

Classe VentilatoreEntity

@Entity (name = "Ventilatore")

@NamedQueries {

@NamedQuery (name = TROVA_TUTTI, query = "SELECT v FROM Ventilatore v"),

@NamedQuery (name = TROVA_PER_ID, query = "SELECT v FROM Ventilatore v " +
"WHERE v.id = ?1"),

@NamedQuery (name = TROVA_PER_NATURA, query = "SELECT v FROM " +
"Ventilatore v WHERE v.natura = ?1")

}

@XmlRootElement

public class VentilatoreEntity implements Serializable {

public static final String TROVA_TUTTI = "Ventilatore.TrovaTutti";

public static final String TROVA_PER_ID = "Ventilatore.TrovaPerId";

public static final String TROVA_PER_NATURA = "Ventilatore.TrovaPerNatura";

// COSTRUTTORE (PIENO E VUOTO)

// GETTER E SETTER

// toString

@Id @GeneratedValue

private int id;

@NotNull

@Column (nullable = false)

private String modello;

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

```
@NotNull @Min(0) @Max(10)  
@Column(nullable=false)  
private int rumore;
```

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

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

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

}

Interface VentilatorESBRemote

@Remote
public interface VentilatorESBRemote {

void createVentilator(VentilatorEntity v);

VentilatorEntity updateVentilator(VentilatorEntity v);

void removeVentilator(VentilatorEntity v);

VentilatorEntity getById(int id);

List<VentilatorEntity> print();

List<VentilatorEntity> printByCountry(String country);

List<VentilatorEntity> printQuickFun();

VentilatorEntity updateRampPerMinute(int rampPerMinute);

}

Interface Counter

@InterceptorBinding

@Target({METHOD, TYPE})

@Retention(RUNTIME)

public @interface Counter { }

Classe VentilatoreESB

@Stateless

@LocalBean

@WebService

public class VentilatoreESB implements VentilatoreESBRemote {

@Inject

private EntityManager em;

@Inject @UpdatePizi

private Event < VentilatoreEntity > e;

@Override

public void createVentilatore(VentilatoreEntity v) {

em.persist(v);
}

@Override

public VentilatoreEntity updateVentilatore(VentilatoreEntity v) {

return em.merge(v);
}

@Override

```
public void removeVentilatore (VentilatoreEntity v) {  
    em.remove(em.merge(v));  
}
```

@Override

```
public VentilatoreEntity getByid (int id) {  
    return em.createNamedQuery (  
        VentilatoreEntity.TROVA_PER_ID, VentilatoreEntity.class)  
        .setParameter (1, id)  
        .getSingleResult();  
}
```

@Override @Counter

```
public List < VentilatoreEntity > printByCountry (String country) {  
    return em.createNamedQuery (  
        VentilatoreEntity.TROVA_PER_NAZIONE, VentilatoreEntity.class)  
        .setParameter (1, country)  
        .getResultList();  
}
```

@Override

```
public List < VentilatoreEntity > print () {  
    return em.createNamedQuery (  
        VentilatoreEntity.TROVA TUTTI, VentilatoreEntity.class)  
        .getResultList();  
}
```

```

@Override
public List<VentilatoreEntity> printQuickFun() {
    return em.createQuery(
        "SELECT v FROM Ventilatore v WHERE " +
        "v.giriMinuto > 2000", VentilatoreEntity.class)
        .getResultList();
}

```

```

@Override
public VentilatoreEntity updateGiriPerMinuto(int countPerMinuto) {
    VentilatoreEntity v = getByid(id);
    v.setGiriMinuto(countPerMinuto);
    // NON OCCORRE FARE IL ROLLBACK POICHÉ L'ENTITY
    // ESSENDO MANAGED È SINCRONIZZATA COL DATABASE
    em.flush(v);
    return v;
}

```

Classe DatabaseProducer

```

public class DatabaseProducer {

```

```

    @Produces

```

```

    @PersistenceContext(unitName = "Esercizio")

