Co je nového v Java EE 6

Petr Adámek

IBA CZ, s.r.o.

Disclaimer

Ačkoliv jsem snažil tuto prezentaci připravit co nejpečlivěji, může obsahovat nepřesnosti nebo dokonce nepravdivé informace. S popisovanými novinkami nemám téměř žádné praktické zkušenosti a většinu informací čerpám ze specifikací, přičemž moje interpretace specifikací nemusí být správná.

(podtržené = součást Java SE)

Java EE 5 (JSR 244, květen 2006)

- EJB 3.0, JPA 1.0, Servlets 2.5, JSP 2.1, JSF 1.2, EL 1.0, JSTL 1.2, Web Services 1.2, JAX-RPC 1.1, JAX-WS 2.0, WS Metadata 2.0, JAXB 2.0, JAXR 1.0, SAAJ 1.3, JMS 1.1, JTA 1.1, Java Mail 1.4, JAF 1.1, Java EE Connector 1.5, JACC 1.1, Common Annotations 1.0, StAX 1.0, JSP Debugging 1.0, Java EE Management API 1.1, Java EE Deployment API 1.2

Java EE 6 (JSR 316, prosinec 2009)

- EJB 3.1, JPA 2.0, Servlets 3.0, JSP 2.2, JSF 2.0, EL 1.1, JSTL 1.2, Web Services 1.3, JAX-RPC 1.1, JAX-WS 2.2, WS Metadata 2.0, JAXB 2.2, JAXR 1.0, SAAJ 1.3, JMS 1.1, JTA 1.1, Java Mail 1.4, JAF 1.1, Java EE Connector 1.6, JACC 1.3, Common Annotations 1.1, StAX 1.0, JSP Debugging 1.0, Java EE Management API 1.1, Java EE Deployment API 1.2, **JAX-RS 1.1**, **Web Beans (JSR-299) 1.0**, **Dependency Injection for Java (JSR-330) 1.0**, **JASPIC 1.0, Bean Validation 1.0**

Profily

- Jedním z cílů specifikace je modularizace platformy.
- Profily reprezentují konfiguraci platformy vhodnou pro určitý typ aplikací (např. Java EE 6 Web Profile).
- Profil může vypustit technologie, které nejsou pro daný typ aplikací potřebné (např. JMS pro Web Profile).
- Profil může zahrnout technologie, které nejsou ve standardní edici Java EE (např. JSR-286 pro hypotetický Java EE Portal Profile).
- Profil může označit nějaké technologie za volitelné.
- Momentálně pouze *Java EE 6 Web Profile*, další přijdou jako samostatné specifikace

Java EE 6 Web Profile

- *Java EE 6 Web Profile Specification* (vznikla v rámci JSR 316 jako samostatná specifikace)
- Povinné technologie: Servlet 3.0, JSP 2.2, EL 2.2 (?), JSP Debugging (JSR-45) 1.0, JSTL 1.2, JSF 2.0, Common Annotations 1.1, EJB 3.1 Lite, JTA 1.1, JPA 2.0, Bean Validation 1.0, Managed Beans 1.0, Interceptors 1.1, Web Beans (JSR-299) 1.0, Dependency Injection for Java (JSR-330) 1.0 (!)
- Žádné volitelné technologie

Typický problém Java EE verze 5:

- Služby jako Dependency Injection fungují pouze u komponent, jejichž životní cyklus řídí kontejner (EJB komponenty, servlety, portlety, apod.).
- Chybí podpora pro POJO komponenty.

Tento problém se v Java EE 6 snaží vyřešit specifikace univerzálního komponentového modelu, který je společný pro celou Java EE 6.

