

Transakcje w systemach budowanych w technologii JEE

Jaroslaw.Blad@e-point.pl

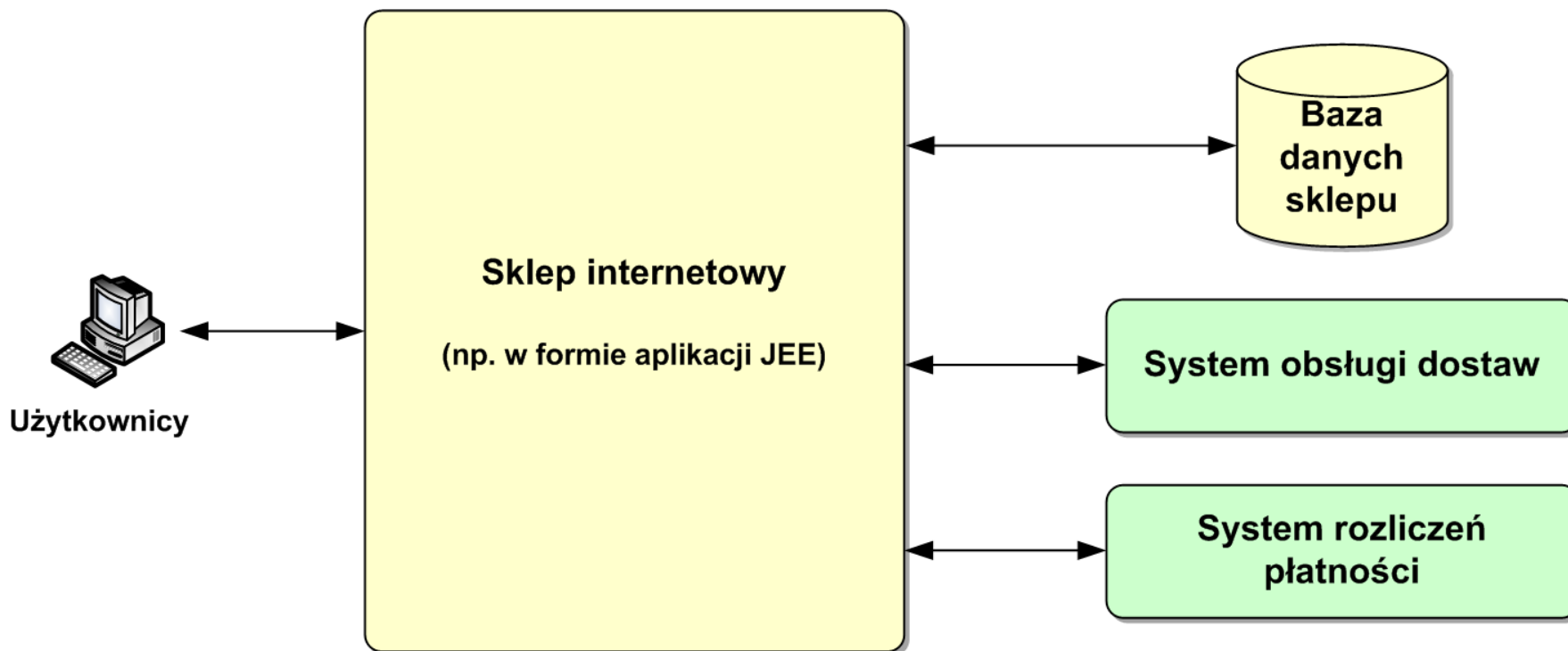


Javarsovia, 4 lipca, 2009

Transakcje i ich właściwości

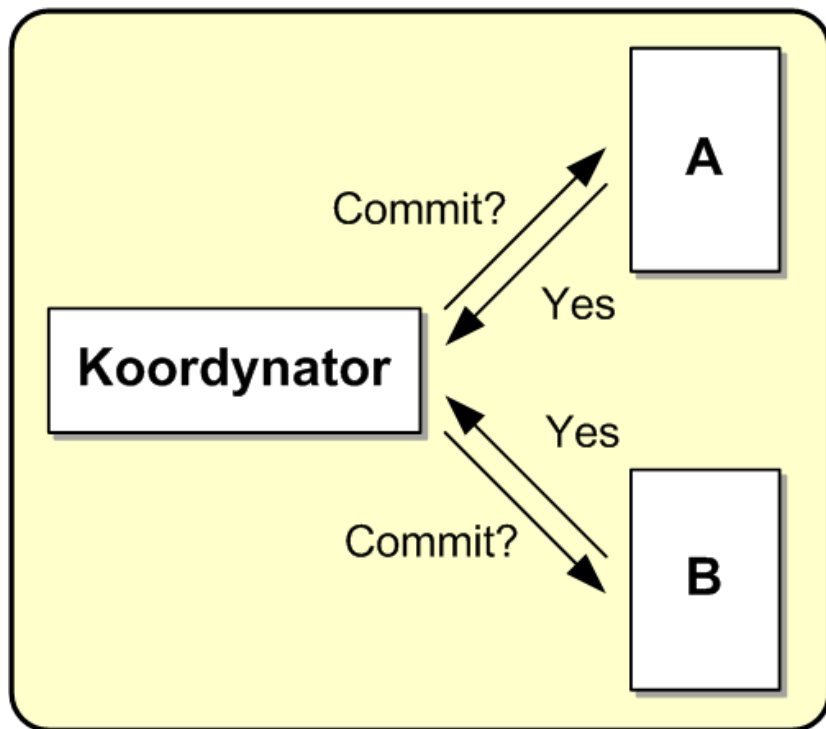
- Definicja transakcji
- Właściwości transakcji (**ACID**)
 - Niepodzielność (Atomicity)
 - Spójność (Consistency)
 - Izolacja (Isolation)
 - Trwałość (Durability)
- Po co budować systemy transakcyjne?

