Transaction API
+ další poznámky

Petr Aubrecht
CA

Co dnes probereme

- Slíbené JTA
- Třívrstvá architektura
- Různé z JEE 6
- Doporučení pro postup práce v JEE

Java Transaction API (JTA)

- Něco definice:
  - Java Transaction API (JTA) specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.
  - The JTA specification was developed by Sun Microsystems in cooperation with leading industry partners in the transaction processing and database system arena. See JSR 907. Also see the Java Transaction Service (JTS) page

Container Managed Transactions

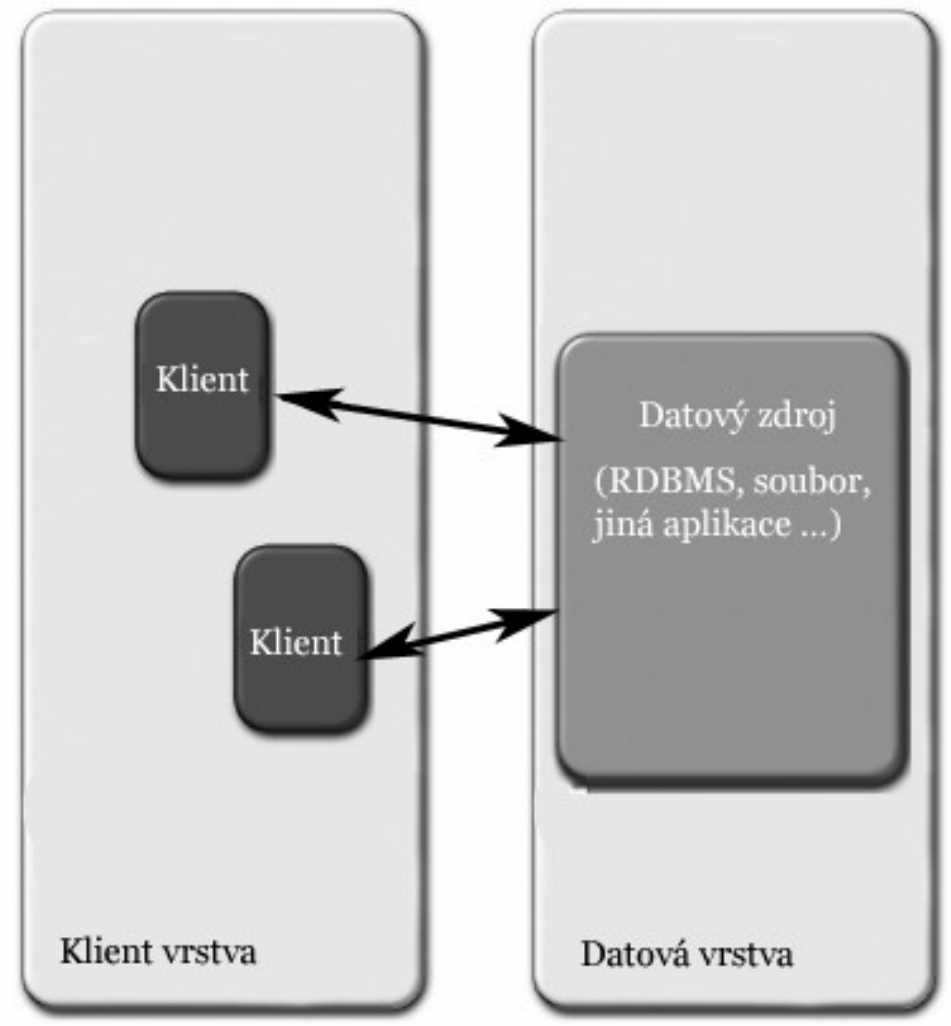
- Probrali jsme u session bean, default nastavení je @Required, tedy transakce je vždy použita pro každou metodu (session beany)

Bean Managed Transaction

- javax.transaction.UserTransaction interface
  - begin()
  - commit()
  - rollback()

