



Lottery #maven

JMS, Message Driven Bean, JMX, Singleton Session Bean

Óbudai Egyetem, Java Enterprise Edition

Műszaki Informatika szak

Labor 6

Bedők Dávid
2018-01-17
v1.0



Feladat: hozzunk létre egy Enterprise Java alkalmazást, mely az ötös lottóhúzás (számok 1-től 90-ig, azonos valószínűséggel) heti eredményeit tárolja (időbélyeggel, aktuális nyereményalappal és a számot kihúzó személy nevével).

- ▶ A kihúzott számokat egy **JMS queue**-n keresztül fogja az alkalmazás megkapni (az öt számot egyben, vesszővel elválasztva szöveges formátumban) (machine-to-machine interface).
- ▶ RESTful interface-en keresztül legyen lehetőség az aktuális és az összes korábbi lottóhúzás adatainak lekérdezésére.
- ▶ A szolgáltatás tartsa karban, hogy az aktuális szabályok szerint milyen a nyereményalap eloszlása (pl. a három találatos szelvények a nyereményalap mekkora részére jogosultak).
- ▶ RESTful interface-en keresztül lehessen öt kihúzott számról megkérdezni, hogy az aktuális sorsolás figyelembe vételével a nyereményalap mekkora részéből remélhet nyereményt (és hogy egyáltalán nyertes szelvényről van-e szó).
- ▶ A szolgáltatást **szabványos management (JMX)** felületen keresztül lehessen finomhangolni (nyereményalap, számhúzó személy, nyeremények eloszlása).



- ▷ JBoss Management Console
 - admin/guest felhasználó létrehozása
- ▷ **J**ava **M**essage **S**ervice (**JMS**)
 - JMS Desination
 - JMS Queue
 - JMS Topic (feladat nem érinti)
 - **M**essage **D**riven **B**ean (**MDB**) (listener)
 - JMS Client
- ▷ **J**ava **M**anagement **eX**tension (**JMX**)
 - Standard **M**anagement **B**ean (MBean)
 - jconsole
- ▷ Singleton Session Bean
 - speciális állapottal rendelkező Session Bean



▷ **lottery** (root project)

- **lot-webservice** (EAR web module)
 - A RESTful webservice-ek project-je (presentation-tier).
- **lot-ejbservice** (EAR ejb module)
 - Üzleti metódusok (service-tier)
- **lot-persistence** (EAR ejb module)
 - ORM réteg, JPA (data-tier)
- **lot-jmsclient** (standalone)
 - JMS kliens alkalmazás
- **lot-jmxclient** (standalone)
 - JMX kliens alkalmazás

Maven esetén egy **lot-ear** project is megjelenik.

A **JMS client** alkalmazás *classpath*-ához jelenleg nem lesz szükség pl. egy **lot-ejbserviceclient** projectre, mivel a kommunikáció szabványos (nem kell pl. remote interface), és szöveges üzeneteket fogunk küldeni (nem kel-
lenek stub-ok). Utóbbi megléte és Object JMS message esetén megjelenne természetesen a **lot-ejbserviceclient** is, mely része lenne mind az EAR mind a JMS client *classpath*ának.

- ▷ <http://localhost:9990/>
- ▷ Használatához egy **admin** jogosultságú management user szükséges (BASIC AUTH)
 - [JBOSS-HOME]/bin/add-user. [bat|sh]
 - Management User (enter)
 - Username: admin (are you sure? yes)
 - Password: AlmafA11#
 - Enter, Yes, Yes, Enter

```
1 > [JBOSS_HOME]/bin/add-user.[bat|sh] admin AlmafA11#
```

- [JBOSS-HOME]/standalone/configuration/mgmt-users.properties
- ▷ **guest** felhasználó létrehozása (később szükség lesz egy sikeresen autentikálható, de admin jogokkal nem rendelkező felhasználóra):

```
1 > [JBOSS_HOME]/bin/add-user.[bat|sh] -a -u jfstestuser  
-p User#70365 -g guest
```

- [JBOSS-HOME]/standalone/configuration/application-users.properties
- [JBOSS-HOME]/standalone/configuration/application-roles.properties



```
1 CREATE TABLE event (  
2   event_id SERIAL NOT NULL,  
3   event_puller CHARACTER VARYING(100) NOT NULL,  
4   event_prizepool INTEGER NOT NULL,  
5   event_date TIMESTAMP WITHOUT TIME ZONE NOT NULL,  
6   CONSTRAINT PK_EVENT_ID PRIMARY KEY (event_id)  
7 );  
8  
9 ALTER TABLE event OWNER TO postgres;  
10  
11 CREATE TABLE drawnumber (  
12   drawnumber_id SERIAL NOT NULL,  
13   drawnumber_event_id INTEGER NOT NULL,  
14   drawnumber_value INTEGER NOT NULL,  
15   CONSTRAINT PK_DRAWNNUMBER_ID PRIMARY KEY (drawnumber_id),  
16   CONSTRAINT FK_DRAWNNUMBER_EVENT FOREIGN KEY (drawnumber_event_id)  
17     REFERENCES event (event_id) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT  
18 );  
19  
20 ALTER TABLE drawnumber OWNER TO postgres;
```

create-schema.sql



- ▷ <http://localhost:8080/lottery/api/service/event/latest>
 - **LotteryRestService** (lot-webservice)
 - EventStub getLatestEvent() throws AdaptorException;
 - **LotteryFacade** (lot-ejbservice)
 - EventStub getLatestEvent() throws AdaptorException;
 - **EventService** (lot-persistence)
 - Event readLatest() throws PersistenceServiceException;
 - ```
1 SELECT e
2 FROM Event e
3 JOIN FETCH e.numbers
4 ORDER BY e.date DESC
```
    - SetMaxResult(1)
    - **Fontos!** A kihúzott számok bekötése miatt a lekérdezés az összes eseményt lekéri az adatbázisból.
  - **EventConverter** (lot-ejbservice): EventStub to(Event event);
- ▷ <http://localhost:8080/lottery/api/service/event/list>
  - List<EventStub>...
  - List<Event>...