```

```

    private EntityManager em;

```

```

}

```


Classe DatabasePopulator

@Singleton

@Startup

@DataSourceDefinition (

className = "org.apache.derby.jdbc.EmbeddedDataSource",

name = "some:global/jdbc/Embedded",

user = "app", password = "app",

databaseName = "Embedded",

properties = {"connectionAttributes"; create = true;"}

}
public class DatabasePopulator {

@Inject

private Ventilatore EJB ejb;

private VentilatoreEntity v1, v2, v3;

@PostConstruct

private void populateDB() {

v1 = new VentilatoreEntity("SoffiaFresco", "Ventilatore per Tutti", 1500, 4, 10, "Italia");

v2 = new VentilatoreEntity("Wind 3000", "Sturm", 2200, 6, 12, "Germania");

v3 = new VentilatoreEntity("Breezelegger", "Fresco", 1000, 5, 11, "Francia");

ejb.createVentilatore(v1);

ejb.createVentilatore(v2);

ejb.createVentilatore(v3);

}

@BeforeClass

```
private void cleanDB() {
```

```
    egb. removeVentilatore(v1);
```

```
    egb. removeVentilatore(v2);
```

```
    egb. removeVentilatore(v3);
```

```
}
```

Class CounterInterceptor

@Counter

@Interceptor

```
public class CounterInterceptor {
```

```
    private int counter = 0;
```

@AroundInvoke

```
public Object logMethod (InvocationContext ic) throws Exception {
```

```
    String methodName = ic.getMethod().getName();
```

```
    if (methodName.equals("printByCounter")) {
```

```
        counter++;
```

```
        System.out.print("Il metodo " + methodName + " è stato chiamato " +  
            counter + " volte.");
```

```
    }
```

```
    return ic.proceed();
```

```
}
```


Classe VentilatoreEJB

@MessageDriven(mappedBy = "jms/gasare7/Topic")
public class VentilatoreEJB implements MessageListener {

@Inject
private VentilatoreEJB esb;

@Inject @UpdateVendite
private Event <VentilatoreEntity> e;

@Override
public void onMessage(Message message) {
try {
int id = message.getIntProperty("id");
int vendite = message.getIntBody(Integer.class);
VentilatoreEntity v = esb.getById(id);
v.setElement.Venduti(v.getElement.Venduti() + vendite);
// NON SERVE FARE L'UPDATE
e.fire(v);
} catch (JMSException e) {
// GESTIRE ECCEZIONE
}
}

Interface UpdateVendite

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

Interface UpdateGiac

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

Classe UpdateNotification

```
public class UpdateNotification {
```

```
    public void logUpdateVendite (@Observer @UpdateVendite VentilatoreEntity v) {  
        System.out.println("Il ventilatore " + v.getId() + " è stato aggiornato." +  
            " stato: " + v.getElement.Venduto());  
    }
```

```
    public void logUpdateGiac (@Observer @UpdateGiac VentilatoreEntity v) {  
        System.out.println("Il ventilatore " + v.getId() + " è stato aggiornato." +  
            " stato: " + v.getGiacMinuto());  
    }
```

```
}
```


Viene creato il file persistence.xml dove:

- definiamo la persistence unit Example;
- specifichiamo il Source del data source all'interno del tag:
`<src-data-source> ... </src-data-source>`
- impostiamo il target del database al valore DERBY.