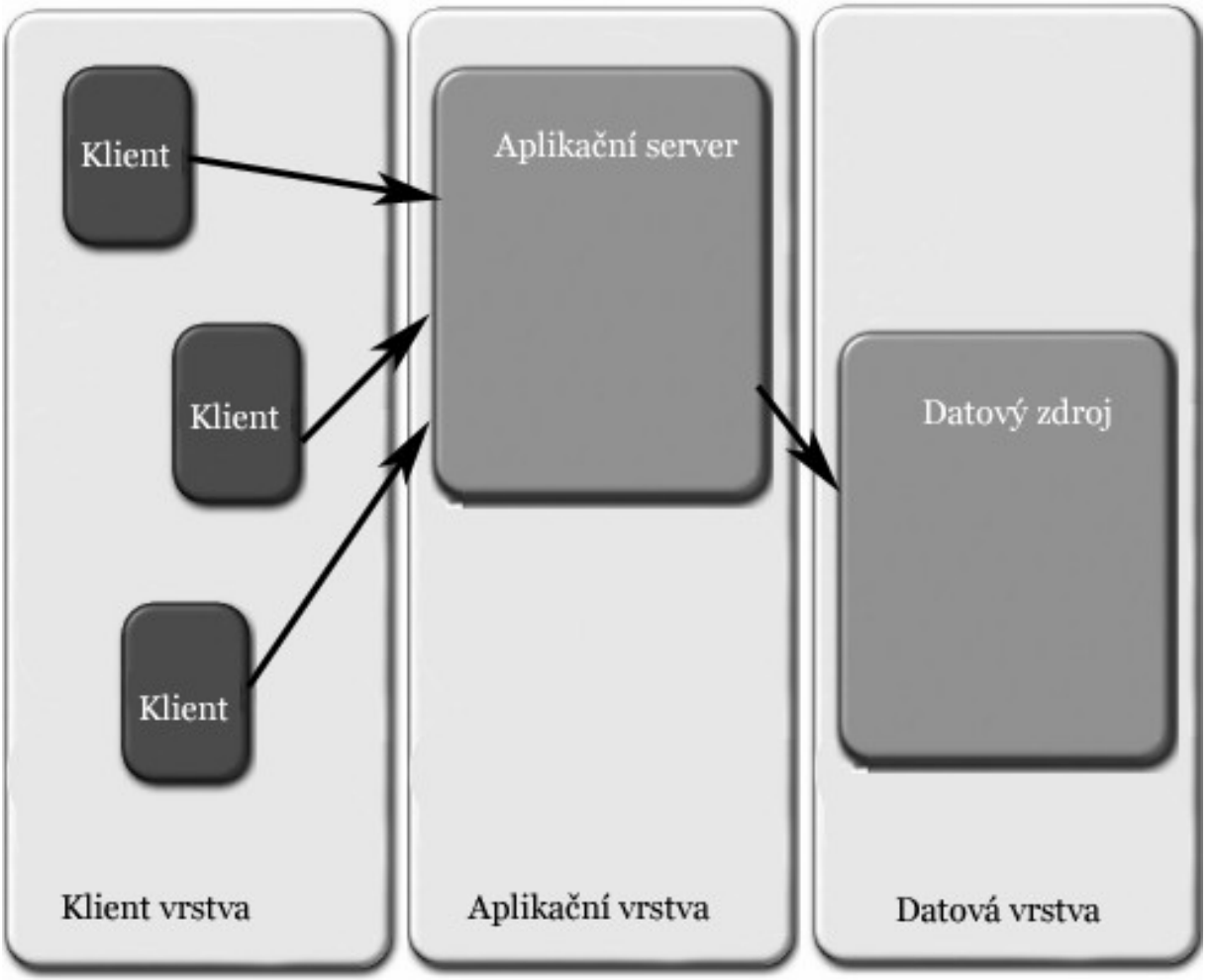
Třívrstvá architektura – 2 vrstvy

- obrázky jsem si půjčil od Dagiho (dagblog)
- 2 vrsvy byly prima v 90. letech
- kde to nestačí?