# JOIN FETCH

## ismétlés

```
1 SELECT
2 event0_.event_id AS event_id1_1_0_,
3 numbers1_.drawnnumber_id AS drawnnum1_0_1_,
4 event0_.event_date AS event_da2_1_0_,
5 event0_.event_prizepool AS event_pr3_1_0_,
6 event0_.event_puller AS event_pu4_1_0_,
7 numbers1_.drawnnumber_event_id AS drawnnum3_0_1_,
8 numbers1_.drawnnumber_value AS drawnnum2_0_1_,
9 numbers1_.drawnnumber_event_id AS drawnnum3_1_0_-,
10 numbers1_.drawnnumber_id AS drawnnum1_0_0_-
11 FROM
12 event event0_
13 INNER JOIN drawnnumber numbers1_ ON
14 event0_.event_id=numbers1_.drawnnumber_event_id
15 ORDER BY
16 event0_.event_date DESC
```

A `Set<DrawnNumber>` **LAZY** módon van bekötve az `Event` entitáshoz. A `JOIN` beköti a lekérdezésbe a táblát (ez esetben a lekérdezés azonos lesz a bemutatottal, kivéve hogy a `SELECT` részben nem lesznek felsorolva a `drawnumber` tábla oszlopai), de nem *attach*-olja az entitásokat (ha bejárjuk, akkor külön lekérdezésben lekéri és csatolja). A `JOIN FETCH` e plusz lekérdezés nélkül *attach*-olja is!



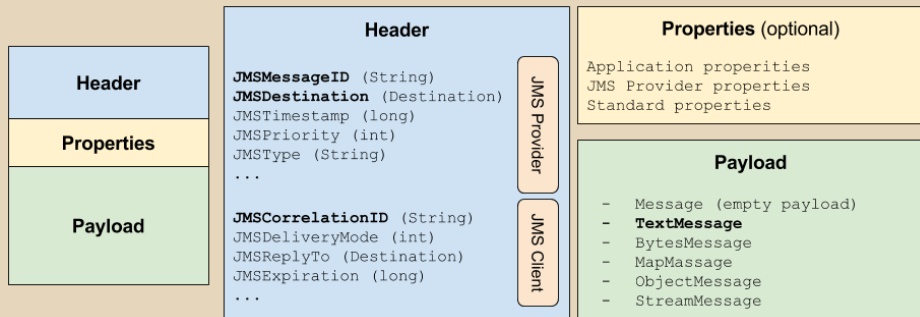
# Java Message Service (JMS)

## Message-Oriented Middleware (MOM)

- ▷ Üzenetküldés alapú kommunikáció
- ▷ "lazán" kötődő komponensek (beékelődik a kommunikációba az üzeneteket kezelő/tároló komponens)
- ▷ **JMS 1.1** (2002, JSR914, JEE6)
- ▷ **JMS 2.0** (2013, JSR343, JEE7)
- ▷ Típusai:
  - **point-to-point (queue)**
    - *producer* üzeneteket küld a queue-ba
    - *consumer* üzenetet kiolvas a queue-ból
    - egy üzenetet egy fogadó dolgoz fel (*ack* küldés is van)
    - *producer* és *consumer* nem kell hogy egy időben "online" legyen
  - **publish-subscribe (topic)**
    - *publisher* üzeneteket küld a topic-ba
    - *subscriber*-ek megkapják a topic-ba küldött üzenetet
    - egy üzenet több fogadó is feldolgoz(hat)
    - *publisher* és *subscriber* között van időbeli függés, "tankönyvi" eset szerint a komponensek egyszerre "online"-ok (de vannak speciális feliratkozási esetek)

- ▷ <http://hornetq.jboss.org/>
- ▷ Deprecated, JBoss 7.x-től **JBoss A-MQ** váltja fel
  - <http://www.jboss.org/products/amq/overview/>
- ▷ **JMS 1.1** és **JMS 2.0** támogatás
- ▷ Verzió: **2.3.25.Final** (JBoss 6.4 esetén)

# JMS üzenet felépítése



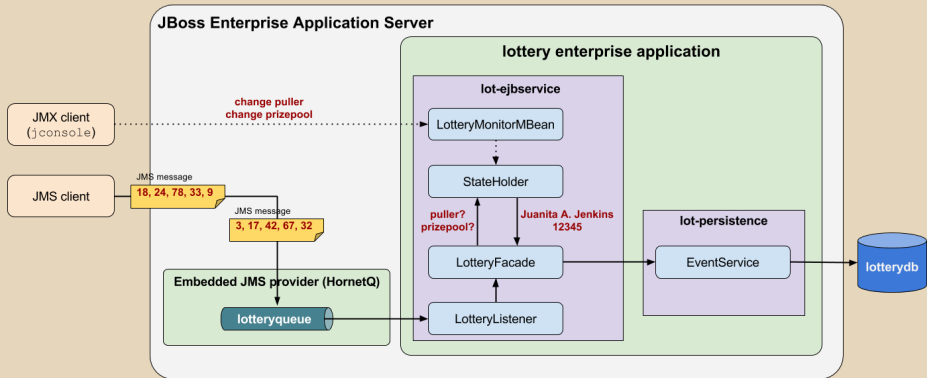
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <messaging-deployment xmlns="urn:jboss:messaging-deployment:1.0">
3 <hornetq-server>
4 <jms-destinations>
5 <jms-queue name="lotteryqueue">
6 <entry name="jms/queue/lotteryqueue" />
7 <entry name="java:jboss/exported/jms/queue/lotteryqueue"
8 />
9 </jms-queue>
10 </jms-destinations>
11 </hornetq-server>
12 </messaging-deployment>
```

java:/jms/queue/lotteryqueue lesz a local JNDI név. A remote JMS client a java:jboss/exported/ előtagot automatikusan fogja használni (JBoss JMS Client jar használata esetén).

lotteryqueue-jms.xml

A file nevének \*-jms.xml-nek kell lennie, és a *deployments* könyvtárba másolással "létrehozható" a destination. Másik megoldás lehet a standalone.xml-en keresztüli definiálás, illetve programozottan (*runtime*) is létrehozható queue/topic.

# Új esemény létrehozása



