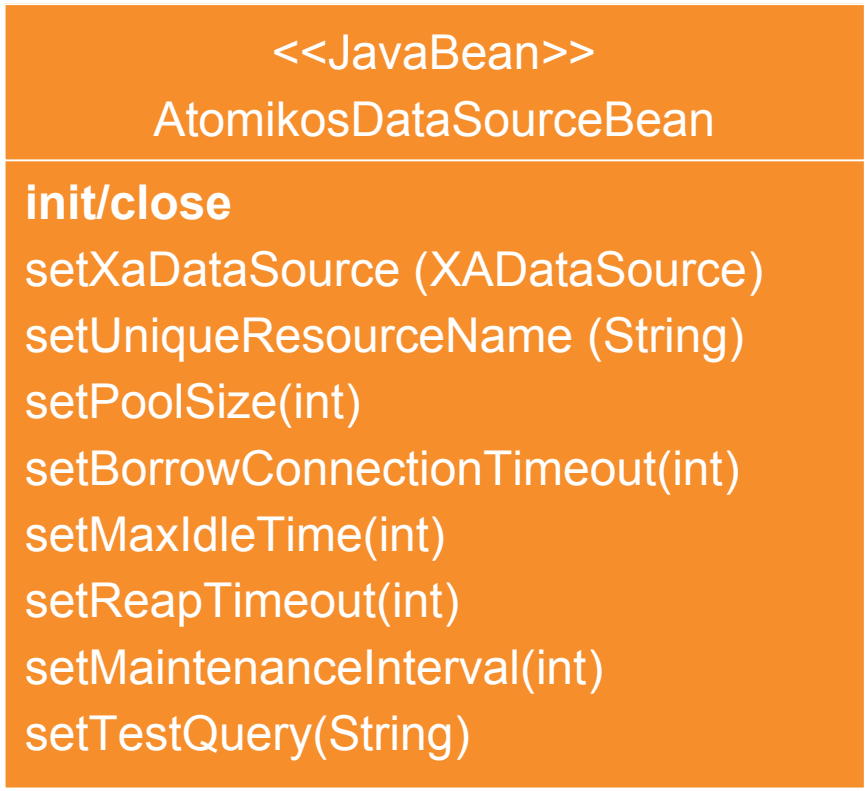
setPoolSize



- Configures the size of the pool
  - no of pre-created connections
  - important for performance
- Optional
  - default is 1

ATOMIKOS

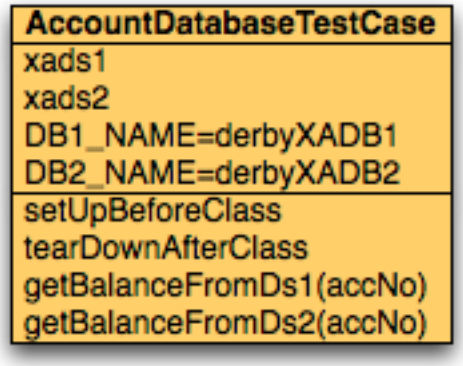
init/close



- Initalization
  - registers database for recovery
  - intializes the connection pool
  - call this before TM starts
- Closing
  - cleans up the connection pool
  - makes database unavailable for two-phase commit or recovery
  - call this after TM shuts down

ATOMIKOS

Database(s) for Exercises



- Extend built-in test class to auto-create test DB
- 2 embedded DBs are created
- Schema:
  - `Accounts (account INTEGER, owner VARCHAR(300), balance BIGINT)`
  - `Java types:`
    - `INTEGER -> int`
    - `VARCHAR -> String`
    - `BIGINT -> long`

