
[License](#) CC0 1.0



-
- -
 -
 -
 -
 -
 -
 -
 -
 -
 -
 -

o
o
o
o
o
■
■
■
o
o
o
o
■
■
o
o
o
o
●
o
o
o
o
o
■
o
■
o
●
o
o
o
■
■
■
o
o
■



o
o
o
o
o
o

■
■
■
■
■
■
■
■
■

•

o
o
o
o
o
o
o
o
o
o
o
o
o
o
o
o

•

o
o
o
o

■
■
■
■
■
■
■
■
■



•
○
○
○
○
○
○
○
○
○
○
○
○
○
○
○
○
○

•
○
○
○
○
○

•
•
•

•
•

•

•

•

•

○

○

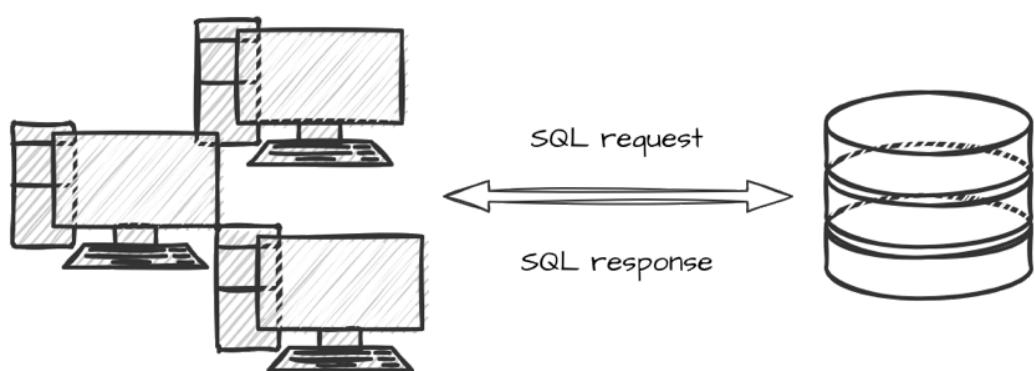
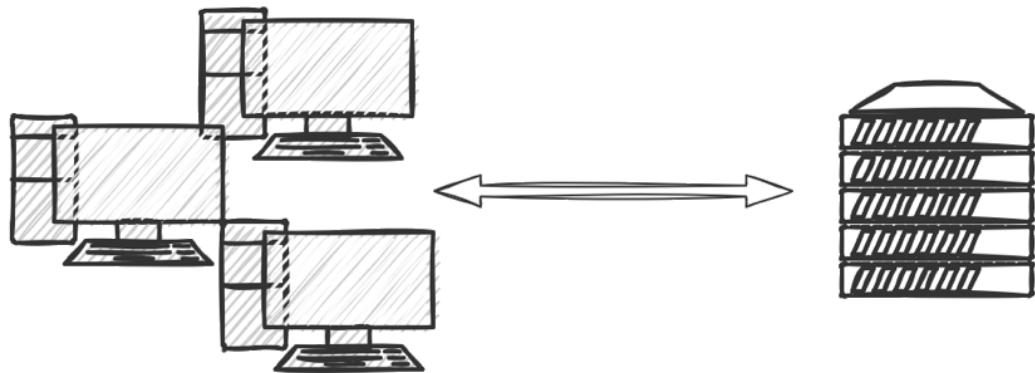
A

B

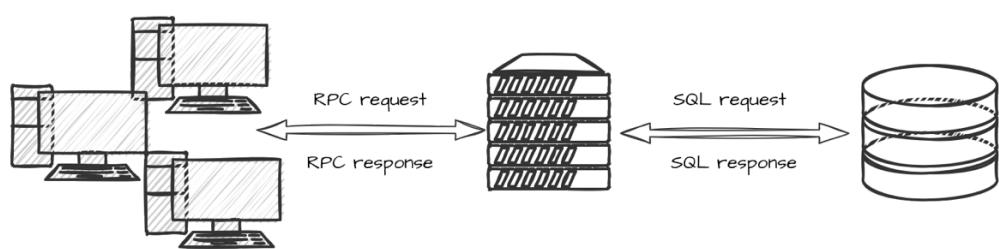
A

B

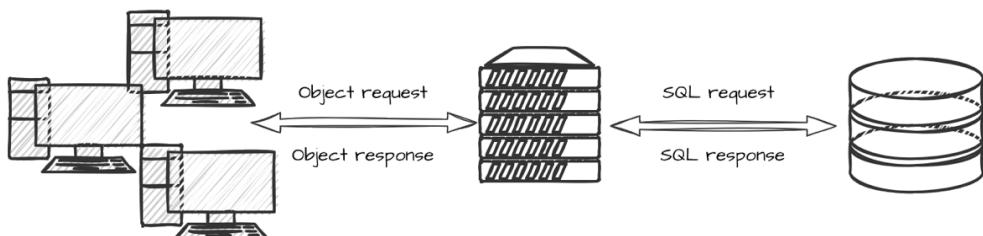




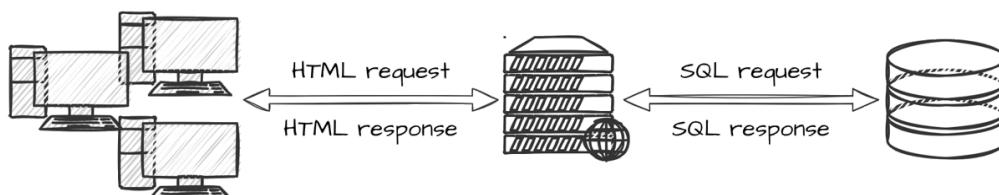
- o

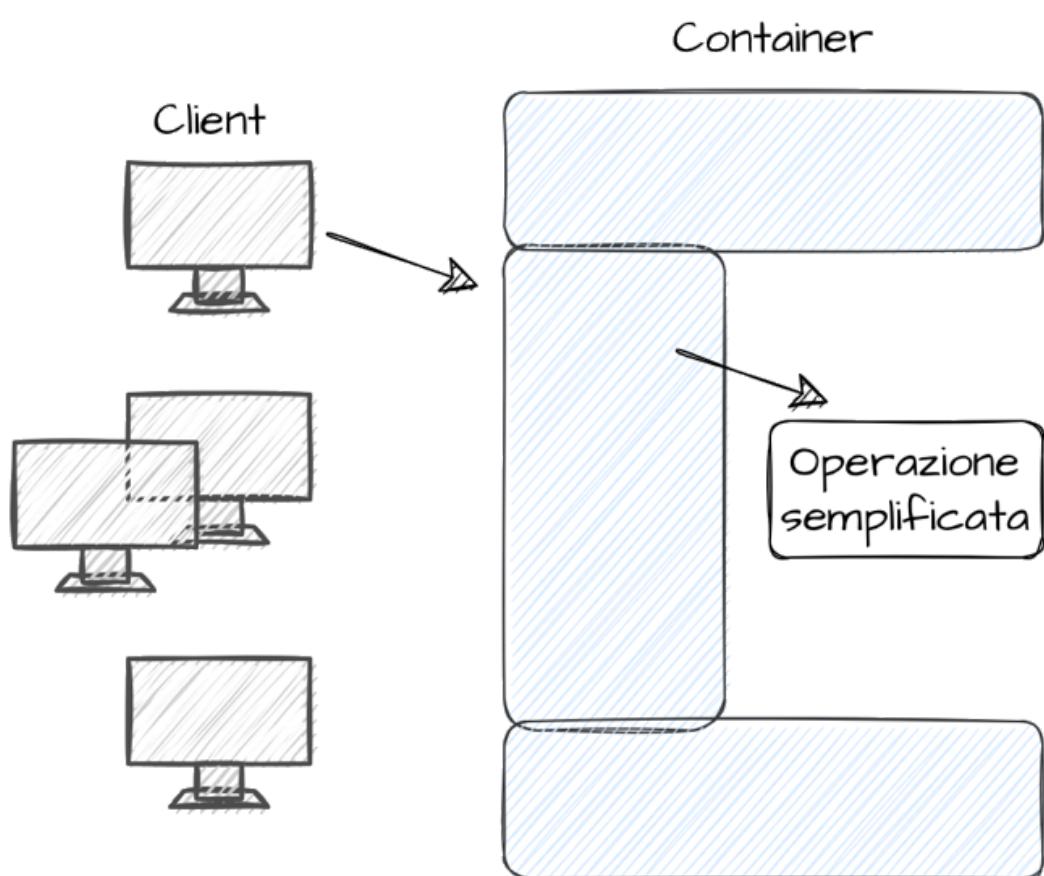


- o



- o





•

•

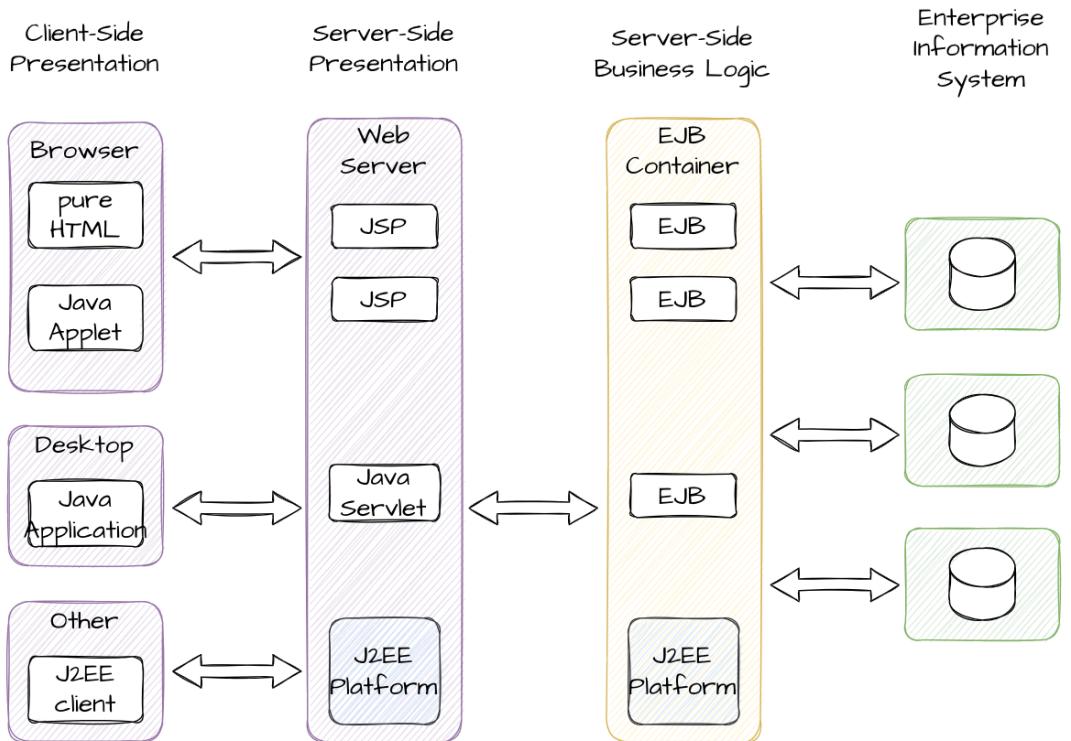
•

•

•

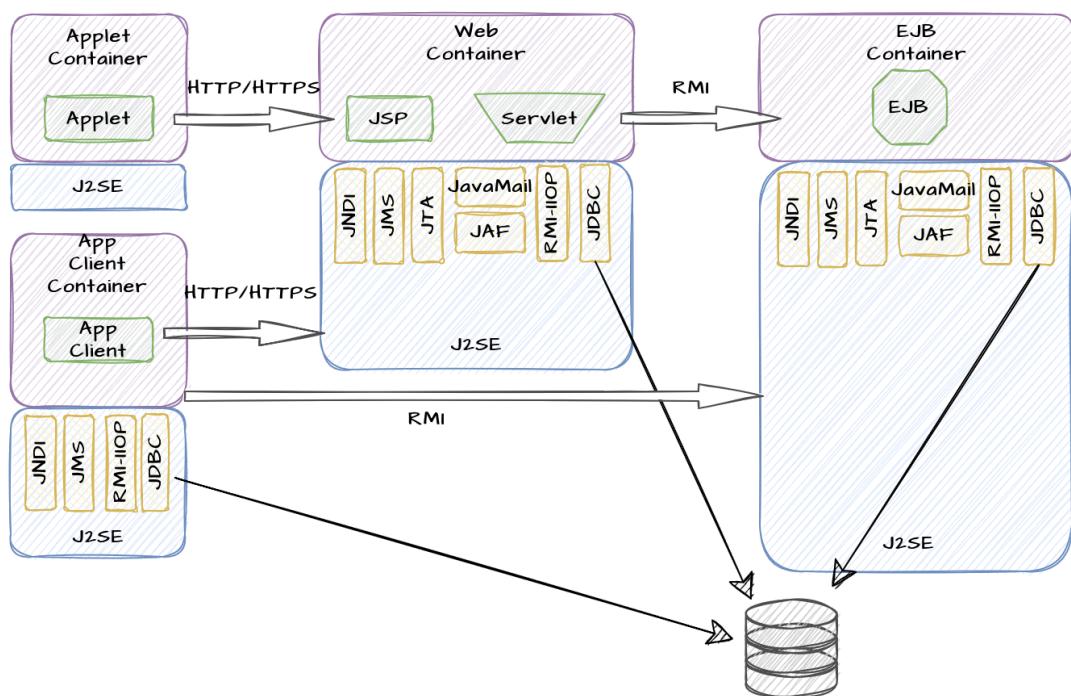
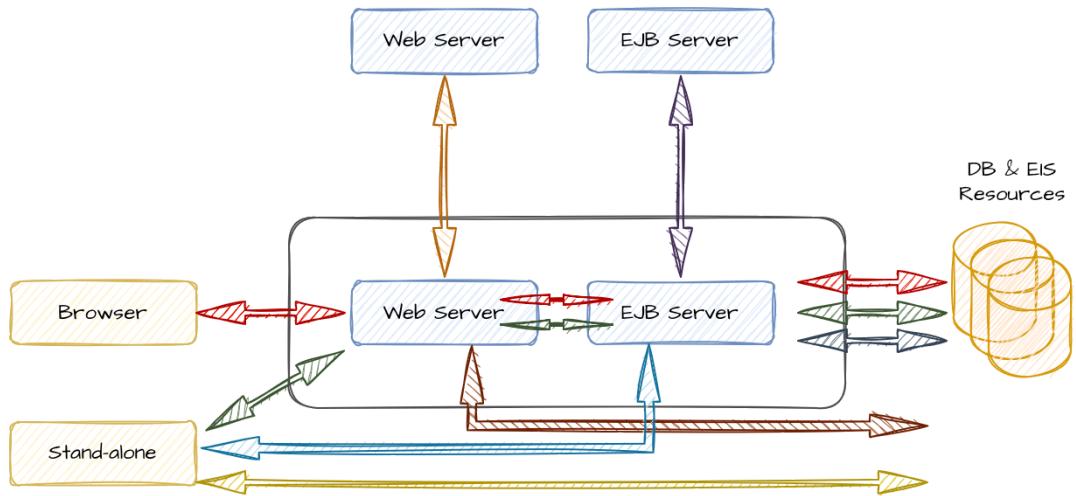
○

○



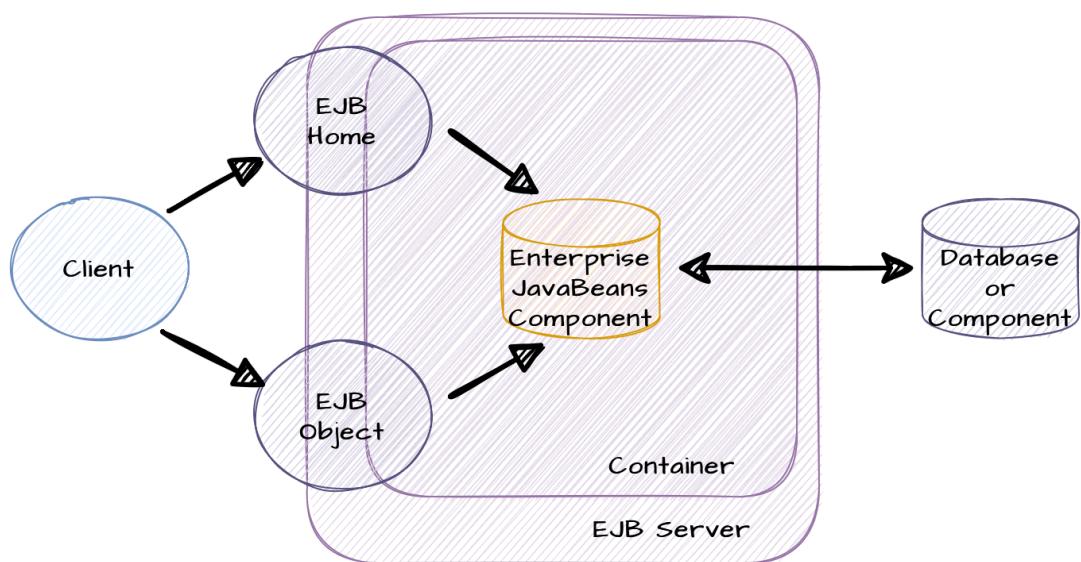
Capitoli 2

Capitolo 5

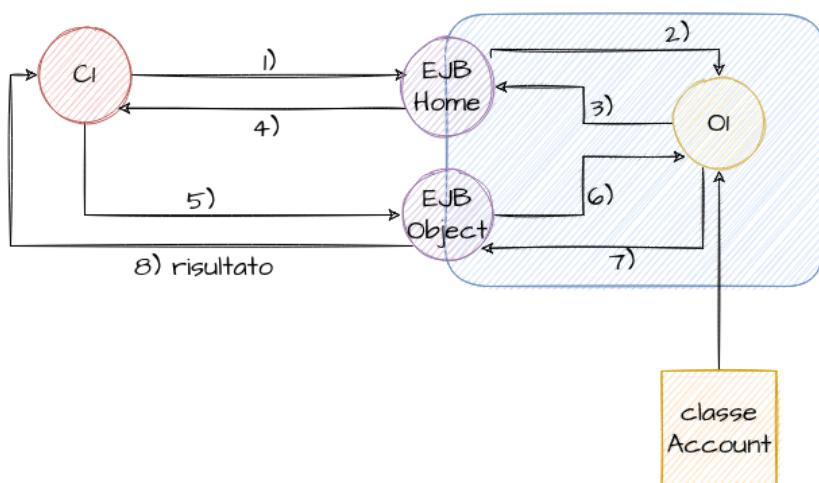


B

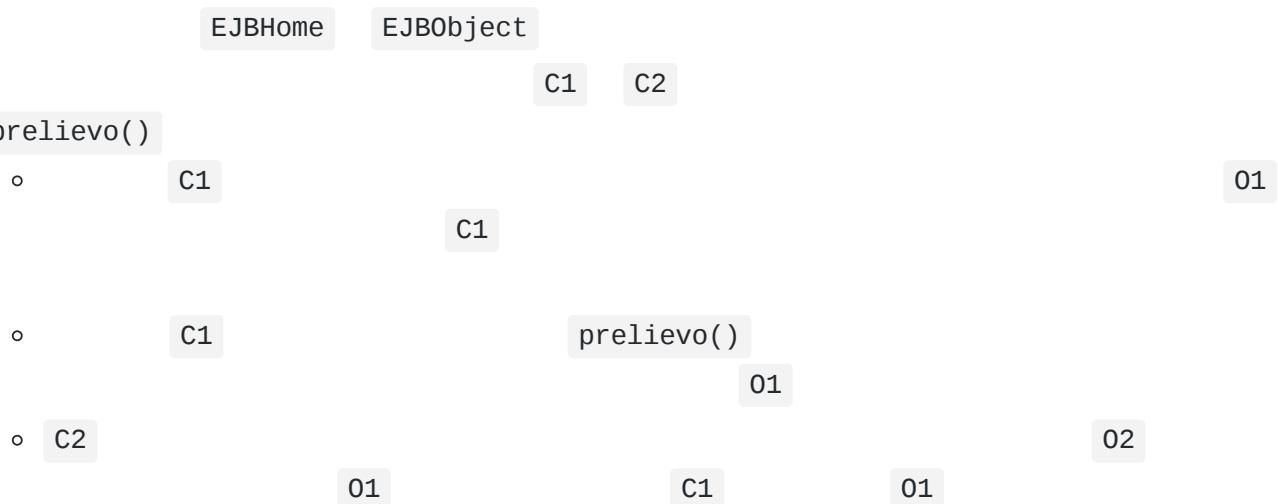
A B A



- EJBHome
- EJBObject



- Account
- prelievo() deposito()



- -

EJBHome

-

EJBObject

-

- -

-

EJBHome

EJBObject

-

-

-

-

EJBHome

EJBObject

•
•
•
•
•
•
•

•

○

■

■

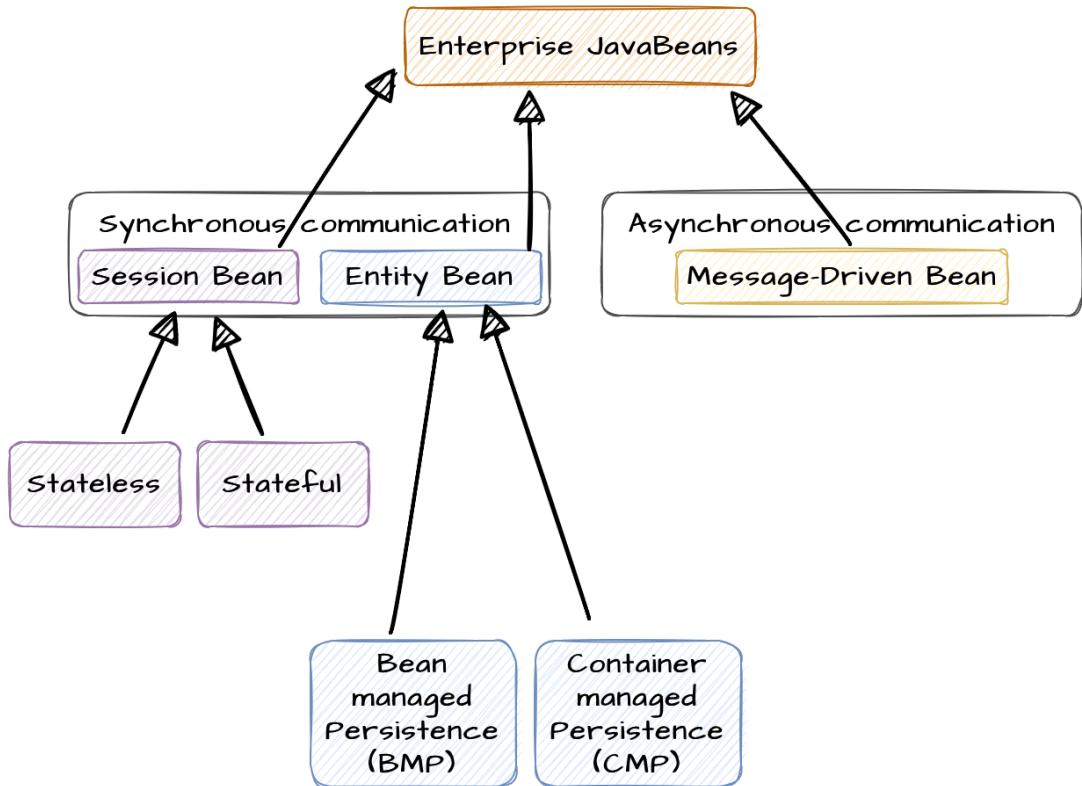
○

■

■

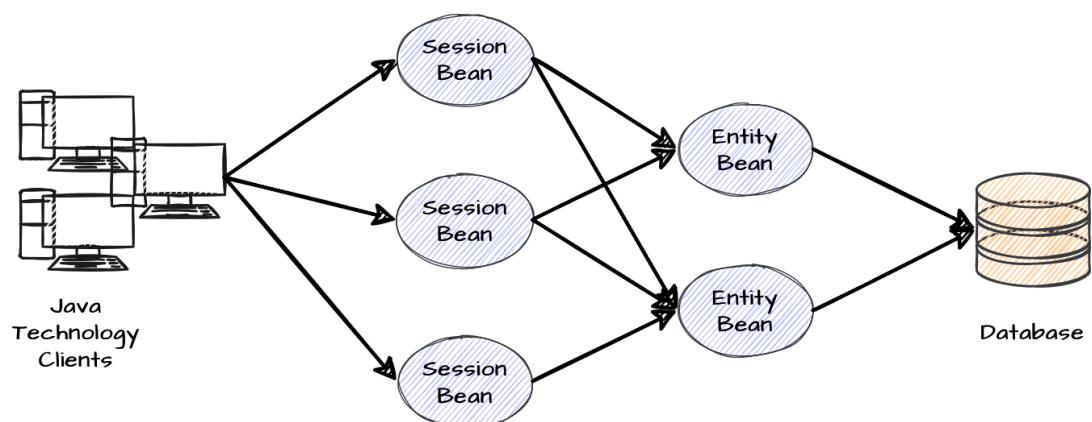
•

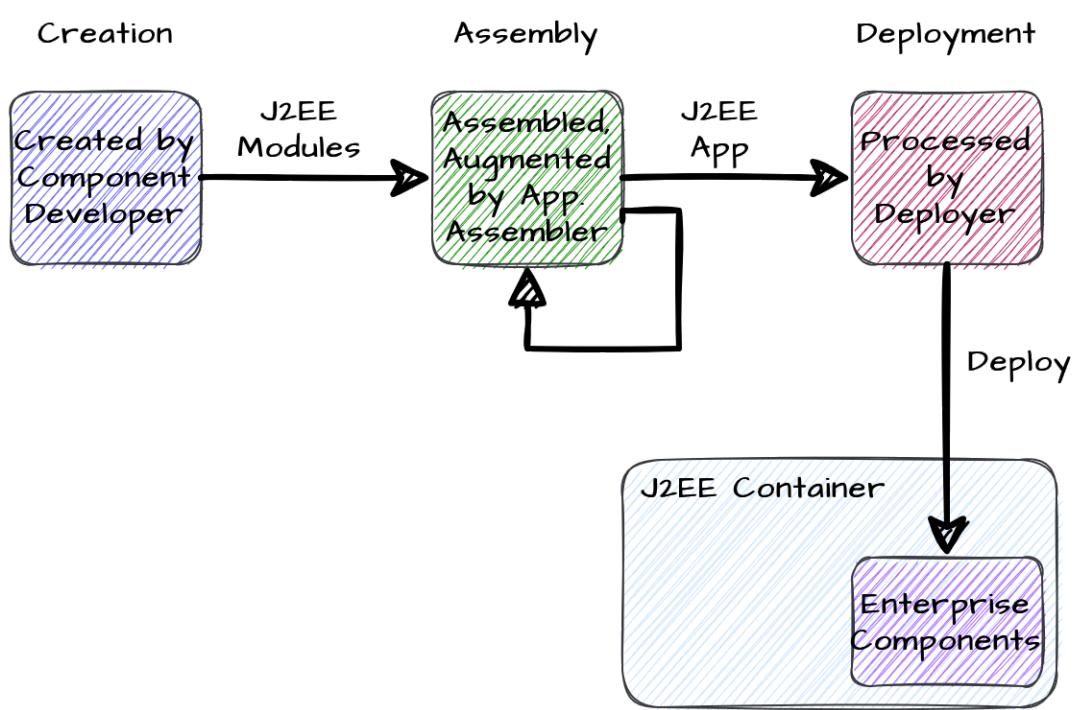
○

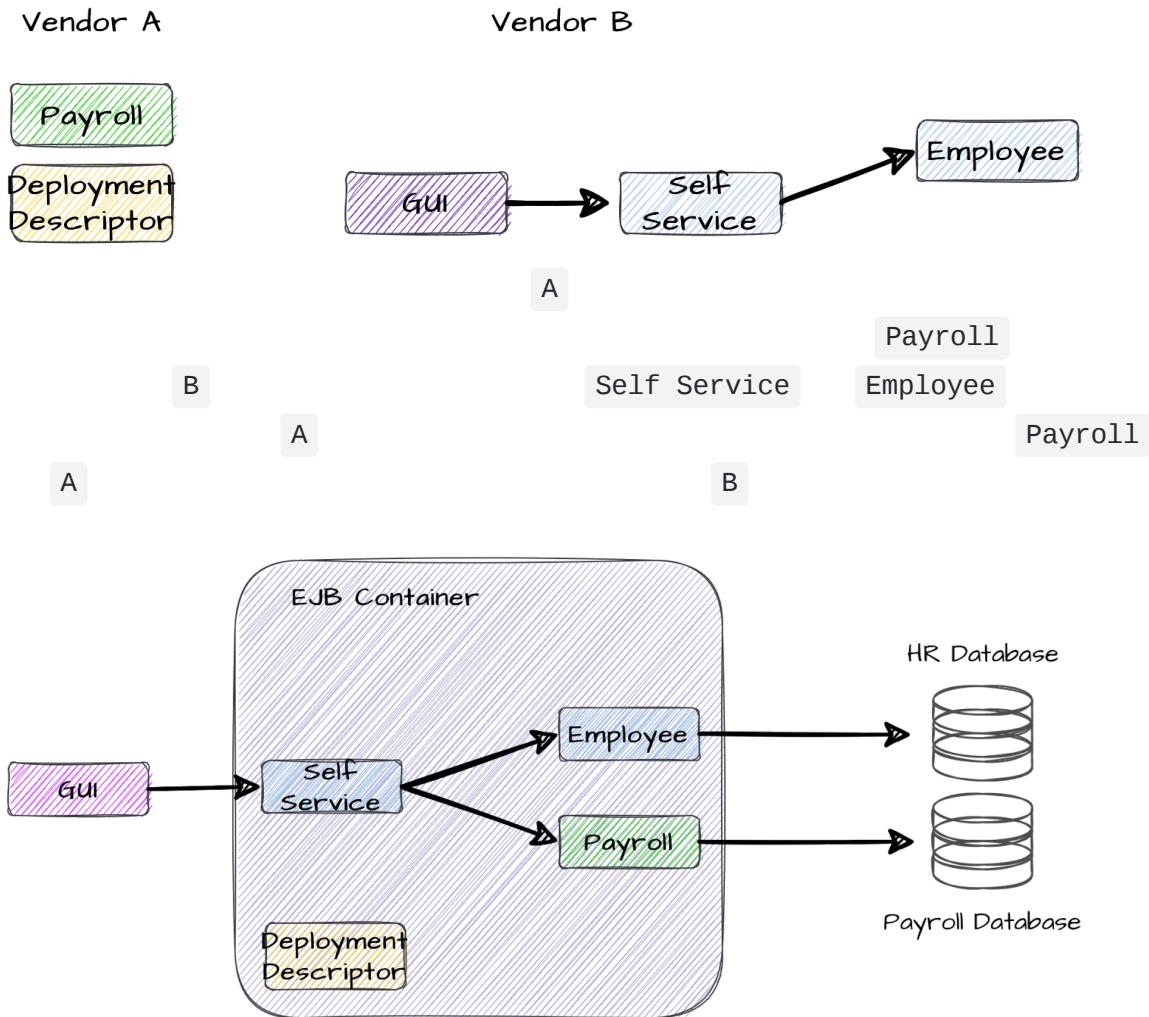


- javax.ejb.SessionBean

Capitolo 7







- **EJBHome**

- **create()** **find()** **remove()**

- **EJBObject**

- **create()** **find()**

EJBHome

EJBObject

```
// EJBHome
package com.ejb_book.interest;

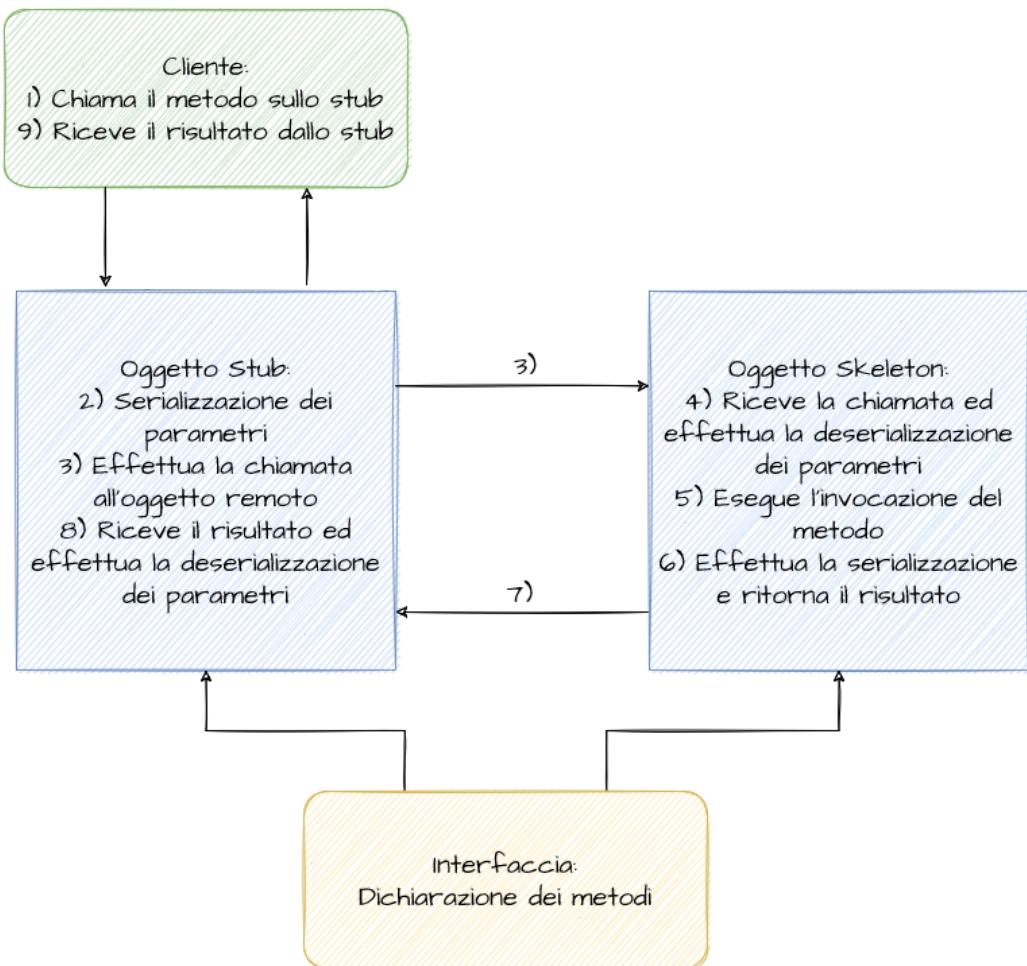
import javax.ejb.*;
import java.rmi.*;

public interface InterestHome extends EJBHome{
    public Interest create() throws CreateException, RemoteException;
}
```

```
// EJBObject
package com.ejb_book.interest;

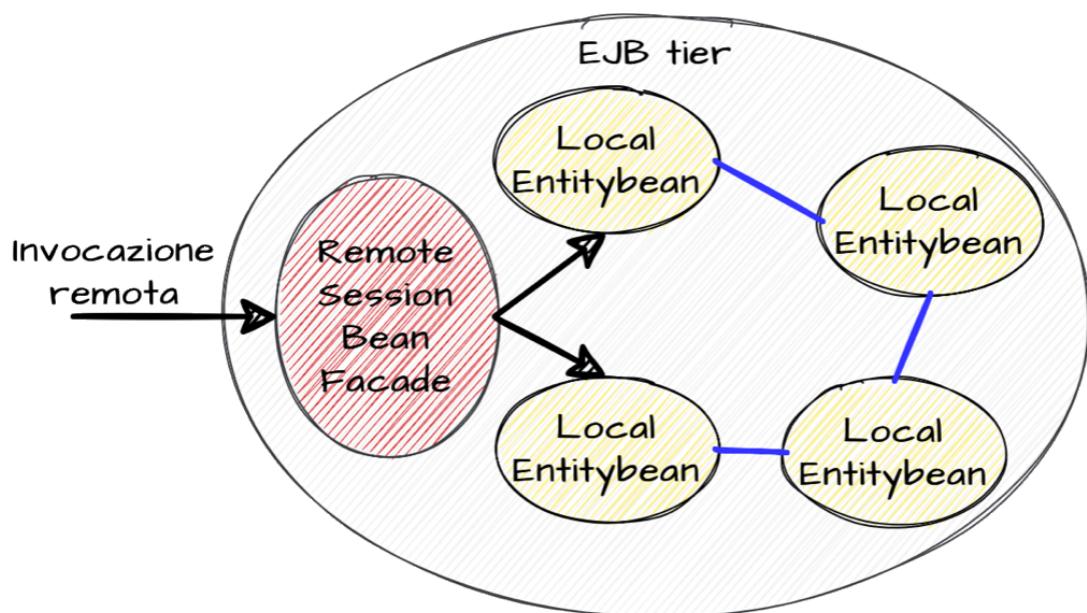
import javax.ejb.*;
import java.rmi.*;

public interface Interest extends EJBObject {
    // Calcola l'interesse da pagarsi ad un dato proprietario, ad uno specifico
    // tasso di interesse (percentuale per term)
    public double getInterestOnPrincipal(double principal, double interestPerTerm, int terms)
        throws RemoteException;
}
```