```
1 package hu.qwaevisz.lottery.ejbservice.listener;
2 [..]
3 @MessageDriven(name = "LotteryListener", activationConfig = { //
4 @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
5 "javax.jms.Queue"),
6 @ActivationConfigProperty(propertyName = "destination", propertyValue =
7 "lotteryqueue"),
8 @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue =
9 "Auto-acknowledge") })
10 public class LotteryListener implements MessageListener {
11
12 @EJB
13 private LotteryFacade facade;
14
15 @EJB
16 private MessageConverter converter;
17
18 @PostConstruct
19 public void initialize() {
20 LOGGER.info("Lottery Listener created...");
21 }
22
23 @Override
24 public void onMessage(final Message message) {
25 [..]
26 }
27 }
```

Amikor a lotteryqueue-ba üzenet érkezik, a LotteryListener MDB aktiválódik, és az onMessage() metódusa meghívásra kerül a feldolgozandó üzenettel. Ha kivételt dob a metódus, az üzenet feldolgozás rollback-elődik, és nem kerül ki a sorból!

# LotteryListener

- ▷ **queue name**: kinyerhető (hasznos, ha egy *listener* több queue-ra figyel egyidejűleg (erre van lehetőség)
- ▷ **correlation id**: tipikusan kliens hozza létre, és elküldi a JMS message-el, hogy később pl. egy async válasz során azonosítani tudja a "választ".

```
1 @Override
2 public void onMessage(final Message message) {
3 try {
4 if (LOGGER.isDebugEnabled()) {
5 final Queue destination = (Queue) message.getJMSDestination();
6 final String queueName = destination.getQueueName();
7 LOGGER.debug("New JMS message arrived into " + queueName + " queue (correlation
8 id: " + message.getJMSCorrelationID() + ")");
9 }
10 if (message instanceof TextMessage) {
11 final TextMessage textMessage = (TextMessage) message;
12 String content = textMessage.getText();
13 if (LOGGER.isDebugEnabled()) {
14 LOGGER.debug("Received message: " + content);
15 }
16 this.facade.createNewEvent(this.converter.toNumbers(content));
17 } else {
18 LOGGER.error("Received message is not a TextMessage (" + message + ")");
19 }
20 } catch (final JMSEXception | AdaptorException | NumberFormatException e) {
21 LOGGER.error(e, e);
22 }
```

- ▷ BytesMessage
- ▷ MapMessage
- ▷ ObjectMessage
- ▷ StreamMessage
- ▷ TextMessage

# RedHat JBoss vs. Oracle WebLogic

|                       | <b>JBoss 6.4</b>         | <b>WebLogic 12.2.1</b> |
|-----------------------|--------------------------|------------------------|
| <b>JavaEE version</b> | JavaEE 6                 | JavaEE 6               |
| <b>Logging</b>        | JBoss Logging, Log4J, .. | JDK Logging, ..        |
| <b>RESTful</b>        | RESTEasy 2.3.10.Final    | Jersey 1.18            |
| <b>JPA Provider</b>   | Hibernate 4.2.18.Final   | EclipseLink 2.5.2      |
| <b>JMS Provider</b>   | HornetQ 2.3.25.Final     | WebLogic JMS           |
| <b>JAXB</b>           | JAXB 2.2.5-redhat-9      | JAXB 2.2.10 Oracle     |



### WebLogic Administration Console

▷ Services | Messaging | JMS Servers

- New

- name: **demoJMSserver**
- persistence store: none
- target: myserver

▷ Services | Messaging | JMS Modules

- New

- name: **demoJMSmodule**
- location in domain: blank
- target: myserver
- Would you like to add resources to this JMS system module? → yes

### WebLogic Administration Console

- ▷ Services | Messaging | JMS Modules
  - demoJMSmodule | Subdeployments | New
    - name: **demoJMSsubmodule**
    - target/server: demoJMSserver
    - target: myserver
    - Would you like to add resources to this JMS system module? → yes
- ▷ Services | Messaging | JMS Modules
  - demoJMSmodule | Configuration | New
    - tpye: **Queue**
    - name: **lotteryqueue**
    - JNDI name: **jms/queue/lotteryqueue**
    - subdeployments: demoJMSsubmodule
    - target/server: demoJMSserver

### WebLogic Administration Console

- ▷ Services | Messaging | JMS Modules
  - demoJMSmodule | Configuration | New
    - type: **Connection Factory**
    - name: **demoConnectionFactory**
    - JNDI name: **jms/demoConnectionFactory**
    - target: myserver

