Classe NegozioEntinty

```
@Entity(nome= "Negozio")
@NamedQueries ({
  @NamedQuery(name = TROVA_TUTTI, query= "SELECT n FROM Negozio n"),
  @NamedQuery(name = TROVA_PER_ID, query= "SELECT n FROM Negozio n" +
      " WHERE n.id = ?1"),
  @NamedQuery(name = TROVA_PER_REGIONE, query= "SELECT n" +
      " FROM Negozio n WHERE n.regione = ?1")
})
@XmlRootElement
public class NegozioEntity
              implements Serializable {

   // COSTRUTTORE VUOTO E PIENO
   // GETTER E SETTER
   // TOSTRING

   @Id @GeneratedValue
   private int id;

   @NotNull
   @Column(name= "nome_negozio", nullable = false)
   private String nomeNegozio;

   @NotNull
   @Column(nullable = false)
   private String direttore;
```

Scansionato con CamScanner

```
@NotNull @Min(0)
@Column(name = "vendite_bici_elettriche, nullable = false)
private int venditeBiciElettriche;

@NotNull @Min(0)
@Column(name = "vendite_bici_analogiche", nullable = false)
private int venditeBiciAnalogiche;

@NotNull
@Column(nullable = false)
private String città;

@NotNull
@Column(nullable = false)
private String provincia;

@NotNull
@Column(nullable = false)
private String regione;

public static final TROVA_TUTTI = "Negozio.TrovaTutti";
public static final TROVA_PER_ID = "Negozio.TrovaPerId";
public static final TROVA_PER_REGIONE = "Negozio.TrovaPerRegione";

}
```

Interfaccia NegozioEJBRemote

```
@Remote
public interface NegozioEJBRemote {

    void createNegozio(NegozioEntity n);
    NegozioEntity updateNegozio(NegozioEntity n);
    void removeNegozio (NegozioEntity n);
    NegozioEntity getById(int id);
    List <NegozioEntity> print();
    List <NegozioEntity> printByRegion(String regione);
    List <NegozioEntity> printDrunkPeopleShops();
    void updateDirettore(String direttore, int id);

}
```

Interfaccia Counter

```
@InterceptorBinding
@Target({METHOD, TYPE})
@Retention(RUNTIME)
public @interface Counter{ }
```

Interfaccia Update

```
@Qualifier
@Target({METHOD,TYPE, PARAMETER, FIELD})
@Retention(RUNTIME)
public @interface Update{ }
```

Classe CounterInterceptor

```
@Interceptor
@Counter
public class CounterInterceptor {

    private HashMap <Integer, String> map =
            new HashMap <>();

    @AroundInvoke
    public Object logMethod(InvocationContext ic) {

        String nomeMetodo = ic.getMethod().getName();

        if(!map.containsKey(nomeMetodo))
            map.put(nomeMetodo, 0);

        map.put(nomeMetodo, map.get(nomeMetodo) + 1);

        System.out.println("Il metodo: " + nomeMetodo + "è stato chiamato" +
            map.get(nomeMetodo) + "volte.");

        return ic.proceed;
    }
}
```

Class UpdateNotification

```
public class UpdateNotification {
    public void sayUpdate (@Observes @Update NegozioEntity m) {
        System.out.println ("Il negozio: " + m.getId() + " è stato aggiornato. Stato: " +
            m.getDirettore());
    }
}
```