EJBLocalObject

EJBLocalHome



```
// EJBHome
package com.ejb_book.interest;

import javax.ejb.*;
import java.rmi.*;

public interface InterestLocalHome extends EJBLocalHome {
```

```
    public InterestLocal create() throws CreateException;  
}
```

```
// EJBObject  
package com.ejb_book.interest;  
  
import javax.ejb.*;  
import java.rmi.*;  
  
public interface InterestLocal extends EJBLocalObject {  
  
    // Calcola l'interesse da pagarsi ad un dato proprietario, ad uno specifico  
    // tasso di interesse (percentuale per term)  
    public double getInterestOnPrincipal(double principal, double interestPerTerm, int terms);  
  
}
```

EJBHome

EJBLocalHome

RemoteException

•

- InitialContext

- lookup

-

- create()

-

```
public class InterestClient {
```

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

```
// passo 1: ottenere un'istanza di EJBHome (in realtà un oggetto
// stub per l'oggetto EJBHome) via JNDI
InitialContext initialContext = new InitialContext();
Object o = initialContext.lookup("Interest");
InterestHome interestHome = (InterestHome) PortableRemoteObject.narrow(o, InterestHome.class);

// passo 2: creare un oggetto EJBObject remoto (in realtà
// uno stub all'oggetto EJBObject remoto
Interest interest = interestHome.create();

double principal = 10000.0;
double rate = 10.0;
int terms = 10;

System.out.println("Principal = $" + principal);
System.out.println ("Rate(%) = " + rate);
System.out.println ("Terms = " + terms);

// passo 3: invocazione metodi di business
System.out.println("Interest = $" + interest.getInterestOnPrincipal(principal, rate, terms));

System.out.println("Total = $" + interest.getTotalRepayment(principal, rate, terms));

// passo 4: clean up
interest.remove();
}

}
```

•

○

○

■

■
■
■

○

●

●

●

●

- `@Override`

```
    @Override  
    public String toString() {  
        ...  
    }
```

- `@Deprecated`

```
    @Deprecated  
    public class ExampleClass { ... }
```

- `@SuppressWarnings`

```
    @SuppressWarnings("unchecked")  
    public void aMethod() {  
        ...  
    }
```

-
-
-

- `@Override`

- `@Deprecated`

- `@SuppressWarnings`

```
@Override  
public String toString() {  
    ...  
}
```

- @Override
- @SuppressWarnings("unchecked")
-
-

```
public @interface GroupTODO {  
    public enum Severity {CRITICAL, IMPORTANT, TRIVIAL} ;  
    Severity severity() default Severity.IMPORTANT;  
    String item();  
    String assignedTo();  
}
```

@interface

```
@GroupTODO ( severity = GroupTODO.Severity.CRITICAL; item = "Figure out the amount of interest per month" assignedTo = "Luca Foschini"; ) public void calculateInterest(float amount, float rate) { ... }
```

severity

IMPORTANT

- • • •

AnnotationA

AnnotationB

AnnotationA

- @Target

```
@Target ( { ElementType.METHOD,ElementType.PACKAGE } )  
public @interface ExampleAnnotation { ... }
```

- `@Documented`

```
@Documented  
public @interface ExampleAnnotation { ... }
```

- `@Inherited`

```
@Target ( { ElementType.METHOD, ElementType.PACKAGE } )  
public @interface ExampleAnnotation { ... }
```

- `@Retention`

```
@Inherited  
public @interface ExampleAnnotation { ... }
```

`@Retention`

- `@Retention(RetentionPolicy.SOURCE)`

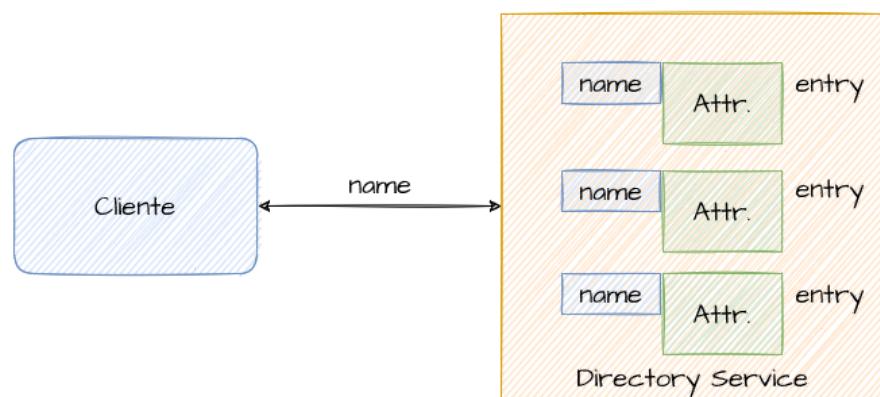
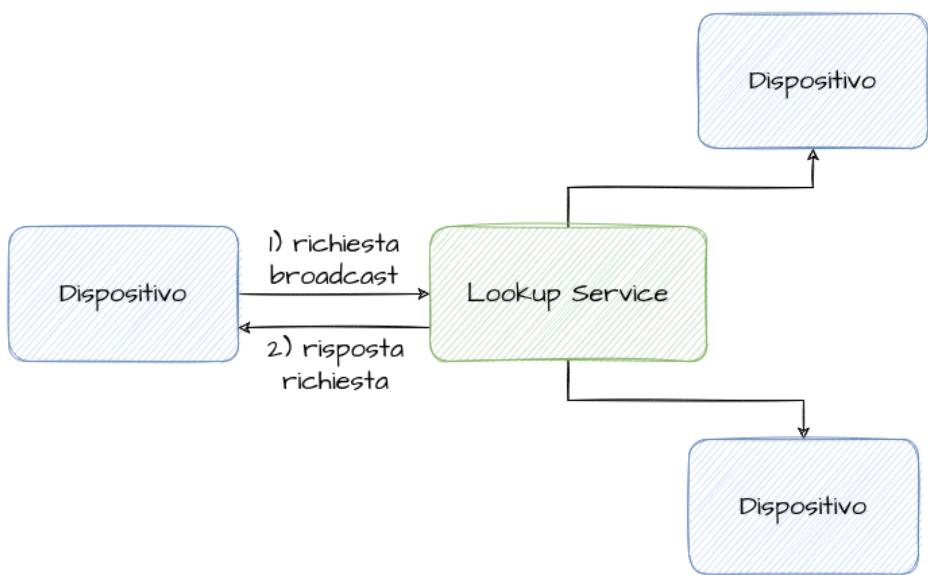
`@Override`

- `@Retention(RetentionPolicy.CLASS)`

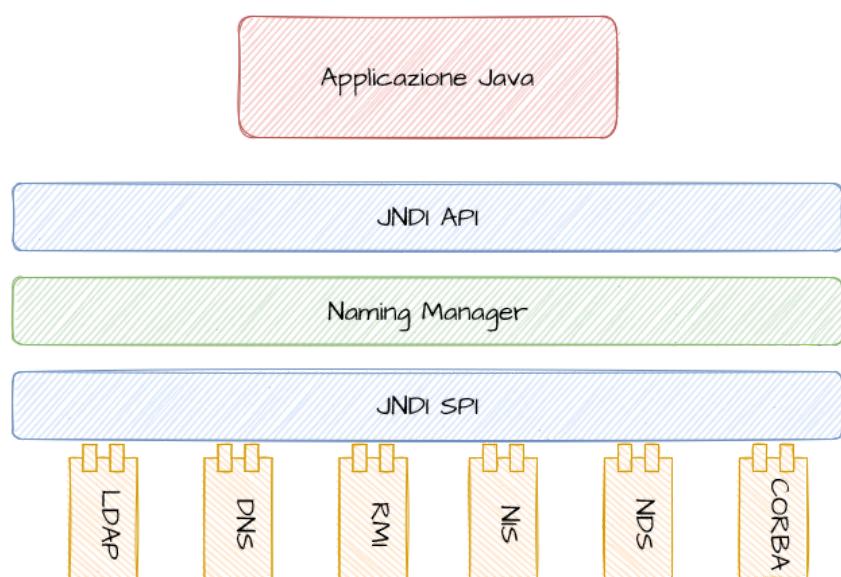
- `@Retention(RetentionPolicy.RUNTIME)`

-
- www.google.com
 -
 -

Rete Locale



-
-



Context

Context

InitialContext

- bind

```
void bind(String stringName, Object object)
```

- rebind

```
void rebind(String stringName, Object object)
```

- lookup

```
Object lookup(String stringName)
```

- unbind

```
void unbind(String stringName)
```

- rename

```
void rename(String stringOldName, String stringNewName)
```

- listBindings

```
NamingEnumeration listBindings(String stringName)
```

InitialContext

DirContext

Context

Context

- bind

```
void bind(String stringName, Object object, Attributes attributes)
```

- rebind

```
void rebind(String stringName, Object object, Attributes attributes)
```

- createSubcontext

```
DirContext createSubcontext(String stringName, Attributes attributes)
```

- getAttributes

```
Attributes getAttributes(String stringName)
```

- getAttributes

```
Attributes getAttributes(String stringName, String [] rgstringAttributeNames)
```

- modifyAttributes

ADD_ATTRIBUTE

REPLACE_ATTRIBUTE REMOVE_ATTRIBUTE

```
void modifyAttributes(String stringName, int nOperation, Attributes attributes)
```

- modifyAttributes

ADD_ATTRIBUTE

REPLACE_ATTRIBUTE REMOVE_ATTRIBUTE

```
void modifyAttributes(String stringName, ModificationItem [] rgmodificationitem)
```

-

Hashtable

```
Hashtable hashtableEnvironment = new Hashtable();
hashtableEnvironment.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
```

- ```
hashtableEnvironment.put(Context.PROVIDER_URL, "ldap://localhost:389/dc=etcee,dc=com");
hashtableEnvironment.put(Context.SECURITY_PRINCIPAL, "name");
hashtableEnvironment.put(Context.SECURITY_CREDENTIALS, "password");
```

#### InitialContext

```
Context context = new InitialContext(hashtableEnvironment);
```

#### InitialDirContext

```
DirContext context = new InitialDirContext(hashtableEnvironment);
```

- 
- 
- 

#### lookup

lookup

- - java.naming.provider.url    java.naming.factory.initial    java.naming
  - java.naming.ldap    java.naming.service
  - java.naming.security.sasl    java.naming.feature
  - com.sun.jndi.ldap.trace.ber
- - InitialContext    HashTable
  - jndi.properties
-

•

---

Capitolo 2

Capitolo 6

@Stateless    @Stateful    @MessageDriven

```
@Remote
public interface Payroll {
 public void setTaxDeductions(int empId, int deductions);
}
```

@Remote    @Local    @WebService

@Remote

@Local

Capitolo 7

```
public interface Payroll {
 public void setTaxDeductions(int empId, int deductions);
}
```

```
// interfaccia locale di EJBHome
public interface PayrollHome extends javax.ejb.EJBLocalHome {
 public Payroll create() throws CreateException;
}
```

```
// interfaccia locale di EJBObject
public interface Payroll extends javax.ejb.EJBLocalObject {
 public void setTaxDeductions(int empId, int deductions);
}
```

```
@Stateless
public class PayrollBean implements Payroll {

 public void setTaxDeductions(int empId, int deductions) {
 ...
 }
}
```

```
public class PayrollBean implements javax.ejb.SessionBean {

 SessionContext ctxt;

 public void setSessionContext(SessionContext ctxt) {
 this.ctxt = ctxt;
 }

 public void ejbCreate() {...}
 public void ejbActivate() {...}
 public void ejbPassivate() {...}
 public void ejbRemove() {...}

 public void setTaxDeductions(int empId, int deductions) {
 ...
 }
}
```

jms.MessageListener

@MessageDriven

```
@MessageDriven
public class PayrollMDB implements javax.jms.MessageListener {

 public void onMessage(Message msg) {
 ...
 }
}
```

```
@EJB
ShoppingCart myCart;
```

...

```
Collection widgets = myCart.startToShop("widgets");
```

...

```
Context initialContext = new InitialContext();
ShoppingCartHome myCartHome = (ShoppingCartHome) initialContext.lookup("java:comp/env/ejb/cart");
ShoppingCart myCart = myCartHome.create();
// utilizzo del bean
Collection widgets = myCart.startToShop("widgets")
```

...

```
// necessario anche il codice per gestire esplicitamente
// l'eccezione javax.ejb.CreateException
```

- @EJB

- @PersistenceContext @PersistenceUnit  
Capitolo 6
- @Resource

@Resource

@Resource

- name name

○

○

- type

○

@Resource

@Resource

- authenticationType  
CONTAINER APPLICATION
- shareable
- mappedName

```
public class SomeClass {
 @Resource
 private javax.sql.DataSource myDB;
}
```

```
public class SomeClass {
 private javax.sql.DataSource myDB;

 ...

 @Resource
 private void setmyDB(javax.sql.DataSource ds) {
 myDB = ds;
 }

 ...
}
```

```
@Resource(name="myMessageQueue", type="javax.jms.ConnectionFactory")
public class SomeMessageBean { ... }
```

name type  
@Resource

- 
- 

@Resources

```
@Resources({
 @Resource(name="myMessageQueue", type="javax.jms.ConnectionFactory"),
 @Resource(name="myMailSession", type="javax.mail.Session")
})
public class SomeMessageBean { ... }
```

// Vista cliente da EJB 3.X di un bean EJB 2.X

```
@EJB
ShoppingCartHome cartHome;

Cart cart = cartHome.create();
cart.addItem(...);
cart.remove();
```

EJBHome

// Vista cliente da EJB 2.X di un bean conforme a EJB 3.X

```
Context initialContext = new InitialContext();
ShoppingCartHome myCartHome = (ShoppingCartHome) initialContext.lookup("java:comp/env/ejb/cart");
```

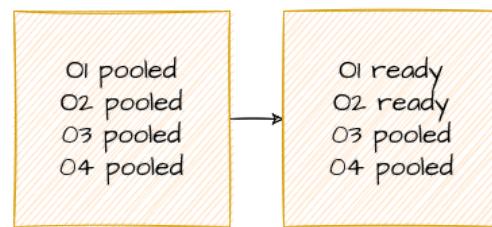
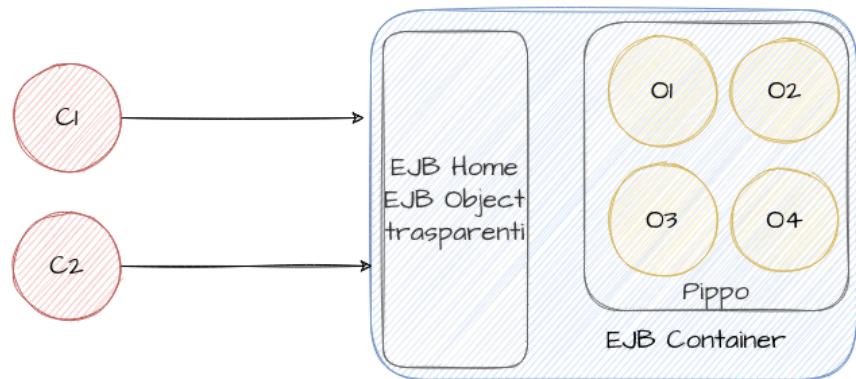
```
ShoppingCart cart = myCartHome.create();
cart.addItem(...);
cart.remove();
```

EJBHome

EJBObject

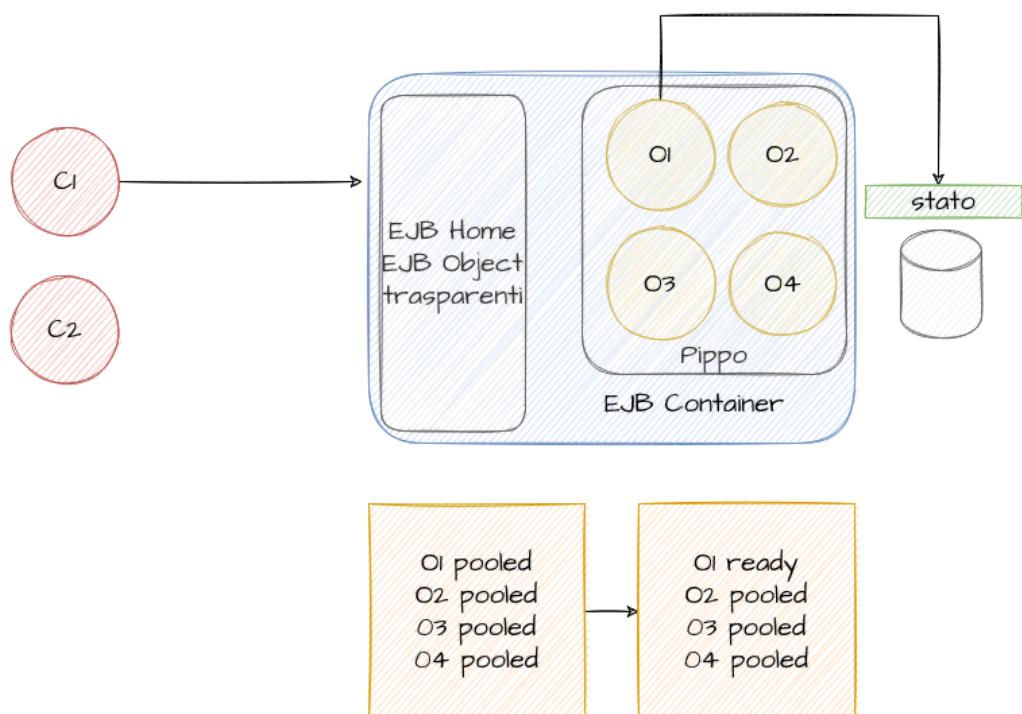
Capitolo 2

- 
- 
- 
-



- Pippo
- 
- 
- C1
- 
- 

C1



@javax.ejb.PostActivate

`@javax.ejb.PrePassivate`

- 
- 
- 
- 

`commit`

`rollback`

| time | T1      | T2     |
|------|---------|--------|
| 1    | R(x)    |        |
| 2    | W(x)    |        |
| ...  |         |        |
| 1000 | R(x500) |        |
| 1001 | commit  |        |
| 1002 |         | R(y)   |
| 1003 |         | W(y)   |
| 1004 |         | commit |

$$\begin{aligned}
 \text{Tempo medio di risposta} &= \\
 (1001 + (1004-1)) / 2 &= \\
 &= 1002
 \end{aligned}$$

| time | T1      | T2     |
|------|---------|--------|
| 1    | R(x)    |        |
| 2    |         | R(y)   |
| 3    |         | W(y)   |
| 4    |         | commit |
| 5    | W(x)    |        |
| ...  |         |        |
| 1003 | R(x500) |        |
| 1004 | commit  |        |

$$\begin{aligned}
 \text{Tempo medio di risposta} &= \\
 (1004 + 3) / 2 &= \\
 &= 503.5
 \end{aligned}$$

•  
•  
•  
•

X X X  
T2 T1

| T1     | X | T2     |
|--------|---|--------|
| R(x)   | I |        |
| X=X-I  | I |        |
|        | I | R(x)   |
|        | I | X=X-I  |
| w(x)   | O |        |
| commit | O |        |
|        | O | w(x)   |
|        | O | commit |

T2

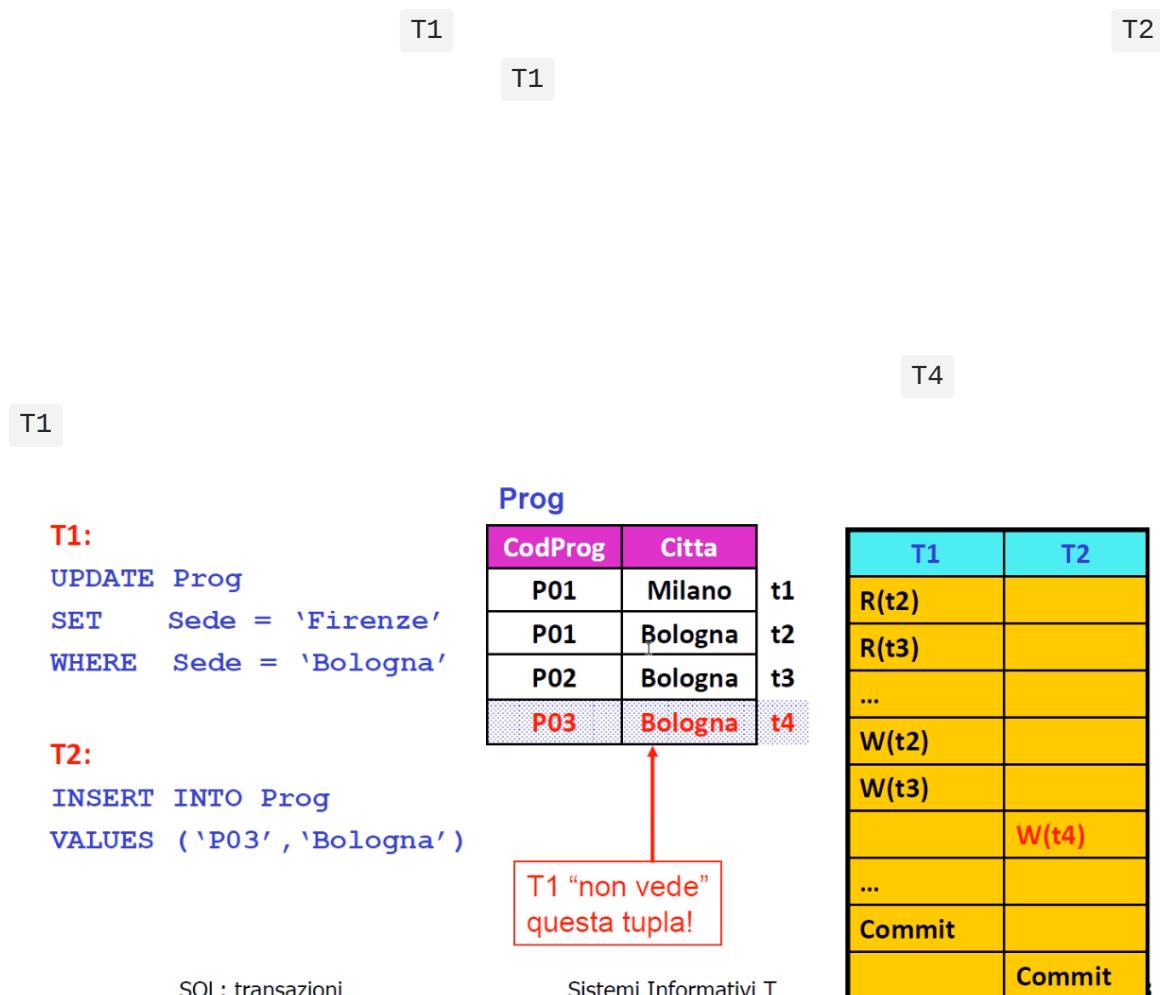
X

T2

T1

| T1       | X | T2     |
|----------|---|--------|
| R(x)     | O |        |
| X=X+1    | O |        |
| W(x)     | I |        |
|          | I | R(x)   |
| rollback | O |        |
|          | O | ...    |
|          | O | ...    |
|          | O | commit |

| T1     | X | T2     |
|--------|---|--------|
| R(x)   | O |        |
|        | O | R(x)   |
|        | I | X=X+I  |
|        | I | W(x)   |
|        | I | commit |
| R(x)   | I |        |
| ...    | I |        |
| commit | I |        |

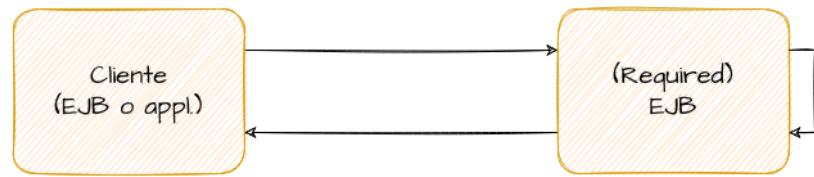


- `ISOLATION_READ_UNCOMMITTED`
- `ISOLATION_READ_COMMITTED`
- `ISOLATION_REPEATABLE_READ`
- `ISOLATION_SERIALIZABLE`
- `@TransactionManagement`
  - `CONTAINER`
  - `BEAN`
- `commit`
- `commit`    `rollback`
  - `java.sql.Connection`
  - `javax.transaction.UserTransaction`
  - `javax.jms.Session`
- `@TransactionAttribute`
  - `BeanA`
  - `BeanB`
- `BeanA`
  - `BeanB`
- `BeanB`
  - `BeanA`

**REQUIRED**

**REQUIRES\_NEW**    **MANDATORY**    **NOT\_SUPPORTED**    **SUPPORTS**    **NEVER**

- REQUIRED



Leggenda

|                     |
|---------------------|
| transazionale<br>T1 |
| transazionale<br>T2 |
| no<br>transazionale |

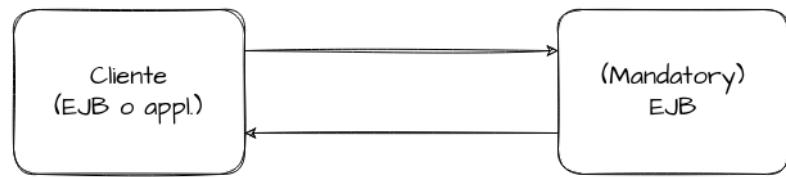
- REQUIRES\_NEW



Leggenda

|                     |
|---------------------|
| transazionale<br>T1 |
| transazionale<br>T2 |
| no<br>transazionale |

- MANDATORY



Leggenda



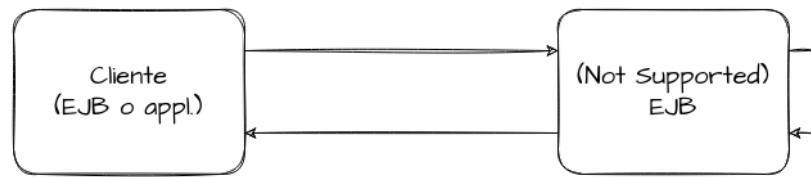
- NOT\_SUPPORTED



Leggenda



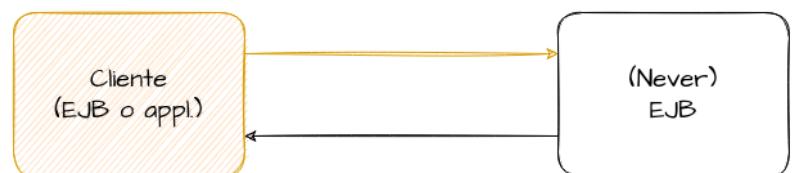
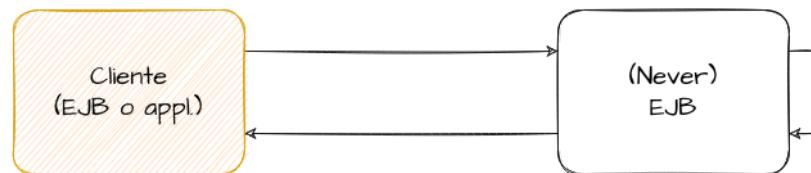
- SUPPORTS



Leggenda



- NEVER



Leggenda



rollback

- **rollback**
- **setRollBackOnly**    EJBContext    EJBContext



```

import static TransactionAttributeType.*;

@Stateless
@TransactionAttribute(NOT_SUPPORTED)
public class TravelAgentBean implements TravelAgentRemote {

 public void setCustomer(Customer cust) { ... }

 @TransactionAttribute(REQUIRED)
 public TicketDO bookPassage(CreditCard card, double price) { ... }
}

```

```

// EJB 3.0: Bean-managed transaction
@TransactionManagement(BEAN)
@Stateless
public class PayrollBean implements Payroll {

 @Resource UserTransaction utx;

```

```
@PersistenceContext EntityManager payrollMgr;
public void setTaxDeductions(int empId, int deductions) {

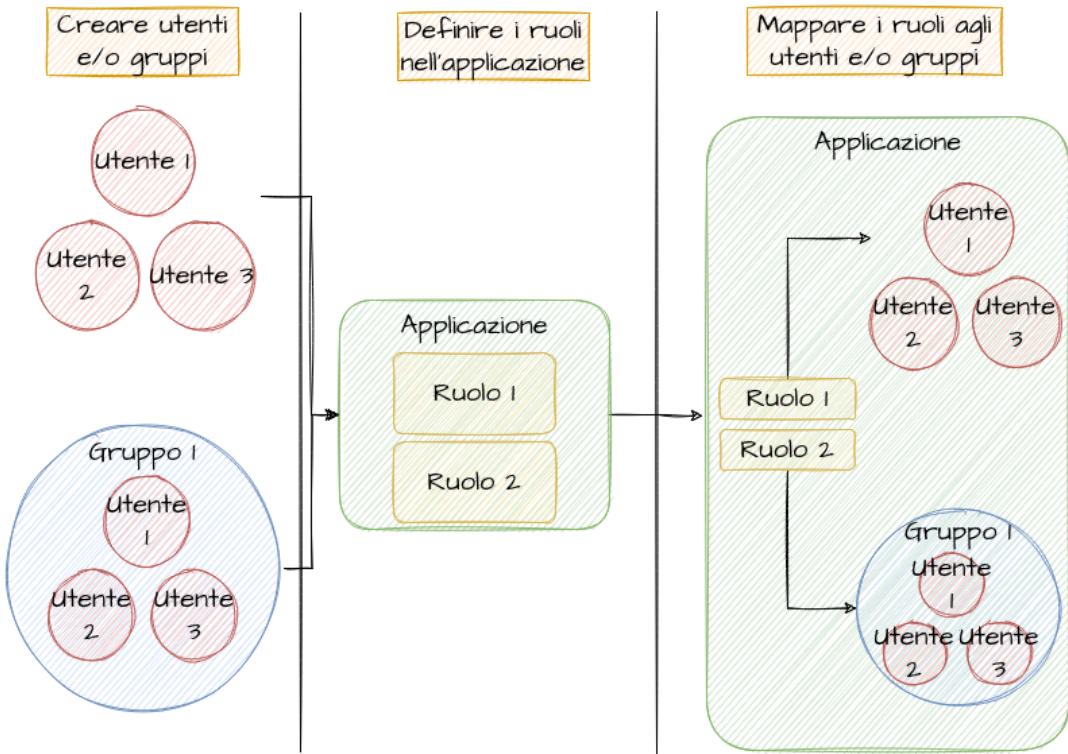
 utx.begin();
 payrollMgr.find(Employee.class, empId).setDeductions(deductions);
 utx.commit();
}

...
}
```

@Resource

Capitolo 6

Capitolo 7



- `@RolesAllowed`
- `@PermitAll`
- `@DenyAll`
- `@RunAs`

```

@Stateless
public PayrollBean implements Payroll {

 public void setBenefitsDeduction(int empId, double deduction) { ... }
 public double getBenefitsDeduction(int empId) { ... }
 public double getSalary(int empid) { ... }

 // setting del salario ha un accesso più restrittivo
 @RolesAllowed("HR_PayrollAdministrator")
 public void setSalary(int empId, double salary) { ... }
}

```

- `@Interceptors`
- `@AroundInvoke`

```

//classe Profiler
public class Profiler {

 @AroundInvoke
 public Object profile() throws Exception {
 ...
 }
}

...
//classe intercettata

```

```
@Interceptors(Profiler.class)
public Object m1(...) throws ... { ... }
```

- - 
  -
-

```
public SampleDAO samplelookup(String id) {

 Connection c = null;
 PreparedStatement ps = null;
 ResultSet rs = null;
 SampleDAO dao = null;
 try {
 c = getDataSource().getConnection();
 ps = c.prepareStatement("SELECT ...");
 ps.setString(1, id);
 rs = ps.executeQuery();
 if (rs.first()) {
 dao = new SampleDAO(id, rs.getString(2), rs.getString(2));
 }
 }
 catch (SQLException se) {
 throw new SampleDAORuntimeException(se);
 }
 finally {
 if (rs != null) try {rs.close();} catch (SQLException se) {}
 if (ps != null) try {ps.close();} catch (SQLException se) {}
 if (c != null) try {c.close();} catch (SQLException se) {}
 }

 return dao;
}
```

- 
- 
-

- `javafx.persistence.Entity`
- `public protected`
- `final`
- `getter`
- `Serializable`
- `private`
- `protected package-private`
- `javax.persistence.Transient`
- `protected Set<Purchase> purchases;`
- `getter setter getProperty() setProperty() isProperty()`
- `Customer`
- `firstName String`
- ```
public String getFirstName() {
    return name;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}
```
- `public Set<Purchase> getPurchases() {
 return purchases;
}`

```
}
```

```
javax.persistence.Id
```

```
@Entity  
public class Project {  
    @Id  
    private long id;  
  
    ...  
}
```

```
javax.persistence.EmbeddedId
```

```
javax.persistence.IdClass
```

```
@Entity @IdClass(ProjectId.class)  
public class Project {  
    @Id  
    private int departmentId;  
    @Id  
    private long projectId;  
  
    ...  
}  
  
public class ProjectId {  
    private int departmentId;  
    private long projectId;  
}
```

```
@IdClass
```

```
@Entity
public class Project {
    @EmbeddedId
    private ProjectId id;
    ...
}

@Embeddable
public class ProjectId {
    private int departmentId;
    private long projectId;
}
```