Vznikla v rámci JSR 316 jako samostatná specifikace ***Managed Beans 1.0 Specification***

Managed Beans 1.0 Specification

- *Managed Beans are container-managed objects with minimal requirements, otherwise known under the acronym "**POJO**s" (Plain Old Java Objects). They support a small set of basic services, such as **resource injection**, **lifecycle callbacks** and **interceptors**. Other, more advanced, aspects will be introduced in companion specifications, so as to keep the basic model as simple and as universally useful as possible.* [Managed Beans 1.0 Specification]
- Jednoduchý a univerzální komponentový model pro platformu Java EE, který je společným základem pro ostatní specifikace.
- Díky společnému základnímu modelu je možné později jednoduché POJO komponenty snadno transformovat na složitější typy komponent (např. EJB)

Managed Bean

Třída označená anotací `javax.annotation.ManagedBean`

Třída nesmí být finální, abstraktní, nebo vnořená nestatická. Nemusí být serializovatelná. Může být pojmenovaná, jméno musí být unikátní v rámci modulu (včetně EJB komponent).

Rozšiřující specifikace může umožnit označení komponenty specifickou anotací (místo `ManagedBean`)

```
@ManagedBean(“cart“)
@Interceptor(
    MyInterceptor.class)
public class ShoppingCart
{
    @Resource
    DataSource ds;

    @PostConstruct
    void init() { … }

    @PreDestroy
    void destroy() { … }

    …
}
```

Contexts and Dependency Injection for the Java EE platform (JSR-299) – Web Beans 1.0

- Snaha o zavedení univerzálního modelu pro Dependency Injection, který bude fungovat pro všechny typy komponent v Java EE (EJB, Servlety, JSP)
- Silně inspirováno frameworky *Seam* (leader je Gavin King) a *Google Guice*
- Podpora pro injektování POJO komponent.
- Web Bean je třída, obsahující business logiku. Životní cyklus Web Bean komponenty řídí Web Bean manažer, který také řeší závisloti mezi jednotlivými Web Bean komponentami.
- http://docs.jboss.org/webbeans/reference/1.0.0.ALPHA1/en/html/intro.html#d0e144
- http://blog.krecan.net/2009/01/09/webbeans/

EJB 3.1 (JSR-318)

- EJB komponenty bez lokálního nebo vzdáleného rozhraní.
- EJB komponenty mohou být zabaleny přímo ve .war souboru.
- Embeddable API pro spouštění EJB komponent přímo v Java SE.
- Singleton Beans (anotace `@javax.ejb.Singleton`).
- Automaticky vytvářené EJB časovače.
- EJB časovače podporují výrazy založené na kalendáři (cron-like).
- Podpora pro asynchronní volání metod u komponent Session EJB.
- EJB 3.1 Lite (pro Java EE 6 Web Profile).
- Standardizovaná globální JNDI jména.

EJB 3.1 – EJB komponenty bez lokálního nebo vzdáleného rozhraní.

S business rozhraním

```
// Component definition
@Stateless
@Local(Catalog.class)
public class CatalogBean
    implements Catalog {…}

// Dependency injection
@EJB Catalog catalog;

// JNDI lookup
@Resource SessionContext ctx;
…
Catalog catalog = (Catalog)
    ctx.lookup(“catalog”);
```

Bez business rozhraní

```
// Component definition
@Stateless
public class CatalogBean {…}

 // Dependency injection
@EJB CatalogBean catalog;

// JNDI lookup
@Resource SessionContext ctx;
…
CatalogBean catalog =
    (CatalogBean)
    ctx.lookup(“catalog”);
```

Embeddable API pro spouštění EJB komponent přímo v Java SE.

- Umožňuje rychle a snadno inicializovat EJB kontejner i v klasické Java SE aplikace.

```
Properties props = new Properties();
props.setProperty(…);
EJBContainer ec = EJBContainer.createEJBContainer(props);

Context ctx = ec.getContext();
String beanName = "java:global/catalog/CatalogBean";
Catalog catalog = (Catalog) ctx.lookup(beanName);
```

Singleton Beans

- Další typ komponenty Session EJB (vedle Stateful a Stateless).
- Anotace `@javax.ejb.Singleton`.

```
@Singleton
public class MySingleton {

    @PersistenceContext
    private EntityManager em;

    @PostConstruct
    private void init() { … };

    @PreDestroy
    private void destroy() { … };
    …
}
```