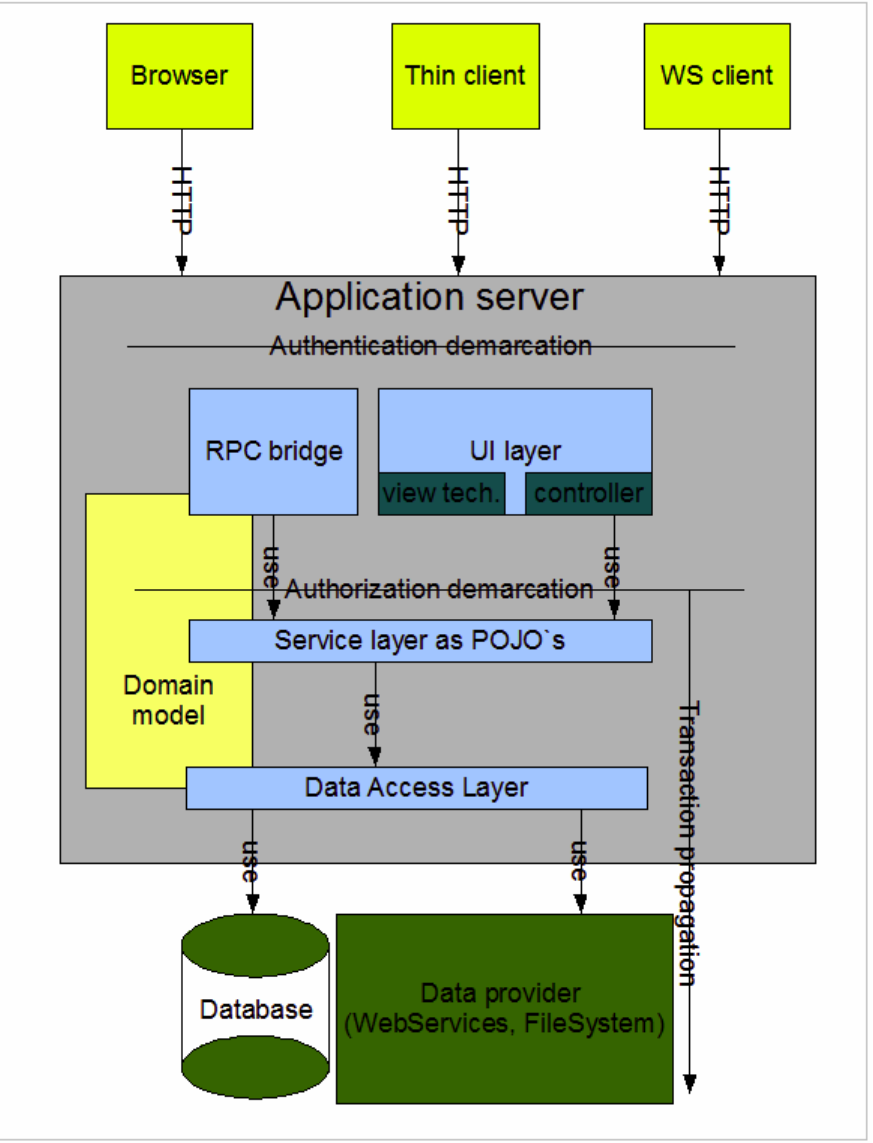
Třívrstvá architektura – 3 vrstvy

- zjednodušený model



Třívrstvá architektura – 3 vrstvy

- dnešní architektura



Local vs Remote

10!	L (ms)	R (ms)
10x	32	13
100x	17	57
1 000x	236	400
10 000x	566	990
100 000x	2,693	4,259
1 000 000x	24,498	42,594

1 000!	L (ms)	R (ms)
10x	3	9
100x	41	29
1 000x	432	868
10 000x	4,326	6,216
100 000x	44,342	42,498
1 000 000x	401,302	418,383

100!	L (ms)	R (ms)
10x	0	1
100x	4	5
1 000x	49	124
10 000x	1,132	758
100 000x	6,543	8,942
1 000 000x	67,017	84,830

10 000!	L (ms)	R (ms)
10x	95	106
100x	565	307
1 000x	3,552	3,874
10 000x	38,518	38,518
100 000x	380,394	387,810
1 000 000x	4,689,103	4,768,913