ATOMIKOS

Exercise 5.1

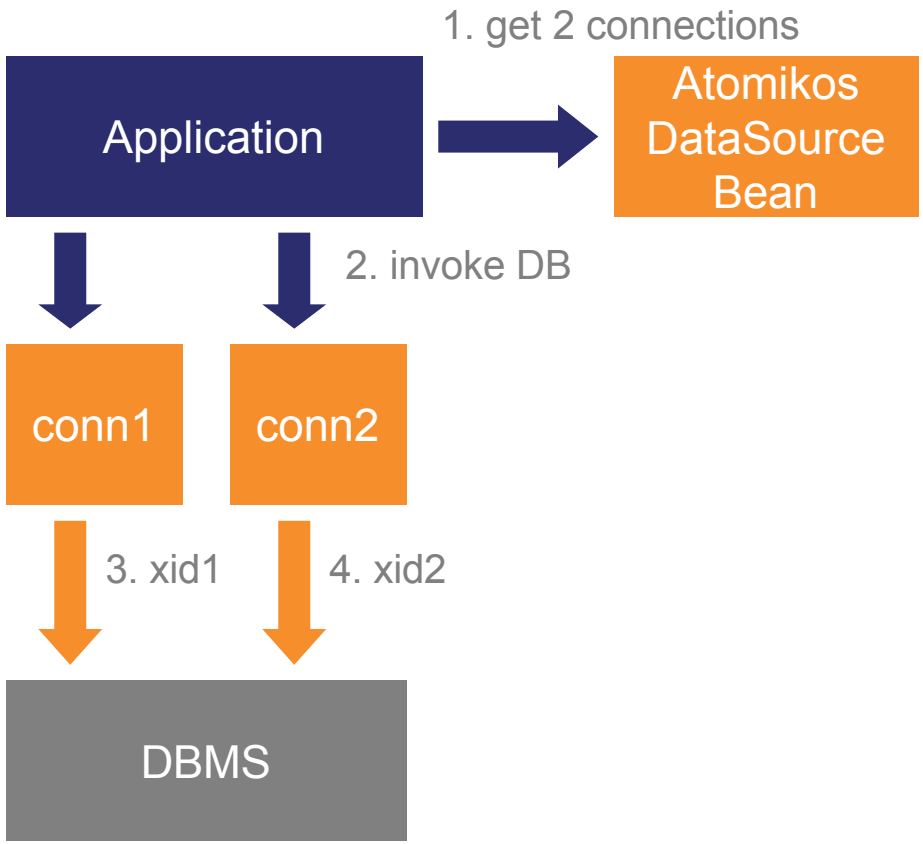
- Extend the utility class "`AccountDatabaseTestCase`"
- Setup an AtomikosDataSourceBean with the predefined XADataSource (Derby) as the vendor-specific driver
- Perform a simple query (OUTSIDE a JTA transaction)
- Make sure to respect the proper shutdown sequence!

ATOMIKOS

Exercise 5.2

- Perform a simple update with LOCAL (JDBC connection-level) transaction management
- Don’t forget to disable autoCommit!
- Update the balance of account 10
- Make sure to test the result
- Suggested approach:
  - get original balance from account 10
  - get connection from datasource
  - set autocommit to false
  - do an SQL update of the balance of account 10
  - commit on the connection
  - get the new balance from account 10 and assert it was changed accordingly (use getBalanceFromDsX in superclass!)

JTA/XA Pitfall



- If 2 connections are open concurrently: different xid will be used
- Even in same transaction!

ATOMIKOS

Exercise 5.3

- Perform the same update within a JTA transaction
- Make sure to test the result:
  - with commit
  - with rollback
  - with timeout/rollback

ATOMIKOS

Exercise 5.4

- Perform the same update with XA directly (using the XA interfaces)
- Make sure to test the result