Časovače

- Automatické spouštění (nastavení v deployment deskriptoru nebo pomocí anotací)
- Výrazy založené na kalendáři (jako Cron)

```
@Singleton
public class MyTimerBean {

    @Schedule(second="0", minute="0", hour="0",
        dayOfMonth="1", month="*", year="*")
    public void generateMonthlyNewsLetter() { … }

    @Schedules({
        @Schedule(hour=”12”, dayOfWeek=”Mon-Thu”),
        @Schedule(hour=”11”, dayOfWeek=”Fri”)
    })
    public void sendLunchNotification() { … }

}
```

Podpora pro asynchronní volání metod u komponent Session EJB

```
@Stateless
public class MyServiceBean {

    @Asynchronous
    public void operationA(…) { … };

    @Asynchronous
    public Future<OperationResult> operationB(…) {
        OperationResult result = …;
        return new AsyncResult<OperationResult>(result);
    }
}
```

Pozor! V prvním případě neexistuje způsob, jak klientovi předat případnou vyhozenou výjimku!

EJB 3.1 Lite

	EJB 3.1 Lite	EJB 3.1 Full
Session Beans	Ano	Ano
Message-driven beans	Ne	Ano
2.x/1.x CMP/BMP beans	Ne	Ano*
Java Persistence 2.0	Ano	Ano
Local/No interface	Ano	Ano
3.0 Remote interface, 2.x Remote Home/Component interface	Ne	Ano
JAX-WS Web Service Endpoint	Ne	Ano
JAX-RPC Web Service Endpoint	Ne	Ano*
EJB Timer Service	Ne	Ano
Asynchronous Session EJB method invocation	Ne	Ano
Interceptors	Ano	Ano
RMI-IIOP interoperability	Ne	Ano
Container-managed/Bean-managed transactions	Ano	Ano
Declarative and Programmatic Security	Ano	Ano
Embeddable API	Ano	Ano**

Standardizovaná globální JNDI jména

- Dříve to bylo specifické pro konkrétní kontejner.
- Nyní `java:global[/<app-name>]/<module-name>/<bean-name>[!<fully-quali-fied-interface-name>]`

Application Server	Proprietary Global Name
JBoss global JNDI name	action-bazaar/PlaceBidBean/remote
GlassFish global JNDI name	PlaceBid
WebSphere Community Edition global JNDI name	action-bazaar-ejb/PlaceBidBean/PlaceBid
Oracle Application Server (OC4J) global JNDI name	PlaceBidBean

Kvíz

```
@EJB Cart cart1;
@EJB Cart cart2;
…
System.out.println(cart1.equals(cart2));
```

Co se vypíše na výstup?

Java Persistence 2.0 (JSR-317)

- Podpora pro kolekce základních typů a objektů typu Embedable.
- Rozšířená podpora map (klíčem může být základní typ, Embedable objekt nebo Entita).
- Podpora pro vnořené třídy typu Embedable.
- Podpora pro odvozené klíče (závislé klíče u slabých entitních množin).
- Jednosměrné OneToMany vazby bez vazební tabulky.
- Podpora pro seznamy s explicitně stanoveným pořadím.
- Orphan Removal.
- Pesimistické zamykání.
- BigInteger a BigDecimal mohou být primárním klíčem.

Java Persistence 2.0 (JSR-317)

- Rozšíření EntityManager API (clear(), getCache(), getFactory(), …)
- Cache API.
- Criteria API.
- Standardní názvy properties (např. `javax.persistence.jdbc.driver` místo `eclipselink.jdbc.driver`)
- Rozšíření JPQL
- Integrace s Bean Validation (včetně napojení na životní cyklus entity)

Bean Validation (JSR-303)

- Framework pro deklarativní definici omezujících podmínek a následnou validaci
- Definice pomocí anotací nebo XML
- Sada předdefinovaných anotací pro standardní omezující podmínky (@Null, @NotNull, @AssertFalse, @AssertTrue, @Min, @Max, @Size, @Pattern)
- Možnost definice vlastní anotací pro další omezující podmínky.

Servlets 3.0, JSP 2.2, JSF 2.0, EL 1.1, …

Pokračování příště

?