Serverske veb tehnologije
- JSP veb aplikacija -

Dragan Ivanović

Katedra za informatiku, Fakultet tehničkih nauka, Novi Sad

2022.

Arhitektura Spring JSP veb aplikacije

- Klijent: controller-i + JSP stranice
- Controller-i pristupaju *service bean*-ovima
  - *Service bean*-ovi se injektuje u atribut *controller*-a pomoću anotacije *@Autowired*
- *Service bean*-ovi pristupaju bazi preko *repository bean*-a
  - *Repository bean* se injektuje u atribut *service bean*-a
- *Repository bean* skladišti i učitava JPA entitete

Spring Primer 19

- Entity klase — osa.spring.pr19.entity.*
- Repository klase — osa.spring.pr19.repository.*
- Service klase — osa.spring.pr19.service.*
- Controller-i — osa.spring.pr19.controller.*
- JSP stranice — osa.spring.pr19.jsp.*

Pozivanje session beanova iz servleta

- Servlet može da uradi JNDI lookup i pronađe session bean
- A može da koristi i dependency injection
- Servlet je po prirodi stateless, pa nema smisla injektovati stateful beanove
- Primer 18

Arhitektura web aplikacije koja koristi EJB

- Klijent: servleti+JSP stranice
- Servleti pristupaju session beanovima
  - SLSB se injektuje u atribut servleta
  - SFSB se pronađe preko JNDI i smesti u HttpSession
- Session beanovi pristupaju entitijima preko EntityManagera
  - EntityManager se injektuje u atribut SB-a

Data Access Object (DAO) sloj

- U praksi su za svaki entity potrebne uobičajene CRUD (create, retrieve, update, delete) operacije
- Njih obično implementiraju posebne DAO klase
- Jedan entity – jedan DAO
- Ima dosta „pešačkog“ posla

Generički DAO: implementacija zajedničkih operacija

```
public interface GenericDao<T, ID extends Serializable> {
  public Class<T> getEntityType();
  public T findById(ID id);
  public List<T> findAll();
  public List<T> findBy(String query);
  public T persist(T entity);
  public T merge(T entity);
  public void remove(T entity);
  public void flush();
  public void clear();
}
```

Generički DAO: implementacija zajedničkih operacija

```
public abstract class GenericDaoBean<T, ID extends Serializable>
    implements GenericDao<T, ID> {
  ...

  @PersistenceContext
  protected EntityManager em;
  ...
}
```

Konkretni DAO za entity User

```
public interface UserDao extends GenericDao<User, Integer> {
  public User login(String username, String password);
}


@Stateless
@Local(UserDao.class)
public class UserDaoBean extends GenericDaoBean<User, Integer>
    implements UserDao {

  public User login(String username, String password) { ... }
}
```

Primer 19

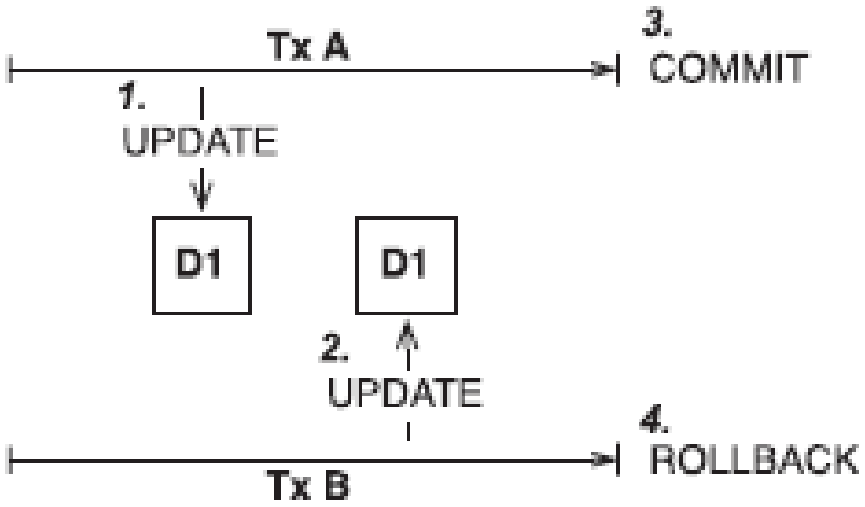
- Entity klase — osa.pr19.entity.*
- DAO klase — osa.pr19.dao.*
- SB klase — osa.pr19.session.*
- servleti — osa.pr19.servlet.*
- JSP stranice — osa.pr19.jsp.*