Project
ProjectId

- ```
• public
• hashCode() equals(Object other)
•
•
```

```
@Entity
public final class LineItemKey implements Serializable {

 public Integer orderId;
 public int itemId;

 public LineItemKey() {
 }

 public LineItemKey(Integer orderId, int itemId) {
 this.orderId = orderId;
 this.itemId = itemId;
 }

 public boolean equals(Object otherOb) {

 if (this == otherOb) {
 return true;
 }

 if (!(otherOb instanceof LineItemKey)) {
 return false;
 }
 }
}
```

```

 }

 LineItemKey other = (LineItemKey) other0b;
 return ((orderId==null ? other.orderId==null : orderId.equals(other.orderId))
 && (itemId == other.itemId));
}

public int hashCode() {
 return ((orderId==null? 0 : orderId.hashCode())^((int) itemId));
}

public String toString() {
 return "" + orderId + "-" + itemId;
}

}

```

```

@Entity
public abstract class Employee {

 @Id
 protected Integer employeeId;

 ...

}

@Entity
public class FullTimeEmployee extends Employee {

 protected Integer salary;

 ...

}

@Entity
public class PartTimeEmployee extends Employee {

```

```
 protected Float hourlyWage;
}
```

```
 @MappedSuperclass
```

```
 @MappedSuperclass
```

```
@MappedSuperclass
```

```
public class Employee {
```

```
 @Id
```

```
 protected Integer employeeId;
```

```
 ...
```

```
}
```

```
@Entity
```

```
public class FullTimeEmployee extends Employee {
```

```
 protected Integer salary;
```

```
 ...
```

```
}
```

```
@Entity
```

```
public class PartTimeEmployee extends Employee {
```

```
 protected Float hourlyWage;
```

```
 ...
```

```
}
```

```
javax.persistence.Inheritance
```

- `InheritanceType.SINGLE_TABLE`

NULL

- InheritanceType.TABLE\_PER\_CLASS

- InheritanceType.JOINED

InheritanceType.SINGLE\_TABLE

@Inheritance

TABLE\_PER\_CLASS

JOINED

- javax.persistence.OneToOne

- javax.persistence.OneToMany

- javax.persistence.ManyToOne

- javax.persistence.ManyToMany

```
@OneToMany
public Set<Purchase> getPurchases() {
 return purchases;
}
```

Ordine

Oggetto

@mappedBy

```
@OneToMany(cascade=REMOVE, mappedBy="customer")
public Set<Order> getOrders() {
 return orders;
}
```

```
@PersistenceContext
```

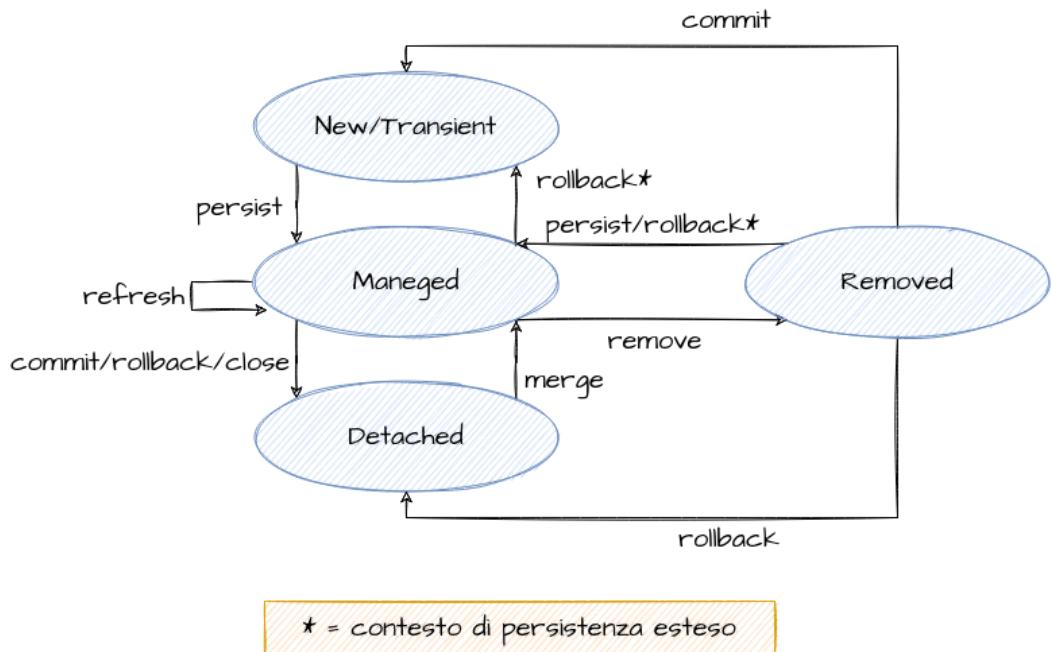
```
@PersistenceContext
EntityManager em;
```

```
createEntityManager()
```

```
javax.persistence.EntityManagerFactory
```

```
@PersistenceUnit
```

```
EntityManagerFactory emf;
EntityManager em = emf.createEntityManager();
```



- 
- 
- 
- 

```

persist()
cascade=PERSIST persist()
 cascade=ALL

```

- `persist()`
- `IllegalArgumentException`

```
persist()
```

```

@PersistenceContext
EntityManager em;

...
public LineItem createLineItem(Order order, Product product) {
 LineItem li = new LineItem(order, product, quantity);
}

```

```
order.getLineItems().add(li);
em.persist(li);

return li;
}

// persist propagata a tutte le Entity in relazione con
// cascade element = ALL o PERSIST

@OneToMany(cascade=ALL, mappedBy="order")
public Collection<LineItem> getLineItems() {
 return lineItems;
}
```

- **remove()**
  - **remove()**  
**IllegalArgumentException**
  - **remove()**
- remove()**  
cascade=REMOVE    cascade=ALL

```
public void removeOrder(Integer orderId) {
 try {
 Order order = em.find(Order.class, orderId);
 em.remove(order);
 }
}
```

**refresh()**

**commit**

**flush()**

**commit**

```
persistence.xml
```

```
<persistence>
 <persistence-unit name="OrderManagement">
 <description> Questa unità gestisce ordini e clienti</description>
 <jta-data-source>jdbc/MyOrderDB</jta-data-source>
 <jar-file>MyOrderApp.jar</jar-file>
 <class>com.widgets.Order</class>
 <class>com.widgets.Customer</class>
 </persistence-unit>
</persistence>
```

OrderManagement

jdbc/MyOrderDB

Order Customer

jar-file class

jta-data-source

createQuery() createNamedQuery()

createQuery()

```
public List findWithName(String name) {
 return em.createQuery(
 "SELECT c FROM Customer c WHERE c.name LIKE :custName")
```

```
.setParameter("custName", name)
.setParameter(10)
.getResultList();
}
```

```
createNamedQuery()
```

```
@NamedQuery
```

```
@NamedQuery(
 name="findAllCustomersWithName",
 query="SELECT c FROM Customer c WHERE c.name LIKE :custName")
```

```
@PersistenceContext
```

```
public EntityManager em;
```

```
...
```

```
customers = em.createNamedQuery("findAllCustomersWithName")
 .setParameter("custName", "Smith")
 .getResultList();
```

```
setParameter()
```

```
:
```

```
:custName
```

```
@OneToMany(cascade=ALL, mappedBy="owner", fetch=EAGER)
```

```
@OneToMany(cascade=ALL, mappedBy="owner", fetch=LAZY)
```

persist

```
@Entity
@EntityListener(com.acme.AlertMonitor.class)
public class AccountBean implements Account {

 Long accountId;
 Integer balance;
 boolean preferred;
 @Transient ClassA obj1;

 public Long getAccountId() { ... }
 public Integer getBalance() { ... }
 public boolean isPreferred() { ... }
 public void deposit(Integer amount) { ... }
 public Integer withdraw(Integer amount) throws NSFException {... }

 @PrePersist
 public void validateCreate() {

 if (getBalance() < MIN_REQUIRED_BALANCE)
 throw new AccountException("Insufficient balance to open an account");
 }

 @PostLoad
 public void adjustPreferredStatus() {
 preferred = (getBalance() >= AccountManager.getPreferredStatusLevel());
 }
}
```

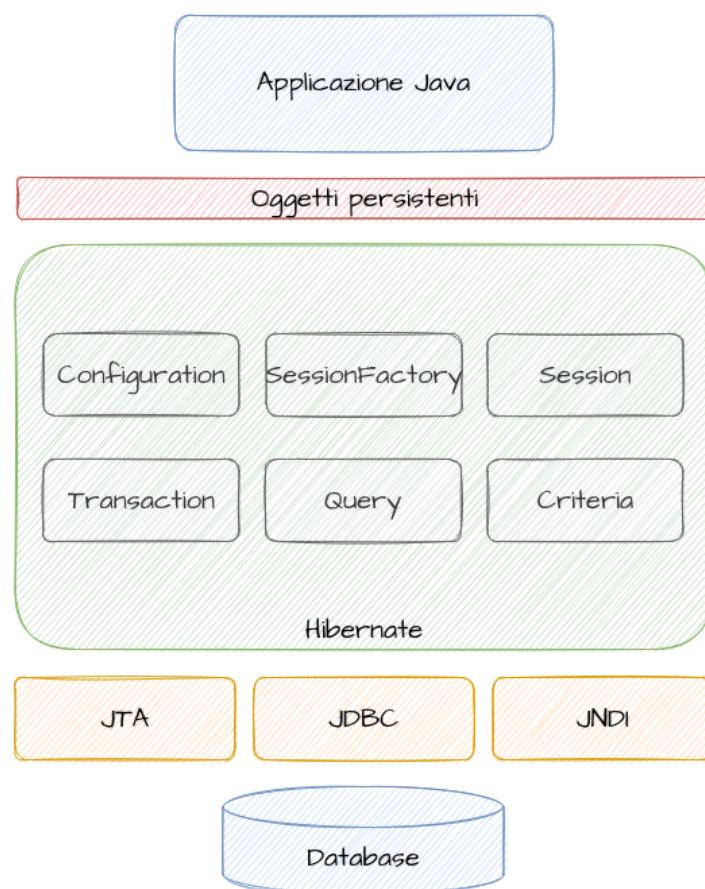
```
public class AlertMonitor {
 @PostPersist
 public void newAccountAlert(Account acct) {
 Alerts.sendMarketingInfo(acct.getAccountId(), acct.getBalance());
```

```
 }
}
```

AccountBean

@EntityListener(com.acme.AlertMonitor.class)

AlertMonitor



SessionFactory  
Session

EntityManagerFactory

SessionFactory

SessionFactory

Session

EntityManager

Transaction

Transaction

Transaction

•

•

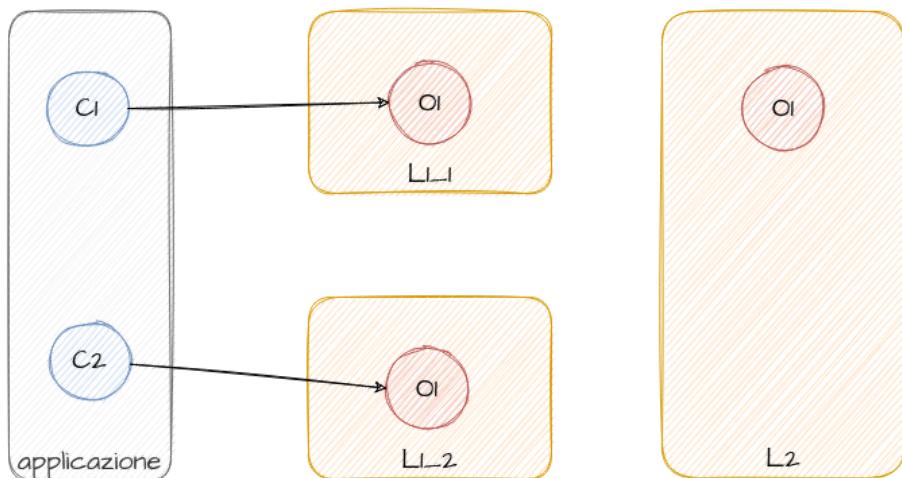
•

Session

Session

SessionFactory

SessionFactory



SessionFactory

Session

Session

$C_1$

01

$C_2$

SessionFactory

Session

$L_{1\_2}$

01

$C_1$

L2

01

$C_2$

01

$C_1$

@Version

```
@Entity
@Table(name = "orders")
public class Order {
 @Id private long id;
 @Version private int version;
 private String description;
 private String status;

 ...
}
```

```
update orders set description=?, status=?, version=? where id=? and
version=?
```

```
update orders set description=?, status=?, version=2 where id=? and
version=1
```

```
update orders set description=?, status=?, version=2 where id=? and
version=1
```

org.hibernate.StaleObjectStateException

## FetchMode

- `FetchMode.DEFAULT` `FetchMode`
- `FetchMode.JOIN`
- `FetchMode.SELECT`
- 
- 

```
// cerca gli oggetti persona tramite un oggetto di esempio
Criteria crit = sess.createCriteria(Person.class);
Person person = new Person();
person.setName("Shin");
Example exampleRestriction = Example.create(person);
crit.add(exampleRestriction);
List results = crit.list();
```

•

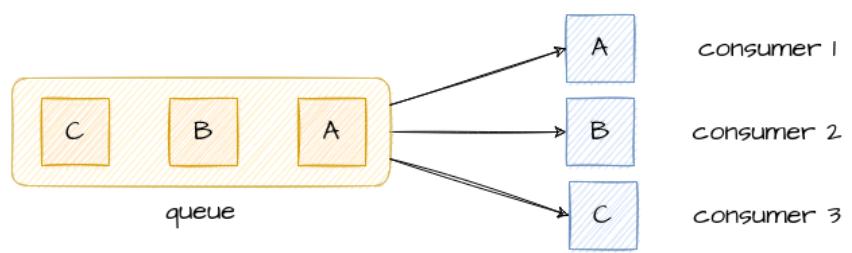
•

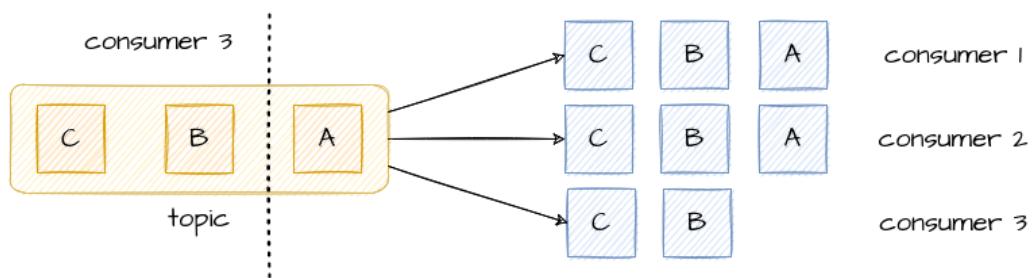
•

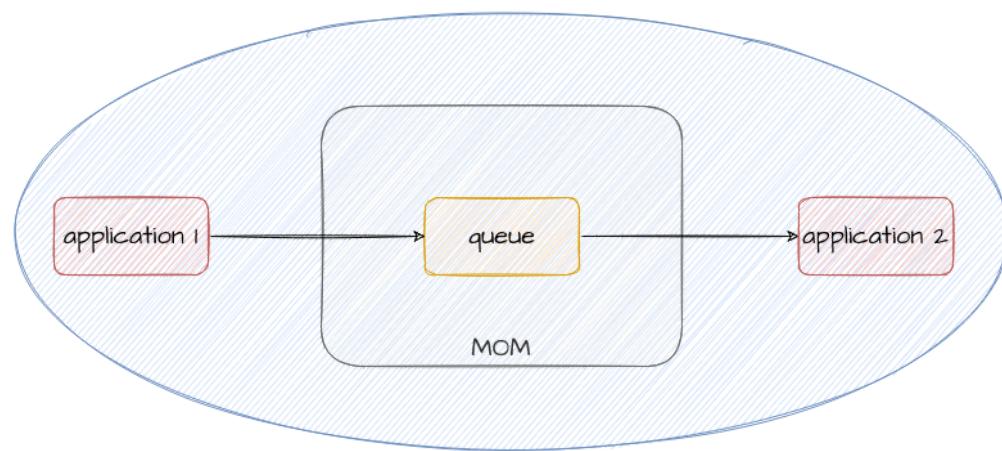
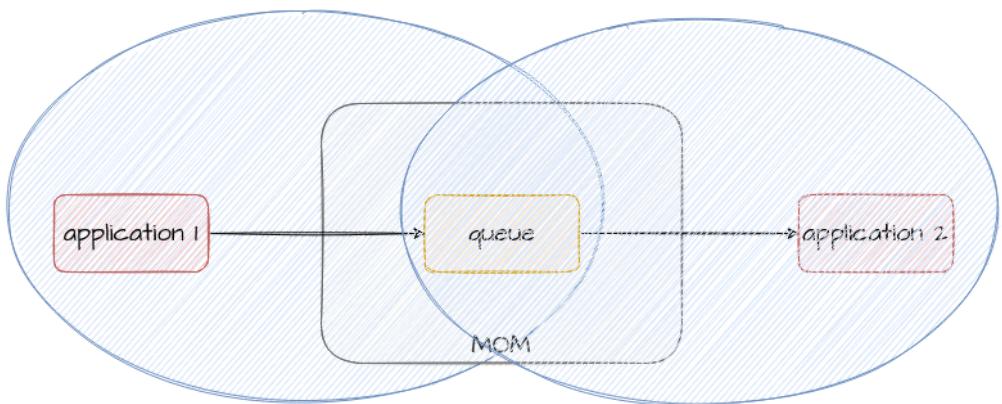
•

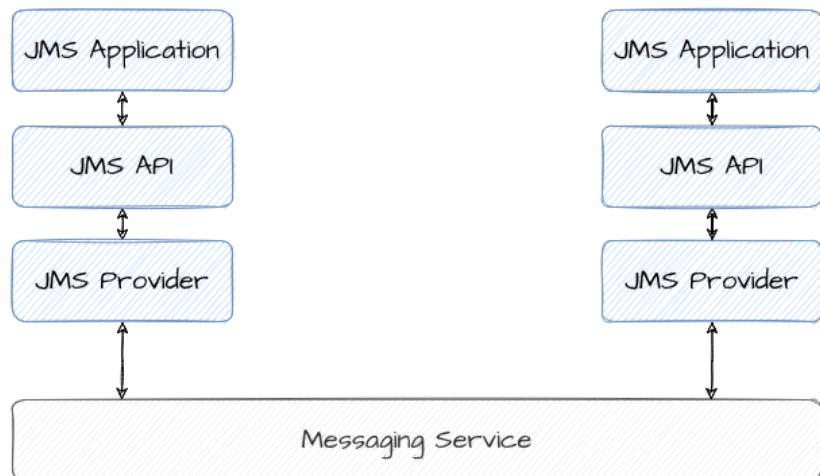
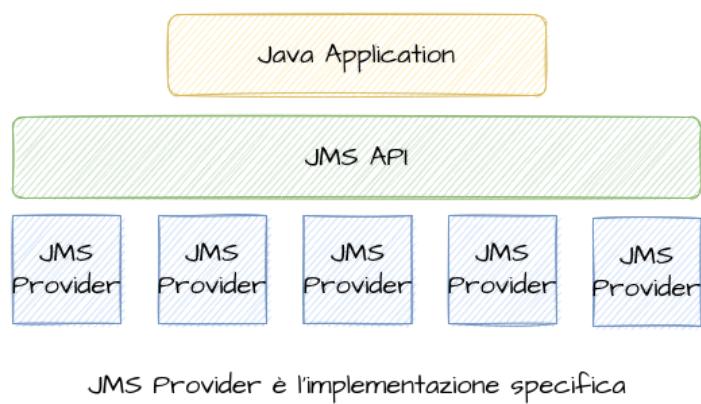
•

•









- 
- 
-

•

•

•

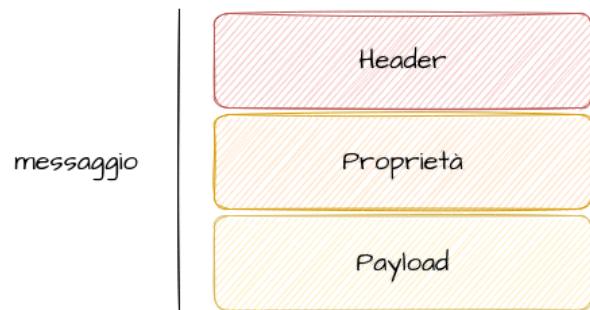
•

•

•

•

•



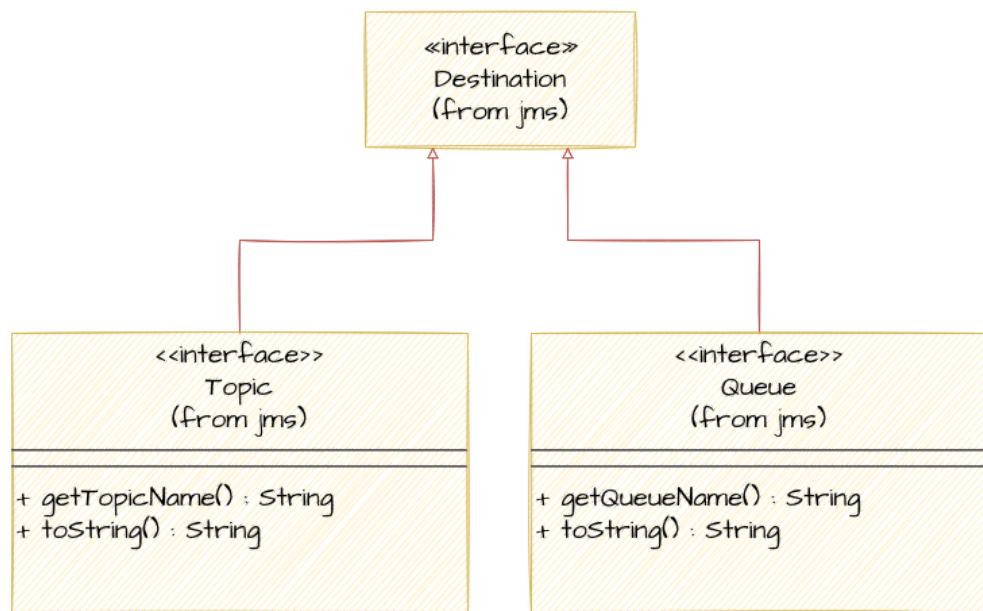
JMSDestination JMSDeliveryMode  
 JMSMessageID JMSTimeStamp JMSRedelivered JMSExpiration JMSPriority  
 JMSCorrelationID JMSReplyTo  
 JMSType

StreamMessage MapMessage TextMessage ObjectMessage  
 BytesMessage

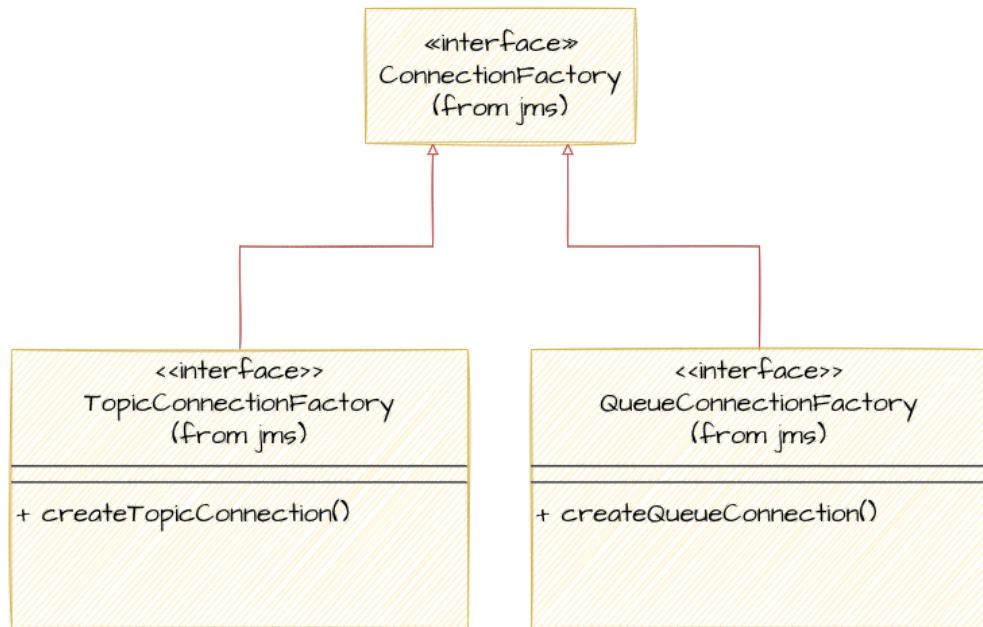
- StreamMessage
- MapMessage
- BytesMessage

Destination

Queue Topic

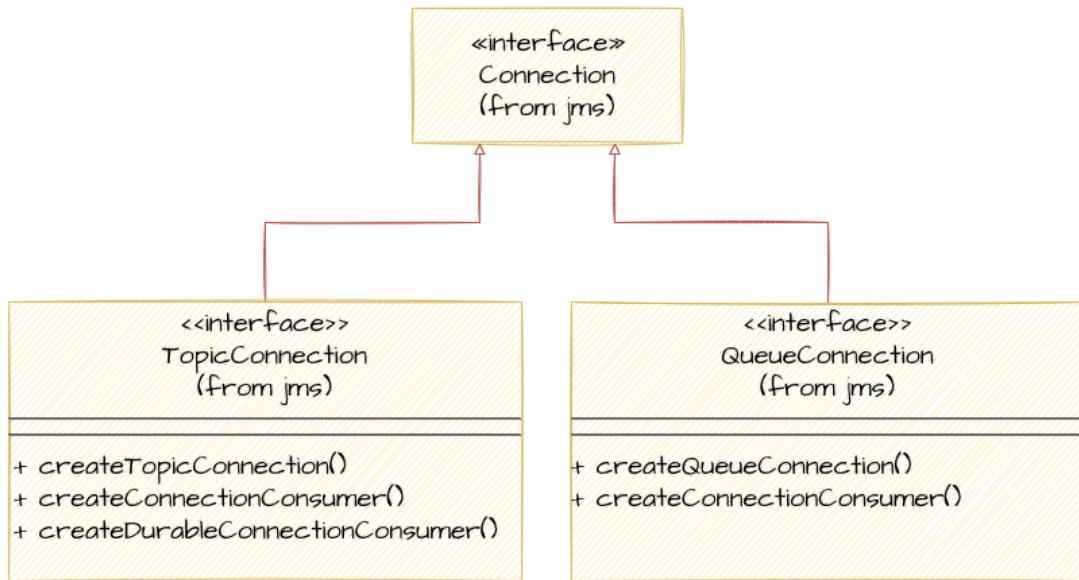


ConnectionFactory  
java.sql.DriverManager  
QueueConnectionFactory      TopicConnectionFactory



Connection

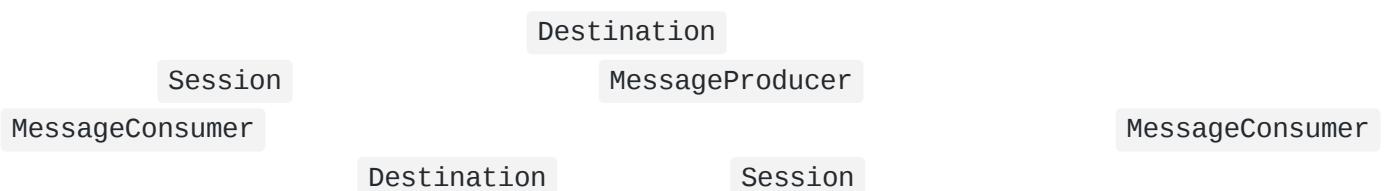
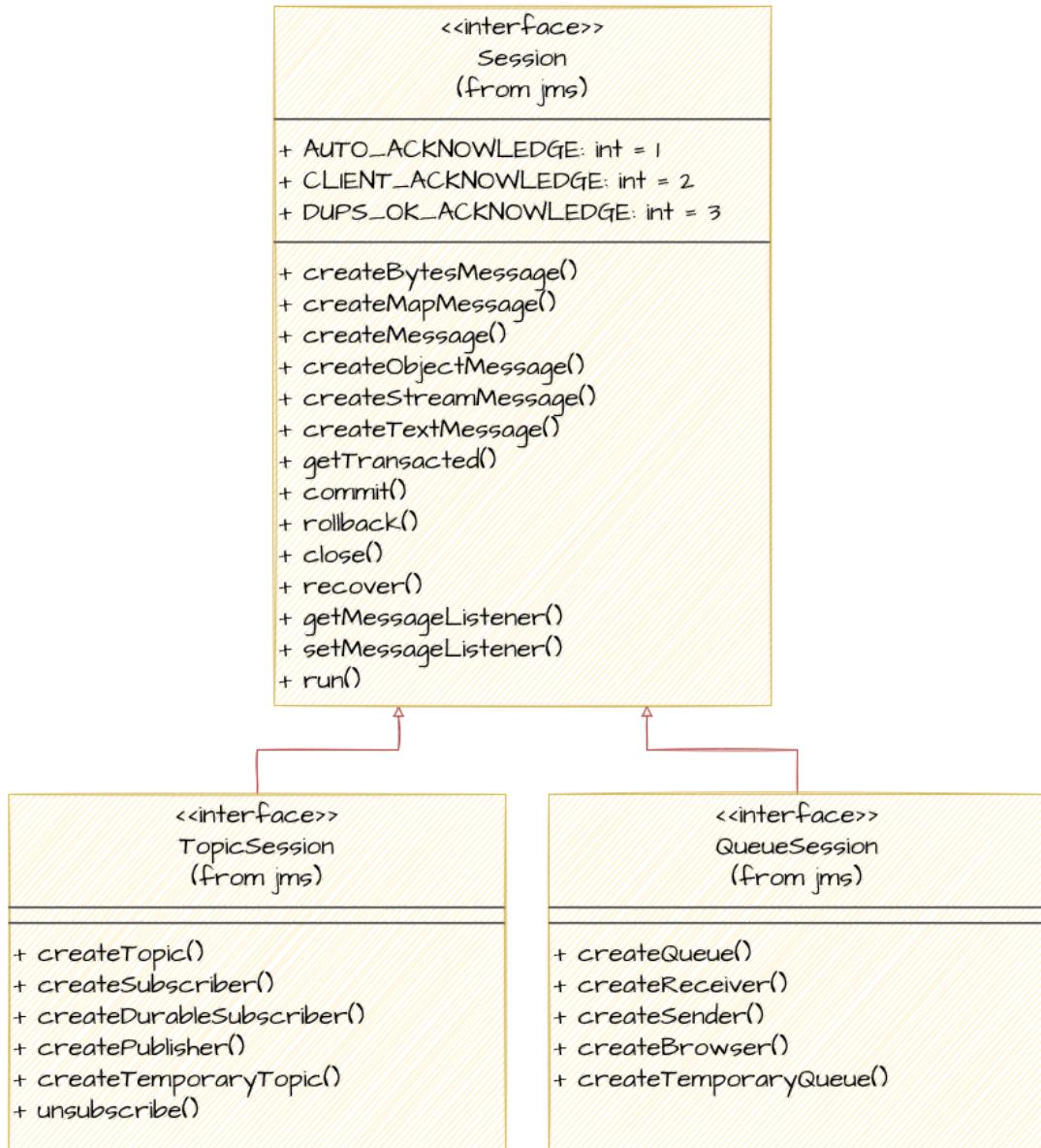
ConnectionFactory