Transakcje rozproszone

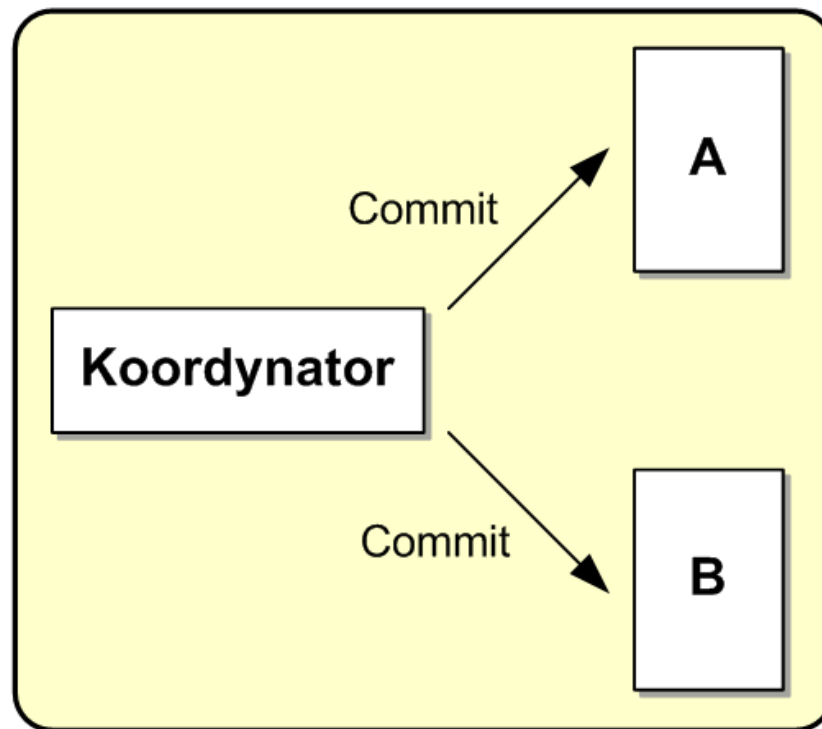


Dwufazowy protokół zatwierdzania transakcji

Faza I

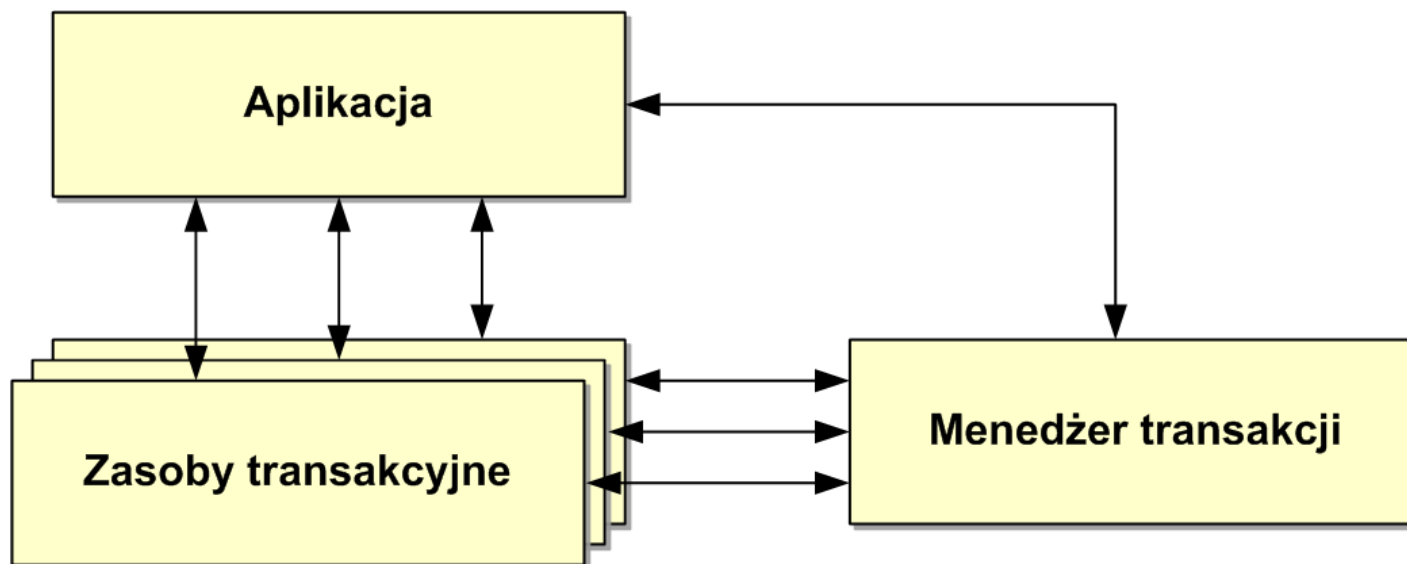


Faza II



Uczestnicy transakcji rozproszonej

- **Aplikacja**
- **Zasoby transakcyjne**
- **Menedżer transakcji**
- **Kontekst transakcji**