Transakcije i konkurentni pristup podacima

- Prilikom istovremenog pristupa podacima može da dođe do štetnog preplitanja rada više transakcija
- Tom prilikom može da se javi više problema

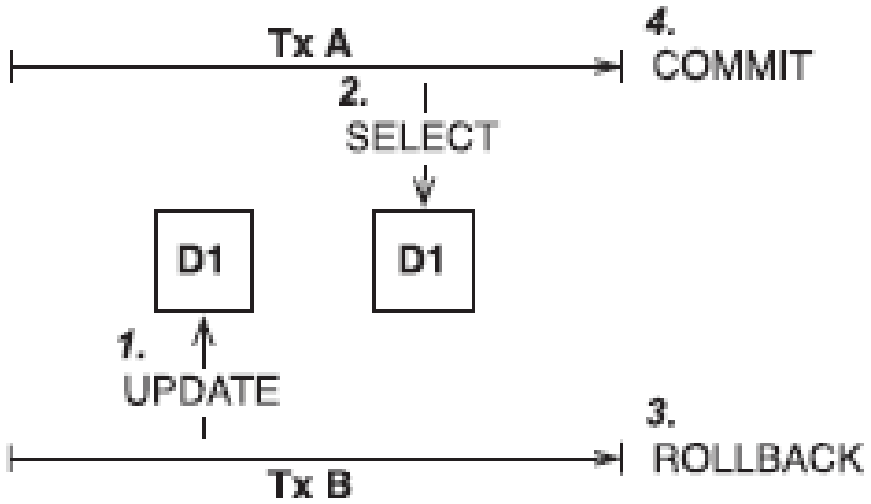
Lost update

- Lost update: dve transakcije menjaju isti podatak bez zaključavanja



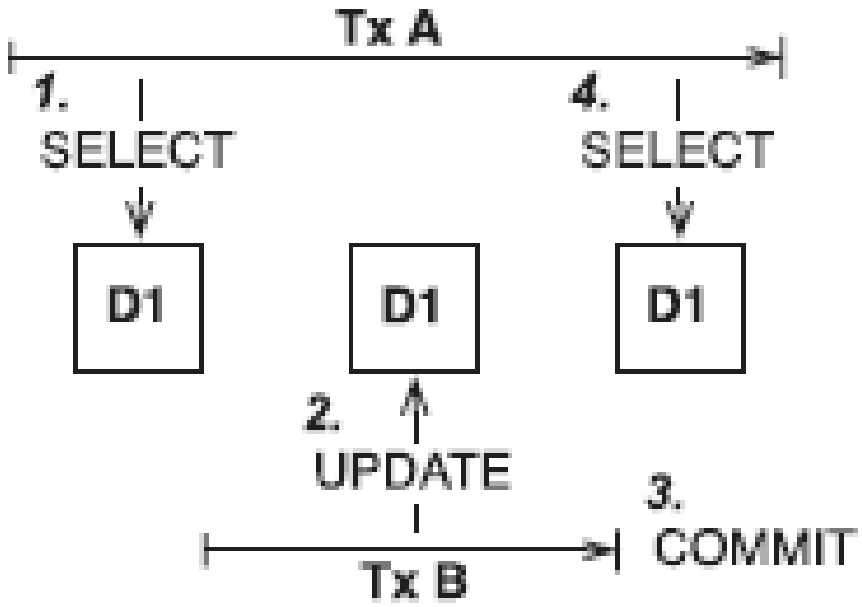
Dirty read

- Dirty read: transakcija A čita podatke pre nego što su commit-ovani



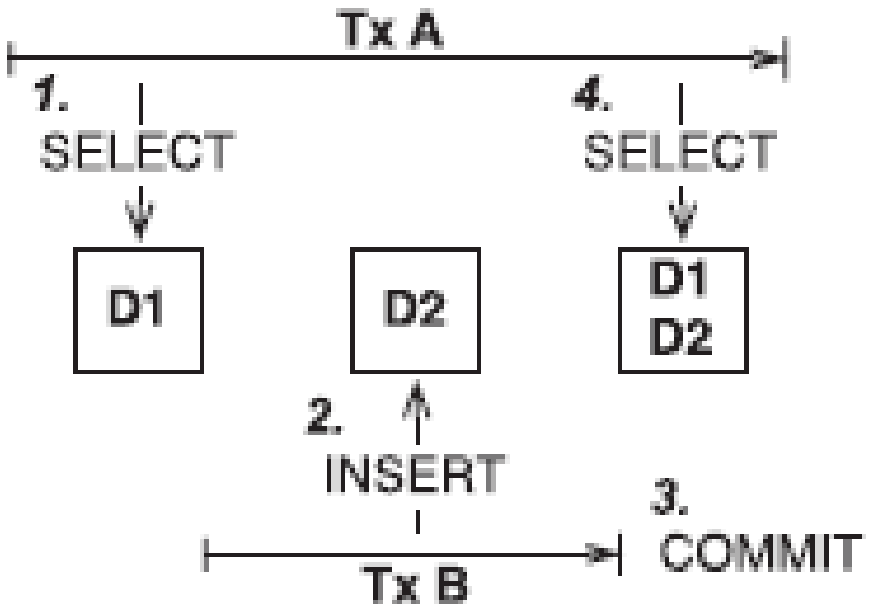
Unrepeatable read

- Unrepeatable read: transakcija A dva puta čita iste podatke i dobija različiti sadržaj



Phantom read

- Phantom read: transakcija A u drugom čitanju dobija i podatke kojih nije bilo prilikom prvog čitanja



Transakcije na nivou JDBC konekcije

- Transakcijama upravlja baza podataka
- Možemo da biramo nivo izolacije transakcija za svaku konekciju
- connection.setTransactionIsolation(...)

Nivo izolacije	**Eliminiše problem**
READ_UNCOMMITTED	lost update
READ_COMMITTED	dirty read
REPEATABLE_READ	unrepeatable read
SERIALIZABLE	phantom read

Ko upravlja transakcijama kod EJB komponenti?

- container-managed tx: transakcijama upravlja kontejner na osnovu anotacija dodeljenih metodama
- bean-managed tx: transakcijama programski upravlja bean (JTA API)
- client-managed tx: transakcijama programski upravlja klijent (JTA API)

Container-managed transakcije

- Anotacija **@TransactionAttribute**

Vrednost	**Značenje**
REQUIRED	metoda se priključuje tekućoj tx, otvara novu ako tx ne postoji
REQUIRES_NEW	metoda uvek pokreće novu tx, ako postoji tekuća tx ona se suspenduje
MANDATORY	metoda mora da se izvršava u tx, koja mora biti ranije pokrenuta; ako je nema javlja se greška
SUPPORTS	metoda će se priključiti tekućoj tx, ako ona postoji; ako ne postoji, izvršava se bez tx
NOT_SUPPORTED	metoda se izvršava bez tx, čak i ako postoji tekuća tx
NEVER	metoda se izvršava bez tx; ako postoji tekuća tx, javlja se greška