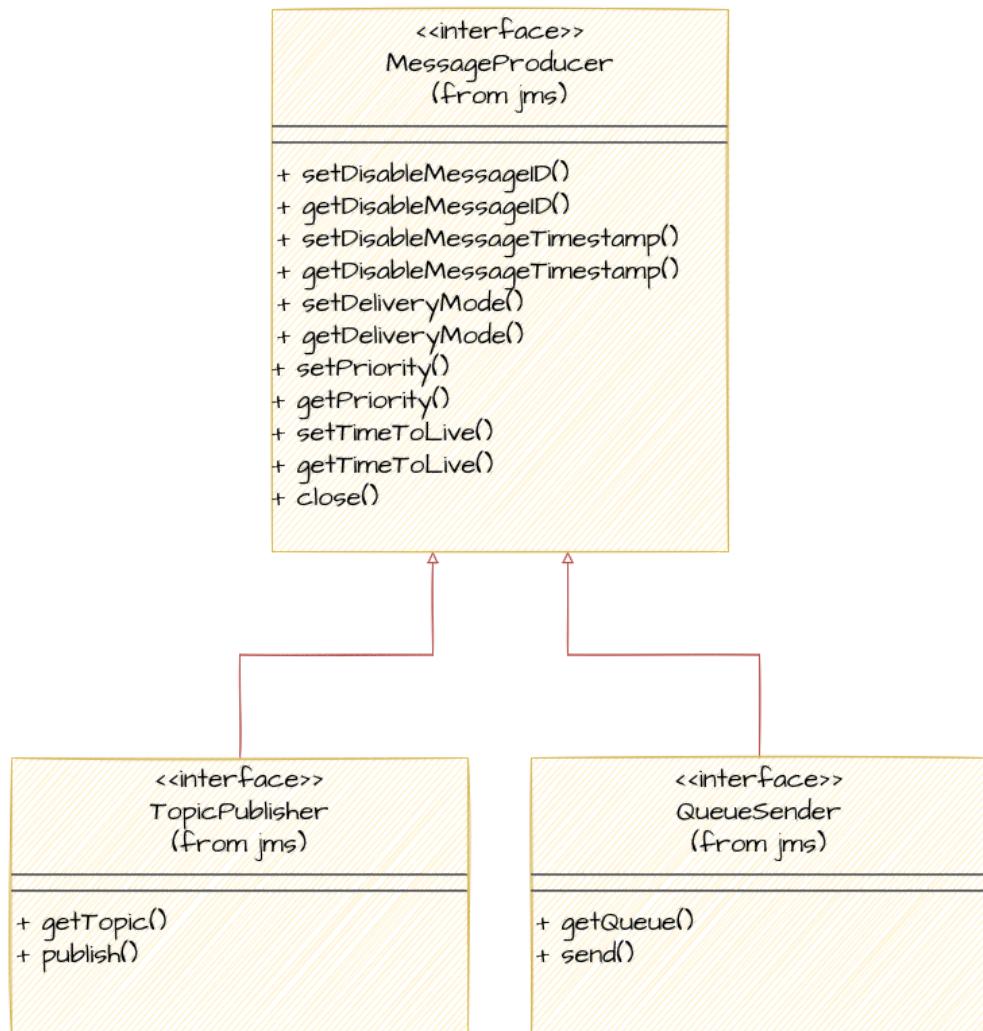


Session

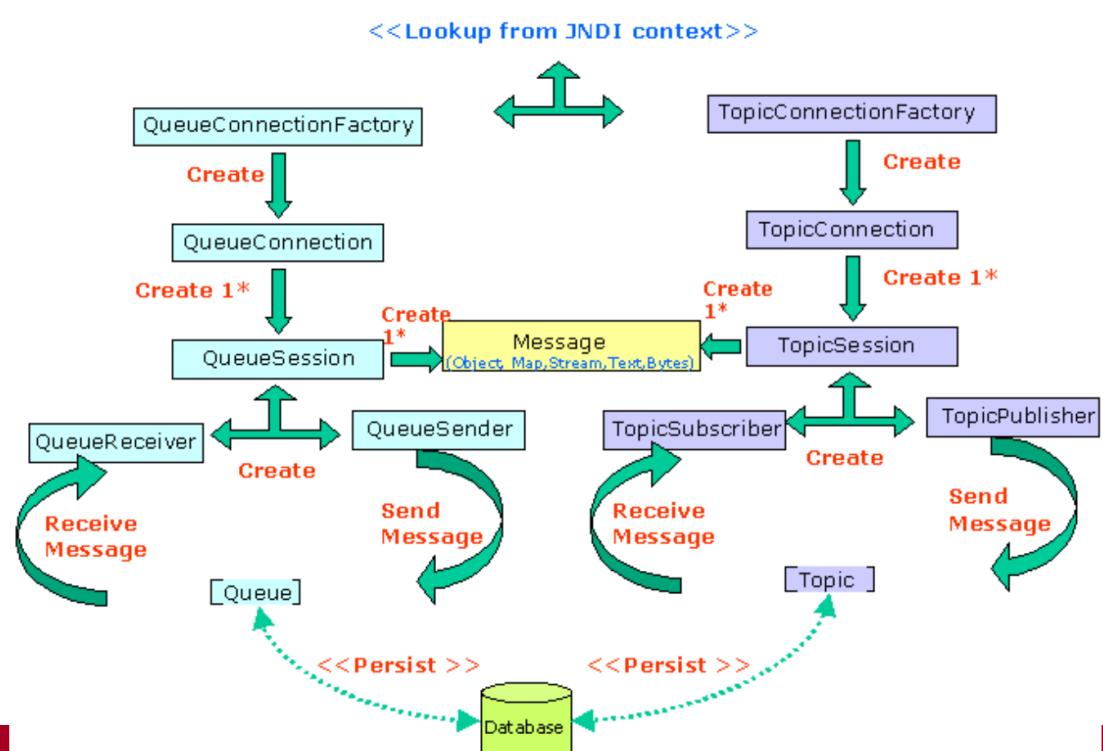
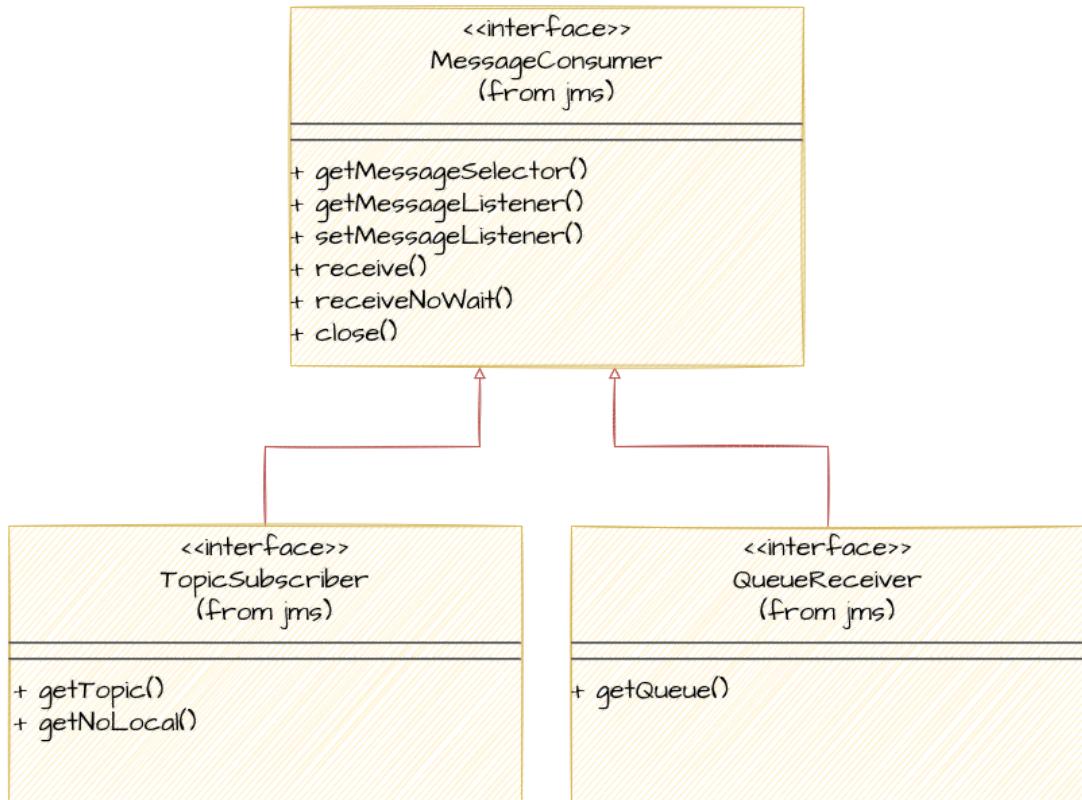
Connection

Connection





- `receive()`
  - `MessageListener`
    - `onMessage()`
- `MessageListener`



- 
- 

- 

ConnectionFactory

Destination

Topic

Queue

```
// Ottiene oggetto InitialContext
Context jndiContext = new InitialContext();

// Trova l'oggetto ConnectionFactory via JNDI
TopicConnectionFactory factory = (TopicConnectionFactory) jndiContext.lookup("MyTopicConnectionFactory");

// Trova l'oggetto Destination via JNDI
// (Topic o Queue)
Topic weatherTopic = (Topic) jndiContext.lookup("WeatherData");
```

- 

Connection

```
// Richiede la creazione di un oggetto Connection
// all'oggetto ConnectionFactory
TopicConnection topicConnection = factory.createTopicConnection();
```

- 

Session

```
// Crea un oggetto Session da Connection:
// primo parametro controlla transazionalità
// secondo specifica il tipo di ack
TopicSession session = topicConnection.createTopicSession(false, session.CLIENT_ACKNOWLEDGE);
```

- 

MessageProducer

TopicPublisher

QueueSender

```
// Richiede la creazione di un oggetto MessageProducer
// all'oggetto Session
// TopicPublisher per Pub/Sub
// QueueSender per Point-to-Point
TopicPublisher publisher = session.createPublisher(weatherTopic);
```

- **Connection**

```
// Avvia la Connection
// Fino a che la connessione non è avviata, il
// flusso dei messaggi non comincia: di solito
// Connection viene avviata prima dell'invocazione
// dei metodi per la trasmissione messaggi
topicConnection.start();
```

- 

```
// Creazione del messaggio
TextMessage message = session.createMessage();
message.setText("text:35 degrees");

// Invio del messaggio
publisher.publish(message);
```

- **Session** **Connection**

```
session.close();
topicConnection.close();
```

- **ConnectionFactory** **Destination** **Topic**

- Queue**

- **Connection**
  - **Session**
  - **MessageConsumer** **TopicSubscriber** **QueueReceiver**

```
// Crea oggetto Subscriber da Session
TopicSubscriber subscriber = session.createSubscriber(weatherTopic);
```

- **MessageListener**

```
// Crea oggetto MessageListener
WeatherListener myListener = new WeatherListener();

// Registra MessageListener per l'oggetto
// TopicSubscriber desiderato
subscriber.setMessageListener(myListener);
```

- **Connection**

- Session Connection

```
createSession()
```

- MessageConsumer.receive()    MessageListener.onMessage()  
return
- acknowledge()
-

AUTO\_ACKNOWLEDGE

CLIENT\_ACKNOWLEDGE

DUPS\_OK\_ACKNOWLEDGE

send()

send()

setDeliveryMode()

MessageProducer

```
// metodo dell'interfaccia MessageProducer
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
```

JMSPriority

TimeToLive

setPriority()

setTimeToLive()

MessageProducer

```
// metodi nell'interfaccia MessageProducer
producer.setTimeToLive(60000);
producer.setPriority(7);
```

Destination

MessageProducer

Session

MessageProducer

MessageProducer

- Session
- Session

Session

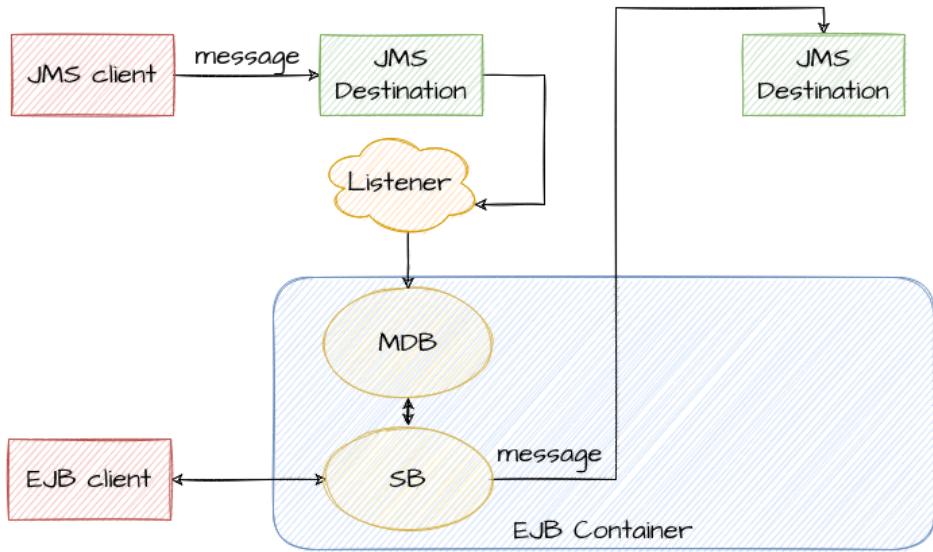
Session.commit()

Session.abort()

QueueConnection.createQueueSession(true, ...)

Session.commit()

Session.rollback()

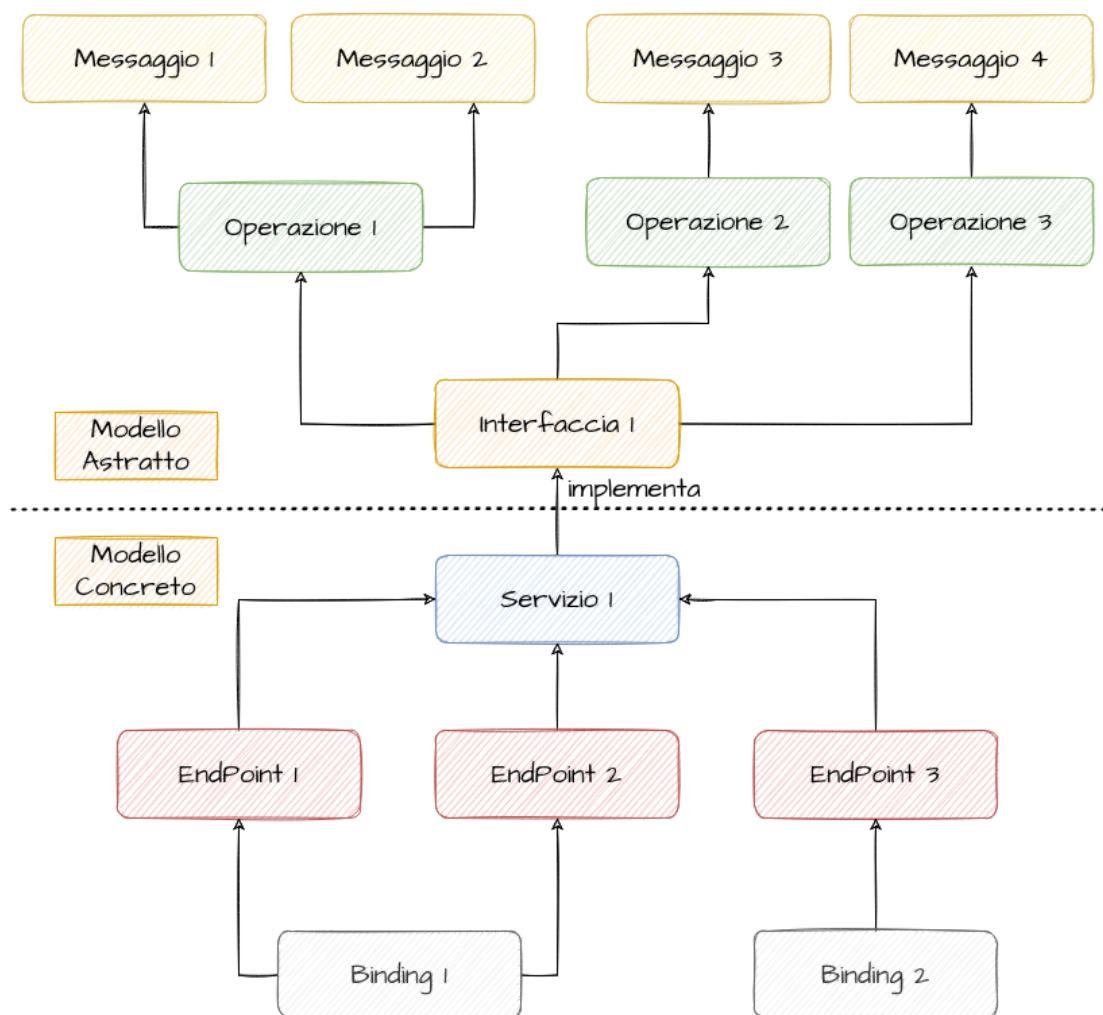


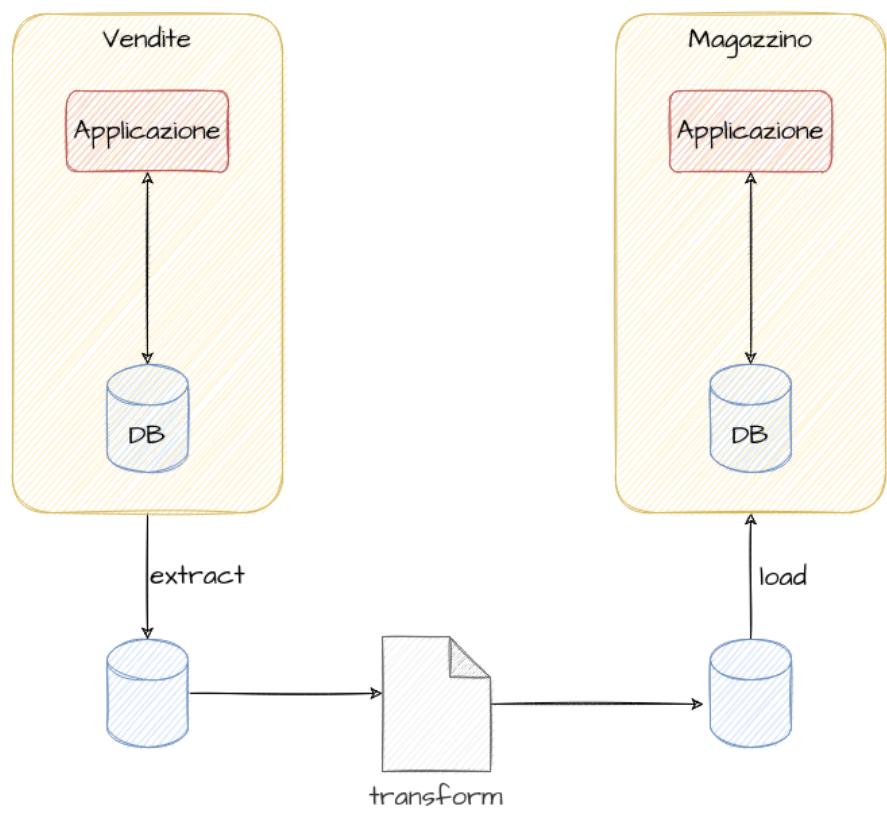
- 
- 
- 
- 
-

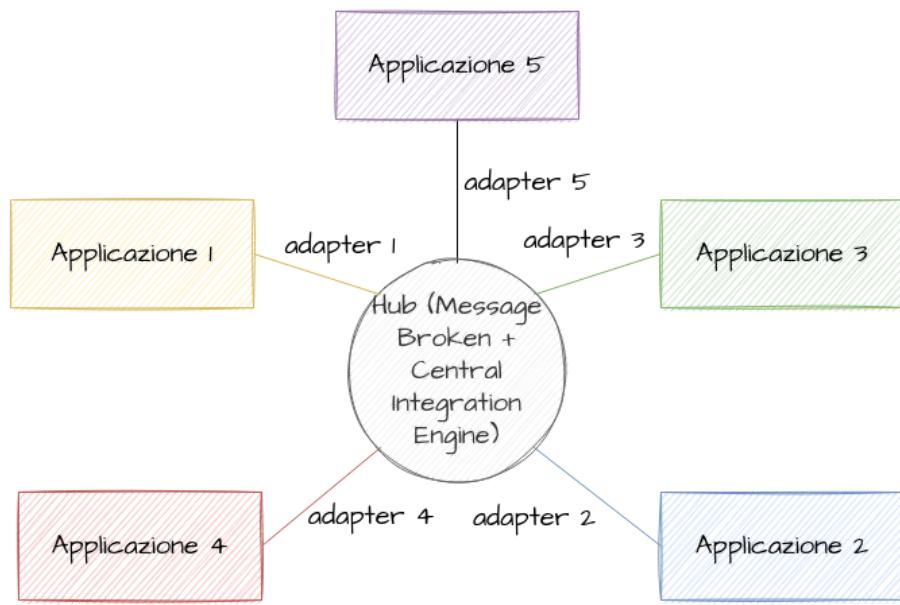
•  
•  
•  
•  
•

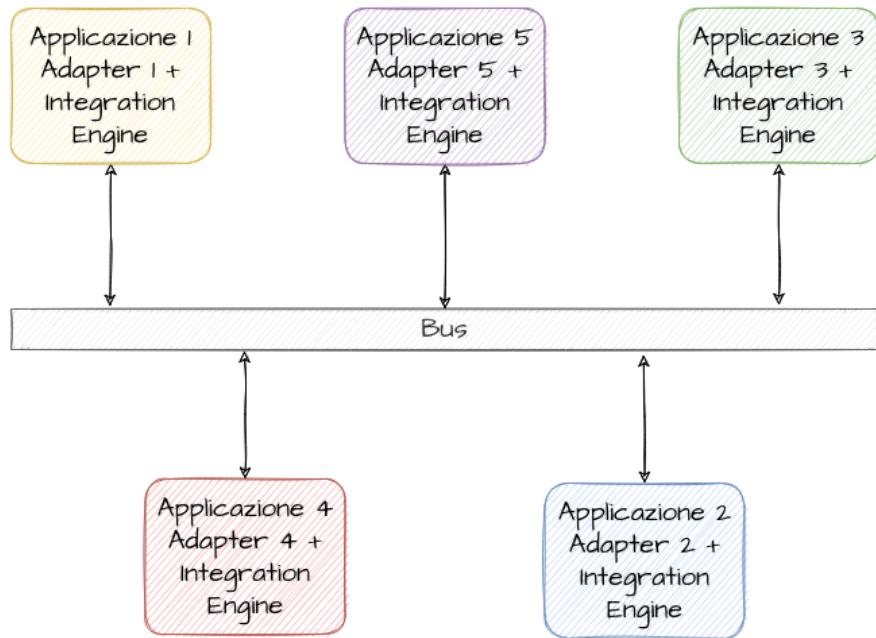
•  
•  
•  
•

•



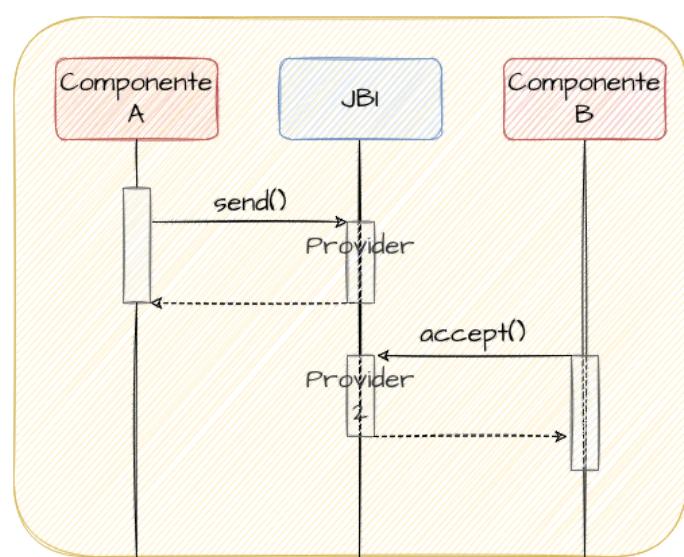






- 
- 
-

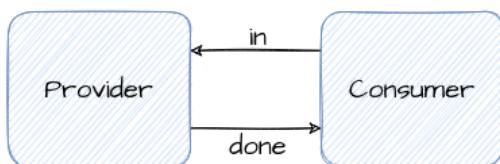
A      B



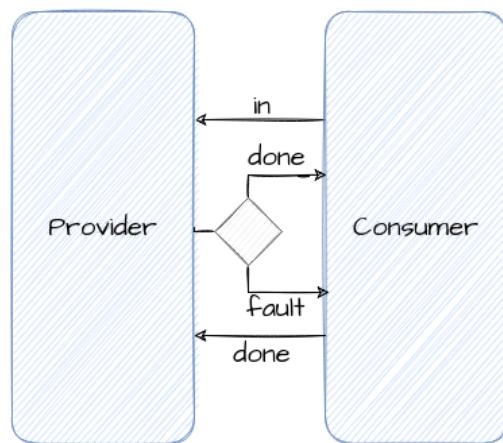
A

B

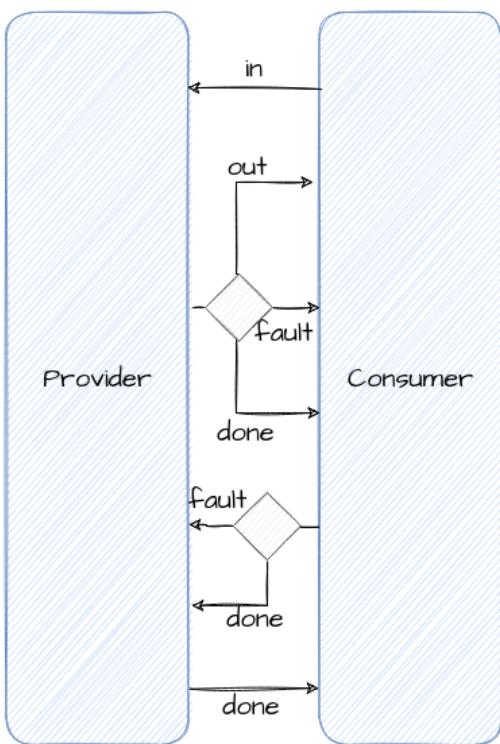
B



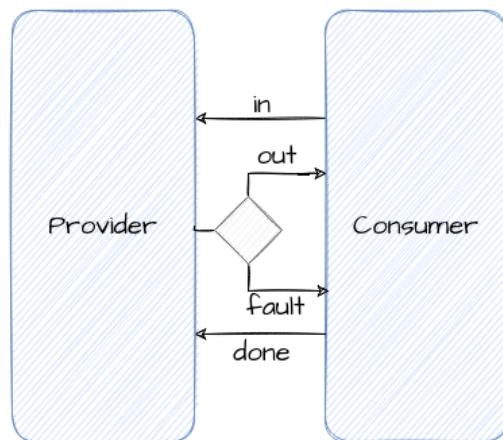
In-Only



Robust in-Only



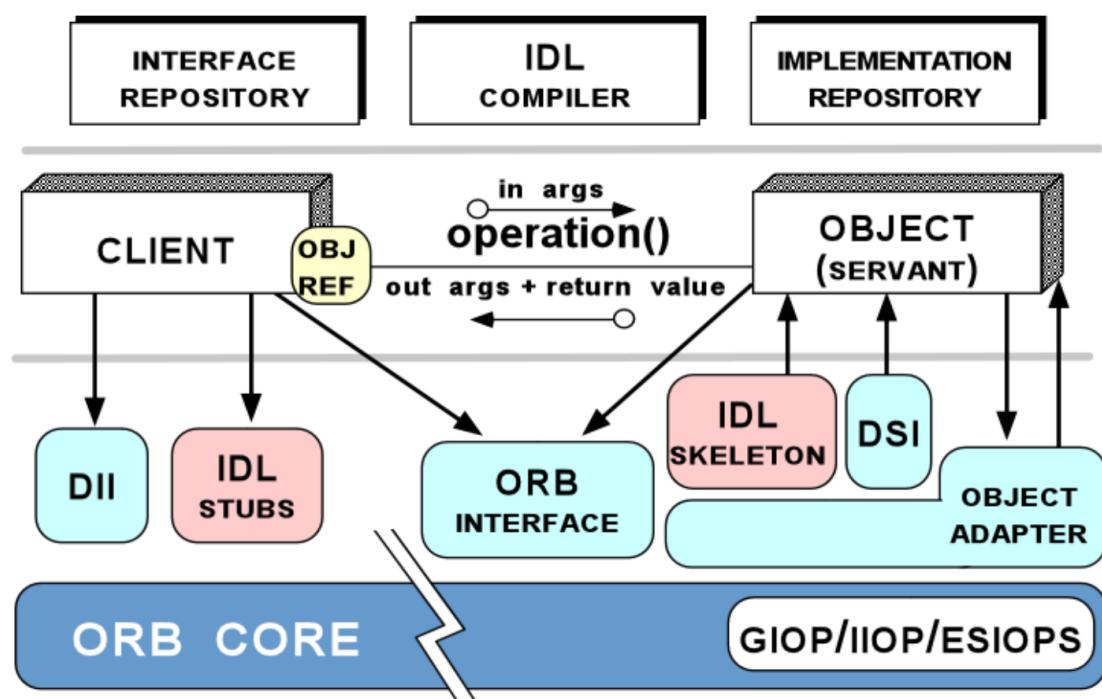
In Optional-Out



In-Out

- 
- 
- 
-

Releases



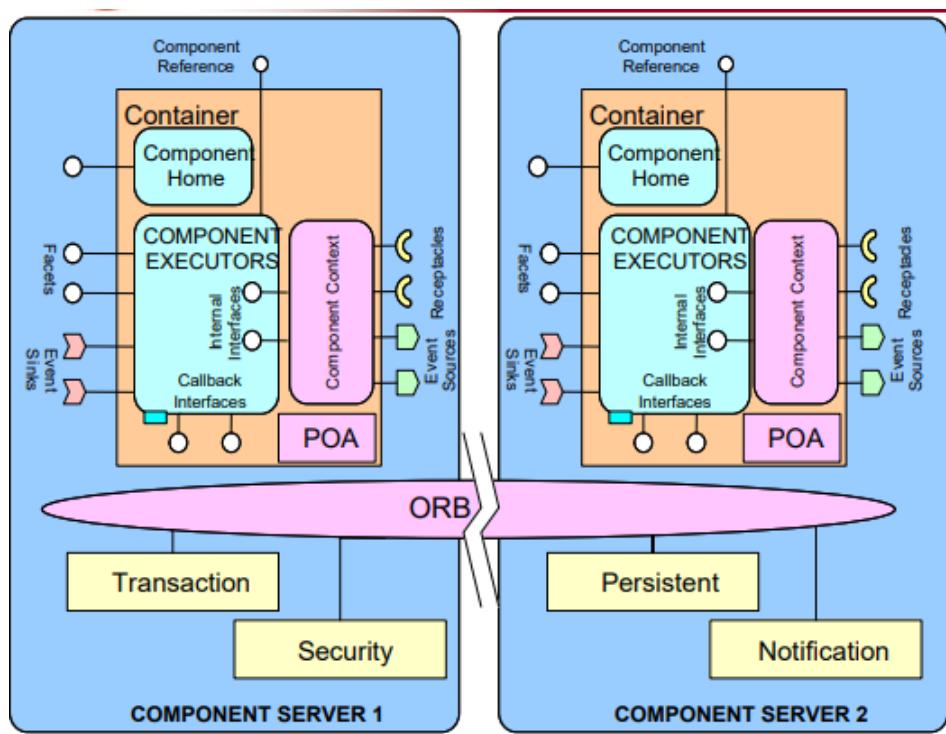
```
interface Foo
{
 parametro in ingresso
 void bar (in long arg);
};
```

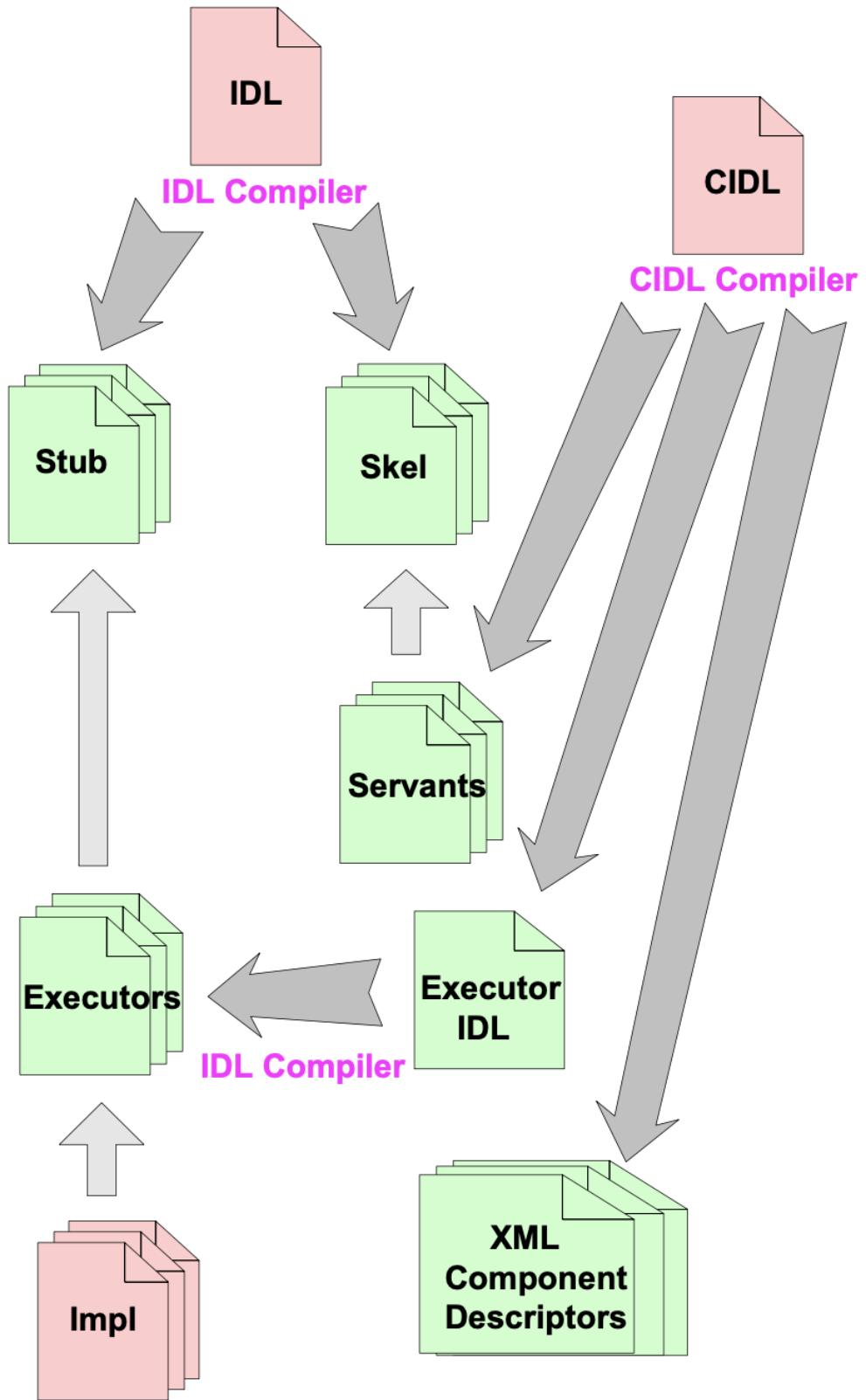
IDL

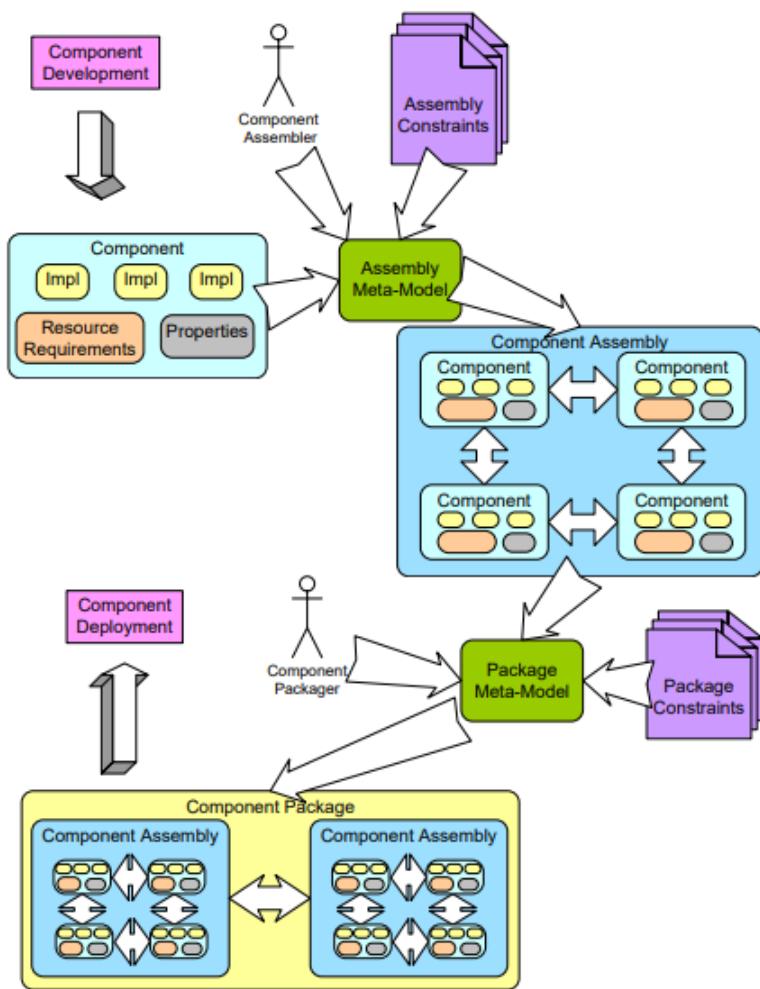


C++

```
class Foo : public virtual CORBA::Object
{
 virtual void bar (CORBA::Long arg);
};
```

