Session Beans

Petr Aubrecht
CA

Vtipy budou tentokrát o krizi:

Co dnes probereme

- Session Beans – mezi perzistentní a prezentační vrstvou
  - Stateless/Stateful
  - Remote/Local
  - Životní cyklus
- Injection
- Transakce
- Security
- Implementace
- Novinky JEE 6
- IDE support (NetBeans), jestli zbyde čas

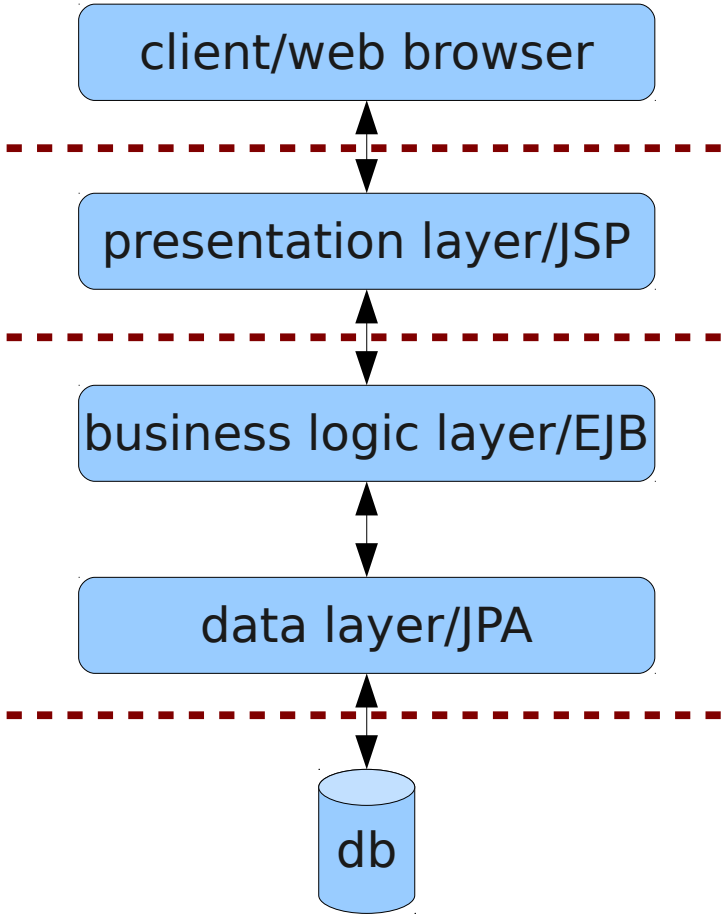
DOWNJONES

Transforming IT Management

Kde je business logic?

- Zatím znáte perzistentní vrstvu pro přístup do databáze (JPA).
- Dále znáte prezentační vrstvu – JSP, servlety.
- Pokud jste příznivci PHP, bude vám to stačit.
- Pokud chcete psát slušné programy, budeme se zabývat aplikační logikou, kterou umístíme mezi tyto vrstvy. Tím získáme logiku bez závislosti na konkrétní perzistenci i na konkrétní prezentaci
  - Tudíž ji můžeme testovat.
  - Použít i jiným způsobem, např. GUI aplikací, web service atp.

Proč business logic



- Příklady ze života
  - mediawiki vs. testovací aplikace
  - pomalá PHP hra
- V podstatě to je middleware (3tier application)
- Objekty jsou ve správě kontejneru
  - budeme tedy potřebovat JEE server
  - pooly, automatické vytváření/rušení, thready
  - load balancing (distribuovaná aplikace)
  - automatické transakce
  - distribuované transakce
  - security

Motivační příklad

- Zkusme vymýšlet aplikaci
  - zadání
  - co bude součástí aplikační logiky?
  - budeme si ukazovat možnosti a jak bychom jich v naší aplikaci využili

Enterprise Java Bean – příklad

```
@Stateless
public class BankOrderBean implements BankOrder {
  public void transfer(Account from, Account to,
                       double amount)...
```

- Třída implementující rozhraní, anotace u třídy
- Proč nemůžeme používat přímo entity?
  - kontrola bezpečnosti (oprávnění)
  - kontrola dostatečných financí
  - práce se dvěma účty najednou+logování
  - transakce

Enterprise Java Beans

- Druhy EJB
  - Entity (už znáte, dnes jsou z EJB vyjmuty)
  - Session bean (pro běžnou práci v rámci session)
    - stateless (nedrží stav)
    - stateful (drží stav)
    - local/remote
  - Message driven bean (pro JMS zprávy)
    - topic
    - queue