Viene creato il file bean.xml dove:

- sostituiamo emulated con all
- definiamo l'interceptor con il nodo
`<interceptors>`
`<class> it.unisa.lanter.Interceptor </class>`
`</interceptors>`

Struttura progetto server = nome progetto VentilatoreServer

SRC

L> it.unisa

L> VENTILATOREENTITY

VENTILATORE EJB RETORE

COUNTER

VENTILATORE EJB

DATABASEPRODUCER

DATABASEPRODUCER

COUNTERINTERCEPTOR

VENTILATORETTDB

UPDATEVENTITE

UPDATE GILI

UPDATE NOTIFICATION

CONF

L> PERSISTENCE.XML

BEAN.XML

LIBRARY

L> JAVA EE API 8

Classe Client_1

```
public class Client_1 {
```

```
    public static void main (String[] args) throws NamingException {
```

```
        Context ctx = new InitialContext();
```

```
        VentilatoreESBRemote esb = (VentilatoreESBRemote)
```

```
            ctx.lookup("java:global/VentilatoreServer/VentilatoreESB/" +  
                "it.unisa.VentilatoreESBRemote");
```

```
        Scanner in = new Scanner(System.in);
```

```
        System.out.println("Inserisci una mazione:");
```

```
        String mazione = in.nextLine();
```

```
        esb.printByConting(mazione).forEach(  
            (c) -> System.out.println(c)
```

```
        );
```

```
    }
```

```
}
```

Struttura progetto client, nome progetto = VentilatoreStdClient1

SRC

↳ it.unisa

↳ CLIENT_1

VENTILATOREENTITY

VENTILATOREESBREMOTE

LIBRARY

↳ GF-CLIENT

JAVA EE 8 API

Classe Client_2

public class Client_2 {

public static void main (String[] args) throws NamingException {

```
Context ctx = new InitialContext();  
VentilatoreESBRemote esb = (VentilatoreESBRemote)  
ctx.lookup("java:global/VentilatoreSensor/VentilatoreESB/" +  
"it.unisa.VentilatoreESBRemote");
```

```
System.out.println("Ventilatore con giri al minuto superiori e 2000");
```

```
esb.printQuickFun().forEach(  
v) -> System.out.println(v)  
);
```

```
}
```

```
}
```

Struttura progetto client, nome progetto = Ventilatore Std Client 2

SRC

↳ it.unisa

↳ CLIENT_2

VENTILATOREENTITY

VENTILATOREESBREMOTE

LIBRARY

↳ GF-CLIENT

SAVE EE 8 APR

Classe SmsClient

```
public class SmsClient
```

```
public static void main (String[] args) throws NamingException {
```

```
    Context ctx = new InitialContext();  
    ConnectionFactory cf = (ConnectionFactory)  
        ctx.lookup("jms/garale7/ConnectionFactory");  
    Destination topic = (Destination)  
        ctx.lookup("jms/garale7/Topic");
```

```
    Scanner in = new Scanner(System.in);  
    System.out.println("Inserisci l'id del ventilatore:");  
    int id = in.nextInt();  
    System.out.println("Inserisci il numero di rendite da aggiungere:");  
    int rendite = in.nextInt();
```

```
    try (JMSContext jmsCtx = ctx.createContext()) {
```

```
        jmsCtx.createProducer()  
            .setProperty("id", id)  
            .send(topic, rendite);
```

```
    }
```

```
}
```


Struttura del progetto, nome = VentilatoreSRIClient

SRE

L> it.uniba

L> SRIClient

LIBRARY

L> SAAEEB API

GF-CLIENT

Classe ClientWS

public class ClientWS {

public static void main (String[] args) {

Scanner in = new Scanner(System.in);
System.out.println("Inserisci l'id del ventilatore:");
int id = in.nextInt();
System.out.println("Inserisci le rotate giri/minuto:");
int giriMinuto = in.nextInt();
updateRPMPerMinute(id, giriMinuto);

}

public void updateRPMPerMinute (int arg0, int arg1) {

it.uniba.VentilatoreESBService service = new it.uniba.VentilatoreESBService();
it.uniba.VentilatoreESBPort port = service.getVentilatoreESBPort();
port.updateRPMPerMinute(arg0, arg1);

}

}

Per ottenere il metodo potete guardare il mio repository e creare un Web Client indicando come riferimento WSDL il progetto VentilatoreService.

Struttura progetto, nome = VentilatoreWSClient

SRE

L> it.unisa

L> CVENTUS

LIBRARY

L> JAVA EE 8 API