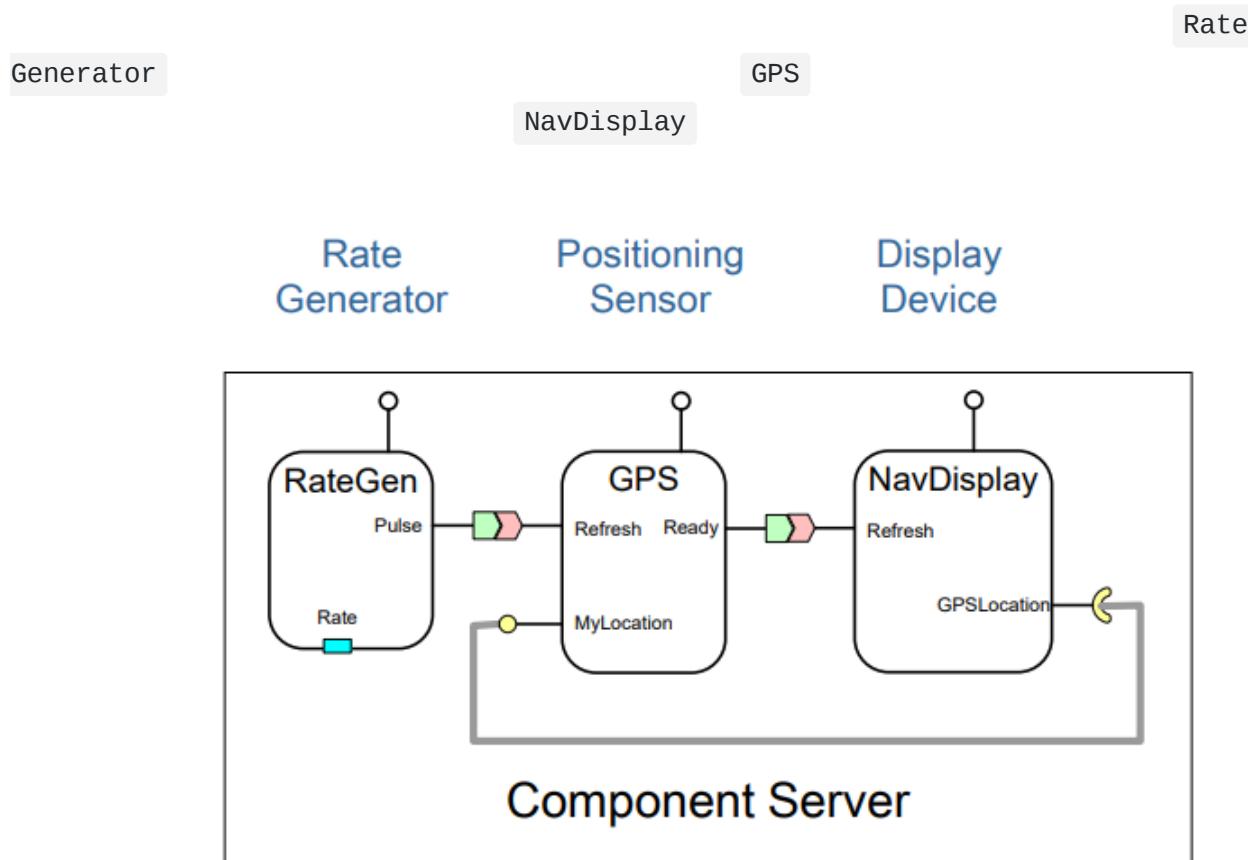




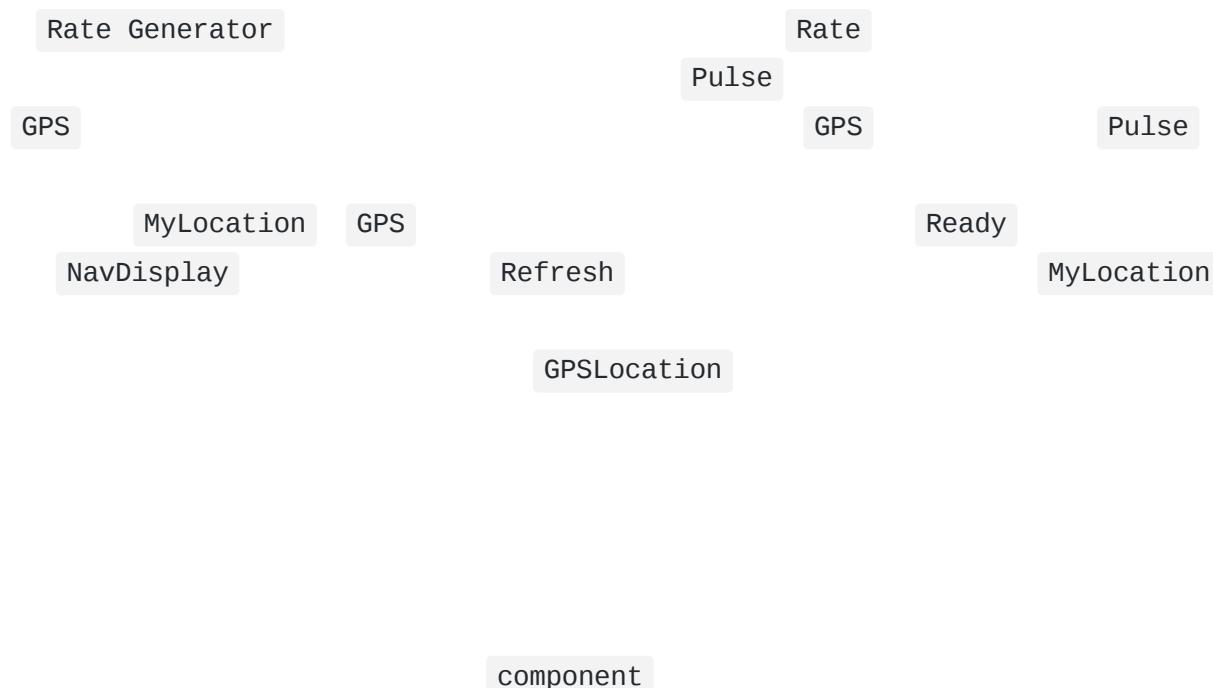


Name	Provider	Open Source	Language	URL
Component Integrated ACE ORB (CIAO)	Vanderbilt University & Washington University	Yes	C++	<a href="http://www.dre.vanderbilt.edu/CIAO/">www.dre.vanderbilt.edu/CIAO/</a>
Enterprise Java CORBA Component Model (EJCCM)	Computational Physics, Inc.	Yes	Java	<a href="http://www.cpi.com/ejccm/">www.cpi.com/ejccm/</a>
K2 	iCMG	No	C++	<a href="http://www.icmgworld.com/products.asp">www.icmgworld.com/products.asp</a>
MicoCCM	FPX	Yes	C++	<a href="http://www.fpx.de/MicoCCM/">www.fpx.de/MicoCCM/</a>
OpenCCM	ObjectWeb	Yes	Java	<a href="http://openccm.objectweb.org/">openccm.objectweb.org/</a>
QoS Enabled Distributed Object (Qedo)	Fokus	Yes	C++	<a href="http://www.qedo.org">www.qedo.org</a>
StarCCM	Source Forge	Yes	C++	<a href="http://sourceforge.net/projects/starccm/">sourceforge.net/projects/starccm/</a>

- 
- 
- 
- 
-



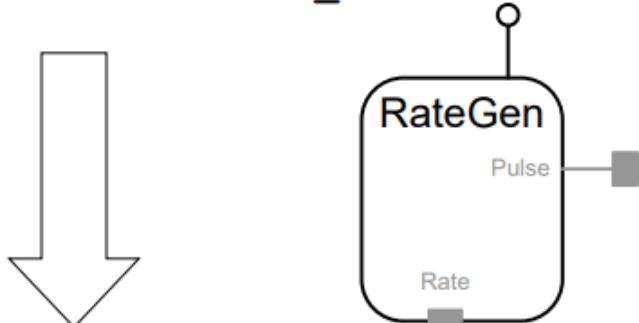
**\$CIAO\_ROOT/examples/OEP/Display/**



```
interface rate_control
{
 void start ();
 void stop ();
};

component RateGen
 supports rate_control {};

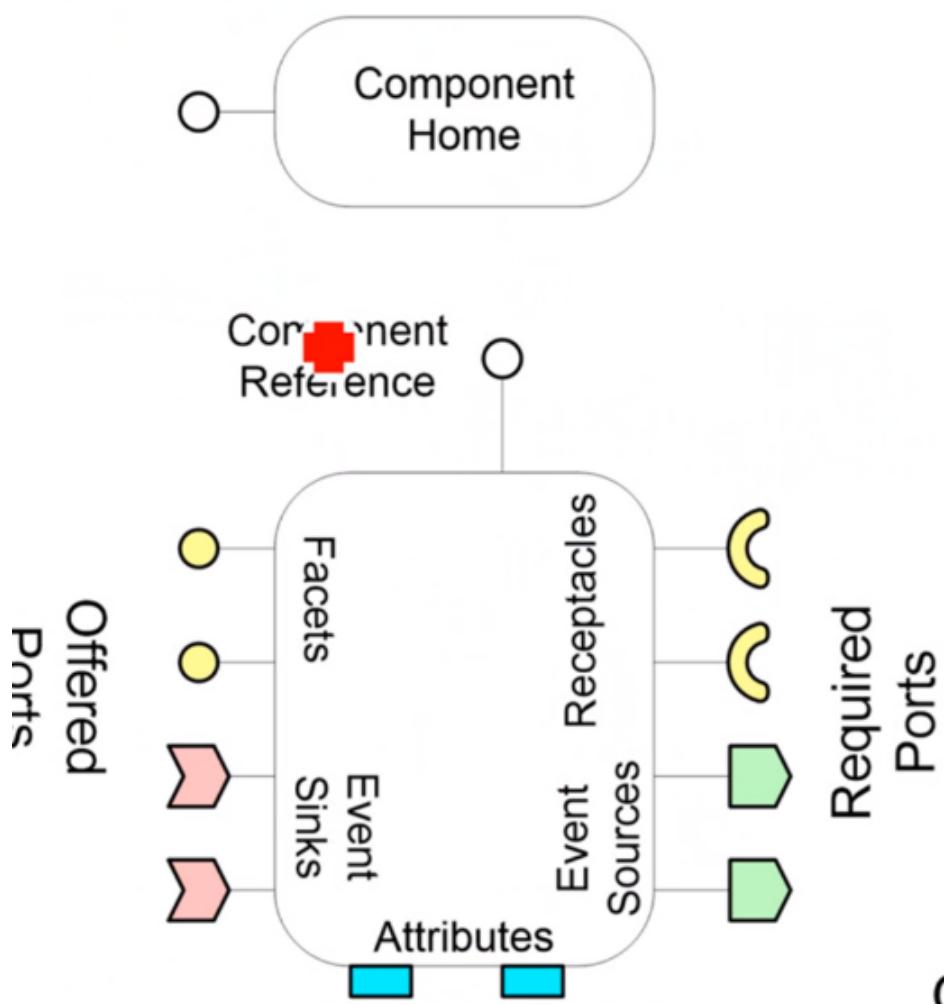
```



RateGen

rate\_control

component



- 
- 
- 
- publishes
- emits
- consumes
- attribute

home

---

```
// IDL 3

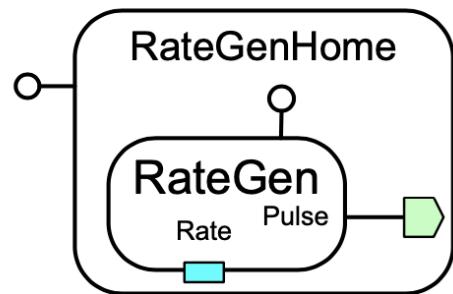
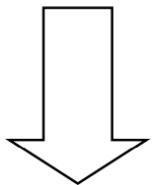
home RateGenHome manages RateGen
{
 factory create_pulser
 (in rateHz r);
};

// Equivalent IDL 2

interface RateGenHomeExplicit
: Components::CCMHome {
 RateGen create_pulser
 (in rateHz r);
}

interface RateGenHomeImplicit
: Components::KeylessCCMHome {
 RateGen create ();
}

interface RateGenHome :
 RateGenHomeExplicit,
 RateGenHomeImplicit {};
```



RateGen

getAllFacets()

getComponent()

```
int
main (int argc, char *argv[])
{
 CORBA::ORB_var orb = CORBA::ORB_init (argc, argv);

 // Get the NameService reference
 CORBA::Object_var o = ns->resolve_str("myHelloHome");
 HelloHome_var hh = HelloHome::_narrow(o.in ());
 HelloWorld_var hw = hh->create();

 // Get all facets & receptacles
 Components::FacetDescriptions_var fd = hw->get_all_facets();
```

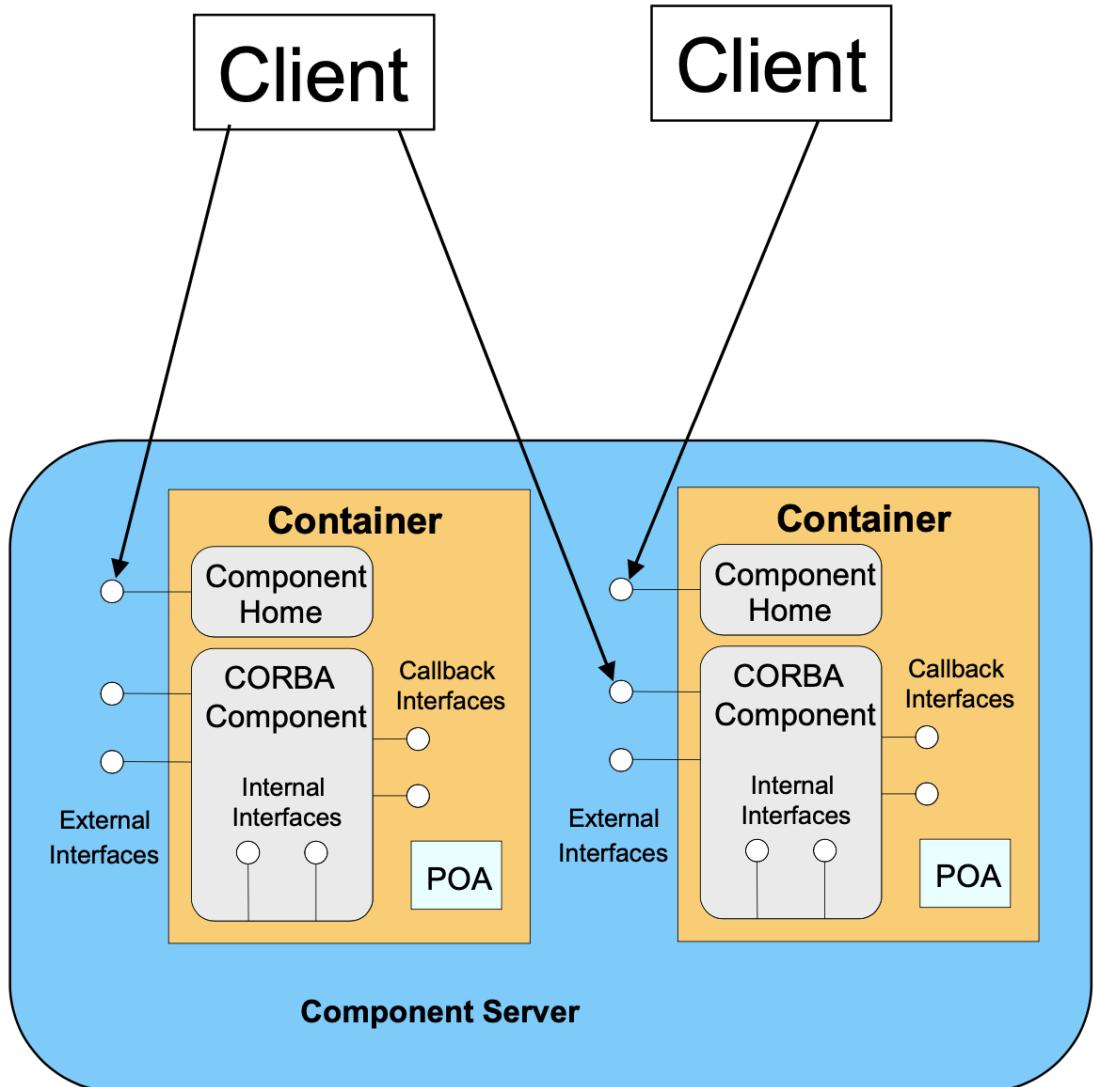
```
Components::ReceptacleDescriptions_var rd = hw->get_all_receptacles();

// Get a named facet with a name "Farewell"
// CORBA::Object_var fobj = hw->provide("Farewell");

// Can invoke sayGoodbye() operation on Farewell after
// narrowing to the Goodbye interface.

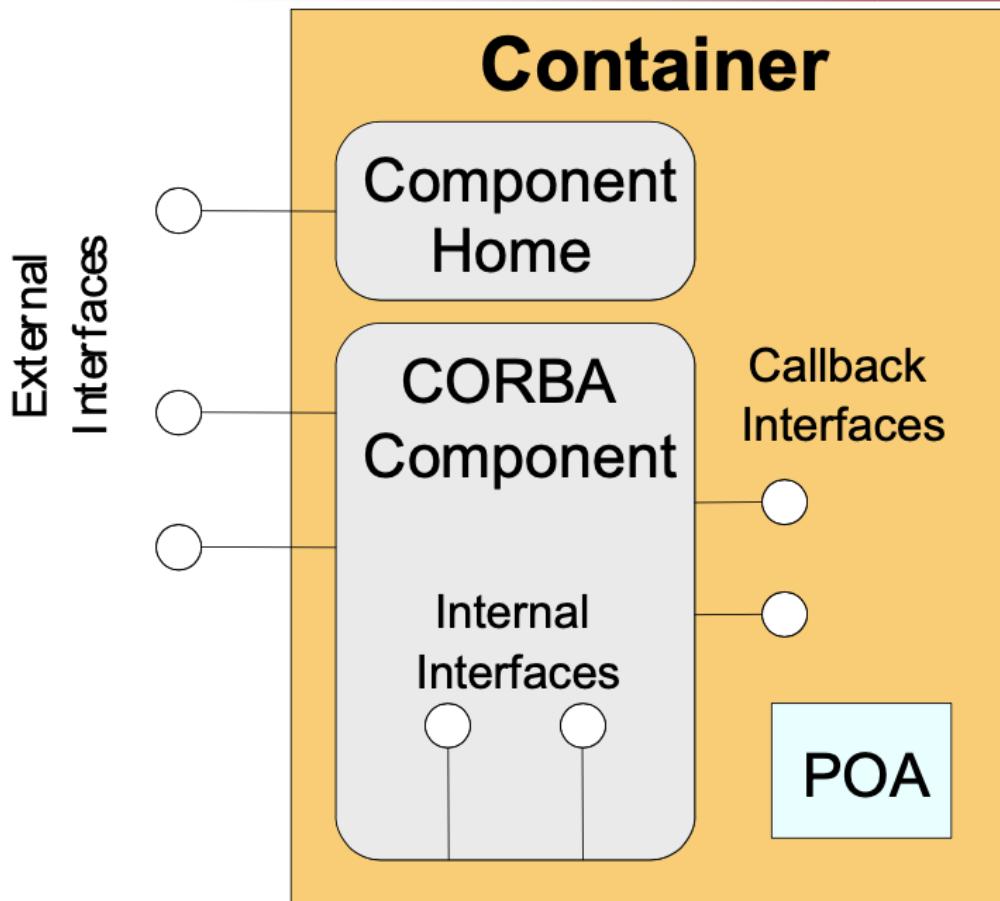
...

return 0;
}
```

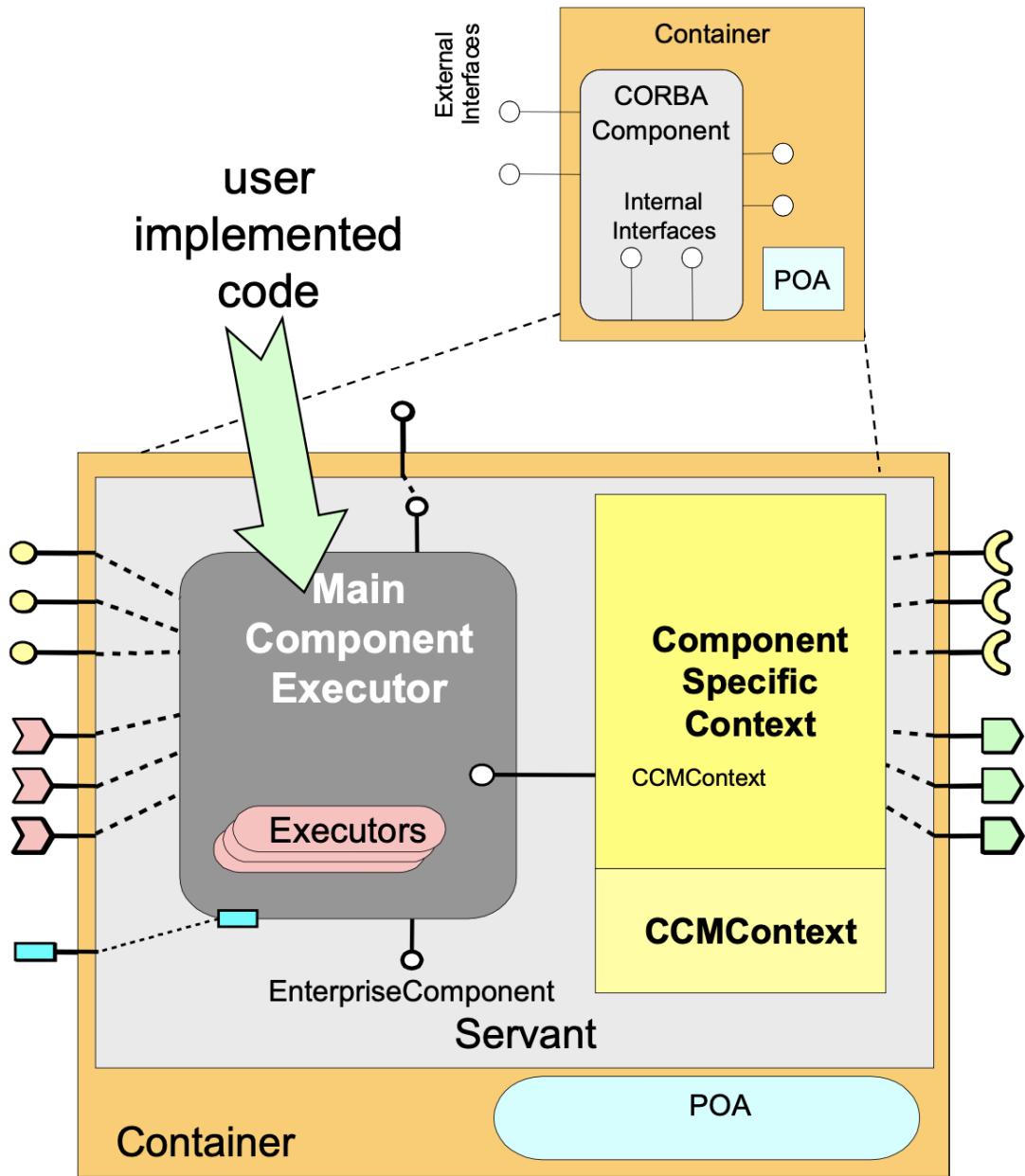


Component category	Container Implementation type	Container type	External Type
Service	Stateless	Session	Keyless
Session	Conversational	Session	Keyless
Process	Durable	Entity	Keyless
Entity	Durable	Entity	Keyful

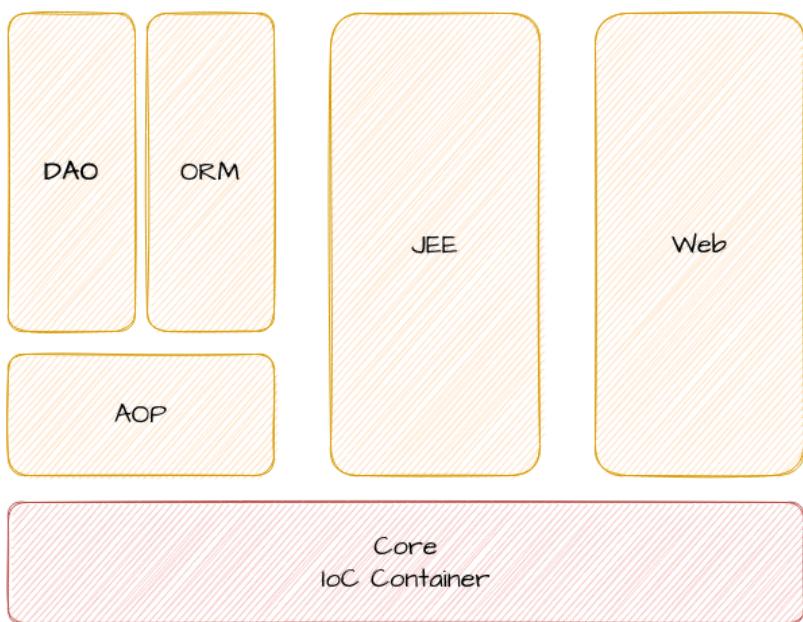
- 
- 
- 
-











• BeanFactory

•

•

•

•





```
public class ConstructorInjection {

 private Dependency dep;

 public ConstructorInjection(Dependency dep) {
 this.dep = dep;
 }
}
```

```
public class SetterInjection {

 private Dependency dep;

 public void setMyDependency(Dependency dep) {
 this.dep = dep;
 }
}
```

setter

setter

BeanFactory

BeanFactory

BeanFactory

XmlBeanFactory

BeanFactory

XmlBeanFactory

DefaultListableBeanFactory

BeanFactory

getBean()

```
public class XmlConfigWithBeanFactory {
 public static void main(String[] args) {
 XmlBeanFactory factory = new XmlBeanFactory(new FileSystemResource("beans.xml"));
 SomeBeanInterface b = (SomeBeanInterface) factory.getBean("nameOftheBean");
 }
}
```

```
public class ConfigurableMessageProvider implements MessageProvider {

 private String message;

 // usa dependency injection per config. del messaggio
 public ConfigurableMessageProvider(String message) {
 this.message = message;
 }
```

```
}
```

```
public String getMessage() {
```

```
 return message;
```

```
}
```

```
}
```

```
<beans>
```

```
 <bean id="provider" class="ConfigurableMessageProvider">
```

```
 <constructor-arg>
```

```
 <value> Questo è il messaggio configurabile</value>
```

```
 </constructor-arg>
```

```
 </bean>
```

```
</beans>
```

provider

ConfigurableMessageProvider

constructor-arg

- 
- 
- 
- 
- 

```
<beans>
```

```
 <bean id="injectSimple" class="InjectSimple">
```

```
 <property name="name">
```

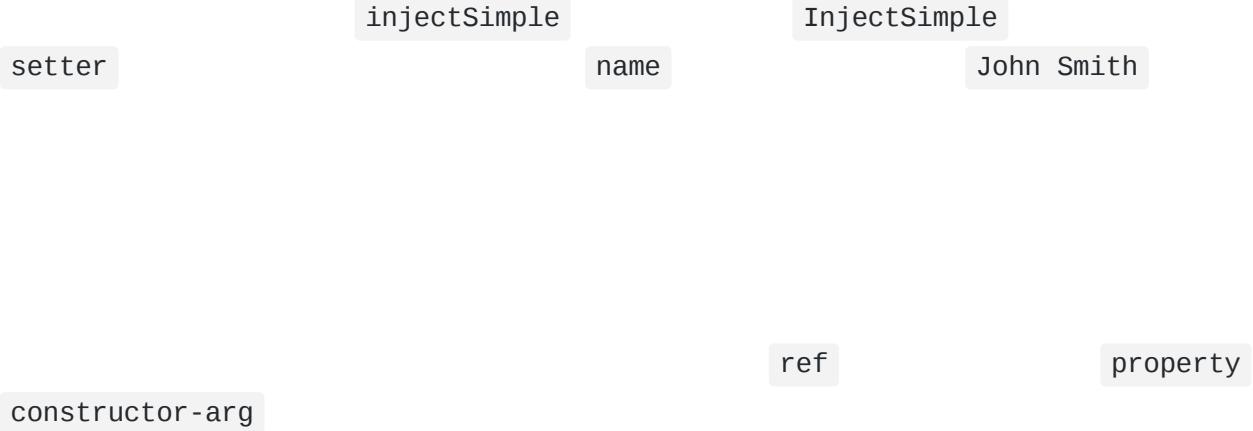
```
 <value>John Smith</value>
```

```
 </property>
```

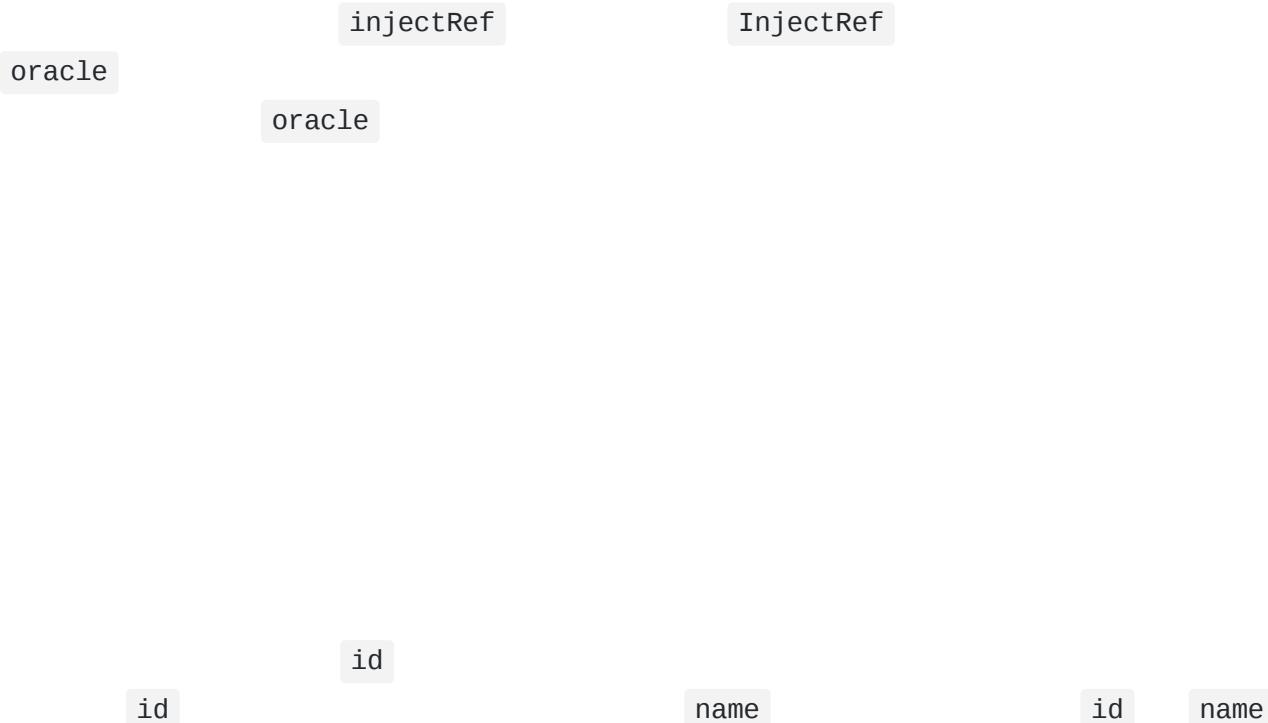
```
 <property name="age">
```

```
 <value>35</value>
```

```
</property>
<property name="height">
 <value>1.78</value>
</property>
</bean>
</beans>
```



```
<beans>
 <bean id="injectRef" class="InjectRef">
 <property name="oracle">
 <ref local="oracle"/>
 </property>
 </bean>
</beans>
```



## BeanFactory

```
public class HelloWorld {

 public static void main(String[] args) {
 System.out.println("Hello World!");
 }

}
```

```
public class HelloWorldMessageProvider {

 public String getMessage() {
 return "Hello World!";
 }

}
```

```
public class StandardOutMessageRenderer {

 private HelloWorldMessageProvider messageProvider = null;

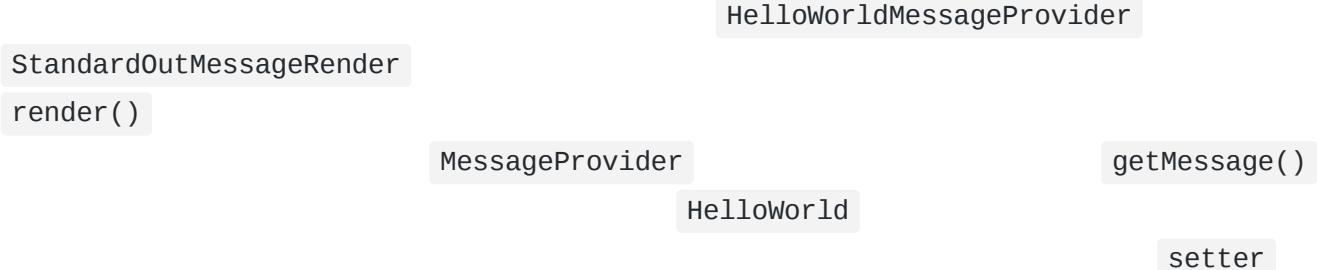
 public void render() {
 if (messageProvider == null) {
 throw new RuntimeException("You must set the property messageProvider of class:" +
 StandardOutMessageRenderer.class.getName());
 }

 System.out.println(messageProvider.getMessage());
 }

 // dependency injection tramite metodo setter
 public void setMessageProvider(HelloWorldMessageProvider provider) {
```

```
this.messageProvider = provider;
}

public HelloWorldMessageProvider getMessageProvider() {
 return this.messageProvider;
}
}
```



```
public class HelloWorldDecoupled {

 public static void main(String[] args) {
 StandardOutMessageRenderer mr = new StandardOutMessageRenderer();
 HelloWorldMessageProvider mp = new HelloWorldMessageProvider();
 mr.setMessageProvider(mp);
 mr.render();
 }
}
```

MessageRenderer      MessageProvider

```
public interface MessageProvider {

 public String getMessage();
}
```

```
public class HelloWorldMessageProvider implements MessageProvider {

 public String getMessage() {
 return "Hello World!";
 }
}
```

```
public interface MessageRenderer {
```

```
public void render();
public void setMessageProvider(MessageProvider provider);
public MessageProvider getMessageProvider();
}
```

```
public class StandardOutMessageRenderer implements MessageRenderer {
 // MessageProvider è una interfaccia Java ora
 private MessageProvider messageProvider = null;

 public void render() {
 if (messageProvider == null) {
 throw new RuntimeException("You must set the property messageProvider of class:" +
 StandardOutMessageRenderer.class.getName());
 }

 System.out.println(messageProvider.getMessage());
 }

 public void setMessageProvider(MessageProvider provider) {
 this.messageProvider = provider;
 }

 public MessageProvider getMessageProvider() {
 return this.messageProvider;
 }
}
```