```
1 package hu.qwaevisz.lottery.ejbservice.listener;
2 [...]
3 @MessageDriven(name = "LotteryListener", activationConfig = { //
4 @ActivationConfigProperty(propertyName = "initialContextFactory", propertyValue =
5 "weblogic.jndi.WLInitialContextFactory"),
6 @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
7 "javax.jms.Queue"),
8 @ActivationConfigProperty(propertyName = "destinationJndiName", propertyValue =
9 "jms/queue/lotteryqueue"),
10 @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue =
11 "Auto-acknowledge") })
12 public class LotteryListener implements MessageListener, MessageDrivenBean {
13 [...]
14 }
```

### LotteryListener.java

## Különbségek

- ▷ A MessageDrivenBean implementálása nem kötelező, de lehetőséget ad az MDB életciklusába való beavatkozásra (setMessageDrivenContext(..) és ejbRemove() metódusok által).
- ▷ initialContextFactory tulajdonság (megadása külső JMS provider esetén fontos)
- ▷ destinationJndiName tulajdonság használata a JBoss által elvárt destination helyett (így az érték is a JBoss *friendly name* helyett a JNDI név lesz)

# JMS Client Application

lot-jmsclient project

Távolról JMS üzenetet küld a `lotteryqueue`-ba, melyet az elindított JBoss EAS által indított **HornetQ** mint JMS MOM fog fogadni.

## Fontos!

Kommunikálni a *JMS Provider*-rel szükséges, de mivel a HornetQ egy **embedded JMS Provider**, ezért csatlakoznunk először a JBoss-hoz kell a Remote EJB-nél már megismert `InitialContext`-en keresztül, hogy elérjük a JBoss **JNDI** fáját. Külső *JMS Provider* esetén mindezek nem így érvényesek!

Ahhoz, hogy meg tudjuk szólítani ezt a szolgáltatást, az alábbiak szükségesek:

▷ **JBoss initial context factory**

- szükséges erőforrás a classpath-on:  
`org.jboss.naming.remote.client.InitialContextFactory`
- compile group:  
'org.jboss.as', name: 'jboss-as-jms-client-bom', version: '7.2.0.Final'

▷ JBoss EAS host-ja (**localhost**) és remote portja (def.: **4447**)

- `standalone.xml` | `socket-binding-group` | `remoting socket-binding` port: 4447

▷ **JMS Connection Factory JNDI neve** (`standalone.xml`: `jms/RemoteConnectionFactory`)

▷ Egy min. guest role-lal rendelkező user sikeres autentikációja (**username** és **password**)

▷ A cél **queue JNDI neve** (`lotteryqueue-jms.xml`: `jms/queue/lotteryqueue`)

▷ Ha `TextMessage` helyett pl. `ObjectMessage`-et küldünk, akkor szükség volna egy "serviceclient.js" ra is, mely tartalmazná a `Serializable` DTO-kat (hasonlóan az `ejb` client-nél alkalmazottak szerint).

```
1 <subsystem xmlns="urn:jboss:domain:messaging:1.4">
2 <hornetq-server>
3 [..]
4 <connectors>[..]</connectors>
5 <acceptors>[..]</acceptors>
6 <security-settings>
7 <security-setting match="#">
8 <permission type="send" roles="guest"/>
9 <permission type="consume" roles="guest"/>
10 [..]
11 </security-setting>
12 </security-settings>
13 <address-settings>[..]</address-settings>
14 <jms-connection-factories>
15 <connection-factory name="InVmConnectionFactory">[..]</connection-factory>
16 <connection-factory name="RemoteConnectionFactory">
17 <connectors>[..]</connectors>
18 <entries>
19 <entry name="java:jboss/exported/jms/RemoteConnectionFactory"/>
20 </entries>
21 </connection-factory>
22 <pooled-connection-factory
23 name="hornetq-ra">[..]</pooled-
24 </jms-connection-factories>
25 <jms-destinations>[..]</jms-destinations
26 </hornetq-server>
</subsystem>
```

Az alapértelmezett biztonsági beállítások végett küldeni és fogadni csak a **guest** szerepkörű autentikált felhasználók tudnak. Az autentikációt a JBoss végzi.

A kapcsolatok *remote* kiépítéséhez a `RemoteConnectionFactory` elérése szükséges, amit JNDI-ből az itt konfigurált `jms/RemoteConnectionFactory` néven tudunk majd elérni.

```
1 final Properties environment = new Properties();
2 environment.put(Context.INITIAL_CONTEXT_FACTORY,
3 "org.jboss.naming.remote.client.InitialContextFactory");
4 environment.put(Context.PROVIDER_URL, "remote://localhost:4447");
5 environment.put(Context.SECURITY_PRINCIPAL, "jmsstestuser");
6 environment.put(Context.SECURITY_CREDENTIALS, "User#70365");
7 final Context context = new InitialContext(environment);
8 final ConnectionFactory connectionFactory = (ConnectionFactory)
9 context.lookup("jms/RemoteConnectionFactory");
10 final Destination destination = (Destination) context.lookup("jms/queue/lotteryqueue");
11 final Connection connection = connectionFactory.createConnection("jmsstestuser",
12 "User#70365");
13 final Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
14 final MessageProducer producer = session.createProducer(destination);
15 connection.start();
16 final TextMessage textMessage = session.createTextMessage("1, 2, 3, 4, 5");
17 producer.send(textMessage);
18 connection.close();
```

## SimpleClient.java

```
1 final Properties environment = new Properties();
2 environment.put(Context.INITIAL_CONTEXT_FACTORY,
3 "weblogic.jndi.WLInitialContextFactory");
4 environment.put(Context.PROVIDER_URL, "t3://localhost:7001");
5 environment.put(Context.SECURITY_PRINCIPAL, "weblogic");
6 environment.put(Context.SECURITY_CREDENTIALS, "AlmafA1#");
7 final Context context = new InitialContext(environment);
8 final ConnectionFactory connectionFactory = (ConnectionFactory)
9 context.lookup("jms/demoConnectionFactory");
10 final Destination destination = (Destination) context.lookup("jms/queue/lotteryqueue");
11 final Connection connection = connectionFactory.createConnection("weblogic",
12 "AlmafA1#");
13 final Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
14 final MessageProducer producer = session.createProducer(destination);
15 connection.start();
16 final TextMessage textMessage = session.createTextMessage("1, 2, 3, 4, 5");
17 producer.send(textMessage);
18 connection.close();
```

## SimpleClient.java



# Remote JMS Client

RedHat JBoss vs. Oracle WebLogic

|                         | JBoss 6.4                                             | WebLogic 12.2.1                                  |
|-------------------------|-------------------------------------------------------|--------------------------------------------------|
| Provider URL            | remote://localhost:4447 6                             | t3://localhost:7001                              |
| User name               | jmstestuser                                           | weblogic                                         |
| Password                | User#70365                                            | AlmafA1#                                         |
| Destination JNDI        | jms/queue/lotteryqueue                                | jms/queue/lotteryqueue                           |
| Initial context factory | org.jboss.naming.remote.client. InitialContextFactory | weblogic.jndi. WLInitialContextFactory           |
| Connection factory JNDI | jms/RemoteConnectionFactory                           | jms/demoConnectionFactory                        |
| Classpath               | org.jboss.as:jboss-as-jms-client-bom:7.2.0.Final      | [WL-HOME]/wlserver/server/lib/wlthint3client.jar |

## Authentikáció

JBoss esetén külön *guest user*-t hoztunk létre, míg WebLogic-nál az *Admin user*-t használhatjuk. A Connection Factory-t JBoss esetén a `standalone-full.xml`-ből előre konfigurálva kaptuk, míg WebLogic esetén mi hoztuk létre *Admin console*-on.

<http://localhost:8080/lottery/QueueServlet?message=1,2,3,4,5>