Transakcje w środowisku serwera aplikacyjnego JEE

- **Modele transakcji**

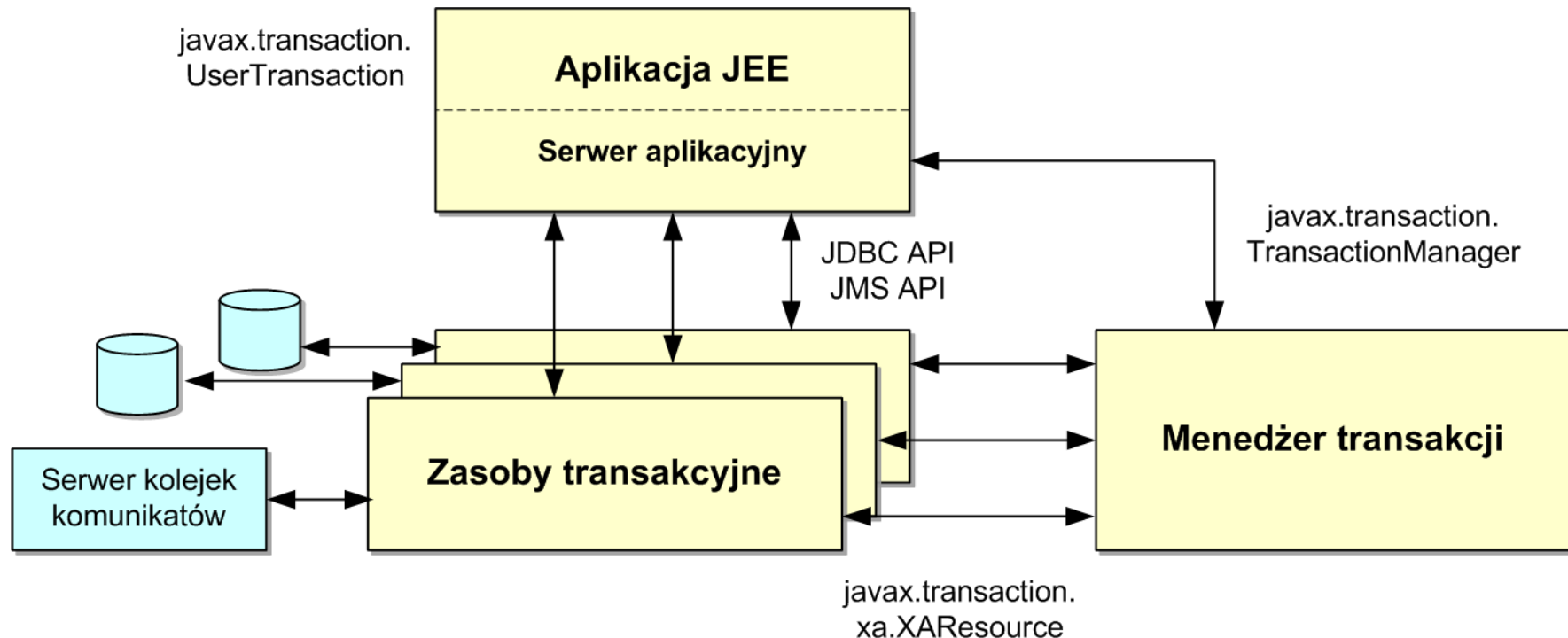
- Lokalne
- Rozproszone - zarządzane przez programistę
- Rozproszone - zarządzane przez kontener

Transakcje w środowisku serwera aplikacyjnego JEE

• Modele transakcji

- Lokalne
- Rozproszone - zarządzane przez programistę
- Rozproszone - zarządzane przez kontener