## main

```
public class HelloWorldDecoupled {

 public static void main(String[] args) {
 MessageRenderer mr = new StandardOutMessageRenderer();
 MessageProvider mp = new HelloWorldMessageProvider();
 mr.setMessageProvider(mp);
 mr.render();
 }

}
```

MessageRenderer

MessageProvider

main

```
public class MessageSupportFactory {

 private static MessageSupportFactory instance = null;
 private Properties props = null;
 private MessageRenderer renderer = null;
 private MessageProvider provider = null;

 private MessageSupportFactory() {
 props = new Properties();
 try {
 props.load(new FileInputStream("msf.properties"));

 // ottiene i nomi delle classi per le interfacce
 String rendererClass = props.getProperty("renderer.class");
 String providerClass = props.getProperty("provider.class");
 renderer = (MessageRenderer) Class.forName(rendererClass).newInstance();
 provider = (MessageProvider) Class.forName(providerClass).newInstance();
 }
 catch (Exception ex) {
 ex.printStackTrace();
 }
 }

 static {
 instance = new MessageSupportFactory();
 }

 public static MessageSupportFactory getInstance() {
 return instance;
 }

 public MessageRenderer getMessageRenderer() {
 return renderer;
 }

 public MessageProvider getMessageProvider() {
 return provider;
 }

}
```

main

```

public class HelloWorldDecoupledWithFactory {

 public static void main(String[] args) {

 MessageRenderer mr = MessageSupportFactory.getInstance().getMessageRenderer();
 MessageProvider mp = MessageSupportFactory.getInstance().getMessageProvider();
 mr.setMessageProvider(mp);
 mr.render();

 }
}

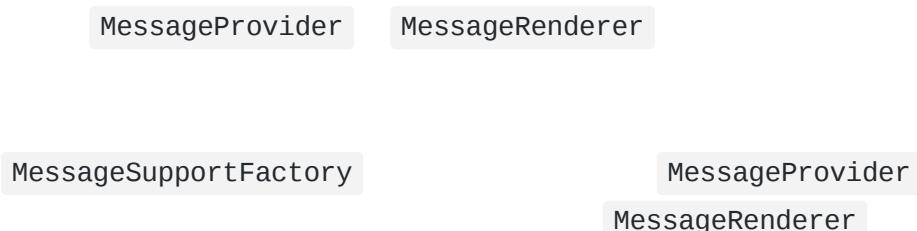
```

```

msf.properties

renderer.class=StandardOutMessageRenderer
provider.class=HelloWorldMessageProvider

```



```

public class HelloWorldSpring {

 public static void main(String[] args) throws Exception {
 // ottiene il riferimento a bean factory
 BeanFactory factory = getBeanFactory();
 MessageRenderer mr = (MessageRenderer) factory.getBean("renderer");
 MessageProvider mp = (MessageProvider) factory.getBean("provider");
 mr.setMessageProvider(mp);
 mr.render();
 }

 // Possibilità di scrivere il proprio metodo getBeanFactory()
 // a partire da Spring DefaultListableBeanFactory class
 private static BeanFactory getBeanFactory() throws Exception {

 DefaultListableBeanFactory factory = new DefaultListableBeanFactory();
 // creare un proprio lettore delle definizioni
 PropertiesBeanDefinitionReader rdr = new PropertiesBeanDefinitionReader(factory);
 // caricare le opzioni di configurazione
 Properties props = new Properties();
 }
}

```

```
 props.load(new FileInputStream("beans.properties"));
 rdr.registerBeanDefinitions(props); return factory;

}
```

MessageSupportFactory

HelloWorld

MessageRenderer

MessageRenderer

getBeanFactory()

HelloWorldMessageProvider

```
File di configurazione

#Message renderer
renderer.class=StandardOutMessageRenderer
Chiede a Spring di assegnare l'effettivo provider alla
proprietà MessageProvider del bean Message renderer
renderer.messageProvider(ref)=provider

#Message provider
provider.class=HelloWorldMessageProvider
```

public class HelloWorldSpringWithDI {

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

 BeanFactory factory = getBeanFactory();
 MessageRenderer mr = (MessageRenderer) factory.getBean("renderer");
 // nota che non è più necessaria nessuna injection manuale
 // del message provider al message renderer
 mr.render();

 }

 private static BeanFactory getBeanFactory() throws Exception {

 DefaultListableBeanFactory factory = new DefaultListableBeanFactory();
 PropertiesBeanDefinitionReader rdr = new PropertiesBeanDefinitionReader(factory);
 }
}
```

```

Properties props = new Properties();
props.load(new FileInputStream("beans.properties"));
rdr.registerBeanDefinitions(props);

return factory;

}
}

main()
MessageRenderer
render()
MessageProvider
MessageProvider
MessageRenderer

```

```

<beans>
 <bean id="renderer" class="StandardOutMessageRenderer">
 <property name="messageProvider">
 <ref local="provider"/>
 </property>
 </bean>
 <bean id="provider" class="HelloWorldMessageProvider"/>
</beans>

```

```

public class HelloWorldSpringWithDIXMLFile {

 public static void main(String[] args) throws Exception {
 BeanFactory factory = getBeanFactory();
 MessageRenderer mr = (MessageRenderer) factory.getBean("renderer");
 mr.render();
 }

 private static BeanFactory getBeanFactory() throws Exception {
 BeanFactory factory = new XmlBeanFactory(new FileSystemResource("beans.xml"));
 return factory;
 }
}

```

MessageProvider

```
<beans>
 <bean id="renderer" class="StandardOutMessageRenderer">
 <property name="messageProvider">
 <ref local="provider"/>
 </property>
 </bean>
 <bean id="provider" class="ConfigurableMessageProvider">
 <constructor-arg>
 <value>Questo è il messaggio configurabile</value>
 </constructor-arg>
 </bean>
</beans>
```

ConfigurableMessageProvider

ConfigurableMessageProvider

MessageProvider

```
public class ConfigurableMessageProvider implements MessageProvider {

 private String message;

 public ConfigurableMessageProvider(String message) {
 this.message = message;
 }

 public String getMessage() {
 return message;
 }
}
```

try-catch

MessageWriter

World

Hello

!

```
public class MessageWriter implements IMessageWriter {

 public void writeMessage() {
 System.out.print("World");
 }

}
```

World

Hello

!

writeMessage()

```
public class MessageDecorator implements MethodInterceptor {

 public Object invoke(MethodInvocation invocation) throws Throwable {
 System.out.print("Hello ");
 Object retVal = invocation.proceed();
 System.out.println("!");
 return retVal;
 }
}
```

ProxyFactory

```

public static void main(String[] args) {

 MessageWriter target = new MessageWriter();
 ProxyFactory pf = new ProxyFactory();
 // aggiunge advice alla coda della catena dell'advice
 pf.addAdvice(new MessageDecorator());
 // configura l'oggetto dato come target
 pf.setTarget(target);
 // crea un nuovo proxy in accordo con le configurazioni
 // della factory MessageWriter
 proxy = (MessageWriter) pf.getProxy();
 proxy.writeMessage();
 // Come farei invece a supportare lo stesso comportamento
 // con chiamata diretta al metodo dell'oggetto target?

}

```

HandlerInterceptorAdaptor

MethodInterceptor

```

public class MyService {

 public void doSomething() {
 for (int i = 1; i < 10000; i++) {
 System.out.println("i=" + i);
 }
 }
}

```

```
public class ServiceMethodInterceptor implements MethodInterceptor {
```

```

 public Object invoke(MethodInvocation methodInvocation) throws Throwable {

 long startTime = System.currentTimeMillis();
 Object result = methodInvocation.proceed();
 long duration = System.currentTimeMillis() - startTime;
 Method method = methodInvocation.getMethod();
 }
}
```

```

 String methodName = method.getDeclaringClass().getName() + "." + method.getName();
 System.out.println("Method '" + methodName + "' took " + duration + " milliseconds to run");

 return null;
 }
}

```

```

<beans>
 <bean id="myService" class="com.test.MyService"></bean>
 <bean id="interceptor" class="com.test.ServiceMethodInterceptor"></bean>
 <bean id="interceptedService" class="org.springframework.aop.framework.ProxyFactoryBean">
 <property name="target">
 <ref bean="myService"/> </property>
 <property name="interceptorNames">
 <list>
 <value>interceptor</value>
 </list>
 </property>
 </bean>
</beans>

```

- PROPAGATION\_REQUIRED
- PROPAGATION\_SUPPORTS
- PROPAGATION\_MANDATORY

- PROPAGATION\_REQUIRE\_NEW
  - PROPAGATION\_NOT\_SUPPORTED
  - PROPAGATION\_NEVER
  - PROPAGATION\_NESTED
- 
- BeanFactory  
    Pippo
- 
- BeanFactory  
    BeanFactory
- 
- getBean()
- 
- BeanFactory
- 
- autowire="name"  
    set()

- autowire="type"  
    set(ArgumentType arg)
  - autowire="constructor"

• 10 •

## BeanFactory

## ApplicationContext

## BeanFactory

## BeanFactory

## ApplicationContext

## BeanFactory

## ApplicationContext

## ApplicationContext

● ● ●

## ApplicationListener

## ApplicationContext

## ApplicationContextAware

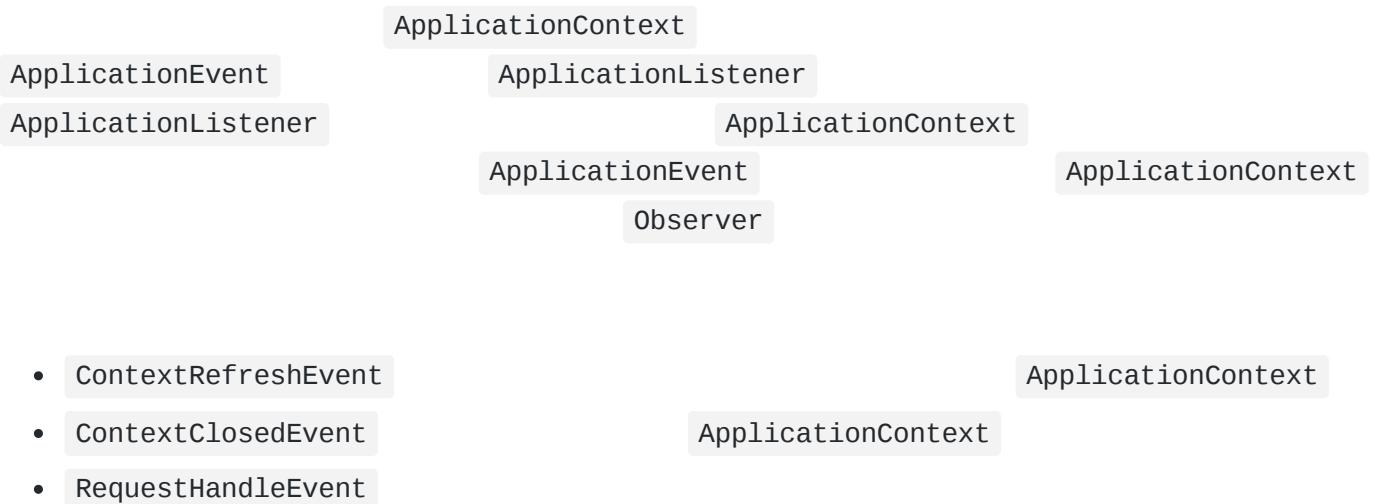
```
setApplicationContext()
```

```
public class Publisher implements ApplicationContextAware {

 private ApplicationContext ctx;

 // Questo metodo sarà automaticamente invocato da IoC container
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {
 this.ctx = applicationContext;
 }

}
```



```
ApplicationContext.xml
```

```
<bean id="emailer" class="example.EmailBean">
 <property name="blackList">
 <list>
 <value>black@list.org</value>
 <value>white@list.org</value>
 <value>john@doe.org</value>
 </list>
 </property>
</bean>

<bean id="blackListListener" class="example.BlackListNotifier">
 <property name="notificationAddress" value="spam@list.org"/>
</bean>
```

```
ApplicationContext
```

```
public class EmailBean implements ApplicationContextAware {

 private List blackList;

 public void setBlackList(List blackList) {
 this.blackList = blackList;
 }

 public void setApplicationContext(ApplicationContext ctx) {
 this.ctx = ctx;
 }

 public void sendEmail(String address, String text) {

 if (blackList.contains(address)) {
 BlackListEvent evt = new BlackListEvent(address, text);
 ctx.publishEvent(evt); return;
 }
 }
}
```

### Notifier

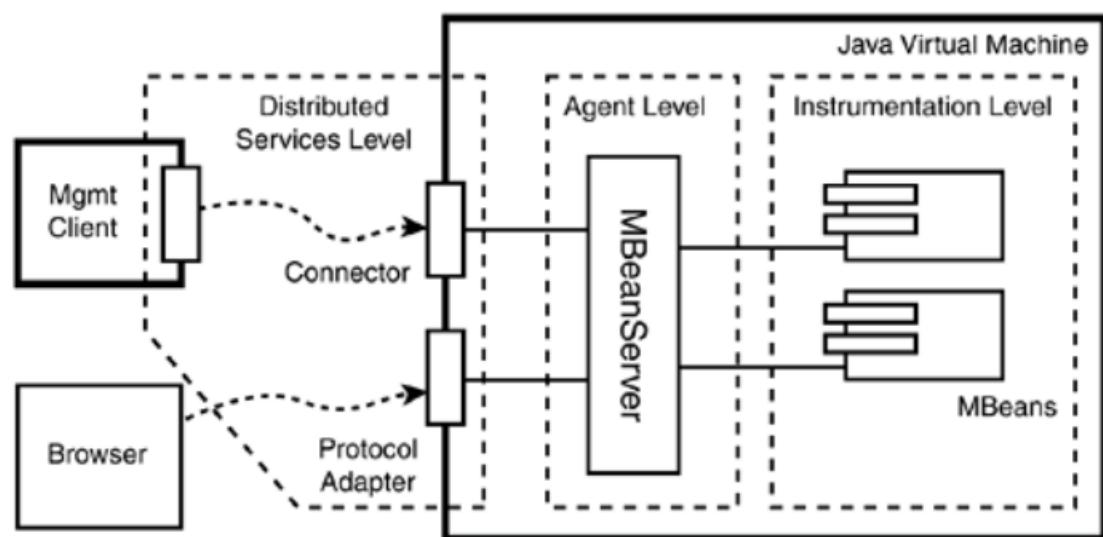
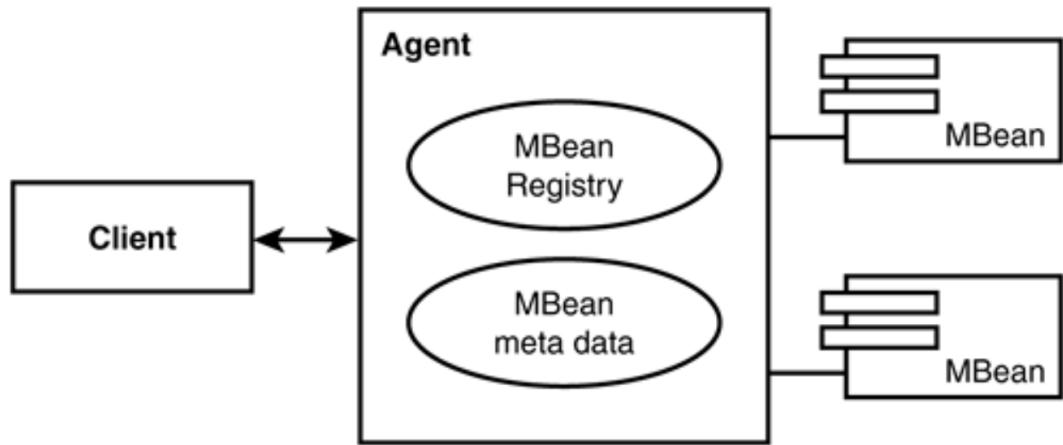
```
public class BlackListNotifier implement ApplicationListener {

 private String notificationAddress;

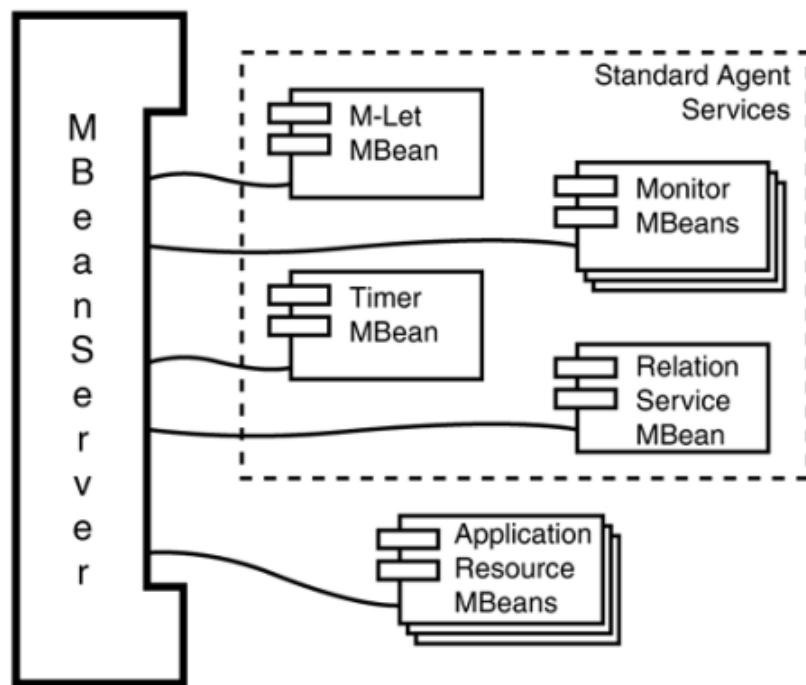
 public void setNotificationAddress(String notificationAddress) {
 this.notificationAddress = notificationAddress;
 }

 public void onApplicationEvent(ApplicationEvent evt) {
 if (evt instanceof BlackListEvent) {
 // invio dell'email di notifica all'indirizzo appropriato
 }
 }
}
```

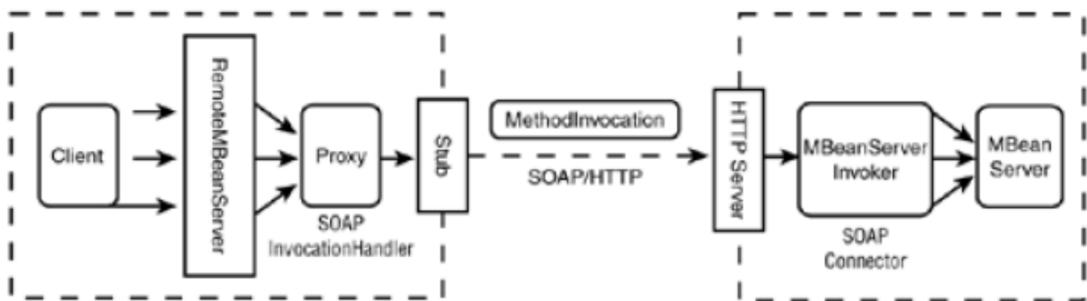








- 
-



```
public interface UserMBean{

 public long getId();
 public void setId(long id);
 public boolean isActive();
 public void setActive(boolean active);
 public String printInfo();
}

public class User implements UserMBean { ... }

public class Student extends User {

 /* anche questa classe può essere registrata come un UserMBean */
```

```
...
```

```
}
```

```
ObjectName
```

```
ObjectName username = new ObjectName("example:name=user1");

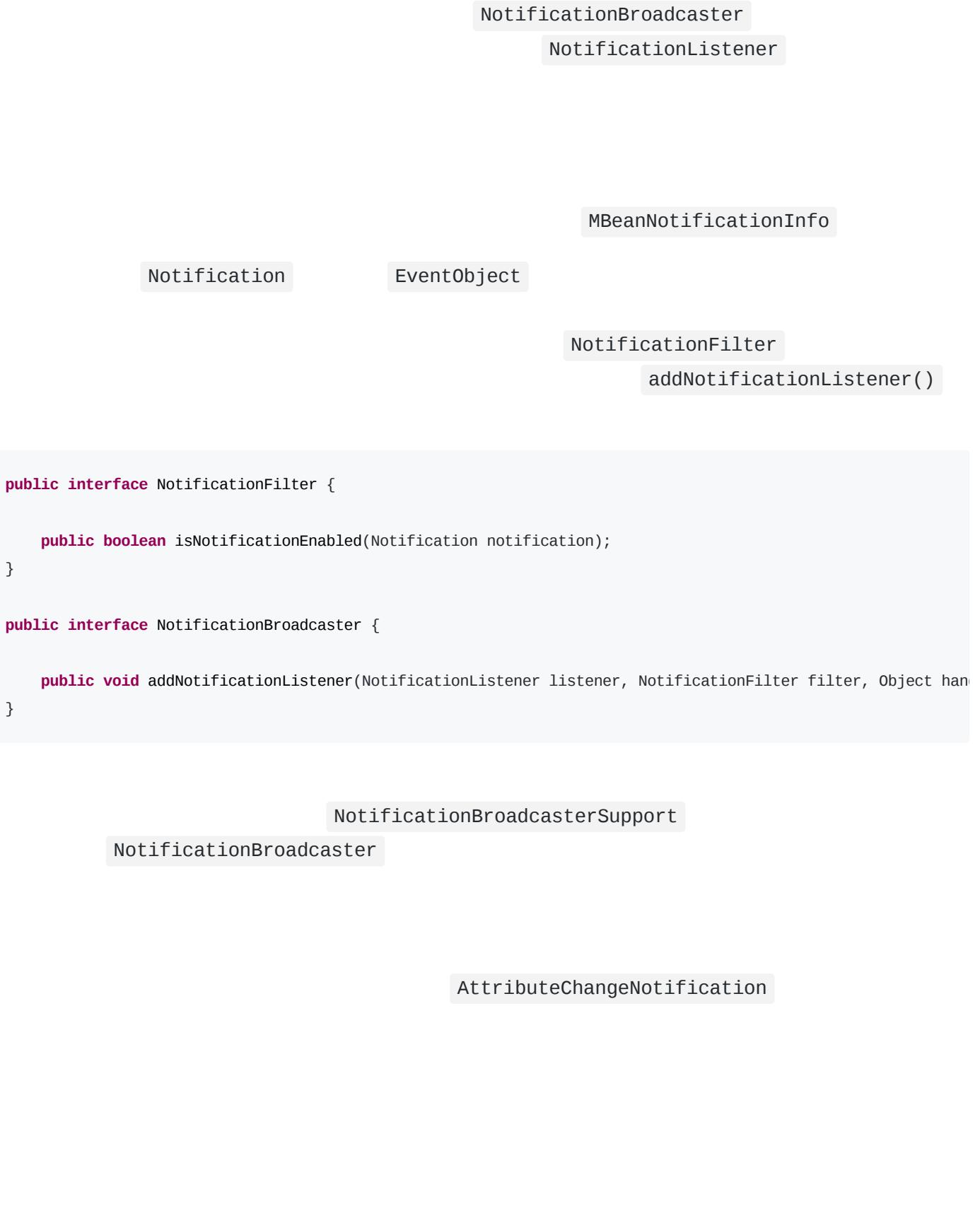
List serverList = MBeanServerFactory.findMBeanServer(null);
MBeanServer server = (MBeanServer)serverList.iterator().next();

/* oppure se prima si deve creare il MBeanServer
MBeanServer server = MBeanServerFactory.createMBeanServer(); */

server.registerMBean(new User(), username);
```

```
ObjectName username = new ObjectName("example:name=user1");

Object result = server.invoke(
 username, // nome MBean
 "printInfo", // nome operazione
 null, // no param
 null); // void signature
```



## Metodi della classe **MBeanInfo**

<code>public String getClassName()</code>	Restituisce il nome della classe di MBean
<code>public String getDescription()</code>	Restituisce una descrizione di MBean
<code>public MBeanAttributeInfo[] getAttributes()</code>	Restituisce un array di oggetti, uno per ogni attributo di management
<code>public MBeanOperationInfo[] getOperations()</code>	Restituisce un array di oggetti, uno per ogni operazione di management
<code>public MBeanConstructorInfo[] getConstructors()</code>	Restituisce un array di oggetti, uno per ogni costruttore pubblico di MBean
<code>public MBeanNotificationInfo[] getNotifications()</code>	Restituisce un array di oggetti, uno per ogni tipo di notifica che MBean può emettere

MBeanInfo

MBeanFeatureInfo

MBeanInfo

getBeanInfo()

BeanInfo

Pippo

```
public class DynamicUser extends NotificationBroadcasterSupport implements DynamicMbean {

 // Attributi
 final static String ID = "id";
 private long id = System.currentTimeMillis();
 public Object getAttribute(String attribute) throws ... {

 if (attribute.equals(ID))
 return new Long(id);
 throw new AttributeNotFoundException("Missing attribute " + attribute);
 }

 // operazioni
 final static String PRINT = "printInfo";

 public String printInfo() {
 return "Sono un MBean dinamico";
 }

 public Object invoke(String actionName, Object[] params, String[] signature) throws ... {

 if (actionName.equals(PRINT))
 return printInfo();
 throw new UnsupportedOperationException("Unknown operation" + actionName);
 }

 // da definire all'interno della classe DynamicUser
 public MBeanInfo getMBeanInfo() {

 final boolean READABLE = true;
 final boolean WRITABLE = true;
 final boolean IS_GETTERFORM = true;
 String classname = getClass().getName();
 String description = "Sono un MBean dinamico";

 MBeanAttributeInfo id = new MBeanAttributeInfo(ID, long.class.getName(),
 "id",
 READABLE,
 !WRITABLE,
 !IS_GETTERFORM);
 MBeanConstructorInfo defcon = new MBeanConstructorInfo("Default", "Creates", null);
 MBeanOperationInfo print = new MBeanOperationInfo(PRINT,
 "Prints info",
 null,
 String.class.getName(),
 MBeanOperationInfo.INFO);
 }
}
```

```
 return new MBeanInfo(classname,description,
 new MBeanAttributeInfo[] { id },
 new MBeanConstructorInfo[] { defcon },
 new MBeanOperationInfo[] { print },
 null);
 }
}
```

invoke MBeanInfo()

MBeanInfo()

ModelMBean

RequiredModelMBean

```
public interface ModelMBean extends DynamicMBean,
PersistentMBean,
ModelMBeanNotificationBroadcaster {

 public void setModelMBeanInfo(ModelMBeanInfo inModelMBeanInfo) throws ... ;

 public void setManagedResource(Object mr, String mr_type) throws ... ;
}

public class RequiredModelMBean implements ModelMBean, ... {

 ...
}
```

Descriptor

Descriptor

DescriptorAccess

```
public interface Descriptor extends Serializable, Cloneable
{
 public String[] getFields();
 public void setField(String name, Object value);
 public void removeField(String name);

 ...
}
```

•  
•  
•  
•

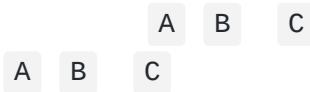
MLetMBean

addURL()

getMBeansFromURL()

addURL()

```
<MLET CODE = class | OBJECT =
 serfile
 ARCHIVE = "archiveList"
 [CODEBASE = codebaseURL]
 [NAME = MBeanName]
 [VERSION = version] >
 [arglist]
</MLET>
```



```
<MLET CODE=com.mycompany.Foo
 ARCHIVE="MyComponents.jar,acme .jar"
 </MLET>
```

### NotificationListener

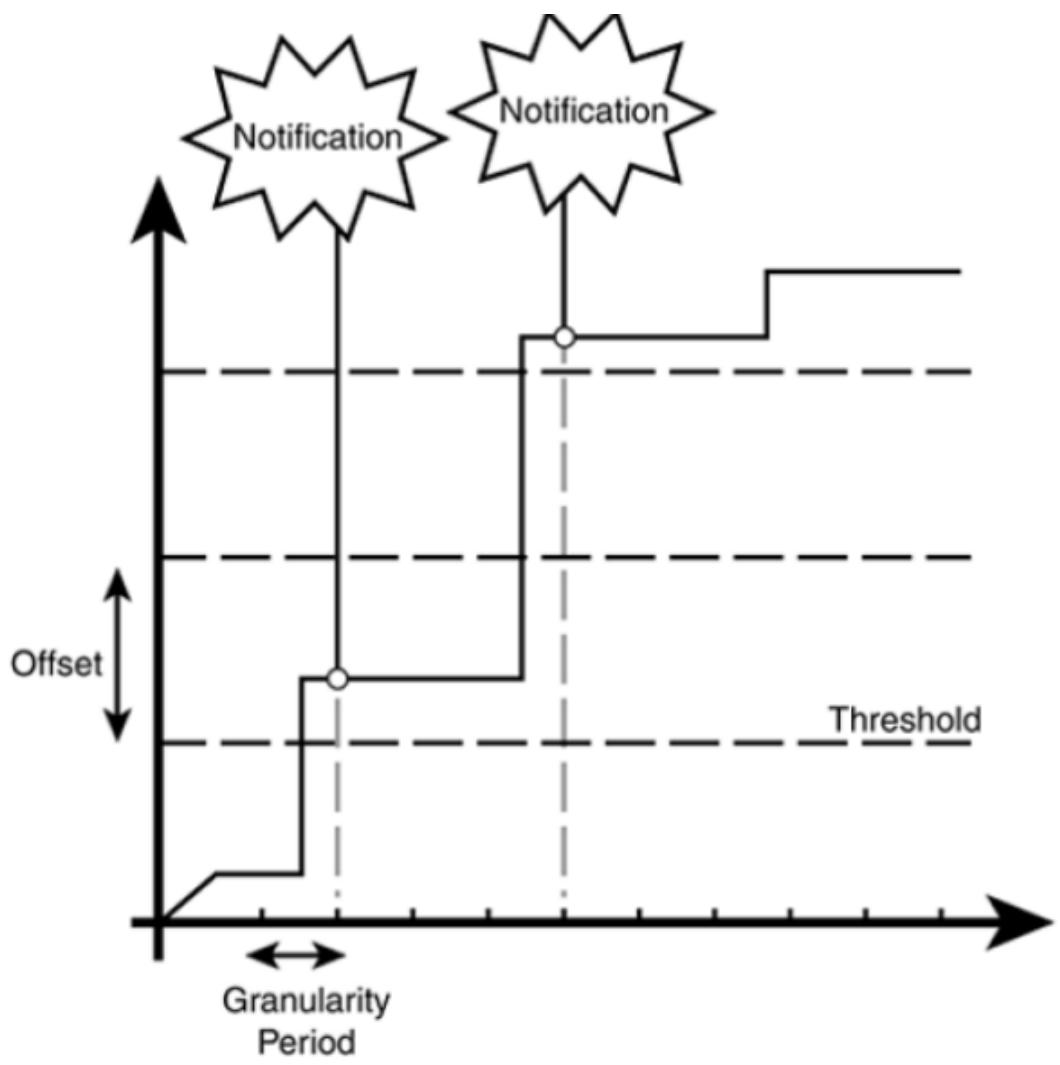
```
// fa partire il servizio di timer
List list = MBeanServerFactory.findMBeanServer(null);
MBeanServer server = (MBeanServer)list.iterator().next();
ObjectName timer = new ObjectName("service:name=timer");
server.registerMBean(new Timer(), timer);
server.invoke(timer, "start", null, null);

// configurazione di notification time
Date date = new Date(System.currentTimeMillis() + Timer.ONE_SECOND * 5);

server.invoke(timer, // MBean
 "addNotification", // metodo
 new Object[] { // args
 "timer.notification", // tipo
 "Schedule notification", // messaggio
 null, // user data
 date}, // time
 new String[] { String.class.getName(),
 String.class.getName(),
 Object.class.getName(), // signature
 Date.class.getName()})
);

// registra il listener MBean
server.addNotificationListener(timer, this, null, null);
```

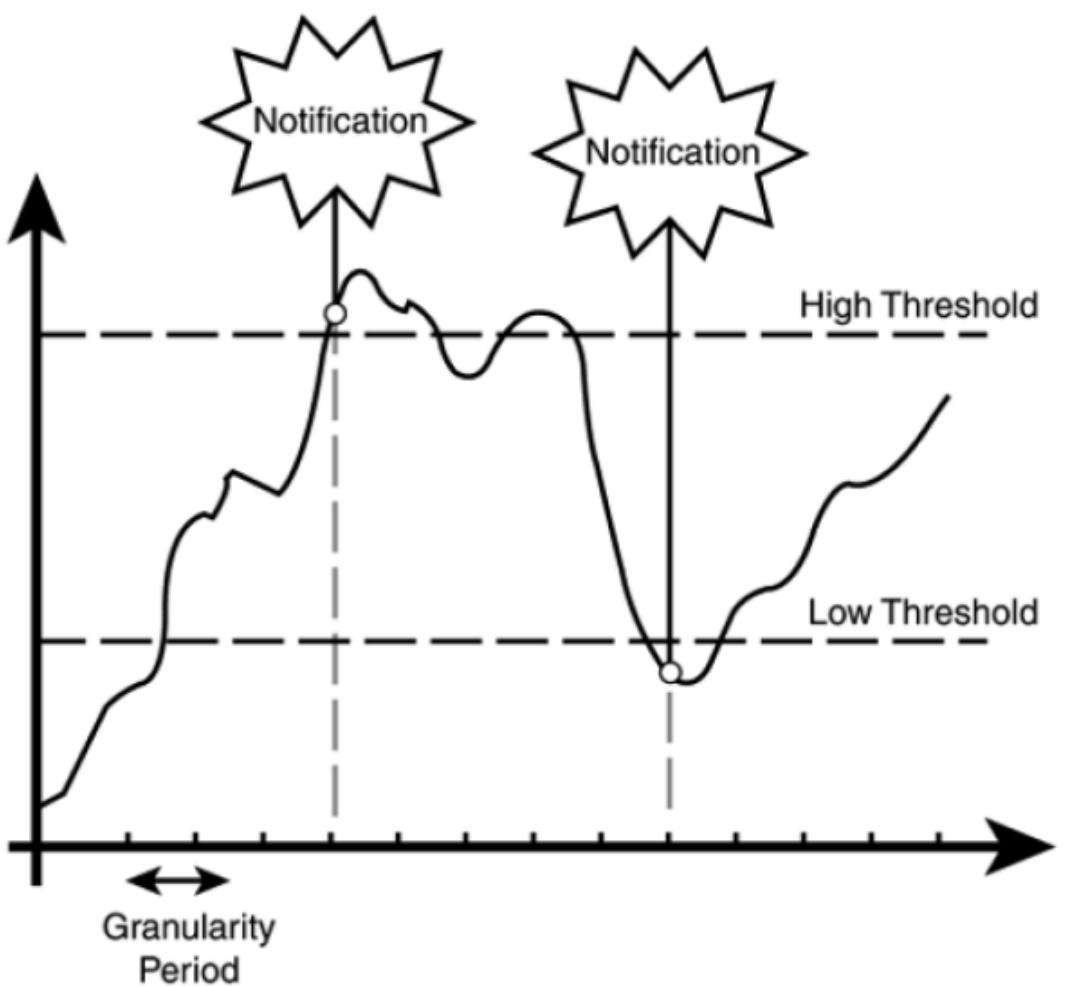
getter      setter



• Integer

x

• Integer      Float



Mbean A

MBean B

MBean A

MBean B

```
// lato cliente

JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://" + "localhost:9999/server");

JMXConnector jmxc = JMXConnectorFactory.connect(url, null);

MBeanServerConnection mbsc = jmxc.getMBeanServerConnection;

mbsc.createMBean(...);
```

```
// lato servitore

MBeanServer mbs = MBeanServerFactory.createMBeanServer();
JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://" + "localhost: 9999/server");
JMXConnectorServer cs = JMXConnectorServerFactory.newJMXConnectorServer(url, null, mbs);
cs.start();
```

JMXServiceURL

JMXConnectorServerFactory

HelloMBean.java:

sayHello add Name CacheSize

```
package com.example.mbeans;

public interface HelloMBean {

 // operazioni (signature)
 public void sayHello();
 public int add(int x, int y);

 // attributi
 public String getName();
 public int getCacheSize();
 public void setCacheSize(int size);
}
```

Hello.java

HelloMBean

```
package com.example.mbeans;

public class Hello implements HelloMBean {

 public void sayHello() {
 System.out.println("hello, world");
 }

 public int add(int x, int y) {
 return x + y;
 }
}
```

```

/* metodo getter per l'attributo Name.
 * Spesso gli attributi sono utilizzati per fornire indicatori di monitoraggio
 * come uptime o utilizzo di memoria. Spesso sono read-only. In questo caso l'attributo è una stringa */
public String getName() {
 return this.name;
}

/* invece anche metodi getter e setter */
/* invece anche metodi getter e setter */
public int getCacheSize() {
 return this.cacheSize;
}

/* perché synchronized? Mantenere uno stato consistente per evitare modifiche concorrenti.
 * No notifiche concorrenti, non ci sono container che si occupano della sincronizzazione
 * quindi serve synchronized, prima non necessario con i container si occupano
 * internamente della sincronizzazione */
public synchronized void setCacheSize(int size) {
 this.cacheSize = size;
 System.out.println("Cache size now " + this.cacheSize);
}

private final String name = "My First MBean";
private int cacheSize = DEFAULT_CACHE_SIZE;
private static final int DEFAULT_CACHE_SIZE = 200;
}

```

Main.java

HelloWorld

```

package com.example.mbeans;
import java.lang.management.*;
import javax.management.*;

public class Main {

 public static void main(String[] args) throws Exception {
 // ottiene il server MBean
 MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
 // costruisce ObjectName per MBean da registrare
 ObjectName name = new ObjectName("com.example.mbeans:type=Hello");
 // crea istanza di HelloWorld MBean
 Hello mbean = new Hello();
 // registra l'istanza
 mbs.registerMBean(mbean, name);
 System.out.println("Waiting forever... ");
 Thread.sleep(Long.MAX_VALUE);
 }
}

```

## Notification

```
package com.example.mbeans;
import javax.management.*;

public class Hello extends NotificationBroadcasterSupport implements HelloMBean {

 public void sayHello() {
 System.out.println("hello, world");
 }

 public int add(int x, int y) {
 return x + y;
 }

 public String getName() {
 return this.name;
 }

 public int getCacheSize() {
 return this.cacheSize;
 }

 public synchronized void setCacheSize(int size) {
 int oldSize = this.cacheSize;
 this.cacheSize = size;

 /* In applicazioni reali il cambiamento di un attributo di solito produce effetti di gestione.
 * Ad esempio, cambiamento di dimensione della cache può generare eliminazione o
 * allocazione di entry */
 System.out.println("Cache size now " + this.cacheSize);
 /* Per costruire una notifica che descrive il cambiamento avvenuto: "source" è ObjectName di MBean
 * che emette la notifica (MBean server sostituisce "this" con il nome dell'oggetto);
 * mantenuto un numero di sequenza */
 Notification n = new AttributeChangeNotification(this,
 sequenceNumber++,
 System.currentTimeMillis(),
 "CacheSize changed",
 "CacheSize",
 "int",
 oldSize,
 this.cacheSize);
 // Invio della notifica usando il metodo sendNotification() ereditato dalla superclasse
 sendNotification(n);
 }

 ...