```
1 [..]
2 InitialContext context = new InitialContext();
3 QueueConnectionFactory connectionFactory = (QueueConnectionFactory)
 context.lookup("ConnectionFactory");
4 QueueConnection connection = connectionFactory.createQueueConnection();
5 QueueSession session = connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
6 Queue queue = (Queue) context.lookup("jms/queue/lotteryqueue");
7 QueueSender sender = session.createSender(queue);
8 TextMessage textMessage = session.createTextMessage();
9 connection.start();
10
11 textMessage.setText("1,2,3,4,5");
12 sender.send(textMessage);
13 sender.close();
14 session.close();
15 connection.close();
16 [..]
```

**WebLogic** esetén egyetlen külön-  
ség a *ConnectionFactory* JNDI neve:  
javax.jms.QueueConnectionFactory.  
**JBoss** esetén az itt megadott JNDI név  
(ConnectionFactory) a standalone-full.xml-ből  
kiolvasható (InVmConnectionFactory).

SendQueueServlet.java

Üzenetet küldeni container-en belül mindig egyszerűbb, lévén InitialContext-et "konfiguráció nélkül" egyszerűen elkérhetjük (hiszen már inicializálva van), illetve nem csak "remote" *connection factory*-kat érünk el.

## Singleton jelleg

Az EJB container garantálja, hogy a Singleton Session Bean-ből ugyanazt a példányt fogja minden szálon használni.

Természetesen nem arról van szó, hogy minden a SSB-t használó klienst szépen sorbaállít a container (ez bottleneck-je lenne az egész rendszernek). Vanak **READ** és **WRITE** lock-kal rendelkező metódusai (kizólag **Container-Managed Concurrency** (CMC) esetén használható):

- ▷ `LockType.READ`: párhuzamosan több szálon is futhat (állapot olvasás)
- ▷ `LockType.WRITE` (def.): kizárólag egy szálon futhat (állapot módosítás)

# Concurrency Management

## Session Beans

### Container-Managed Concurrency (CMC) (def.)

`@ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)`

### Bean-Managed Concurrency (BMC)

`@ConcurrencyManagement(ConcurrencyManagementType.BEAN)`

### Bean managed

Kizárólag **Singleton Session Bean**-ek esetén van értelmezve a *Bean-Managed Concurrency*, és ez esetben engedélyezett pl. a `synchronized` és a `volatile` kulcsszavak használata.

# StateHolder

Számhúzó, nyereményalap és nyereményeloszlás tárolása és kezelése

```
1 package hu.qwaevisz.lottery.ejb.service.holder;
2 [..]
3 @Singleton(mappedName = "ejb/lotteryState", name = "lotteryState")
4 @ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)
5 public class LotteryStateHolderImpl implements LotteryStateHolder {
6 private String puller;
7 private Integer prizePool;
8 private PrizeDistribution distribution;
9
10 @PostConstruct
11 public void initialize() {
12 this.puller = "Juanita A. Jenkins";
13 this.prizePool = 12345;
14 this.distribution = new PrizeDistribution();
15 }
16
17 @Override
18 @Lock(LockType.READ)
19 public String getCurrentPuller() {
20 return this.puller;
21 }
22
23 @Override
24 @Lock(LockType.WRITE)
25 public void setCurrentPuller(String name) {
26 LOGGER.info("Change Puller: " + name);
27 this.puller = name;
28 }
29 [..]
30 }
```

A `prizePool` tárolása és `getter/setter` üzleti metódusa mindezzel teljesen azonosan elkészíthető. A SSB-nek természetesen "illendő" interface-t készíteni (`LotteryStateHolder`), mely jelen esetben a `@Local` annotációt megkapja.

# LotteryFacade

lot-ejbservice project

```
1 package hu.qvaevisz.lottery.ejbservice.facade;
2 [...]
3 @Stateless(mappedName = "ejb/lotteryFacade")
4 public class LotteryFacadeImpl implements LotteryFacade {
5
6 private static final Logger LOGGER = Logger.getLogger(LotteryFacadeImpl.class);
7
8 @EJB
9 private EventService eventService;
10
11 @EJB
12 private LotteryStateHolder stateHolder;
13
14 @Override
15 public void createNewEvent(int[] numbers) throws AdaptorException {
16 try {
17 String puller = this.stateHolder.getCurrentPuller();
18 Integer prizePool = this.stateHolder.getCurrentPrizePool();
19 this.eventService.create(puller, prizePool, numbers);
20 } catch (final PersistenceServiceException e) {
21 LOGGER.error(e, e);
22 throw new AdaptorException(e.getLocalizedMessage());
23 }
24 }
25 [...]
26 }
```

Az EventService a *persistence* rétegben egy tranzakción belül be kell hogy beszúrja az új eseményt (event), illetve a hozzá tartozó 5 új drawnumber sort.

LotteryFacadeImpl.java



# EventService

lot-persistence project

**Fontos!** Az Event entitás `Set<DrawnNumber> numbers` field-je `@OneToMany` annotációjában a `cascade` értéke `CascadeType.ALL` vagy `PERSIST` legyen!

