Java Message Services (JMS) a další užitečné věci, co se jinam nevešly

16. prosince 2020

V tomto semináři si představíme machnismus zasílání zpráv, který umožňuje velmi rozvolnit vazby mezi jednotlivými komponentami a zjednodušit tak vývoj aplikace. Dále si představíme dvě velmi užitečné funkce, kterými jsou volání metod v daném čase a přístup k rozhraní elektronické pošty.

1 Java Message Services

Charakteristickým rysem Java EE je snaha o co největší oddělení jednotlivých komponent tak, aby mohly být

(i) vyvíjeny a testovány odděleně, což zjednodušuje vývoj a umožňuje jej rozdělit mezi více týmů,

(ii) provozovány na více strojích, což umožňuje lepší škálování aplikace.

Nejdále v této snaze jdou Java Message Services (JMS), které umožňují standardizované zasílání zpráv nejen mezi jednotlivými komponentami či moduly jedné aplikace, ale dokonce mezi jednotlivými aplikacemi. Narozdíl od standardního volání metod se jedná o asynchronní komunikaci, tj. klient odešle zprávu a pokračuje dál, a není vyžadována znalost žádného specifické rozhraní pro komunikaci.

Klíčovým prvkem architektury postavené na zasílání zpráv je *message broker*, který se stará o příjem zpráv od odesílatelů a jejich distribuci k adresátům. Jako message broker může vystupovat buď samostatný software nebo aplikační server, pokud tuto funkcionalitu zahrnuje.

Zasílání zpráv lze realizovat ve dvou režimech.

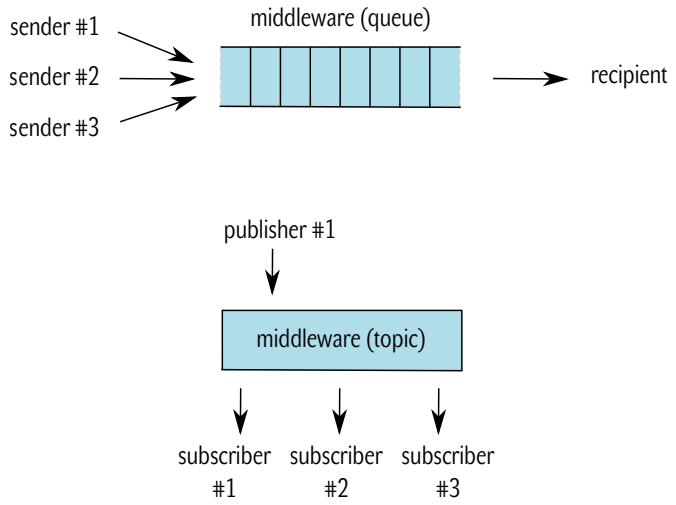
(i) V režimu *point-to-point* odesílatel odešle zprávu do fronty, odkud si ji vyzvedne příjemce[1]. V režimu point-to-point je možné mít několik odesílatelů současně, to je výhodné, pokud přichází požadavky z různých zdrojů, např. webová aplikace, mobilní aplikace, interní informační systém. Současně je možné mít několik příjemců, přičemž každá zpráva je doručena právě jednomu příjemci, to je vhodné v situacích, kdy máme velké množství požadavků a potřebujeme rozložit jejich zpracování.

(ii) V režimu označeném jako *publisher-subscriber* odesílatel odešle zprávu identifikovanou nějakým tématem a příjemci přihlášení k odběru tohoto tématu obdrží danou zprávu. To umožňuje velice jednoduše přidávat další funkcionalitu bez zásahu do existujícího kódu.

[1] Zpráva z fronty zmízí až je potvrzeno, že ji příjemce převzal.

Uvažujme například redakční systém, kde máme funkci *vložení článku*. Při vložení článku do systému je odeslána zpráva s tématem "Nový článek", která obsahuje nadpis článku, autora, text, apod. A nyní podle potřeby můžeme přidávat funkce. Jeden příjemce může aktualizovat RSS-feed, další příjemce aktualizovat full-textové vyhledávání, jiný rozeslat upozornění na email. Všechna tato funkcionalita je oddělena od samotného vložení článku, jde ji tedy vyvíjet odděleně a není potřeba nijak zasahovat do kódu, který se stará o vložení článku. Bude-li někdo chtít třeba po zveřejnění článku automaticky vložit link na Twitter, je to opět jen otázka vložení nového příjemce a není potřeba zasahovat do existujícího kódu. V tomto směru zasílání zpráv nabízí velkou volnost v používání a konfiguraci aplikací.

Oba režimy ilustruje Obrázek 1.



Obrázek 1: Režimy práce JMS. (i) point-to-point, nahoře; (ii) publisher-subscriber, dole.

1.1 Message-driven Beans

Pro zasílání a příjem zpráv nabízí Java EE velmi jednoduchý mechanismus postavený Enterprise JavaBeans. My si jeho použití ukážeme společně s AS Payara/Glassfish v základním nastavení, i když v praxi se zejména s ohledem na výkon a dostupnost používají složitější konfigurace.