Container-managed transakcije

- Primer 20: osa.pr20.container.*

Bean-managed transakcije

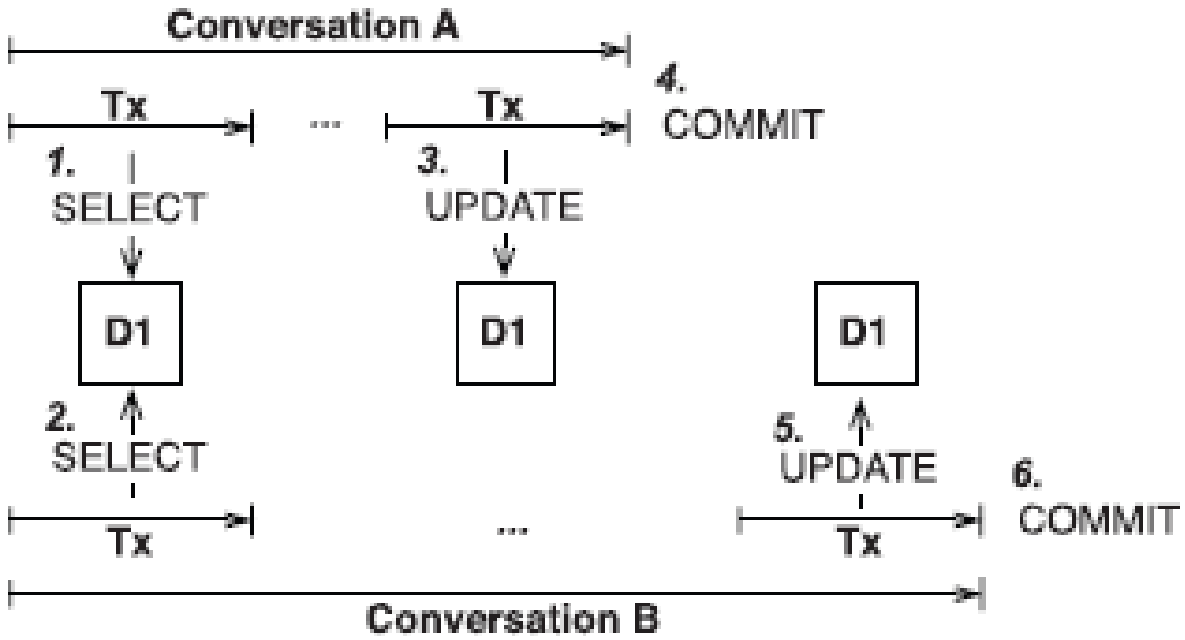
- Class-level anotacija **@TransactionManagement(BEAN)**
- Injekcija UserTransaction objekta pomoću @Resource anotacije
- Ručno pozivanje metoda
  - UserTransaction.begin()
  - UserTransaction.commit()
  - UserTransaction.rollback()
- Primer 20: osa.pr20.bean.*

Client-managed transakcije

- Klijent dobija UserTransaction preko JNDI lookup-a
- UserTransaction tx = (UserTransaction)ctx.lookup("java:comp/UserTransaction");
- Ručno pozivanje metoda
  - tx.begin()
  - tx.commit()
  - tx.rollback()
- Primer 20: osa.pr20.client.*
- (Beanovi koji se pozivaju obično su označeni kao bean-managed tx)

Optimističko i pesimističko zaključavanje

- Problem: operacija B će pregaziti izmene koje napravi operacija A, ne znajući za njih



Optimističko i pesimističko zaključavanje

- Rešenje 1 – pesimističko zaključavanje: svaka operacija treba da zaključa podatke **i za čitanje i za pisanje** sve dok se ne završi
  - u prethodnom primeru operacija B bi bila blokirana sve dok A ne otključa podatke
- Rešenje 2 – optimističko zaključavanje: svaka operacija pre izmene podataka treba da proveri da li je podatke neko drugi u međuvremenu menjao
  - poredi verziju podataka koje je pročitala sa onim što se trenutno nalazi u bazi
  - ovo poređenje mora da se izvodi u režimu pesimističkog zaključavanja
  - ako su podaci menjani, prijavi se greška korisniku

Optimističko i pesimističko zaključavanje

- Pesimističko zaključavanje garantuje ispravan rad
- Ali ima loše performanse
  - čak i ako dve transakcije pristupaju različitim redovima u tabeli može doći do blokiranja
- Optimističko zaključavanje polazi od pretpostavke da u praksi do kolizije dolazi jako retko
  - a situacije kada dođe do kolizije se otkrivaju i kontrola se vraća korisniku

Implementacija optimističkog zaključavanja

- Varijanta 1: poredimo sve vrednosti objekta sa vrednostima u bazi
  - nezgodno ako tabela ima puno kolona
- Varijanta 2: dodamo novu kolonu koja služi kao brojač izmena
  - na svaku izmenu u datom redu tabele inkrementiramo njenu vrednost

Optimističko zaključavanje i JPA

- Implementacija pomoću „varijante 2“
- Entity dobija još jedan atribut tipa int koji se označava anotacijom **@Version**
- Atribut se mapira na novu kolonu u tabeli
- Ako dođe do kolizije generiše se OptimisticLockException
- Primer 21 – osa.pr21.optimistic.*

Pesimističko zaključavanje i JPA

- Učitani entity može da se zaključa za čitanje pomoću EntityManagera:
- em.lock(entity, READ);
- Entity je zaključan do kraja transakcije
- Druga transakcija koja proba da zaključa objekat dobiće izuzetak
- Primer 21 – osa.pr21.pessimistic.*