```
1 package hu.qvaevisz.lottery.persistence.service;
2 [...]
3 @Stateless(mappedName = "ejb/eventService")
4 @TransactionManagement(TransactionManagementType.CONTAINER)
5 @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
6 public class EventServiceImpl implements EventService {
7
8 @PersistenceContext(unitName = "lot-persistence-unit")
9 private EntityManager entityManager;
10
11 @Override
12 public void create(String puller, Integer prizePool, int[] numbers) throws
13 PersistenceServiceException {
14 try {
15 final Event event = new Event(puller, prizePool);
16 for (final int number : numbers) {
17 event.addNumber(number);
18 }
19 this.entityManager.persist(event);
20 } catch (final Exception e) {
21 throw new PersistenceServiceException("Unknown error when persisting Events! " +
22 e.getLocalizedMessage(), e);
23 }
24 }
25
26 public void addNumber(Integer number) {
27 this.numbers.add(new DrawnNumber(number, this));
28 }
29 }
```

**persist:** egy új (vagy egy törlésre jelölt) entitás létrehozása, és egyben *managed* állapotba emelése // **merge:** egy *detached* (nem *managed*) entitás létrehozása (a metódus visszaadja a *managed* entitást, az átadott *detached* példányt nem bántja)

EventServiceImpl.java

# Java Management eXtension (JMX)

JavaSE

- ▷ A **JMX** technológia a JavaSE része, és természetesen a JavaEE is támogatja, szerver oldali komponensek monitorozására is használható.
- ▷ **Managed Bean**-ek létrehozása szükséges hozzá (**MBean**), melyeket az *MBean server* észlel és kezel.
- ▷ JMX klienst könnyedén írhatunk, de a szabvány csatorna lévén erre legtöbbször nincsen szükség (pl. **jconsole** egy Java SE-vel szállított kliens alkalmazás).
- ▷ Az MBean-eknek követniük kell a JMX specifikációban leírt szabályokat (JMX kliensek szabvány elérése ezáltal garantált).



# MBean készítés szabályai

- ▷ Ha az implementáció `Something` class, akkor az interface `SomethingMBean` kell hogy legyen.
- ▷ Az MBean-ben **műveleteket** és **attribútumokat** definiálhatunk.
- ▷ MBean attribútum (**attribute**) definiálása:
  - Read-only A típusú xyz attribútum esetén léteznie kell egy `A getXyz()` metódusnak.
  - Írható/olvasható A típusú xyz attribútum esetén léteznie kell egy `A getXyz()` és egy `void setXyz( A value )` metódusnak.
  - Nem lehet egy attribútumhoz tartozó getter/setter-t másra használni, nem lehet azonos névvel overload-olt gettert készíteni, nem lehet más az összetartozó getter/setter paraméterezése és visszatérési értékének típusa.
- ▷ MBean művelet (**operation**) definiálása:
  - Minden olyan metódus, mely nem szabványos accessor (getter) illetve mutator (setter), automatikusan műveletnek számít (pl. `B getItem( C value )`).
- ▷ Egyszerű esetben a használható/javasolt típusok a java **primitívek**, **tömbök**, és a **String**, de létezik komplexebb típus is (pl. `TabularData`).

```
1 package hu.qwaevisz.lottery.ejbsservice.management;
2 [...]
3 public class LotteryMonitor implements LotteryMonitorMBean {
4 @EJB
5 private LotteryStateHolder stateHolder;
6
7 @Override
8 public String getPuller() { return this.stateHolder.getCurrentPuller(); }
9
10 @Override
11 public void setPuller(String name) { this.stateHolder.
12
13 @Override
14 public Integer getPrizePool() { [...] }
15
16 @Override
17 public void setPrizePool(Integer value) { [...] }
18
19 @Override
20 public int getDistribution(int hit) { return this.stateHolder.getDistribution(hit); }
21
22 @Override
23 public int[] getDistributions() { [...] }
24 @Override
25 public void setDistribution(int hit, int value) { [...] }
26 public void start() throws Exception { LOGGER.info("Start Lottery MBean"); }
27 public void stop() throws Exception { LOGGER.info("Stop Lottery MBean"); }
28 }
```

A start() és a stop() metódusok a JMX MBean életciklusa során meghívódnak. Használatuk opcionális.

```
1 package hu.qwaevisz.lottery.ejbservice.management;
2 [...]
3 public class LotteryMonitor extends StandardMBean implements LotteryMonitorMBean {
4
5 public LotteryMonitor() {
6 super(LotteryMonitorMBean.class, false);
7 }
8
9 @Override
10 public ObjectName preRegister(MBeanServer server, ObjectName name) throws Exception {
11 return name;
12 }
13
14 @Override
15 public void postRegister(Boolean registrationDone) {}
16
17 @Override
18 public void preDeregister() throws Exception {}
19
20 @Override
21 public void postDeregister() {}
22
23 }
```

Az ős (StandardMBean) ctor-a az MBean interface osztályát várja!

A MBeanRegistration interface használata lehetővé teszi az életciklusba való beavatkozást.

LotteryMonitor.java