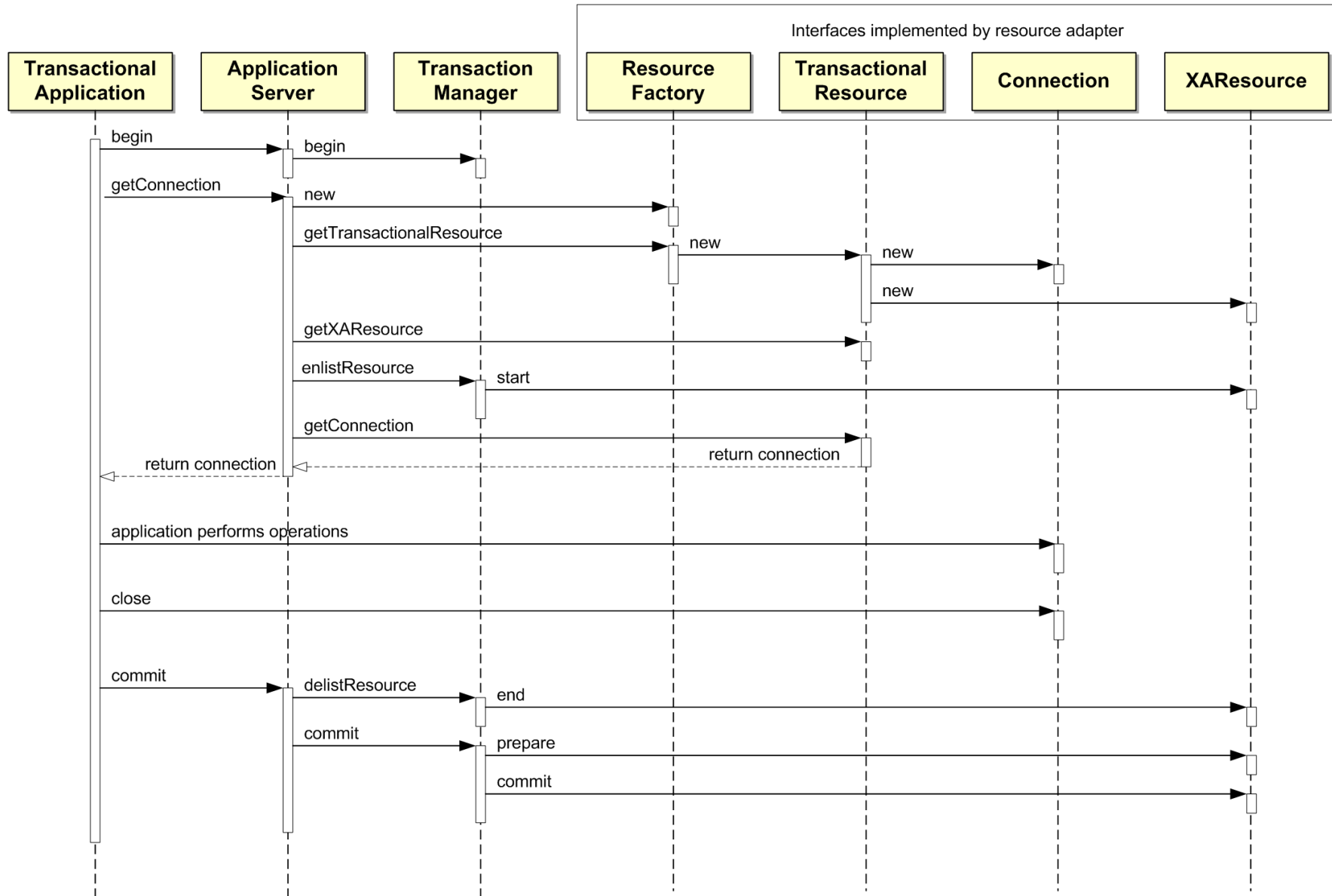
• Interfejsy programowe



Interfejsy programowe - UserTransaction

```
1 void begin();
2
3 void commit();
4
5 void rollback();
6
7 void setRollbackOnly();
8
9 int getStatus();
10
11 void setTransactionTimeout(int seconds);
```

Przebieg transakcji w serwerze aplikacyjnym



Transakcje w serwerze aplikacyjnym JEE a bazy danych

Lokalne transakcje JDBC

```
1  ...
2  Context ctx = new InitialContext();
3  DataSource ds = (DataSource)ctx.lookup("jdbc/ds");
4  Connection con = ds.getConnection();
5  Statement stmt1 = con.createStatement();
6  Statement stmt2 = con.createStatement();
7
8  con.setAutoCommit(false);
9
10 stmt1.executeUpdate(...);
11 stmt2.executeUpdate(...);
12
13 con.commit();    //lub con.rollback(); jeśli się coś nie powiodło
14
15 stmt1.close();
16 stmt2.close();
17 con.close();
18 ...
```

Transakcje w serwerze aplikacyjnym JEE a bazy danych

Uczestnictwo w transakcjach rozproszonych

- **Warunki uczestnictwa w transakcji rozproszonej**
 - Wsparcie bazy danych
 - Driver JDBC XA
 - Konfiguracja serwera aplikacyjnego
- **Sposób użycia**

Transakcje w serwerze aplikacyjnym JEE a bazy danych

Uczestnictwo w transakcjach rozproszonych - przykład

```
1 Context ctx = new InitialContext();
2 UserTransaction ut = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
3
4 ut.begin();
5
6 DataSource ds1 = (DataSource)ctx.lookup("jdbc/ds1");
7 DataSource ds2 = (DataSource)ctx.lookup("jdbc/ds2");
8 Connection con1 = ds1.getConnection();
9 Connection con2 = ds2.getConnection();
10 Statement stmt1 = con1.createStatement();
11 Statement stmt2 = con2.createStatement();
12
13 stmt1.executeUpdate(...);
14 stmt2.executeUpdate(...);
15
16 stmt1.close();
17 stmt2.close();
18 con1.close();
19 con2.close();
20
21 ut.commit(); // lub ut.rollback(); jeśli się coś nie powiodło
```