 @Override
 /* metadescrizione */
 public MBeanNotificationInfo[] getNotificationInfo() {
```

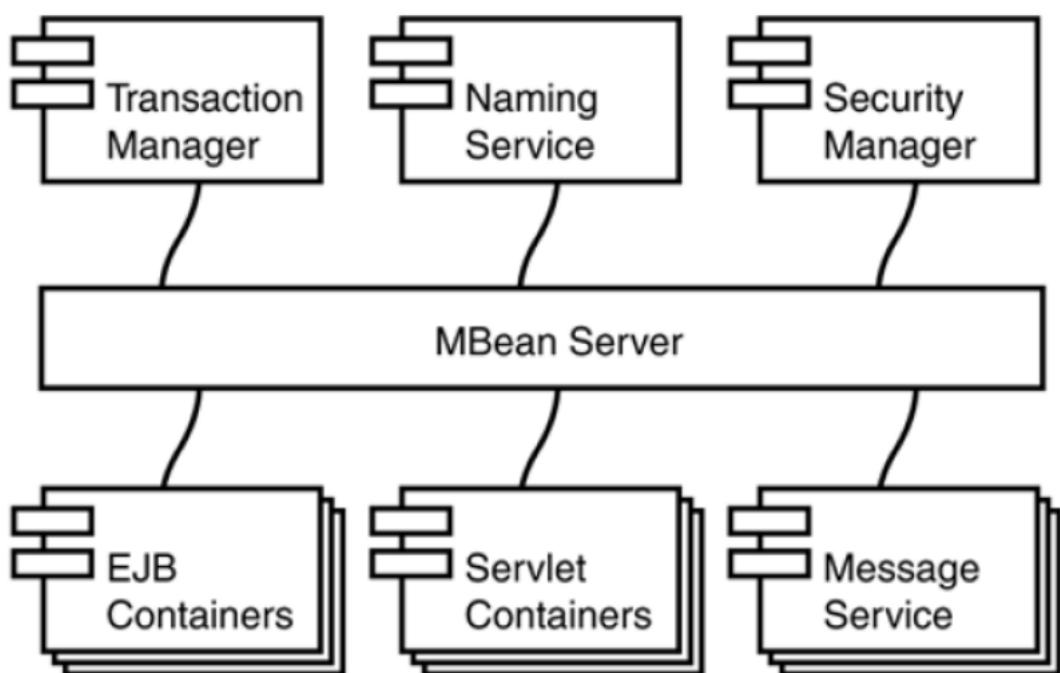
```

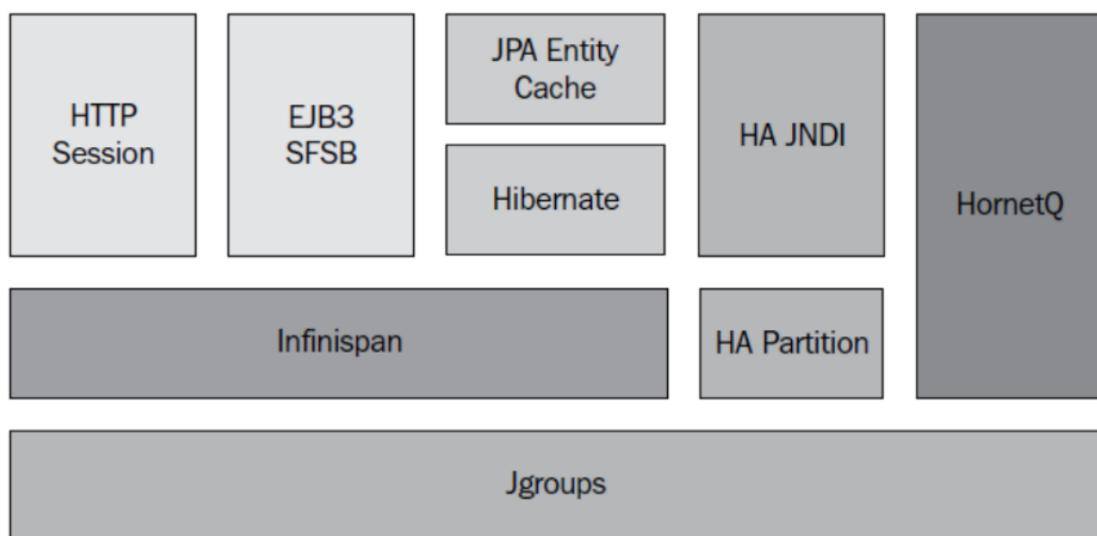
String[] types = new String[] { AttributeChangeNotification.ATTRIBUTE_CHANGE };
String name = AttributeChangeNotification.class.getName();
String description = "è stato cambiato un attributo!";
MBeanNotificationInfo info = new MBeanNotificationInfo(types, name, description);

return new MBeanNotificationInfo[] {info};
}

private final String name = "My first MBean";
private int cacheSize = DEFAULT_CACHE_SIZE;
private static final int DEFAULT_CACHE_SIZE = 200;
private long sequenceNumber = 1;
}

```





```
run.bat -c all
./run.sh -c all
```

JGroups.jar

jbosscache.jar

•  
•

cluster-service.xml

deploy

PartitionConfig

ClusterPartition

```
<mbean code="org.jboss.ha.framework.server.ClusterPartition"
name="jboss:service={jboss.partition.name:DefaultPartition}">
```

```

...
<attribute name="PartitionConfig">
<Config>
<UDP mcast_addr="${jboss.partition.udpGroup:228.1.2.3}"
mcast_port="${jboss.hapartition.mcast_port:45566}"
tos="8"

...
<!-- ping per scoprire i membri che appartengono al cluster -->
<PING timeout="2000"
down_thread="false" up_thread="false"
num_initial_members="3"/>

...
<!-- per fondere gruppi già scoperti -->
<MERGE2 max_interval="100000"
down_thread="false" up_thread="false"
min_interval="20000"/>

...
<!-- timeout per failure detection -->
<FD timeout="10000" max_tries="5"
down_thread="false" up_thread="false" shun="true"/>

<!-- questo protocollo verifica se un membro sospetto è realmente morto eseguendo nuovamente
il ping di quel membro. -->
<VERIFY_SUSPECT timeout="1500" down_thread="false"
up_thread="false"/>

...
<pbcast.STATE_TRANSFER down_thread="false" up_thread="false"/>
```

PING

MERGE2

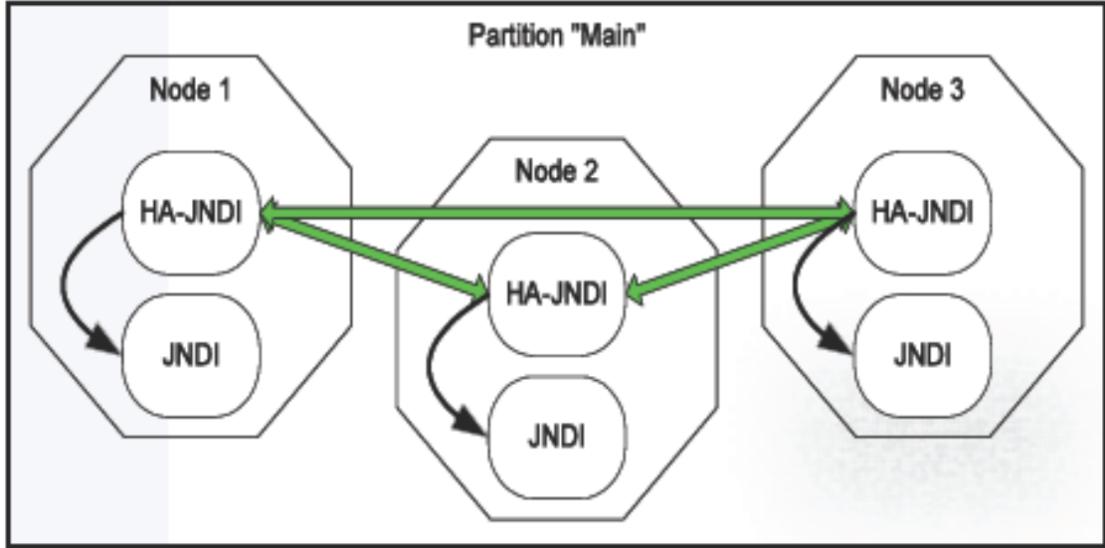
FD

```
<mbean code="org.jboss.ha.framework.server.ClusterPartition"
name="jboss:service=DefaultPartition">
 <attribute name="PartitionName">${jboss.partition.name:DefaultPartition}</attribute>
 <!-- indirizzo usato per determinare il nome del nodo -->
 <attribute name="NodeAddress">${jboss.bind.address}</attribute>
 <!-- deadlock detection abilitata o no -->
 <attribute name="DeadlockDetection">False</attribute>
 <!-- Max time (in ms) di attesa per il completamento del trasferimento di stato -->
 <attribute name="StateTransferTimeout">30000</attribute>
 <!-- configurazione protocolli JGroups -->
 <attribute name="PartitionConfig">...</attribute>
</mbean>
```

PartitionName

PartitionConfig

NameNotFoundException



Nodo 1

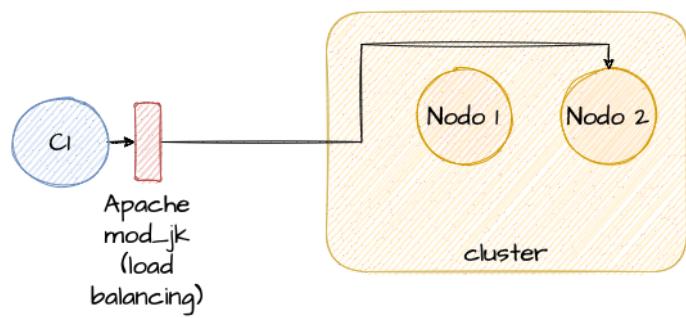
Nodo 1

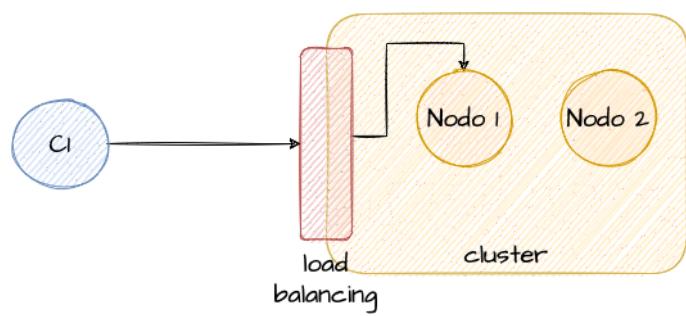
Nodo 2

Nodo 1

Nodo 3

•





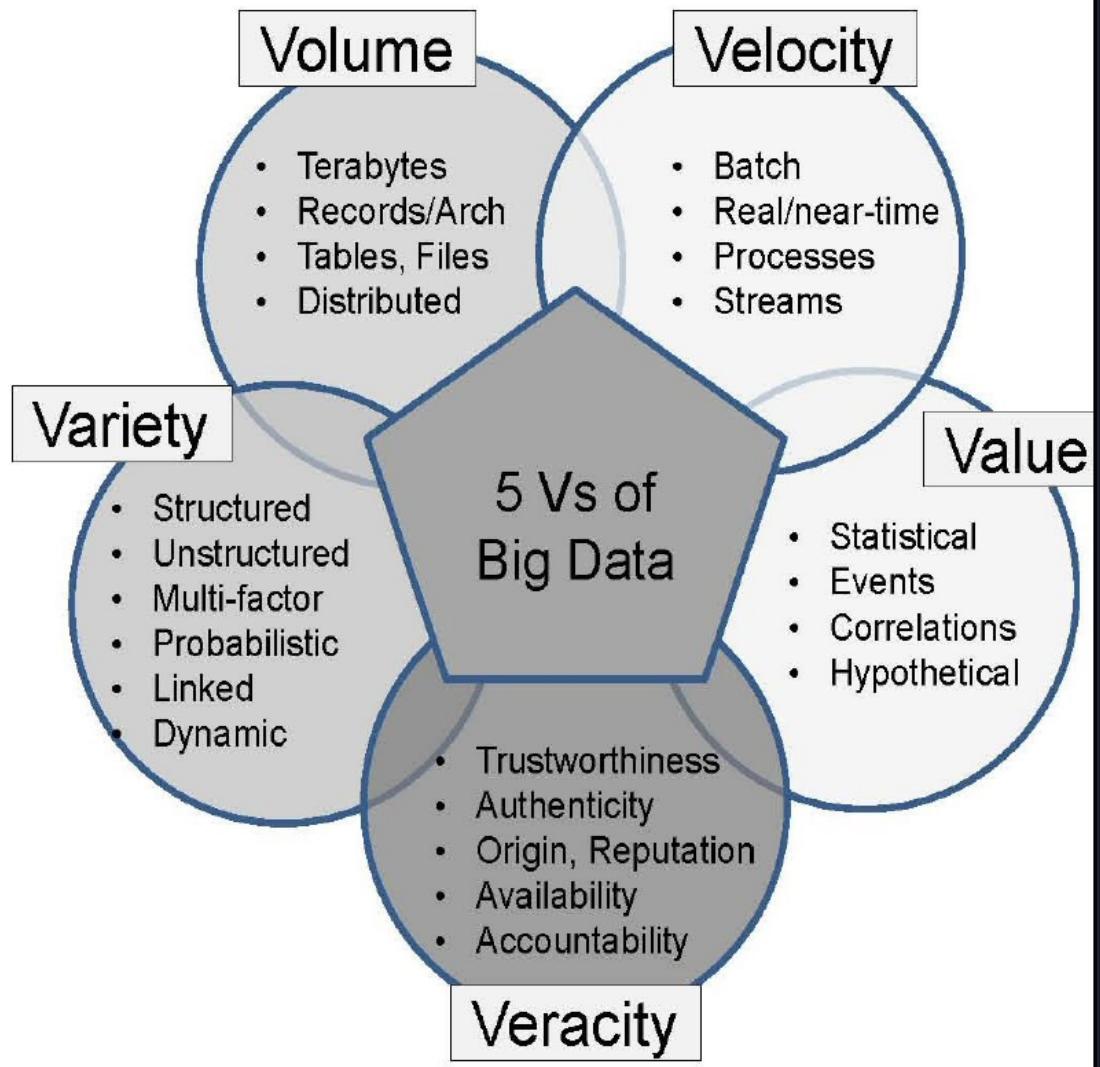
- 
- 
- 
-

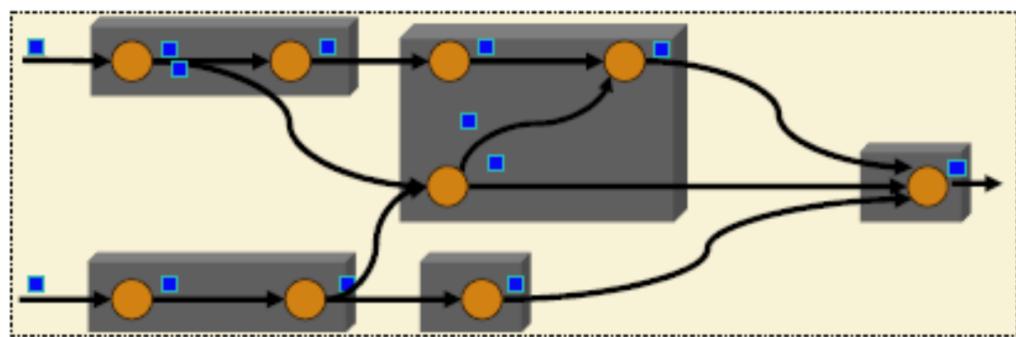
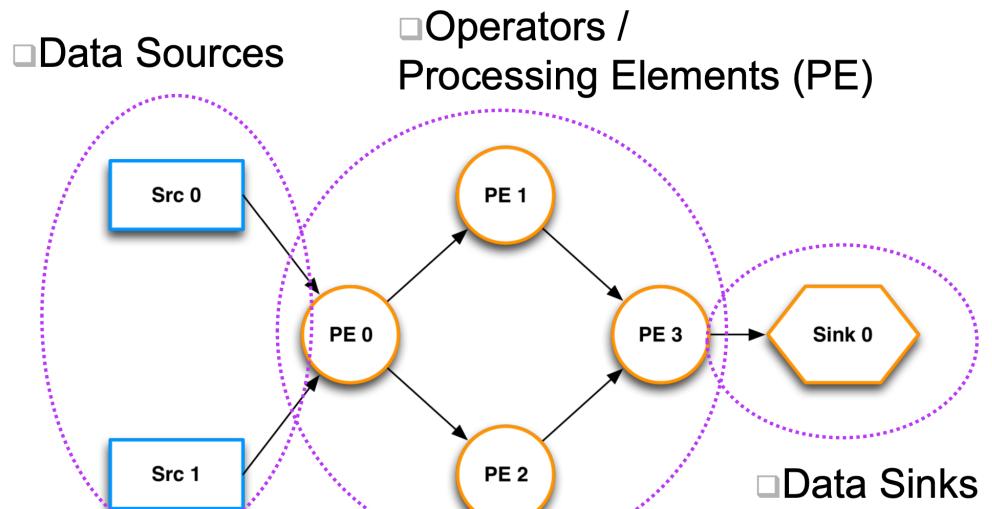
•  
•  
•

- @Clustered
- 
- 
- 
- 
- standalone.xml
- standalone-full.xml
- standalone-ha.xml
- standalone-full-ha.xml

- standalone-load-balancer.xml
- 

- - 
  - 
  - - 
    - 
    - 
    -
  - 
  -





## Instance

### Job

#### Node

PE  
operator

PE

Stream 2

Stream 1

Stream 1

PE  
operator

Stream 3

Stream 3

Stream 4

Stream 5

#### Node

istanza1

C D

instance2

A B

istanza1

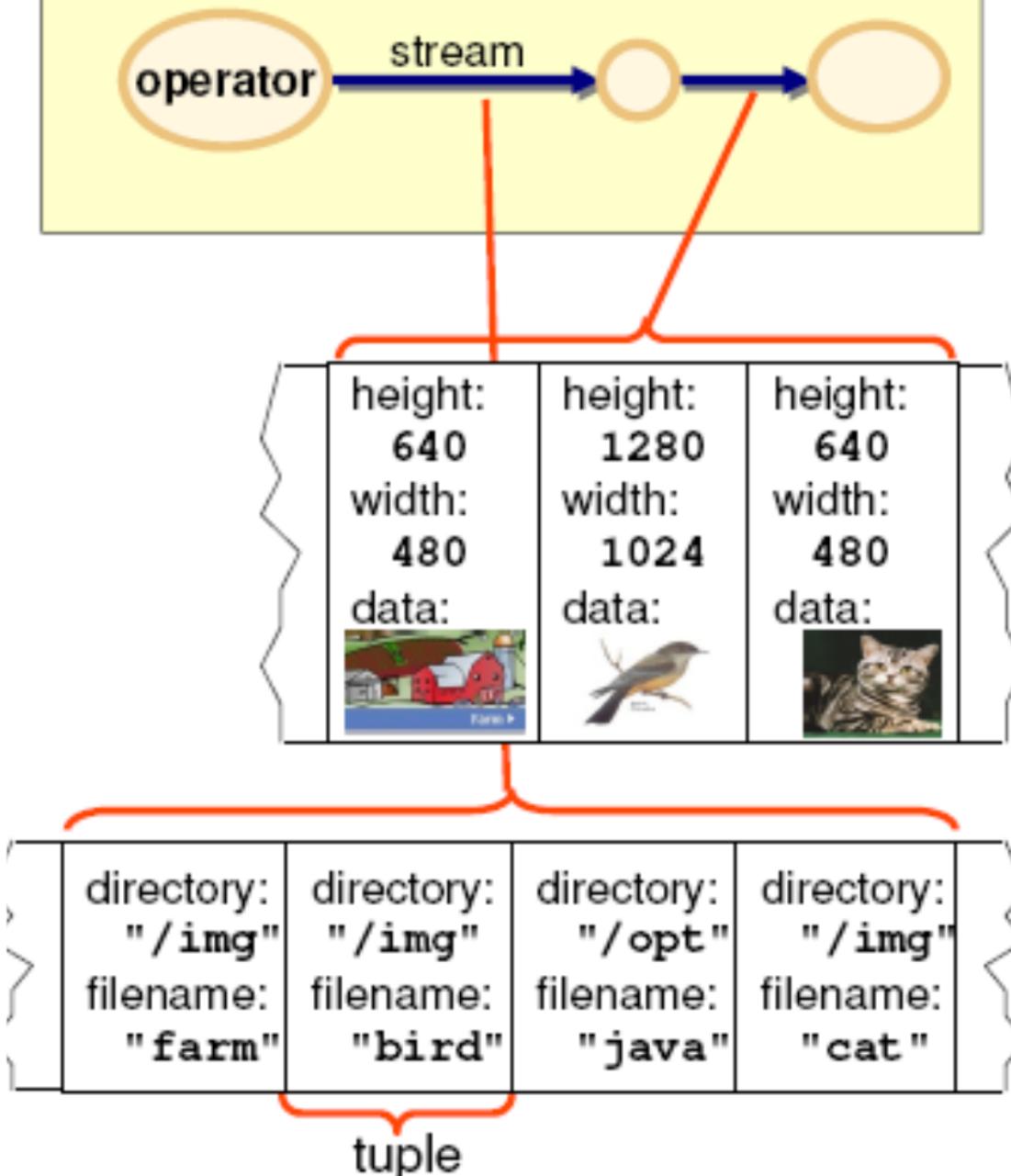
A B C D

istanza2

B

B

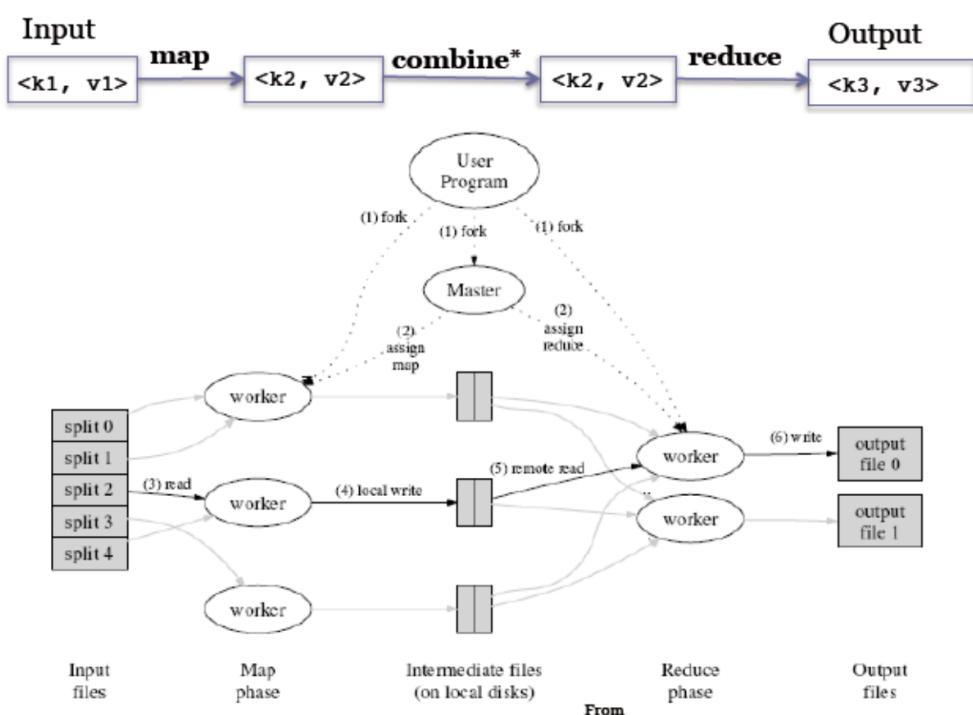
## Streams Application



•  
•  
•

•  
•

•



$\langle \text{key}, \text{ value} \rangle$

- hello world hello moon
- goodbye world goodnight moon

```
<hello, 1>
<world, 1>
<hello, 1>
<moon, 1>
```

```
<goodbye, 1>
<world, 1>
<goodnight, 1>
<moon, 1>
```

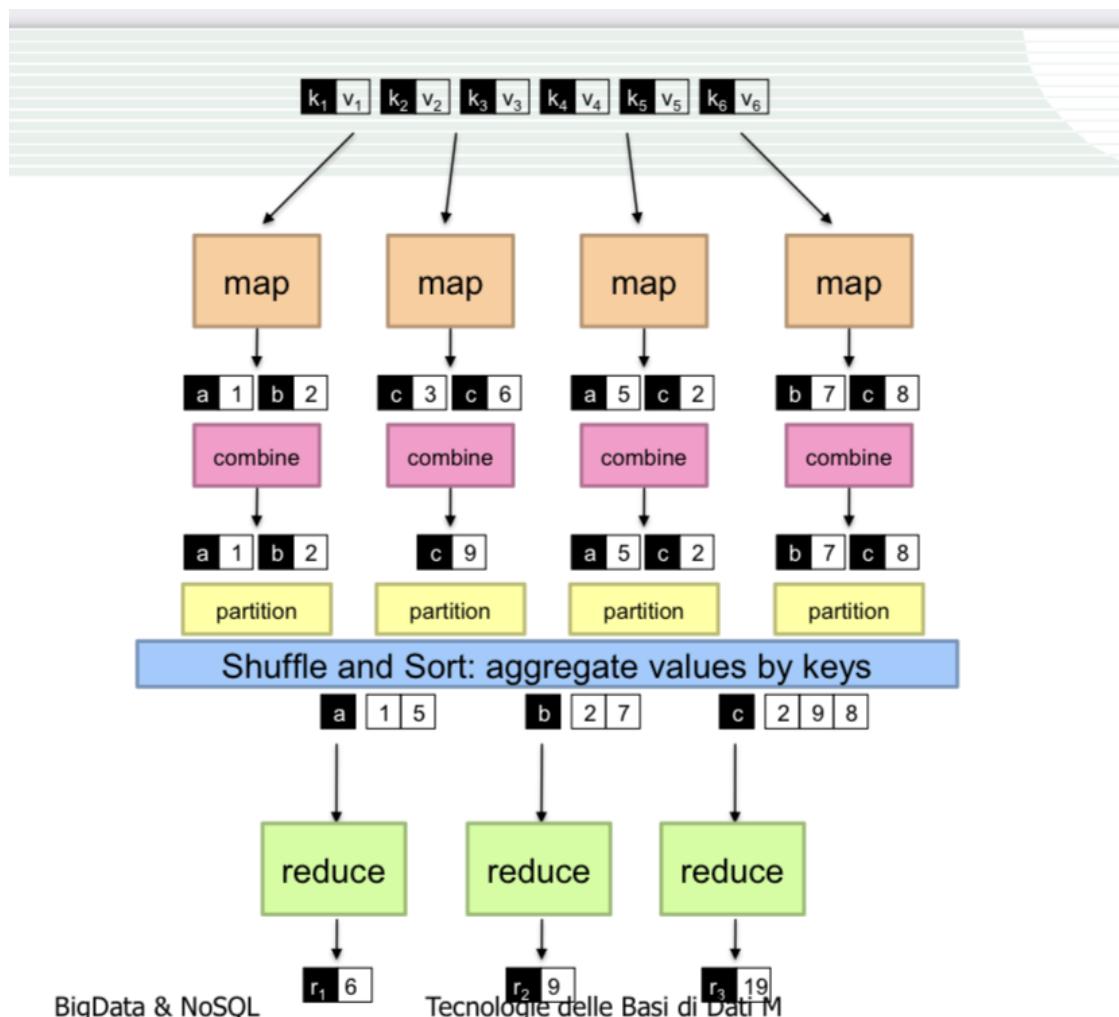
```
<moon, 1>
<world, 1>
<hello, 2>
```

```
<goodbye, 1>
<world, 1>
<goodnight, 1>
<moon, 1>
```

```
<goodbye, 1>
<goodnight, 1>
<moon, 2>
```

```
<world, 2>
<hello, 2>
```

- 
- 
- 





select

<b>Thread</b>	<b>Asynchronous Event-driven</b>
Blocca applicazione/richieste con listener-worker thread	Un solo thread, che fa ripetutamente fetching di eventi da una coda
Usa modello incoming-request	Usa una coda di eventi e processa eventi presenti
Multithreaded server potrebbe bloccare una richiesta che coinvolge eventi multipli	Salva stato e passa poi a processare il prossimo evento in coda
Usa context switching	No contention e NO context switch
Usa ambienti multithreading in cui listener e worker thread spesso acquisiscono incoming-request lock	Usa framework con meccanismi per cosiddetto I/O asincrono (callback, NO poll/select, O_NONBLOCK)

sendReply

```
request = readRequest(socket);
reply = processRequest(request);
sendReply(socket, reply);
```

readRequest

read

```
startRequest(socket);
listen("requestAvail", processRequest);
listen("processDone", sendReplyToSock);
```

startRequest

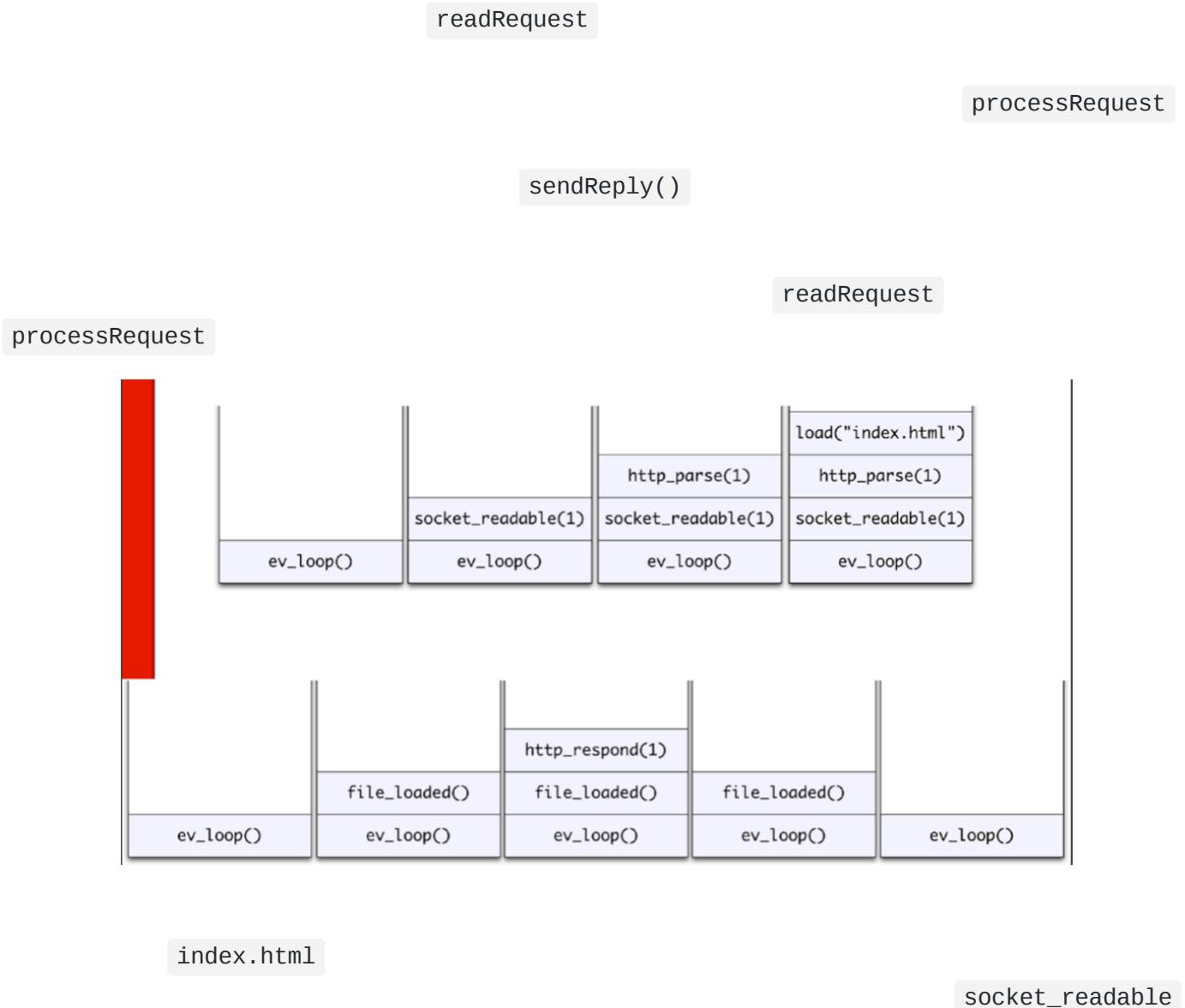
listen

processDone