```
1 package hu.qvaevisz.lottery.ejbservice.management;
2 [...]
3 public class LotteryMonitor extends StandardMBean implements LotteryMonitorMBean {
4
5 private static final String LOTTERY_STATE HOLDER_JNDI =
6 "java:global.lottery-1.0.lot-ejbservice.lotteryState";
7
8 private LotteryStateHolder getStateHolder() {
9 LotteryStateHolder holder = null;
10 try {
11 InitialContext context = new InitialContext();
12 holder = LotteryStateHolder.class.cast(context.lookup(LOTTERY_STATE HOLDER_JNDI)
13);
14 } catch (NamingException e) {
15 LOGGER.log(Level.SEVERE, e.getMessage(), e);
16 }
17 return holder;
18 }
19
20 @Override
21 public String getPuller() {
22 LotteryStateHolder holder = this.getStateHolder()
23 return holder != null ? holder.getCurrentPuller() : "";
24 }
25 }
```

WebLogic esetén az MBean-ek nem lesznek részei az *EJB context*-nek (JBoss esetén igen), így az *@EJB* annotáció nem használható. Minden ilyen esetben a megoldás az EJB kikérése a JNDI tree-ből.

LotteryMonitor.java

### WebLogic Administration Console

▷ Environment | Server | Configuration

• General tab → **VIEW JNDI Tree** link

◦ *Overview tab* Binding Name tulajdonság értéke

Deploy-olt alkalmazások EJB-inek elérése (a JNDI fa felépítése container specifikus, bár a JavaEE szabvány megpróbálta már szabványosítani):

`java:global.lottery-1.0.lot-ejbservice.lotteryState`

▷ `java:global`

▷ `lottery-1.0` → deployolt EAR neve (a pont miatt a browser ketté szedi))

▷ `lot-ejbservice` → az EJB module neve az EAR-on belül

▷ `lotteryState` → az EJB name attribútuma (def: impl. osztály neve)

src | main | **resources** | META-INF | jboss-service.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <server xmlns="urn:jboss:service:7.0"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4 xsi:schemaLocation="urn:jboss:service:7.0 jboss-service_7_0.xsd">
5 <mbean code="hu.qwaevisz.lottery.ejbsservice.management.LotteryMonitor"
6 name="lottery.mbean:service=LotteryMonitorMBean"></mbean>
</server>
```

jboss-service.xml

**lottery.mbean** lesz a topológiában a helye (name attribútum), **LotteryMonitorMBean** pedig ezen belül az MBean neve (name attribútum). A **code** értékénél az osztályt kell megadni, mely megfelel mindenben a JMX MBean szabványoknak!

Ez egy *JBoss* *specifikus* állomány, neve kötelezően `jboss-service.xml` kell hogy legyen.

**Gradle** esetén amit a gradle ear plugin által kezelt modul (jelen esetben ez a *root project*) `src/main/application` könyvtárba teszünk, az része lesz az *archive*-nak (az EAR-nak).

src | main | **application** | META-INF | weblogic-application.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <weblogic-application xmlns="http://xmlns.oracle.com/weblogic/weblogic-application">
3 <listener>
4 <listener-class>hu.qvaevisz.lottery.ejb.service.management.ApplicationMBeanLifeCycleList
5 </listener>
6 </weblogic-application>
```

weblogic-application.xml

WebLogic esetén az MBean regisztrációt **programozottan** kell elvégezni, és ezen programozott konfigurációs osztályt az EAR container specifikus leírójában (ez esetben ez lesz a `weblogic-application.xml`) kell meghivatkozni.

```
1 package hu.qwaevisz.lottery.ejbservice.management; [..]
2 public class ApplicationMBeanLifecycleListener extends ApplicationLifecycleListener {
3 private static final String MBEAN_SERVER_JNDI = "java:comp/jmx/runtime";
4 private static final String OBJECT_PACKAGE =
5 "hu.qwaevisz.lottery.ejbservice.management";
6 @Override
7 public void postStart(ApplicationLifecycleEvent evt) throws ApplicationException {
8 try {
9 InitialContext context = new InitialContext();
10 MBeanServer mbeanServer = MBeanServer.class.cast(
11 context.lookup(MBEAN_SERVER_JNDI));
12 LotteryMonitor mbean = new LotteryMonitor();
13 ObjectName oname = this.buildObjectName();
14 mbeanServer.registerMBean(mbean, oname);
15 }
16 catch (Exception e) {
17 LOGGER.log(Level.SEVERE, e.getMessage(), e);
18 }
19 }
20 private ObjectName buildObjectName() throws MalformedObjectNameException {
21 return new ObjectName(OBJECT_PACKAGE +
22 ":type="+LotteryMonitor.class.getSimpleName()+",name="+LotteryMonitor.class.getName());
23 }
24 @Override
25 public void preStop(ApplicationLifecycleEvent evt) throws ApplicationException {
26 [..]
27 }
28 }
```

Az ApplicationLifecycleListener osztály miatt a [WL-HOME]/wlserver/server/lib/wls-api.jar-t el kell helyezni a classpath-on!



```
1 > [JRE_HOME]/bin/jconsole.[bat|sh]
```

DE: JBoss esetén a `jconsole` classpath-ához hozzá kell fűzni további osztályokat (pl. `jboss-cli-client.jar`), ezért a

```
1 > [JBOSS_HOME]/bin/jconsole.[bat|sh]
```

paranccsal indítsuk el (mely hivatkozik a `[JRE_HOME]`-ban lévőre.

A JBoss AS látszódní fog a *Local process*-ek között (de ugyanezen klienssel *Remote JVM*-hez is tudunk csatlakozni).

## Megjegyzés

MAC OS-en (vagy bármilyen más rendszeren) ha a JBoss nem találja meg a `jconsole.sh` futtatásakor a `[JRE_HOME]`-ot, lefuttatva a JBoss alatti `jconsole.sh`-t a `CLASSPATH` környezeti változót beállíthatjuk az aktuális terminálban, majd ugyanebben a terminálban elindítjuk a `[JRE_HOME]` alatti `jconsole`-t.

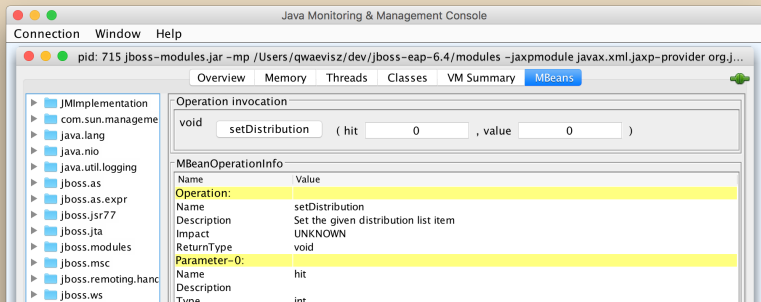
Hasonlóan a JBoss-hoz, a WebLogic esetén is szükséges a jconsole classpath-ához container specifikus JAR-ok hozzáadása.

