Komponenty obchodní logiky EJB

Tomáš Vlk (vlktoma5@fit.cvut.cz)

May 30, 2019

Úvod

Enterprise Java Beans 3[1] jsou řízené, serverové komponenty umožňující modulární tvorbu podnikových aplikací. Specifikace EJB3 je součástí množiny aplikačního programového rozhraní[2] definující Java Enterprise Edition.

Hlavním cílem EJB je oddělit je oddělit business logiku aplikace od prezentační[3] a persistentní vrstvy[4], ale také zajistit předpoklady pro integraci s ostatními technologiemi[5].

Session Beans

Stateless session beans

Bezstavové beany neuchovávají stav relevantní pro klienta mezi obsluhou jeho jednotlivých požadavků. Pro obsloužení každého požadavku klienta je mu vždy na serveru přidělena samostatná instance, která je vyhrazena pouze tomuto klientovi v rámci jednoho daného požadavku. Důsledkem tohoto chování je thread safety.

Většinou se uchovávají v poolu ze kterého jsou odebrány na vykonání požadavku a poté jsou zase vráceny zpět. Nutná anotace jako **@Stateless**.

Při tvorbě instance beanu je možná injekce případných referencí a také možnost volat metodu **@PostConstruct**, která může zprostředkovat věci nutné k fungování beany[6]. Opakem metody **@PostConstruct** je metoda **@PreDestroy**, která se volá před smazáním beany. Vpřípadě systémové vyjímky se metoda **@PreDestroy** nevolá.

[1] EJB3
[2] API
[3] Například JSP
[4] Persistentní vrstva zajišťuje CRUD operace.
[5] Například JMS, JNDI a CORBA
[6] Například otevření přístupu do databáze

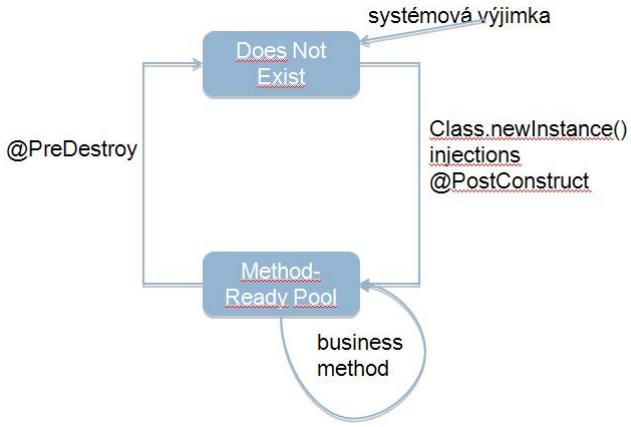

Figure 1: Životní cyklus stateless beanu

Stateful session bean

Jsou schopné uchovat svůj stav v rámci jedné session mezi jednotlivými voláními klienta. Každý klient má tedy vlastní instanci, která odpovídá pouze jemu a uchovává informace z předchozích interakcí. Může být ve stavu pasivace[7], pokud je nutné uvolnit paměť na serveru.

Musí být anotována pomocí **@Stateful**. Tvorba a odstraňování beanu probíhá stejně jako u stateless verze. Jediným rozdílem je pasivace, kdy bude zavolána metoda **@PrePasivate** před pasivací a následně metoda **@PostActivate** po opětovném obnovení beanu. Je zde také možnost nastavit beanu timeout po kterém bude automaticky odstraněn.