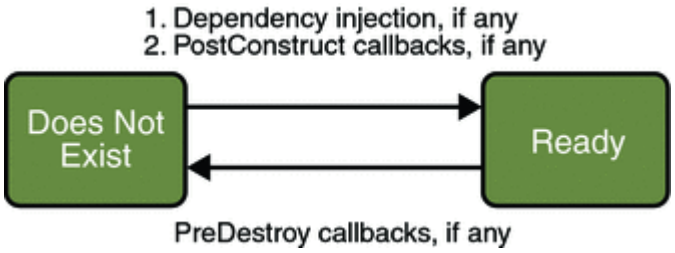
Local/Remote

- Protože jde o potenciálně distribuovanou aplikaci, je potřeba říci, co lze volat lokálně a co vzdáleně:
  - @Local interface ManzelkaBean {...}
    - nelze volat vzdáleně
    - výkonnější (performance)
  - @Remote interface TchyneBean {...}
    - EJB kontejner se může rozhodnout volat beanu na jiném fyzickém serveru vzdáleně kvůli load balancingu
    - při lokálním volání je zbytečná režie
  - určuje se tím způsob volání beany
  - umožňuje snadno dělat distribuovanou aplikaci

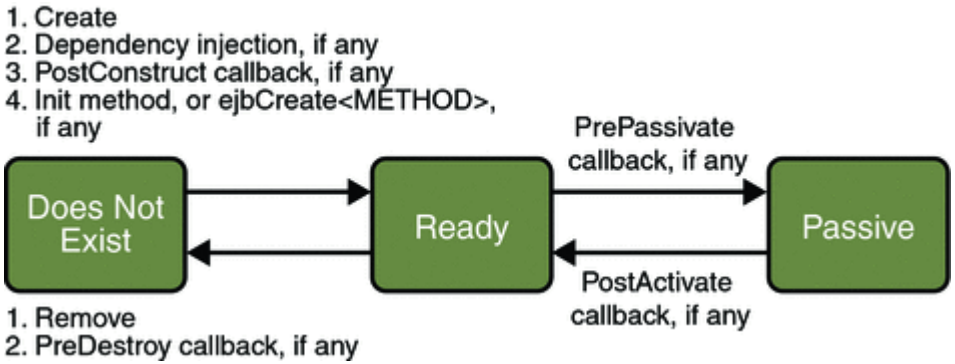
Životní cyklus Session Beans

- klient (např. webová session) má přístup právě k jedné instanci session bean
- kontejner se stará o pool bean
- životní cyklus

Stateless



Stateful



2M

Transforming IT Management

Injection

- o mnoho věcí se u EJB nemusíme starat, třeba o inicializaci EJB, které potřebujete:

```
@EJB potrebnaBeana;
```

- o její inicializaci se postará kontejner
- totéž přístup k entity manageru:

```
@PersistenceContext()
private EntityManager em;
```

- tyto věci jsou nastavené v serveru, NetBeans je umí při deploy automaticky zaregistrovat (podle definice v Services/Database)

Volání EJB

```
public class ManzelkaTest extends HttpServlet {
    @EJB
    private ManzelkaLocal manzelkaBean;
...
    manzelkaBean.odevzdat(vyplata);
...
```

- @EJB říká kontejneru, že chci použít EJB a že má injectovat nějakou instanci, která implementuje interface ManzelkaLocal. Dál se používá jako normální POJO.

A teď ty jednoduché věci – transakce

```
@Remote public interface BankOpRemote {...}

@Stateless
public class BankOpBean extends BankOpRemote {
  public void transfer(Account from, Account to, double amnt) {
    from.sub(amnt);
    to.add(amnt);
  }
```

- metoda transfer je defaultně opatřena transakcí!
- chování můžeme ovlivnit anotací

```
@TransactionAttribute(REQUIRES_NEW)
```

- možnosti: REQUIRED, REQUIRES_NEW, MANDATORY, NOT_SUPPORTED, SUPPORTS, NEVER
- http://java.sun.com/javaee/5/docs/tutorial/doc/bncij.html

Jednoduché zabezpečení

```
@DeclareRoles({"Administrator", "Manager", "Employee"})
@Stateless
public class BankOpBean extends BankOpRemote {
  @RolesAllowed("Manager")
  public void static transfer(Account from, Account to, double amnt) {
```

- případně přistoupíme k session:

```
@Resource SessionContext ctx;
        if (!ctx.isCallerInRole("Manager")) {
                throw new SecurityException(...);
        }
```

- http://java.sun.com/javaee/5/docs/tutorial/doc/bnbyk.html

Implementace (1)

- Každá (session) beana je „obalena“ speciální třídou, která dodává požadovanou funkcionalitu:
  - Session bean je za běhu zděděna, všechny metody přetíženy a přetěžující metoda obsahuje zpracování transakce, kontroly bezpečnosti ap.
  - Obsah vygenerované obalující metody je řízen anotacemi (příp. XML souborem).
  - Buď jde o jednoduché zavolání předka, nebo o stub dovolující vzdálené volání.