1.1.1 Vytvoření tématu/fronty

Nejdříve na straně serveru vytvoříme téma a frontu, se kterými budou naše ukázkové příklady pracovat. V *konfiguraci AS* → *Resources* → *Destination Resources* vytvoříme frontu[2] pojmenovanou[3] `jms/Lecture08Queue` a téma[4] `jms/newUserTopic`. U fronty i tématu je potřeba uvést pojmenování v rámci message brokeru, kde stačí zadat vhodné jméno. Pokud fronta nebo téma daného jména neexistuje, message broker se postará o její vytvoření.

[2] `javax.jms.Queue`
[3] JNDI name
[4] `javax.jms.Topic`

1.1.2 Message-driven beans

Pro příjem zpráv bez ohledu na to, jestli se vztahují k frontě nebo tématu, slouží Message-driven Enterprise JavaBeans, což jsou třídy, které implementují rozhraní `MessageListener` a mají anotaci `@MessageDriven`, kde je parametrem `mappedName` určeno z jakého tématu či fronty daná komponenta odebírá zprávy.

Zpracování zpráv se děje v metodě `void onMessage(Message message)`, kde pomocí metody `<T> T Message.getBody(Class<T> clazz)` můžeme získat obsah zprávy, která je typu `clazz`.

1.2 Zasílání zpráv

Pro zasílání zpráv nejdříve musíme získat objekt typu `JMSContext`, který zajišťuje spojení a jednotlivá sezení při komunikaci s massage brokerem.

```
@Inject @JMSConnectionFactory("java:comp/DefaultJMSConnectionFactory")
private JMSContext context;
```

A následně pomocí anotací `@Resource` získáme od AS objekt fronty, resp. tématu.

```
@Resource(mappedName = "jms/Lecture08Queue")
private Queue textQueue;

@Resource(mappedName = "jms/newUserTopic")
private Topic newUserTopic;
```

Nyní již můžeme posílat zprávy, jak ukazuje následující příklad.

```
context.createProducer().send(textQueue, firstName + " " + secondName);
context.createProducer().send(newUserTopic, new NewUserMessage(firstName, secondName));
```

První řádek odesílá textový řetezec do fronty `textQueue`, druhý odesílá objekt představující nového uživatele všem příjemcům tématu `jms/newUserTopic`.

2 Plánování akcí

Běžně potřebujeme provádět opakující se úkoly, které nejsou závislé na akci uživatele, ale na konkrétním čase, např. jednou za týden vypočítat souhrny z dat, provést údržbu, rozeslat emaily. K tomu se často používají externí nástroje jako `cron`, ale platforma Java EE pokrývá i tuto funkcionalitu. Stačí nám k tomu použít EJB a přidat k metodám, které se mají vyvolat v učitém čase anotaci `@Schedule`, jak ukazuje následující příklad.

```
/** EJB vyvolavajici udalosti v danem casovem okamziku */
@Singleton
@Startup
```

```
public class HooyerBean {

    @Schedule(minute="*/10", hour="*")
    public void reminder() {
        System.out.println("It's time!");
    }
}
```

Máme v něm EJB typu singleton, navíc vybavenou anotací `@Startup`, která zajistí její vytvoření při spuštění aplikace[5] a v ní metodu `reminder`, která bude vyvolána každých deset minut a vypíše na standardní výstup text `It's time!`. Samozřejmě můžeme použít i jiný čas, např. můžeme metodu volat vždy ve dvě hodiny odpoledne a to následovně.

```
@Schedule(minute="0", hour="14", dayOfWeek = "*")
```

3 Posílání emailů

Mezi další běžně používané funkce, které je potřeba (nejen) v informačních systémech implementovat, je rozesílání emailů. Java EE pro to nabízí velmi pohodlné prostředky.

Nejdříve na straně AS nakonfigurujeme připojení k poštovním serverům (viz *konfigurace AS* → *Resources* → *JavaMail Sessions*). Předpokládejme, že jsme si nastavili JavaMail session pojmenovanou `mail/lecture08`.

Přístup k této JavaMail session necháme do JavaBean vložit AS následovně.

```
@Resource(lookup = "mail/lecture08")
private Session session;
```

A nyní již můžeme pracovat s poštou přes rozhraní JavaMail. Můžeme vytvořit zprávu a nechat ji poslat, například následovně.

```
MimeMessage message = new MimeMessage(session);
message.setRecipient(javax.mail.Message.RecipientType.TO,
                     new InternetAddress("john-doe@example.com"));
message.setSubject("New message");
message.setSender(new InternetAddress("noreply@example.com"));
message.setText("Lorem ipsum");
Transport.send(message);
```

Při vývoji a ladění je dobré zapnout si debugování.

```
session.setDebugOut(System.out);
session.setDebug(true);
```

[5] Jinak může EJB kontejner odložit vytvoření na pozdější (podle něj vhodnější) dobu.