Global JNDI Names

- v současnosti jsou JNDI názvy specifické pro servery, takže je testování a deployment poněkud magie
  - Glassfish: com.os.ent.CustomerManagerRemote
  - JBoss: ejb3sampleapp/CustomerManager/remote
  - Oracle: CustomerManager
- ve 3.1 bude univerzání (alternativní) pojmenování:
  - java:global[/<application-name>]/<module-name>/<bean-name>#<interface-name>

Calendar Based Timer Services

- možnost spouštět akce v určitý čas, např. každé pondělí v 6:15
- podobně jako cron, diskuze se vede, jestli podporovat syntaxi cron (protože ji každý zná)

```
@Stateless
public class NewsLetterBean {
   @Schedule(minute=”15”, hour=”6”, dayOfWeek=”Mon”)
   public void sendNewsLetter() {
      ...
   }
}
```

EJB 3.1 Embeddable Container

- Problém s deploymentem lze řešit pomocí tzv. embeddable containeru. Neběží jako server, je iniciován kódem. Must have pro testování:

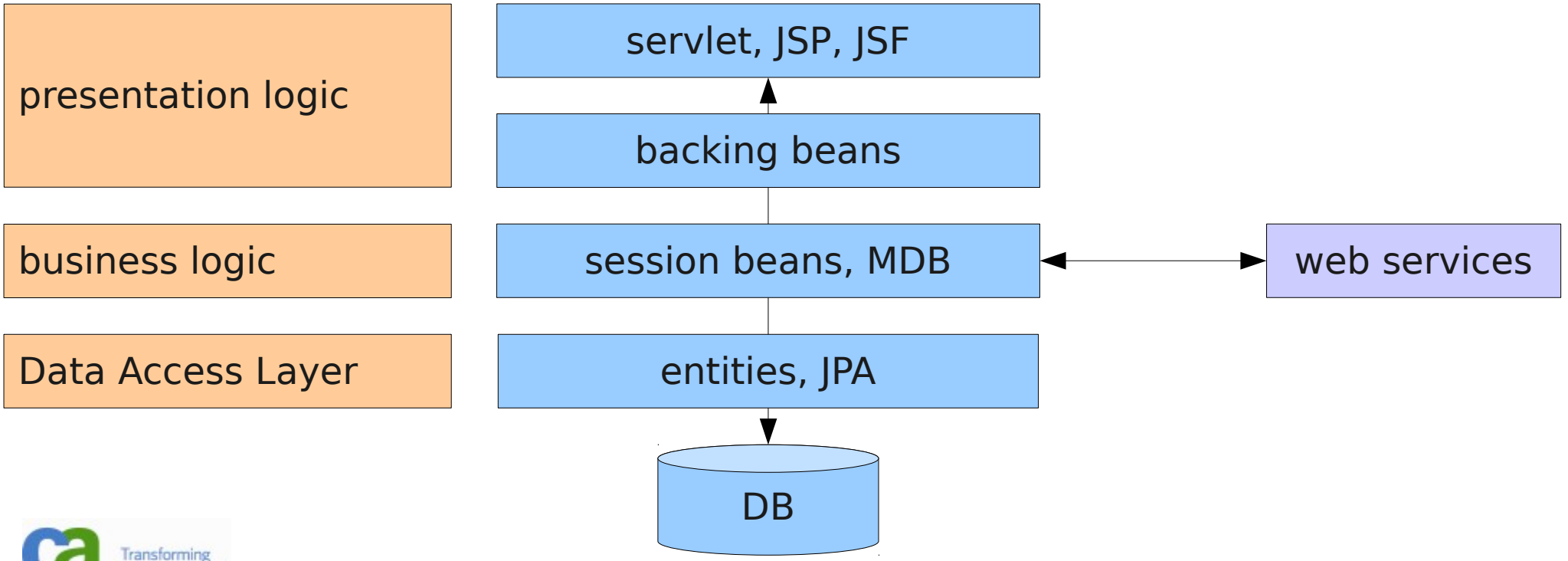
```
@Test
    public void testStates() throws Exception {
        Properties props = new Properties();
        props.setProperty(Context.INITIAL_CONTEXT_FACTORY,
                "org.apache.openejb.client.LocalInitialContextFactory");

        InitialContext context = new InitialContext(props);

        StateRegistry stateOne = (StateRegistry) context
                .lookup("StateRegistryBeanLocal");

        StateRegistry stateTwo = (StateRegistry) context
                .lookup("StateRegistryBeanLocal");

        stateOne.setState("MaryLand", "MD");
        stateTwo.setState("Virginia", "VA");
```