Implementace (2)

```
public void transfer(Account from, Account to, double amnt) {
    try {
        if (!ctx.isCallerInRole("Cashier")) {
                throw new SecurityException(...);
      }
      transaction.begin();
      super.transfer(from, to, amnt);
      transaction.commit();
    } catch (Exception e) {
      log.error(e);
      transaction.rollback();
    }
  }
```

Novinky v JEE 6

- No-interface View
- Singleton
- Asynchronous Session Bean Invocation
- Simplified Packaging, EJB

No-interface View

```
@Stateless
   public class HelloBean {
```

- Nyní není třeba používat rozhraní.
- V případě lokálního rozhraní stejně nemá velký význam.
- Klient samozřejmě nesmí použít new, reference se dělá normálně pomocí @EJB.

Singleton

- Nový typ session beany, který je v celé aplikaci pouze jednou.

```
@Singleton
   public class PropertiesBean {
```

- Dostupné jsou jako obvykle – přes @EJB.
- Metody jsou thread-safe, takže kontejner zajistí, že nejsou volány z více threadů najednou.
- Bohužel, není definováno chování v clusteru, pouze v rámci jedné JVM.

Asynchronous Session Bean Invocation

```
@Asynchronous
public java.util.concurrency.Future<Integer> performCalc(...) {
        // ... do calculation
        Integer result = ...;
        return new AsyncResult<Integer>(result);
}
```

- Metody mohou vracet výsledky asynchronně.
- Další podpora dlouhotrvajících operací.

Simplified Packaging, EJB

- Není potřeba balit jar a war do ear, stačí pouze war.
- EJB Lite je podmnožina EJB 3.1, která stačí ve většině případů (zapadá do kontextu profilů).
- EJB 3.1 mají nyní Embeddable API, které dovoluje iniciovat EJB i v desktopových aplikacích (vhodné především pro testování).
- Anotace @Startup zařídí, že singleton je vytvořen při startu aplikace, takže @PostConstruct je volán na začátku.

Náměty k diskuzi

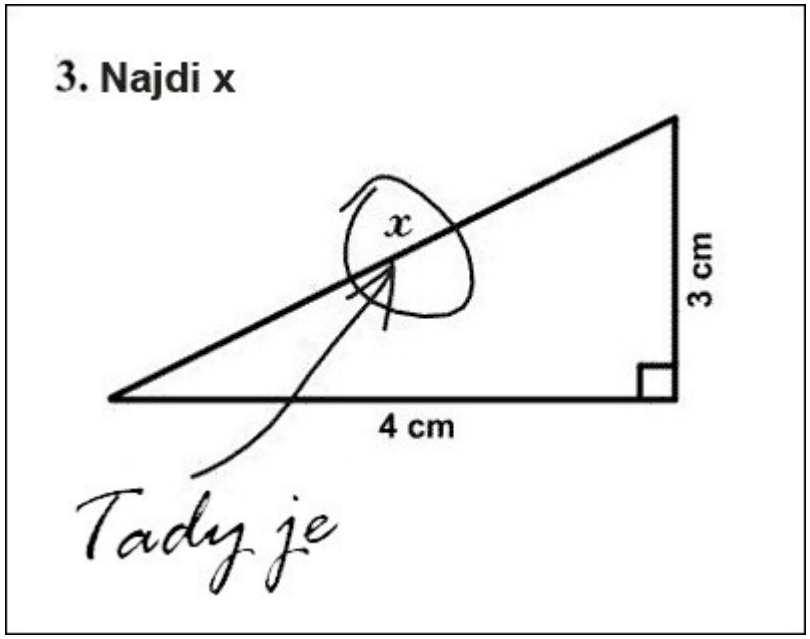
- proč nestačí business logika v backing beanách?

Linky na Session bean

- http://java.sun.com/javaee/5/docs/tutorial/doc/
  - bible, součástí zkoušky bude doslovná znalost každého slova
- kniha Enterprise Java Beans 3.0

Apple

Ještě dva, jestli si je zaslouží!



Přijde zákazník do obchodu s PC s reklamací.
Povídá prodavači:
"Mám nainstalované Windows 95 a mám s nimy problémy."
Prodavač: "To už jste říkal."