Transakcje w serwerze aplikacyjnym JEE a bazy danych

Poziomy izolacji

- **READ_UNCOMMITTED**
- **READ_COMMITTED**
- **REPEATABLE_READ**
- **SERIALIZABLE**

Transakcje w środowisku JEE a systemy kolejkowania

Lokalne transakcje JMS

```
1 Context ctx = new InitialContext();
2 QueueConnectionFactory qcf = (QueueConnectionFactory)ctx
3     .lookup("java:comp/env/jms/QCF");
4 Queue queue = (Queue)ctx.lookup("java:comp/env/jms/myQueue");
5
6 conn = qcf.createQueueConnection();
7 session = conn.createQueueSession(true, Session.AUTO_ACKNOWLEDGE);
8 QueueSender sender = session.createSender(queue);
9
10 TextMessage msg1 = session.createTextMessage("message1");
11 TextMessage msg2 = session.createTextMessage("message2");
12
13 sender.send(msg1);
14 sender.send(msg2);
15
16 session.commit(); // lub session.rollback(); jeśli się coś nie powiodło
17
18 session.close();
19 conn.close();
```

Transakcje w środowisku JEE a systemy kolejkowania

Wysyłanie komunikatów JMS a transakcje rozproszone

```
1 Context ctx = new InitialContext();
2 UserTransaction ut = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
3
4 ut.begin();
5
6 QueueConnectionFactory qcf = (QueueConnectionFactory)ctx
7                               .lookup("java:comp/env/jms/XAQCF");
8 Queue queue = (Queue)ctx.lookup("java:comp/env/jms/myQueue");
9
10 conn = qcf.createQueueConnection();
11 session = conn.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
12 QueueSender sender = session.createSender(queue);
13
14 TextMessage msg = session.createTextMessage("message");
15 sender.send(msg);
16
17 session.close();
18 conn.close();
19
20 ut.commit(); // lub ut.rollback(); jeśli się coś nie powiodło
```

Transakcje w środowisku JEE a systemy kolejkowania

Odbieranie komunikatów JMS a transakcje rozproszone

```
1 @TransactionManagement(TransactionManagementType.CONTAINER)
2 public class MessageBean implements MessageListener {
3     public void onMessage(Message msg)
4         // zrób coś z komunikatem
5     }
6 }
```

```
1 @TransactionManagement(TransactionManagementType.BEAN)
2 public class MessageBean implements MessageListener {
3     @Resource
4     UserTransaction tx;
5
6     public void onMessage(Message msg)
7         tx.begin();
8
9         // zrób coś z komunikatem
10
11     tx.commit();
12 }
13 }
```

Transakcje zarządzane przez kontener

- **Modele zarządzania transakcjami**

Transakcje zarządzane przez kontener

- Modele zarządzania transakcjami
- W starym stylu

```
1 ...  
2 <container-transaction>  
3   <method>  
4     <ejb-name>OrderBean</ejb-name>  
5     <method-name>sendOrder</method-name>  
6   </method>  
7   <trans-attribute>RequiresNew</trans-attribute>  
8 </container-transaction>  
9 ...
```

Transakcje zarządzane przez kontener

- Modele zarządzania transakcjami
- W starym stylu

```
1 ...
2 <container-transaction>
3   <method>
4     <ejb-name>OrderBean</ejb-name>
5     <method-name>sendOrder</method-name>
6   </method>
7   <trans-attribute>RequiresNew</trans-attribute>
8 </container-transaction>
9 ...
```

- W nowym stylu (od EJB 3.0)

```
1 ...
2 @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
3 public void sendOrder(...) {
4     ...
5 }
6 ...
```

