Modularyzacja EJB

Karol Gostek
Tetiana Zhvanko
Maksym Blazhyievskyi

Typy sesyjnych EJB

- Singleton
- Stateless
- Stateful

Definiowanie EJB

- **No-interface**

```
@Stateless public class Invoice {
 ... implementation
}
```

- **Local**

```
@Local public interface InvoiceInterface {
      ... abstract interface methods
}
@Stateless public class Invoice
     implements InvoiceInterface {
   ... implementation
}
```

- **@Remote**

```
public interface InvoiceInterface {
      ... abstract interface methods
}
@Stateless
@Remote(InvoiceInterface.class)
public Invoice
        implements InvoiceInterface {
      ... implementation
}
```

Dostęp do EJB

Dostęp do lokalnych EJB z klienta

```
public class SomeCdiManagedClass {

        @EJB
        private SomeEjbInterface theEjb;

        // or, for no-interface EJBs
        @EJB
        private SomeEjbClass theEjb;
    }
```

Dostęp do EJB ze zdalnym interfejsem

```
package book.jakarta8.testEjbServer;
public interface SomeEjbRemote {  String tellMe();      }
```

```
package book.jakarta8.testEjbServer;
import javax.ejb.Remote;
import javax.ejb.Stateless;

@Stateless()
@Remote(SomeEjbRemote.class)
public class SomeEjb implements SomeEjbRemote {
@Override
public String tellMe() {    return "Hello World";  }          }
```

Projekty EJB

Zmiana deklaracji opakowania

```
<groupId>book.jakarta8</groupId>
<artifactId>testEjbServer</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>ejb</packaging>
```

Zmiana aspektów i przeniesienie interfejsów

org.eclipse.wst.common.project.facet.core.xml

Book.jakarta8.ejbproj.ejb <- Implementation

book.jakarta8.ejbproj.ejb.interfaces <- Interfaces

Zmiana deklaracji opakowania

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-ejb-plugin</artifactId>
<version>3.0.1</version>
<configuration>
        <generateClient>true</generateClient>
        <ejbVersion>3.2</ejbVersion>
                <clientExcludes><clientExclude>
                        book/jakarta8/ejbproj/ejb/*
                </clientExclude></clientExcludes>
</configuration>
</plugin>
```

EJB z dependencjami

Sposoby dodawania dependencji do serwera

Globalnie:

- Nie da się dodawać i usuwać dependencji bez wyłączenia serwera
- Takie rozwiązania są globalne
- Takie rozwiązania nie są przenośne

Za pomocą EAR:

- Utrudnia administrację
- Standardowe rozwiązanie

Asynchroniczne wywołanie EJB

Przykład asynchronicznej metody EJB

```
import java.util.concurrent.Future;
import javax.ejb.AsyncResult;
import javax.ejb.Asynchronous;
import javax.ejb.Singleton;

@Singleton
public class SomeEjb {
	@Asynchronous
	public Future<String> tellMeLater() {
		// Simulate some long-running calculation
		try {
			Thread.sleep(2000);
		} catch (InterruptedException e) {

		}

		return new AsyncResult<String>("Hi from tellMeLater()");
	}
}
```

Przykład wywołania asynchronicznej metody EJB

```
...
@EJB
private SomeEjb someEjb;
...
Future<String> f = someEjb.tellMeLater();
try {
	String s = f.get();
	System.err.println(s);
} catch (Exception e) {
	e.printStackTrace(System.err);
}
```

EJB ze stoperami

Przykłady wykorzystania timerów

```
@Schedule(second="10", minute="0", hour="0")                    // O 00:00:10 każdego dnia
@Schedule(minute="30", hour="0", dayOfWeek="Tue")               // O 00:30:00 we wtorki
@Schedule(minute="11", hour="15", dayOfWeek="Mon,Tue,Fri")      // O 15:11:00 w poniedziałki, wtorki i piątki
@Schedule(minute="*/10", hour="*")                              // O 10 minucie, każdą godzinę
@Schedule(minute="25/10", hour="1")                             // O 01:25, 01:35, 01:45 oraz 01:55
@Schedule(hour="*", dayOfMonth="1,2,3")                         // Każdą godzinę 1-go, 2-go, i 3-go każdego miesiąca
@Schedule(hour="*/10")                                          // Każde 10 godzin
@Schedule(month="Feb,Aug")                                      // O 00:00 każdy luty i sierpień
@Schedule(dayOfMonth="1", year="2020")                          // O 00:00 1-go każdego miesiąca w 2020 roku
@Schedule(dayOfMonth="1-10")                                    // O 00:00 od 1-go do 10-go każdego miesiąca
```

Przykład połączenia kilku timerów

```
@Schedules({
        @Schedule(hour="*"),
        @Schedule(hour="0", minute="30")
})
private void someMethod(Timer tm) {
        ...
}
```

Programowe wykorzystanie timerów

```
@Singleton
@Startup
public class Timer1 {
	@Resource
	private SessionContext context;

	@PostConstruct
	public void go() {
		context.getTimerService().createSingleActionTimer(5000, new TimerConfig());
	}

	@Timeout
	public void timeout(Timer timer) {
		System.err.println("Hello from " + getClass());
	}
}
```

Dziękujemy za uwagę