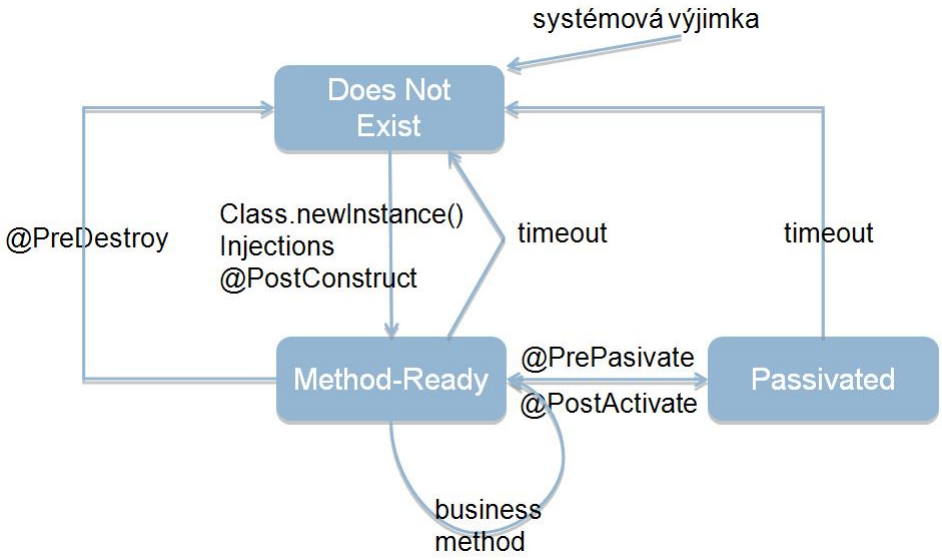

Figure 2: Životní cyklus statefull beanu

[7] Uložen pomocí perzistentní vrstvy

Singleton session bean

Objekty s globálně sdíleným stavem. Přístup je řízen buď kontejnerem, nebo samotným beanem pomocí anotace **@Lock** a parametrem pro zámku pro čtení nebo zápis při volání metod. Anotace **@Startup** umožnuje vznik singletonu už při vzniku EJB kontejneru[8]. Singleton je nutné anotovat jako **@Singleton**.

Message Driven Beans

Beany řízené zprávami, jsou asynchroní, tudíž se používají k operacím, které nevyžadují okamžitou odpověď. Umožnují událostně řízené zpracování uvnitř EJB kontejneru.

EJB kontejner

Dedikovaný virtuální prostor v aplikačním serveru pro EJB komponenty. Hlavní problematika řešená pomocí EJB kontejneru je:

- **Komunikace se vzdáleným klientem** - zjednodušuje komunikaci mezi klientem a aplikací
- **Dependency injection** - zajišťuje naplnění deklarovaných proměnných[9]
- **Řízení stavu** - kontejner udržuje v paměti stavy jednotlivých stavových (stateful) beanů a tím i vzdálený stav u klienta, kterému se jeví stav jakoby uložen lokálně
- **Pooling** - vytváření poolu instancí pro bezstavové beany a message-driven beany
- **Řízení životního cyklu** - stará se o vytváření, inicializaci a destrukci instancí beanů a další události
- **Messaging** - umožňuje MDB poslouchat na JMS destinacích a konzumovat zprávy a zároveň odstiňuje programátora od komplikovaného API Java Messaging Service
- **Management transakcí** - beany deklarují transakční vlastnosti metod, kontejner řeší commit a rollback
- **Bezpečnost** - deklarace přístupů na úrovni tříd a metod
- **Podpora souběžného zpracování** - vývojář nemusí řešit problémy synchronizace souběžných přístupů ke sdíleným datům
- **Správa interceptorů** - komponenty umožňující odchytávat okamžik před a po volání metody
- **Asynchronní volání metod**

Rozhraní

Rozhraní slouží pro definování metod, které bude muset bean implementovat a poskytovat navenek pro manipulaci s jeho datovými atributy. Každý EJB musí implementovat aspoň jedno z níže uvedených rozhraní:

- **Lokální rozhraní** - pokud není žádné rozhraní u třídy definováno, má se za to, že bean je lokálního typu, což v praxi znamená, že běží a jeho metody jsou volány pouze v rámci jednoho Java Virtual Machine. Anotuje se jako @Local.
- **Vzdálené rozhraní** - značeno anotací @Remote. Používá se při volání metod takovéhoto EJB z jiného vzdáleného JVM. K volání metod slouží distribuovaný protokol. Tímto způsobem lze vytvářet plnohodnotné klientské aplikace, které nejsou vázané pouze na webový prohlížeč.

[8] Vhodné pro načítaní statických hodnot.
[9] Například dalšími EJBeany, JMS, datovými zdroji, atd.

- **Endpoint rozhraní** - slouží k označení rozhraní přístupného pro webové služby. Anotováno @javax.jws.WebService.
- **Message rozhraní** - implementováno Message Driven Beany. Rozhraní MD beanu je voláno Java Message Službou, když je doručena nová zpráva do fronty obsluhována nějekým Message-oriented-middlewarem.

Embedded kontejner

Nabízí možnost spouštět EJB aplikace v prostředí Java SE.

Injektování EJB

Do EJB můžeme injectovat další EJB. Dochází k tomu při vytvoření beany, to je zajištěno pomocí EJB kontejneru. Anotace **@EJB** se používá pro injektování jiné beany, a anotace **@Resource** pro injektování zdrojů dat nebo kontextu.