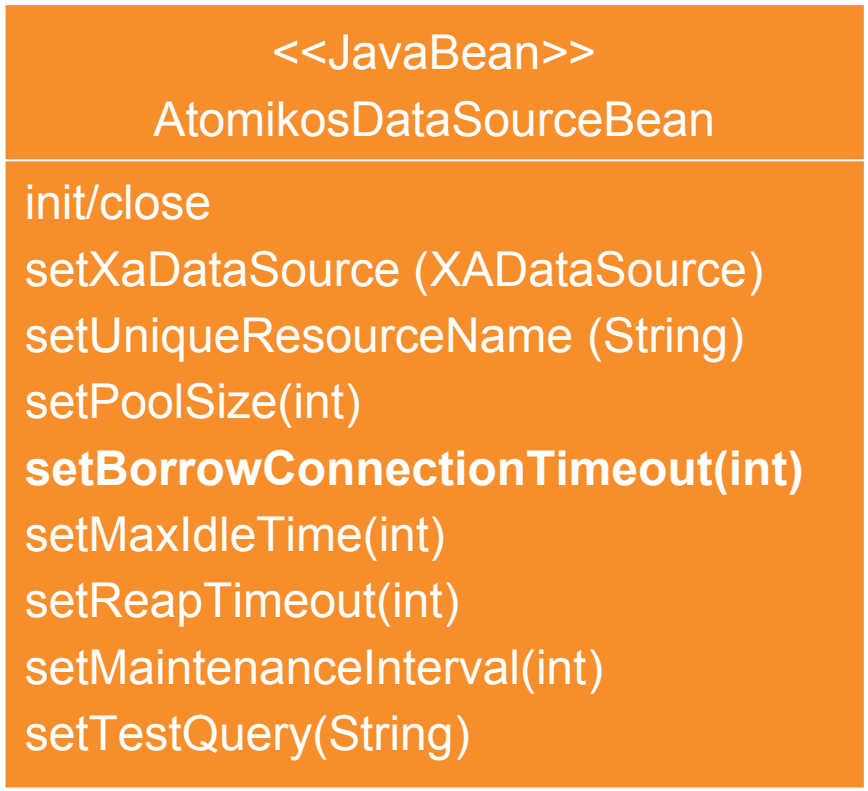
ATOMIKOS

minPoolSize and maxPoolSize

- Actual poolSize varies between:
  - minPoolSize at init time
  - for peak loads: new connections are added when needed, until
  - maxPoolSize is reached
  - once maxPoolSize is reached, new connection requests will **block/wait** until the pool can reuse an exising one
- If you call setPoolSize, both are set to the same value
- Tuning maxPoolSize:
  - Should be more or less the number of concurrent transactions you expect (cf com.atomikos.icatch.max_actives!)
  - But: check your DBA for the upper limit of connections allowed

ATOMIKOS

setBorrowConnectionTimeout



- The number of seconds to wait when pool is exhausted
  - when maxPoolSize is reached
- Optional
  - default is 30 secs

ATOMIKOS

setMaxIdleTime

<<JavaBean>> AtomikosDataSourceBean
init/close
setXaDataSource (XADataSource)
setUniqueResourceName (String)
setPoolSize(int)
setBorrowConnectionTimeout(int)
setMaxIdleTime(int)
setReapTimeout(int)
setMaintenanceInterval(int)
setTestQuery(String)

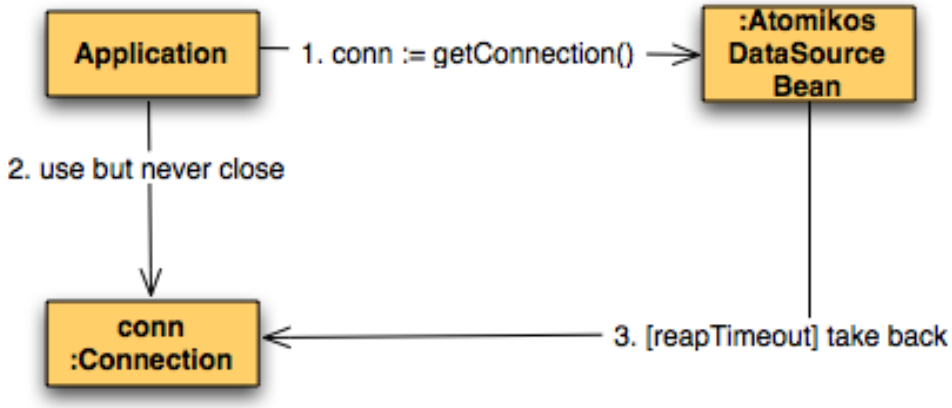
- How long will unneeded connections be kept in the pool?
  - Before the pool shrinks to minPoolSize again
- Optional
  - default is 60 secs

ATOMIKOS

setReapTimeout

<<JavaBean>> AtomikosDataSourceBean
init/close
setXaDataSource (XADataSource)
setUniqueResourceName (String)
setPoolSize(int)
setBorrowConnectionTimeout(int)
setMaxIdleTime(int)
setReapTimeout(int)
setMaintenanceInterval(int)
setTestQuery(String)