Profily JEE

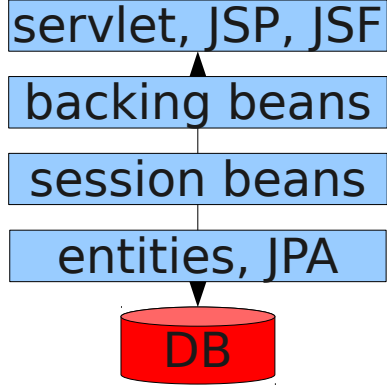
- Již nyní je JEE opravdu široká technologie s mnoha požadavky, které většina aplikací ani nevyužije (viz oblíbený terč CORBA)
- Pro JEE 6 je navržena myšlenka profilů, kdy kontejner musí implementovat určité technologie; není potřeba implementovat úplně všechny.
- Aktuálně je definován pouze Web Profile, který obsahuje webové technologie, EJB mimo JMS a JavaMail, ale nemusí obsahovat CORBA, web services, management a security.
- Objevují se nové technologie, které budou později začleněny jako nepovinné (např. SIP Servlety, JAX-RS).

End-to-end

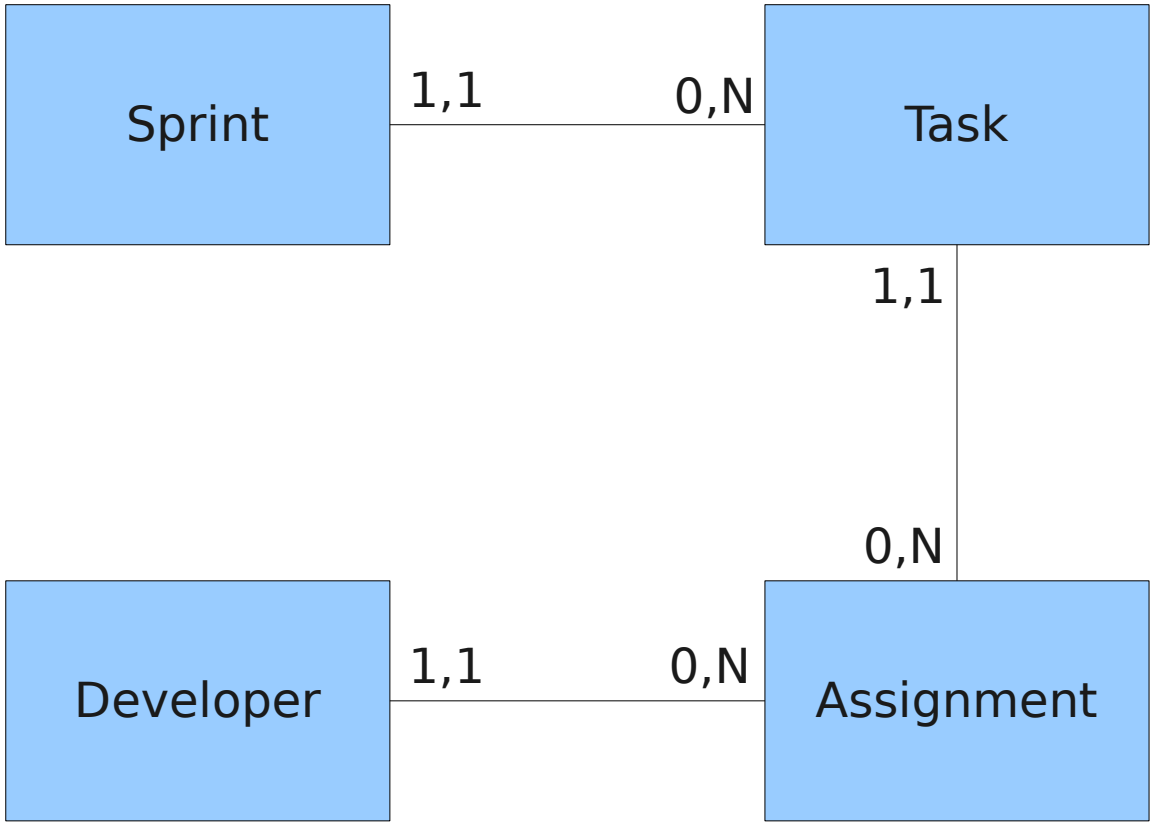
- Postup od začátku do konce... všechno



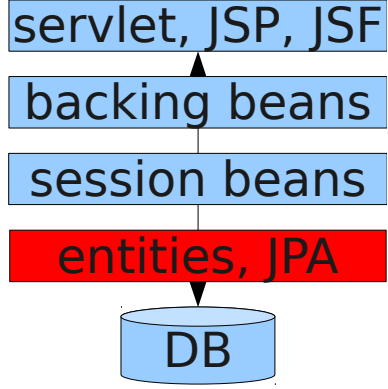
Databáze



- ER

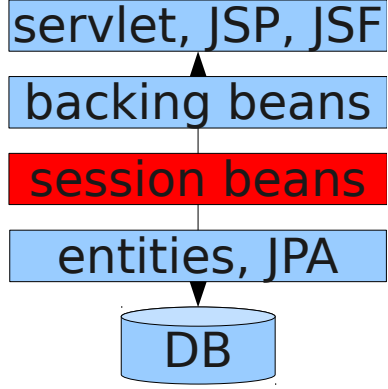


JPA (DAO, DAL)



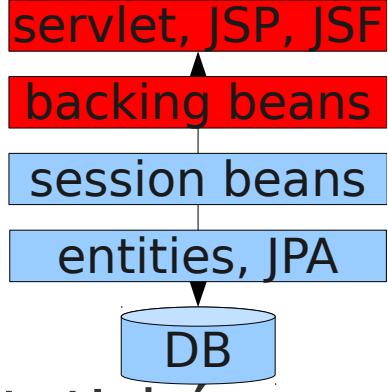
- prostě vygenerujeme
- doplníme @GeneratedValue(strategy=GenerationType.IDENTITY) a případně opravíme
- doporučuji mít dopředu připravená data (snáze se kontroluje)

Session beans (business logic)



- budeme potřebovat přístup k Entity manageru:
  @PersistenceContext
  private EntityManager em;
- samozřejmě necháme IDE generovat tyto části