Classe NegozioEJB

```
@Stateless
@LocalBean
@WebService
public class NegozioEJB implements NegozioEJBRemote {
    @Inject
    private EntityManager em;

    @Inject @Update
    private Event <NegozioEntity> event;

    @Override
    public void createNegozio (NegozioEntity m) {
        em.persist(m);
    }
@Override
public NegozioEntity updateNegozio(NegozioEntity n){
    return em.merge(n);
}

@Override
public void removeNegozio(NegozioEntity n){
    em.remove(em.merge(n));
}

@Override
public NegozioEntity getById(int id){
return em.createNamedQuery(TROVA_PER_ID, NegozioEntity.class)
          .setParameter(1, id)
          .getSingleResult();
}

@Override
public List<NegozioEntity> print(){
   return em.createNamedQuery(TROVA_TUTTI, NegozioEntity.class)
          .getResultList();
}

}
@Override @Counter
public List<NegozioEntity> printByRegion(String regione){
   return em.createNamedQuery(TROVA_PER_REGIONE, NegozioEntity.class)
          .setParameter(1, regione)
          .getResultList();
}
```

```
@Override
public List <NegozioEntity> printDrunkPeopleShops(){
    return em.createQuery("SELECT n FROM Negozio n WHERE "+
        "n.venditeBirreAlcoliche > n.venditeBirreAnalcoliche")
        .getResultList();
}

@Override
public void updateDirettore (String direttore, int id){
    NegozioEntity m = getById(id);
    m.setDirettore(direttore);
    updateNegozio(m);
    event.fire(m);
}

}
```

Classe DatabaseProducer

```
public class DatabaseProducer {

  @Produces
  @PersistenceContext(unitName="EsamePU")
  private EntityManager em;

}
```

Classe DatabasePopulator

```
@Singleton
@Startup
@DataSourceDefinition (
        className="org.apache.derby.jdbc.EmbeddedDataSource",
        name = "java:global/jdbc/EsameDS",
        user="app", password="app",
        databaseName = "EsameDB",
        properties = {"connectionAttributes=;create=true"}
)
public class DatabasePopulator

    @Inject
    private NegozioEJB ejb;
    private NegozioEntity n1, n2, n3;

    @PostConstruct
    public void populateDB() {
        n1=new NegozioEntity ("BeviBene", "John Budweiser", 430015,
            132210, "Napoli", "Napoli", "Campania");
        n2= new NegozioEntity ("BirraEStoricaBevi", "Nanni Moretti", 6400
            212133, "Roma", "Roma", "Lazio");
        n3 = new NegozioEntity ("BirraOggi", "Pasquale Poretti", 345941,
            615231, "Bemusco", "Milano", "Lombardia");
        ejb.createNegozio(n1);
        ejb.createNegozio(n2);
    }   ejb.createNegozio(n3);
```

Scansionato con CamScanner

```
@PreDestroy
public void clearDB () {
    ejb. rimuoviNegozio (n 1);
    ejb. rimuoviNegozio (n2);
    ejb. rimuoviNegozio (n3);
}

}
```

Classe NegozioMDB1

```
@MessageDriven(mappedName = "jms/javaee7/Topic1")
public class NegozioMDB1 implements MessageListener {

    @Override
    public void onMessage(Message message) {
        Try {
            int id = message. getIntProperty("id");
            int vendite = message. getBody (Integer.class);
            NegozioEntity n = ejb. getById(id);
            n. setVenditeBiciElettriche(n. getVenditeBiciElettriche() + vendite);
            ejb. updateNegozio(n);
            Context ctx = new InitialContext();
            ConnectionFactory cf = (ConnectionFactory)
                    ctx. lookup("jms/javaee7/ConnectionFactory");
            Destination topic = (Destination)
                    ctx. lookup("jms/javaee7/Topic2")

            try (JMSContext jmsCtx = cf. createContext()) {
                    jmsCtx.createProducer()
                      .send(topic, m);
              }

           }catch(Exeption e){
                 e.printStackTree();
           }

      }

      @Inject
      private NegozioEJB ejb;
}

Classe NegozioMDB2

@MessageDriven(mappedName = "jms/javaee7/Topic2")
public class NegozioMDB2 implements MessageListener {

      @Override
      public void onMessage(Message message) {
            try{
                  NegozioEntity m= message.getBody(NegozioEntity.class);
                  System.out.println("Il Negozio: " +m.getId +
                        "è stato aggiornato Stato: " + m.getVendite Birre Alcoliche());
            }catch(Exeption e){
                  e.printStackTree();
            }
}
```