```
1 > jconsole
 -J-Djava.class.path=%JAVA_HOME%\lib\jconsole.jar;%JAVA_HOME%\lib\
 -J-Djmx.remote.protocol.provider.pkgs=weblogic.management.remote
 -debug
```

A classpath beállításának egyszerű módja, ha lefuttatjuk a `setDomainEnv.[sh|cmd]` parancsot, majd ezt követően (azonos terminalban/command ablakban) a jconsole-t elindítjuk.

```
1 >
 [WL-HOME]/user_projects/domains/mydomain/bin/setDomainEnv.[sh|cmd]
2 > jconsole
```

# MBean leírások, argumentum nevek



The screenshot shows the Java Monitoring & Management Console interface. The title bar reads "Java Monitoring & Management Console". The menu bar includes "Connection", "Window", and "Help". The main window title is "pid: 715 jboss-modules.jar -mp /Users/qwaevisz/dev/jboss-eap-6.4/modules -jaxpmodule javax.xml.jaxp-provider org.j...". The "MBeans" tab is selected, showing a tree view on the left with categories like JMImplementation, com.sun.management, java.lang, java.nio, java.util.logging, jboss.as, jboss.as.expr, jboss.jsr77, jboss.jta, jboss.modules, jboss.msc, jboss.remoting.hand, and jboss.ws. The main area displays "Operation invocation" for the `void setDistribution ( hit 0 , value 0 )`. Below this, the "MBeanOperationInfo" table is shown:

| Name                | Value                                |
|---------------------|--------------------------------------|
| <b>Operation:</b>   |                                      |
| Name                | setDistribution                      |
| Description         | Set the given distribution list item |
| Impact              | UNKNOWN                              |
| ReturnType          | void                                 |
| <b>Parameter-0:</b> |                                      |
| Name                | hit                                  |
| Description         |                                      |
| Type                | int                                  |

Egy MBean attribútum/művelet leírása hasznos lehet, ahogy az sem volna előnytelen ha egy művelet argumentumainak forráskóban szereplő neve megjelenne a jconsole-ban, azonban sajnos ezen adatok átküldése kliens oldalra nem automatikus. Az MBean leszármazhat a `javax.management.StandardMBean` osztályból, melynek van számos metódusa, melyek arra szolgálnak hogy a kliens ezen adatok lekérdezésekor ezeket meghívja (egyesével). Ezen metódusok alapértelmezett viselkedését egyszerűen felülírhatjuk (*override*), ám ez még mindig egy nagyon kényelmetlen, hosszú `switch case` rengeteget jelentene, ráadásul a leírások *maintenance* feladata is rémálom volna.

# Annotált MBean készítése

lot-ejbservice project

```
1 package hu.qvaevisz.lottery.ejbservice.management;
2
3 @Description("Lottery Monitor")
4 public interface LotteryMonitorMBean {
5
6 @Description("Get current Puller")
7 String getPuller();
8 [...]
9 @Description("Set the given distribution list item")
10 void setDistribution(@PName("hit") int hit, @PName("value") int value);
11 }
```

LotteryMonitorMBean.java

## Technika

Néhány saját annotáció bevezetésével (@Description és @PName) illetve egy köztes AnnotatedStandardMBean osztály készítésével (mely öröklődik a StandardMBean-ből és mostantól a saját MBean-jeink őse lesz) megoldhatjuk a problémát karbantartható módon is. A megoldás során **Reflection API**-t használunk.

# Lottószelvény ellenőrzése

POST `http://localhost:8080/lottery/api/service/verify`

HTTP Request (application/json):

```
1 [5, 15, 20, 42, 40]
```

HTTP Response (application/json):

```
1 370
```

```
1 @Path("/service")
2 public interface LotteryRestService {
3 @POST
4 @Path("/verify")
5 @Consumes("application/json")
6 @Produces("application/json")
7 int verifyTicket(int[] numbers) throws AdaptorException;
8 }
```

LotteryRestService.java

## Szelvény nyereményalapjának kiszámítása

Aktuális nyerőszámok: 10, 20, 30, 40, 50

Nyereményalap: 12345

Nyeremény eloszlás: { 1, 3, 6, 10, 80 }

Két találatos szelvény:  $12345 * 0.03 = 370.35$

# Deploy - I

## Alkalmazás telepítése fejlesztési környezetben (lot-ear module)



```
1 <project [..]>
2 [..]
3 <build>
4 <finalName>${project.parent.artifactId}-${project.version}</finalName>
5 <plugins>
6 [..]
7 <plugin>
8 <groupId>org.apache.maven.plugins</groupId>
9 <artifactId>maven-dependency-plugin</artifactId>
10 <version>${version.plugin.dependency}</version>
11 <executions>
12 <execution>
13 <id>copy</id>
14 <phase>package</phase>
15 <goals>
16 <goal>copy</goal>
17 </goals>
18 </execution>
19 </executions>
20 <configuration>
21 [CONFIGURATION]
22 </configuration>
23 </plugin>
24 </plugins>
25 </build>
26 </project>
```

A **maven-dependency-plugin** verziószáma a *root pom.xml*-ben van definiálva (3.0.2).

A Maven **package** fázisához fog kötődni a másolás.

pom.xml



A [JBOSS\_HOME] környezeti változó beállítása az alkalmazott OS függvényében:  
NIX: export JBOSS\_HOME=~ /dev/jboss-eap-6.4 (.bash\_profile)  
WIN: JBOSS\_HOME=c:\apps\jboss-eap-6.4 (environment variable)

```
1 <configuration>
2 <artifactItems>
3 <artifactItem>
4 <groupId>${project.groupId}</groupId>
5 <artifactId>${project.artifactId}</artifactId>
6 <version>${project.version}</version>
7 <type>${project.packaging}</type>
8 <overwrite>true</overwrite>
9 <destFileName>${project.parent.artifactId}-${project.version}.ear</destFileName>
10 </artifactItem>
11 </artifactItems>
12 <outputDirectory>${env.JBOSS_HOME}/standalone/deployments/</outputDirectory>
13 </configuration>
```

pom.xml

```
1 > mvn clean package
```