- How long before borrowed connections are 'reaped' from the application?
- Optional
  - default is 0 (disabled)



ATOMIKOS

setMaintenanceInterval

<<JavaBean>> AtomikosDataSourceBean
init/close
setXaDataSource (XADataSource)
setUniqueResourceName (String)
setPoolSize(int)
setBorrowConnectionTimeout(int)
setMaxIdleTime(int)
setReapTimeout(int)
setMaintenanceInterval(int)
setTestQuery(String)

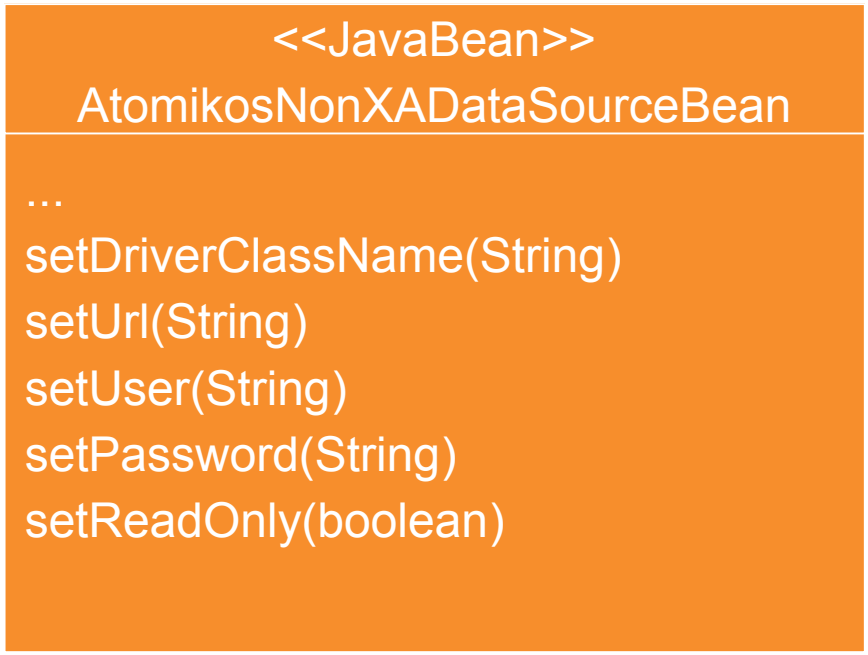
- How long between pool cleanup runs?
- Should be smaller than prior 2 to work fine
- Optional
  - default is 60 secs

ATOMIKOS

Exercise 6.1

- Transfer 100 from Account 10 in DB1 to Account 10 in DB2
- Do this inside a JTA transaction
- Make sure to test the result

AtomikosNonXADataSourceBean



- Use this if a JDBC vendor doesn't support XADataSource
- Can be used if you don't need JTA but rather just JDBC connection pooling
- Safe to use ONLY IF
  - there is no JTA transaction, OR
  - there is no other data source involved in the transaction, OR
  - in READ-ONLY mode

ATOMIKOS

Configuring AtomikosNonXA DataSourceBean

```
import com.atomikos.jdbc.nonxa.AtomikosNonXADataSourceBean;
...
AtomikosNonXADataSourceBean ds  = new AtomikosNonXADataSourceBean();
ds.setUniqueResourceName( "NonXADB" );
ds.setUser ( sa );
ds.setPassword ( "" );
ds.setUrl ( "jdbc:HypersonicSQL:tmp/NonXaAccountDB" );
ds.setDriverClassName ( "org.hsql.jdbcDriver" );
ds.setPoolSize ( 1 );
ds.setBorrowConnectionTimeout ( 60 );
```


ATOMIKOS

Q&A

- Why is the application-level close operation on the JDBC connection required?
- Opening a connection should be done in the *try* part and closing it should be in the finally part. Why?
- If your RDBMS vendor does not support XADataSource, what will you do?
- What is the limitation associated with using Non-XA data sources?
- If you have performance problems: what is the first thing to check?
- Why is there no default testQuery value?

ATOMIKOS