Transakcje zarządzane samodzielnie przez bean'a

```
1  @Stateless
2  @TransactionManagement(TransactionManagementType.BEAN)
3  public class OrderBean implements ... {
4      @Resource
5      UserTransaction tx;
6
7      public OrderBean() {
8      }
9
10     public void sendOrder(...) {
11         tx.begin();
12
13         // zrób coś w transakcji
14
15         tx.commit();
16     }
17 }
```

Strategie obsługi transakcji w aplikacjach JEE

- **Transakcje lokalne**
- **Transakcje rozproszone realizowane za pomocą komponentów EJB**
 - Zarządzanie transakcjami na poziomie warstwy klienta
 - Zarządzanie transakcjami na poziomie warstwy logiki biznesowej
- **Transakcje rozproszone bez komponentów EJB**

Każde żądanie HTTP objęte transakcją

```
1  public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) {
2      UserTransaction ut = (UserTransaction)ctx.lookup("java:comp/UserTransaction")
3
4      ut.begin();
5      boolean chainOk = false;
6      try {
7          chain.doFilter(req, res);
8          chainOk = true;
9      } finally {
10         if (chainOk) {
11             if (ut.getStatus() == Status.STATUS_ACTIVE) {
12                 ut.commit();
13             } else if (ut.getStatus() == Status.STATUS_MARKED_ROLLBACK) {
14                 ut.rollback();
15             } else { ... }
16         } else {
17             if (ut.getStatus() == Status.STATUS_ACTIVE ||
18                 ut.getStatus() == Status.STATUS_MARKED_ROLLBACK) {
19                 ut.rollback();
20             } else { ... }
21         }
22     }
23 }
```

Transakcje w serwerze aplikacyjnym JEE a usługi sieciowe

- **Wywołania usług sieciowych nie uczestniczą w transakcjach rozproszonych**
- **Jakie to rodzi problemy?**
- **Próby wprowadzenia standardów**
 - OASIS Business Transactions Protocol (Hewlett-Packard, Oracle, BEA)
 - Web Services Coordination oraz Web Services Transactions (IBM, Microsoft, BEA)
 - Web Services Atomic Transaction oraz Web Services Business Activity (IBM, Microsoft, BEA)
 - Web Services Composite Application Framework (IONA, Oracle, Sun)
- **Mechanizmy kompensacji**

Problemy związane z timeout'ami transakcji

- **Działanie użytkownika w systemie nie przynosi efektów**

Problemy związane z timeout'ami transakcji

- Działanie użytkownika w systemie nie przynosi efektów
- Rozwiązania (**wszystkie mają wady**)
 - Zwiększyć timeout transakcji
 - Nie wysyłać odpowiedzi aż do zatwierdzenia transakcji

Problemy związane z timeout'ami transakcji

- Działanie użytkownika w systemie nie przynosi efektów
- Rozwiązania (**wszystkie mają wady**)
 - Zwiększyć timeout transakcji
 - Nie wysyłać odpowiedzi aż do zatwierdzenia transakcji
- Integracja z systemem, który nie wspiera transakcji rozproszonych

```
1 userTransaction.begin();
2
3 // operacje na bazie systemu
4
5 // nietransakcyjne wywołanie obcego systemu
6
7 userTransaction.commit();
```

Inne ciekawe problemy związane z transakcjami

- **Zakleszczenia w transakcjach rozproszonych**
- **Obciążenie serwera aplikacyjnego a problemy ze współbieżnością**
- **Poziomy izolacji**
- **Serwer aplikacyjny nie wstaje po awarii**
- **Transakcje a aplikacje internetowe**

Podsumowanie

- **Systemy transakcyjne - wymaganie współczesnego biznesu**
- **Na szczęście w serwerze JEE nie jest to trudne**
- **Menedżer transakcji - największa wartość dodana serwera aplikacyjnego**
- **Literatura**
 - „Java Transaction Processing design and implementation”
 - „Java Transaction Design Strategies”
 - Specyfikacja JTA oraz API
 - „Transaction Processing: Concepts and Techniques”

Czy są jakieś pytania?

Zapraszam do dyskusji