```

readRequest(socket, function(request) {
 processRequest(request,
 function (reply) {
 sendReply(socket, reply);
 });
}

```



```

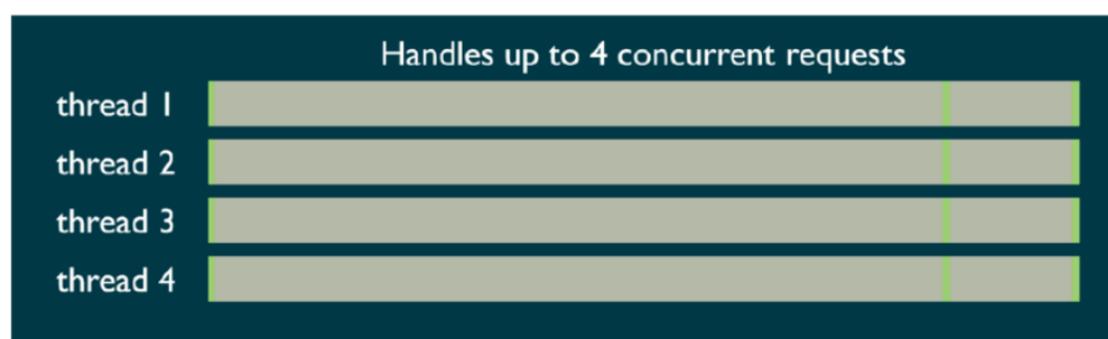
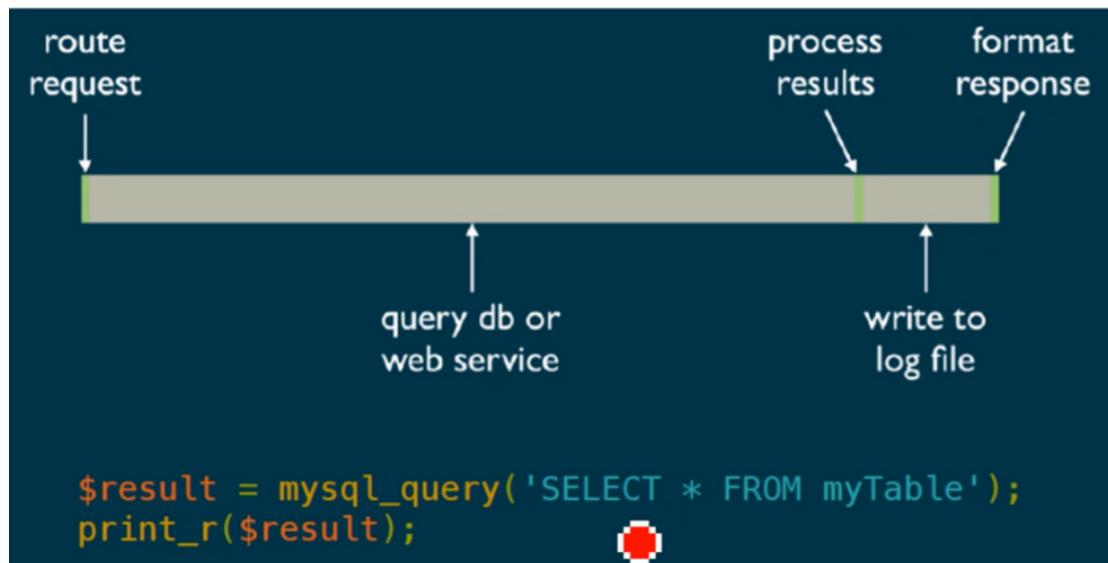
while(true) {
 if (!eventQueue.notEmpty()) {

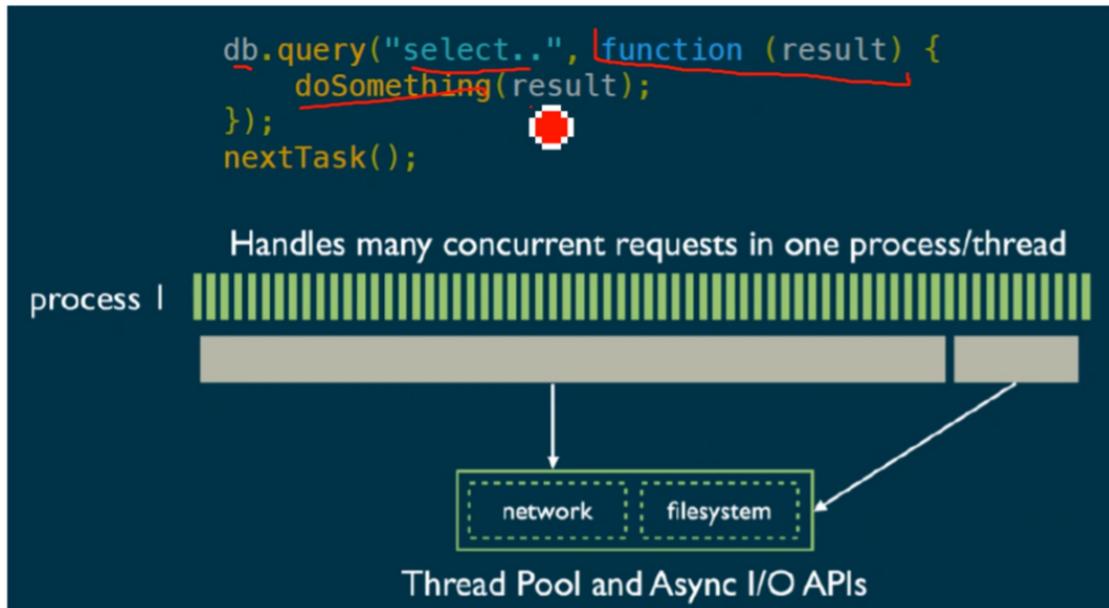
```

```
 eventQueue.pop().call();
}
}
```

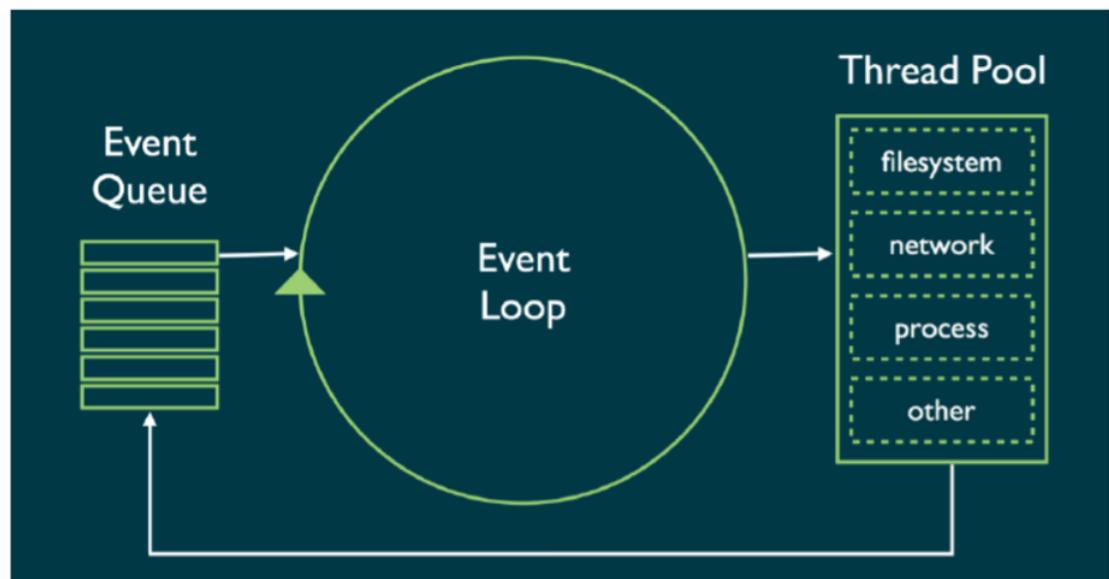
```
 call()
```

```
 push()
```





doSomething



```
require
```

```
// carica il modulo http per creare un http server
var http=require('http');
// configura HTTP server per rispondere con Hello World
var server=http.createServer(function(request,response) {
 response.writeHead(200, {"Content-Type":"text/plain"});
 response.end("Hello World\n");
});
// ascolta su porta 8000
server.listen(8000);
// scrive un messaggio sulla console terminale
console.log("Server running at http://127.0.0.1:8000/");
```

World

Hello

```
require
```

```
read
```

```
var fs = require("fs");
// modulo fs richiesto oggetto fs fa da wrapper a chiamate bloccanti sui file
// read() a livello SO è sincrona bloccante mentre
// fs.readFile è non-bloccante
fs.readFile("smallFile", readDoneCallback); // inizio lettura

function readDoneCallback(error, dataBuffer) {
 // convenzione Node per callback: primo argomento è oggetto
 // js di errore
 if (!error) {
 console.log("smallFile contents", dataBuffer.toString());
 }
}
```

```
var readableStreamEvent = fs.createReadStream("bigFile");
readableStreamEvent.on('data', function (chunkBuffer) {
 console.log('got chunk of', chunkBuffer.length, 'bytes');
});
//operazione eseguita ogni volta che arriva un chunck di dati

readableStreamEvent.on('end', function() {
 // Lanciato dopo che sono stati letti tutti i datachunk fine dello stream
 console.log('got all the data');
});

readableStreamEvent.on('error', function (err) {
 console.error('got error', err);
});
//gestione a evento dell'errore
```

```
var writableStreamEvent = fs.createWriteStream('outputFile');
writableStreamEvent.on('finish', function () {
 console.log('file has been written!');
});

writableStreamEvent.write('Hello world!\n');
writableStreamEvent.end();
```

```
var net = require('net');
net.createServer(processTCPconnection).listen(4000);
```

### processTCPconnection

```
// lista di client connessi
var clients = [];
function processTCPconnection(socket) {

 // aggiunge il cliente alla lista
 clients.push(socket);
 socket.on('data', function (data) {
 // invia a tutti i dati ricevuti
 broadcast("> " + data, socket);
 });

 socket.on('end', function () {
 // remove socket
 clients.splice(clients.indexOf(socket), 1);
 });
}

// invia messaggio a tutti i clienti
function broadcast(message, sender) {

 clients.forEach(function (client) {
 if (client === sender)
 return;
 client.write(message);
 });
}
```

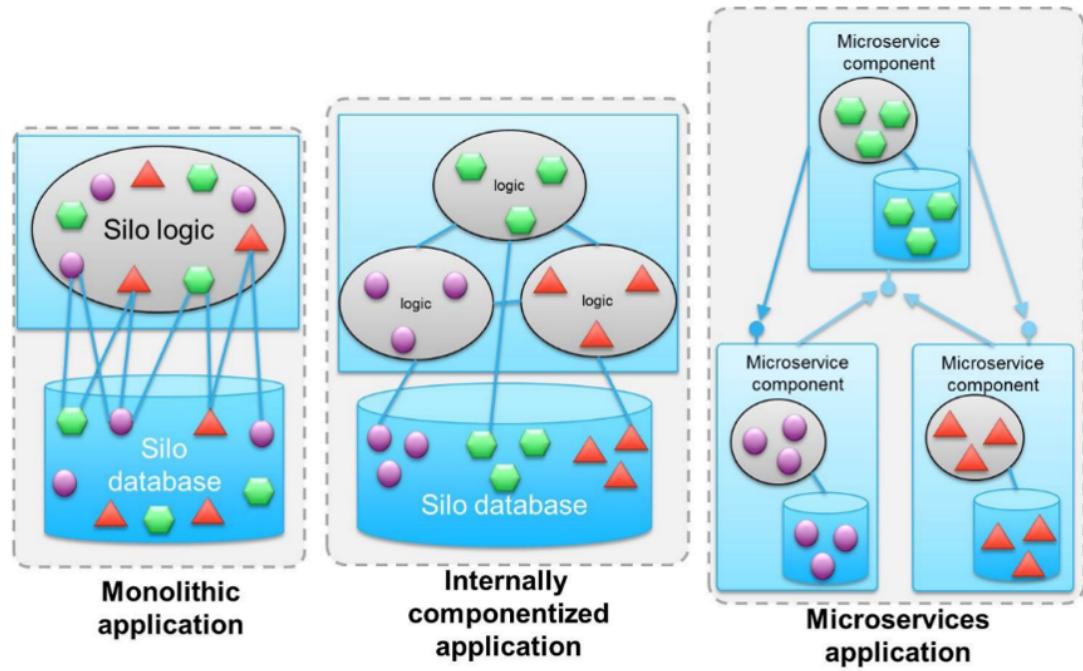
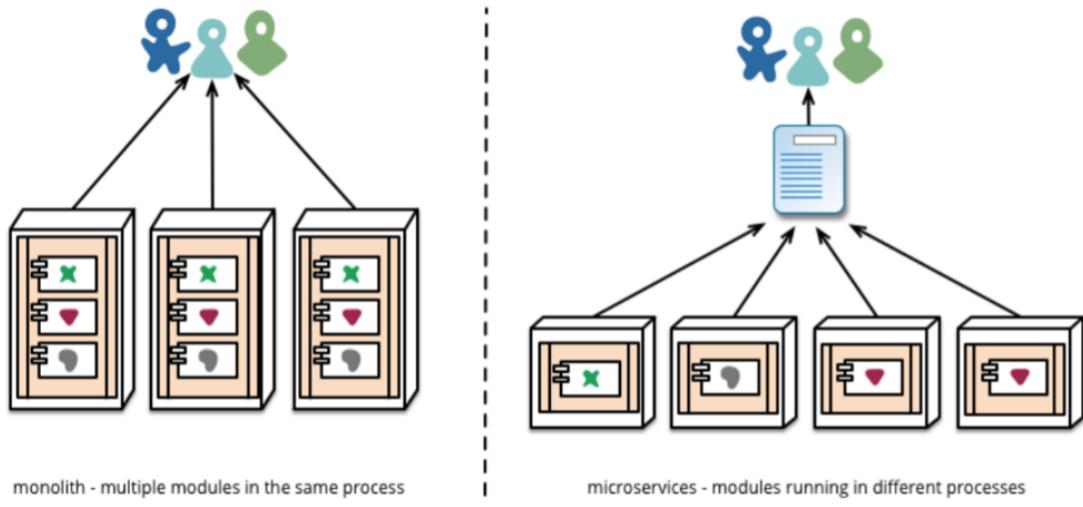
### processTCPConnection

## Sinatra

```
var express=require('express');
var app=express();
app.get('/', function(req,res) {res.send('Hello World!'); });
var server=app.listen(3000,function() {
 var host=server.address().address;
 var port=server.address().port;
 console.log('Listening at http://%s:%s',host,port);
});
```

Hello World!

---

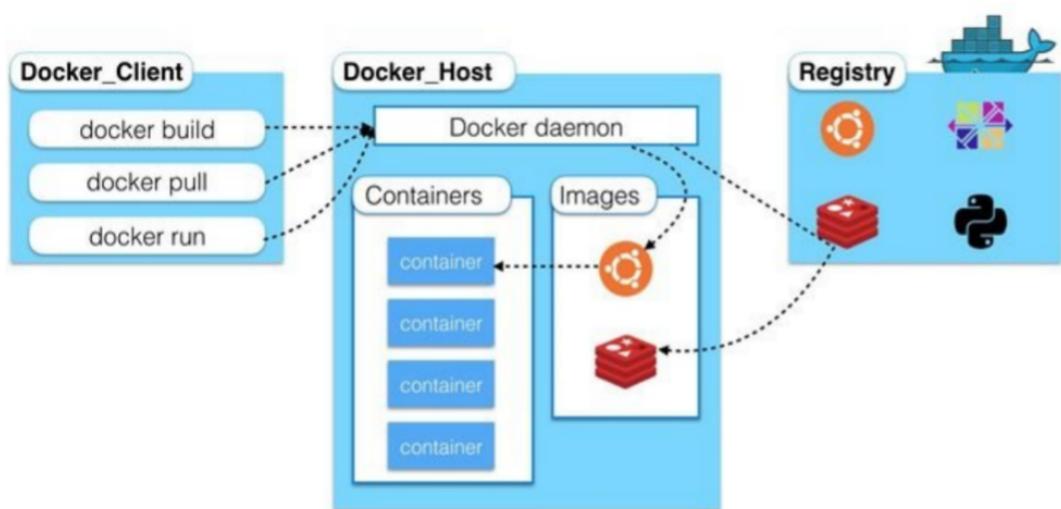




visor

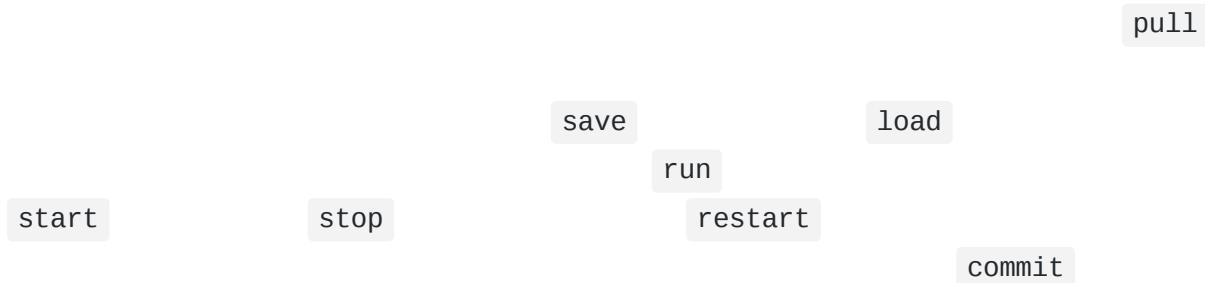
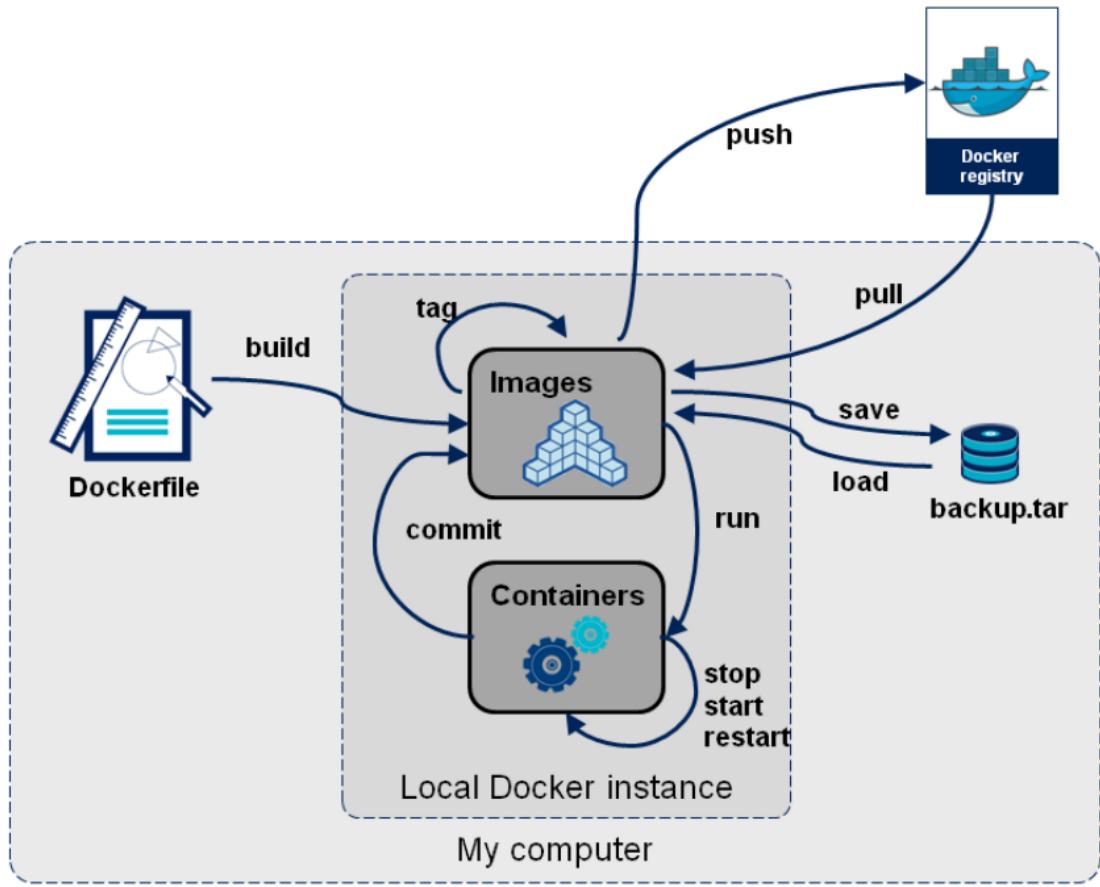
- Container engine

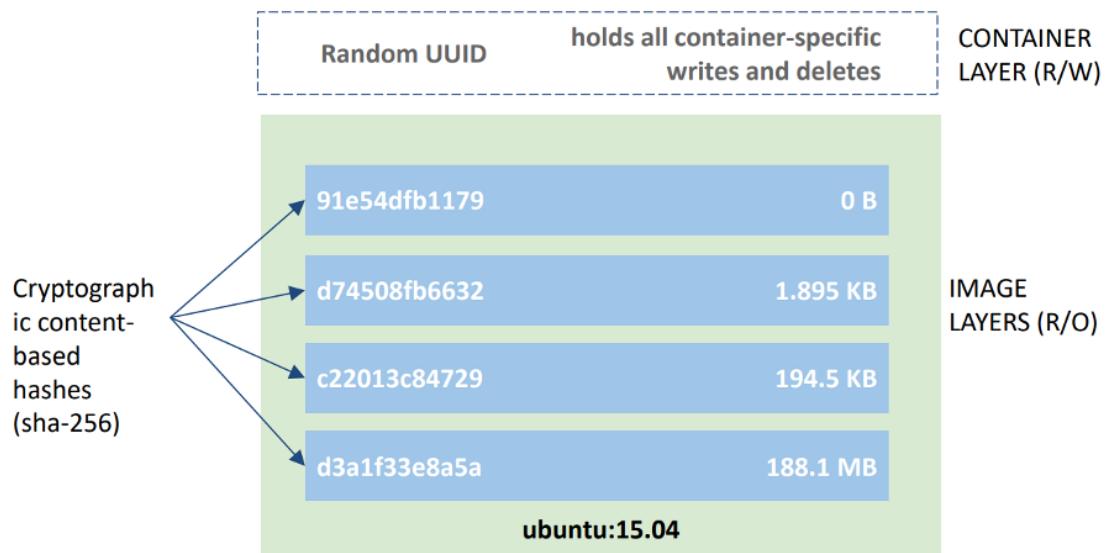
	<b>Process</b>	<b>Container</b>	<b>VM</b>
<b>Definition</b>	A representation of a running program.	Isolated group of processes managed by a shared kernel.	A full OS that shares host hardware via a hypervisor.
<b>Use case</b>	Abstraction to store state about a running process.	Creates isolated environments to run many apps.	Creates isolated environments to run many apps.
<b>Type of OS</b>	Same OS and distro as host,	Same kernel, but different distribution.	Multiple independent operating systems.
<b>OS isolation</b>	Memory space and user privileges.	Namespaces and cgroups.	Full OS isolation.
<b>Size</b>	Whatever user's application uses.	Images measured in MB + user's application.	Images measured in GB + user's application.
<b>Lifecycle</b>	Created by forking, can be long or short lived, more often short.	Runs directly on kernel with no boot process, often is short lived.	Has a boot process and is typically long lived.



`docker run`

run   started   stopped   moved   deleted

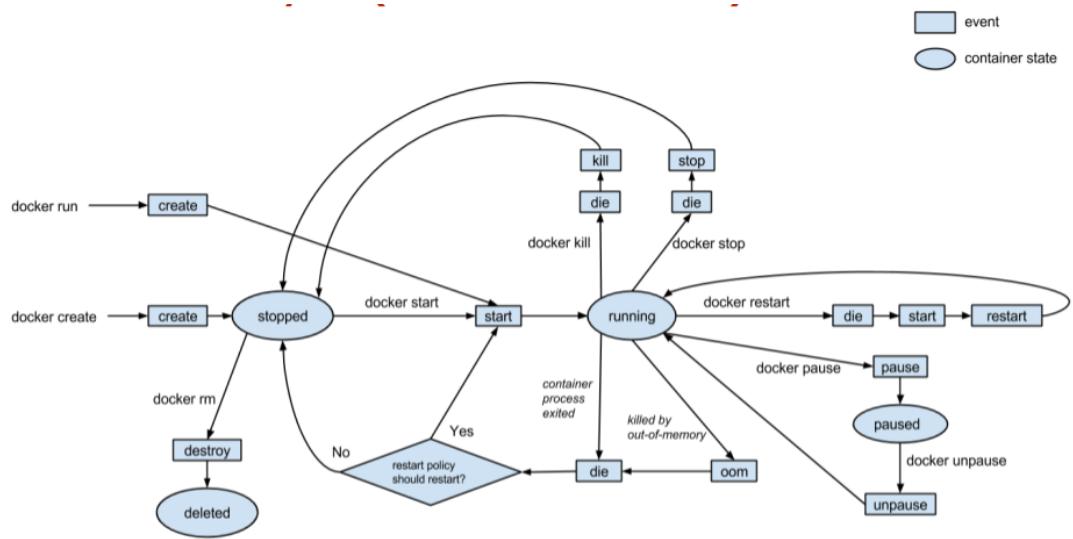


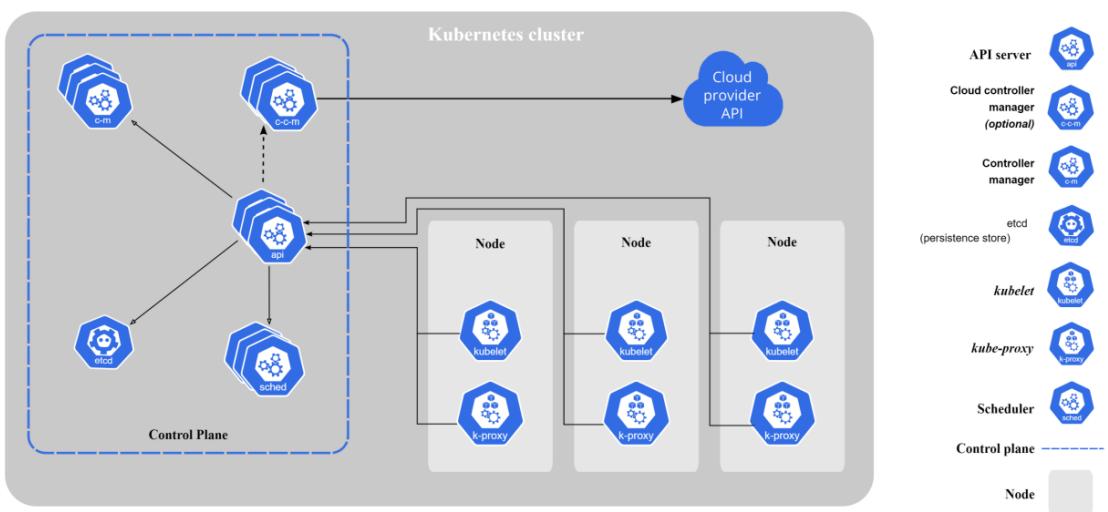


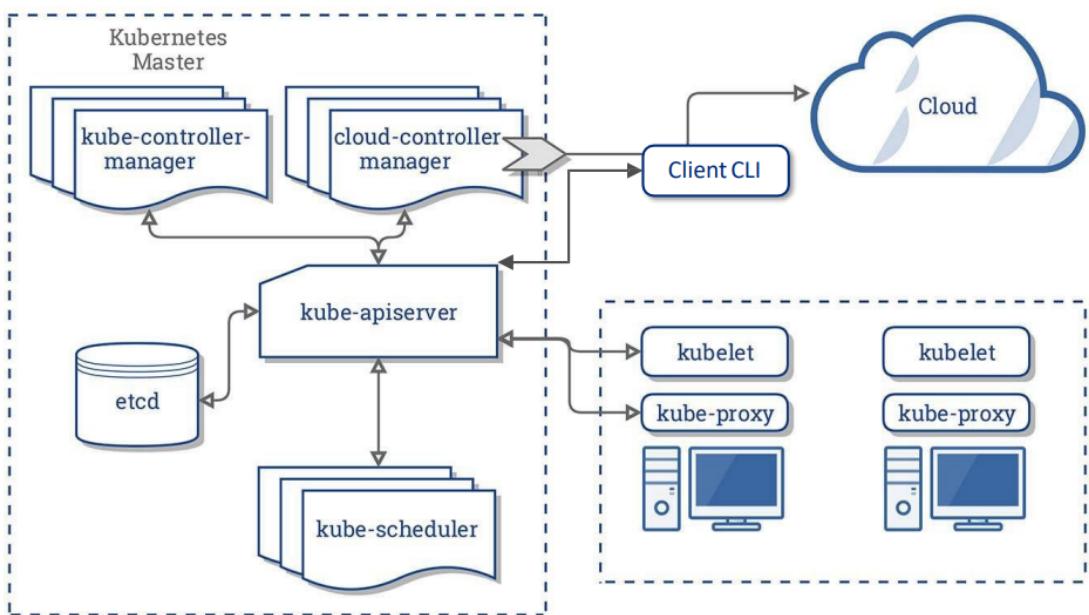
[hostname[:port]]/[username]/reponame[:tag]

run









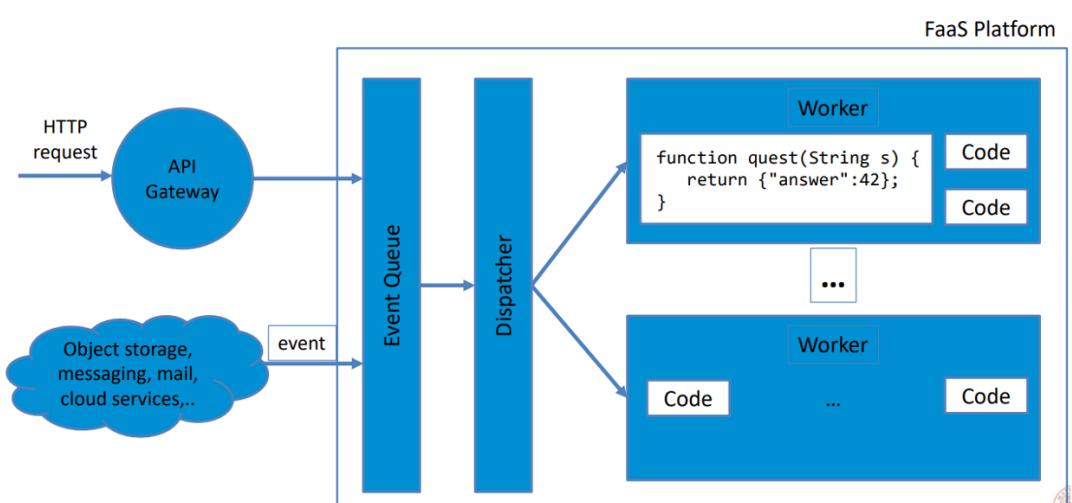
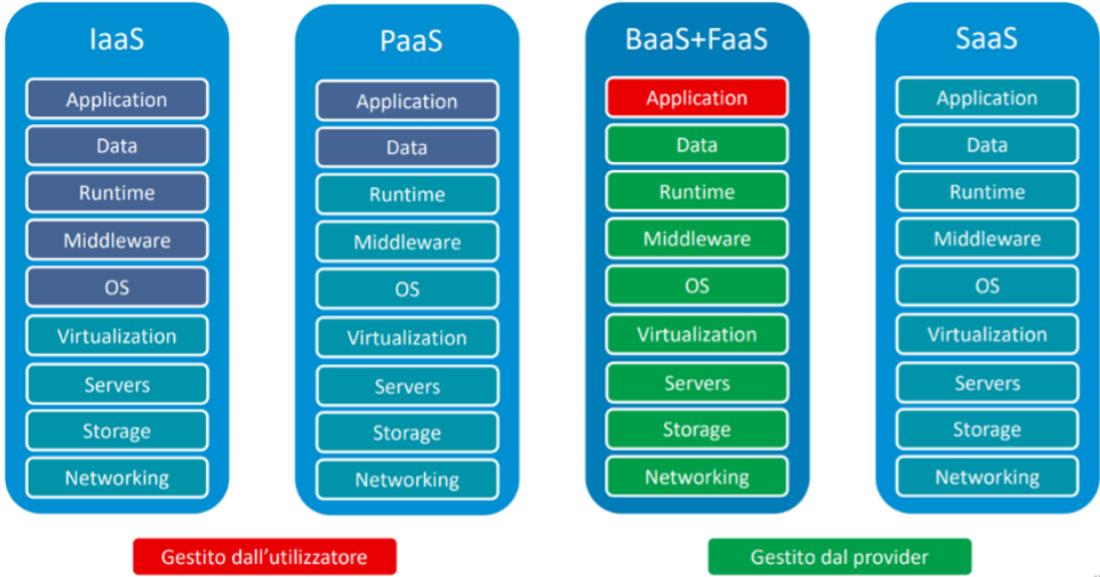




•  
•  
•  
•  
•







•  
•  
•  
  
•  
•