- funčnost ověříme pomocí web services (buď z webového interface Glassfishe nebo necháme NB vygenerovat ws clienta)

Mock prezentační vrstvy



- vyrobíme webové stránky, jak mají vypadat, se statickým obsahem.
- Postupně je budeme nahrazovat funkčním obsahem a psát backing beany.
- Nepište velké části bez dalších návazností, předejdete tak velkým přepisům.
- Pište vždy funkcionalitu vertikálně, od session beans, backing bean až po JSF.

A rozšiřujeme...

- Budeme dále rozšiřovat; když už máme základ aplikace, přidávání dalších stránek je radost!
- Postupujte inkrementálně, práce roste viditelně pod rukama a motivuje vás k dalšímu vývoji! Rozdělená práce, kde dlouho nevidíte výsledek, demotivuje!
- Nesnažte se mít úchvatnou aplikaci hned z počátku, strávíte příliš času nad nedůležitými částmi a nakonec nebude aplikace hotová.
- Soustřeďte se na předem dohodnutou funkcionalitu. Až ji budete mít hotovou, můžete ji zkusit vylepšit, případně dodat další.

Linky

- http://java.sun.com/javaee/
- http://java.sun.com/developer/technicalArticles/JavaEE/JavaEE6Overview.html
- http://java.sun.com/javaee/javaserverfaces/
- http://download-book.net/jsf-2.0-book-pdf.html
- Beginning Java EE 6 with GlassFish 3
- JavaServer Faces 2.0, The Complete Reference