

Enterprise Java Beans 2 (EJB)



EJB Nedir?

- Enterprise uygulamaların server tarafında çalışan **distributed java bileşenleridir**
- Uygulama sunucuları EJB bileşenlerinin çalışabilmesi için **ihtiyaç duydukları ortamı** sağlar
- Buna **EJB Container** adı verilir
- Uygulama sunucuları, distribution, thread yönetimi, güvenlik, transaction, veritabanı bağlantıları, mesajlaşma gibi **altyapısal servisler** sunar

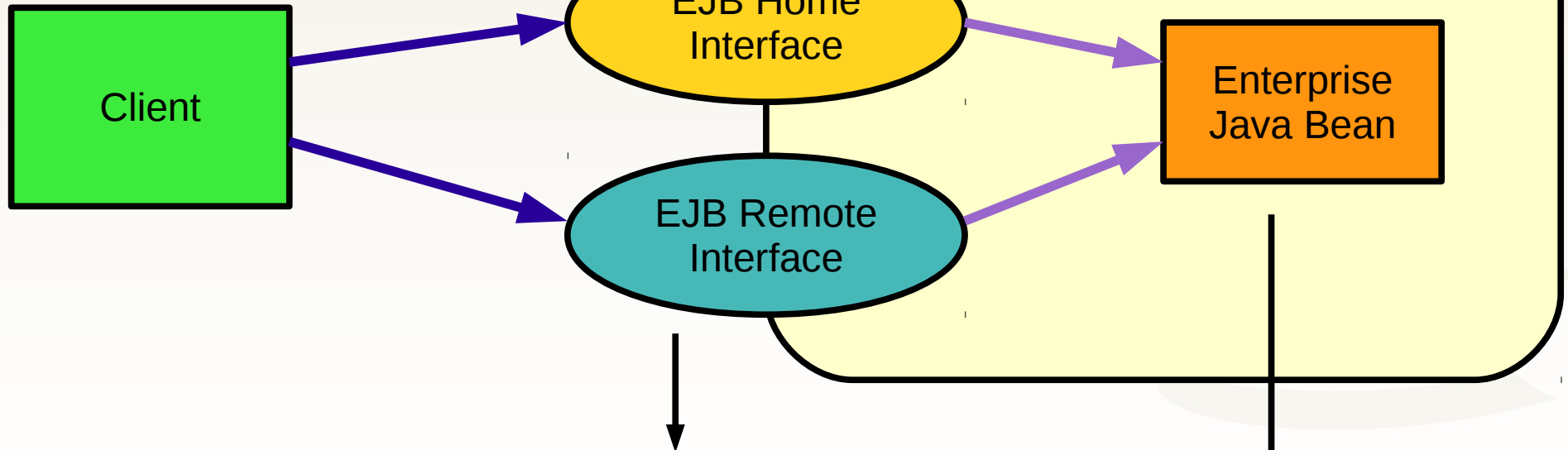
EJB'nin Faydaları

- EJB geliştiricileri bu altyapısal servislerle ilgilenmek yerine **iş mantığını implement etmeye odaklanabilirler**
- Altyapısal servislerin ve bileşenlerin container tarafından yönetilmesi ile **daha esnek ve ölçeklenebilir sistemler üretilmesi** mümkün hale gelmektedir

EJB Mimarisi

Home interface, EJB ile ilgili lifecycle metotlar sunar
Bileşenin yaratılması, bileşene erişilmesi, bileşenin
Silinmesi gibi.

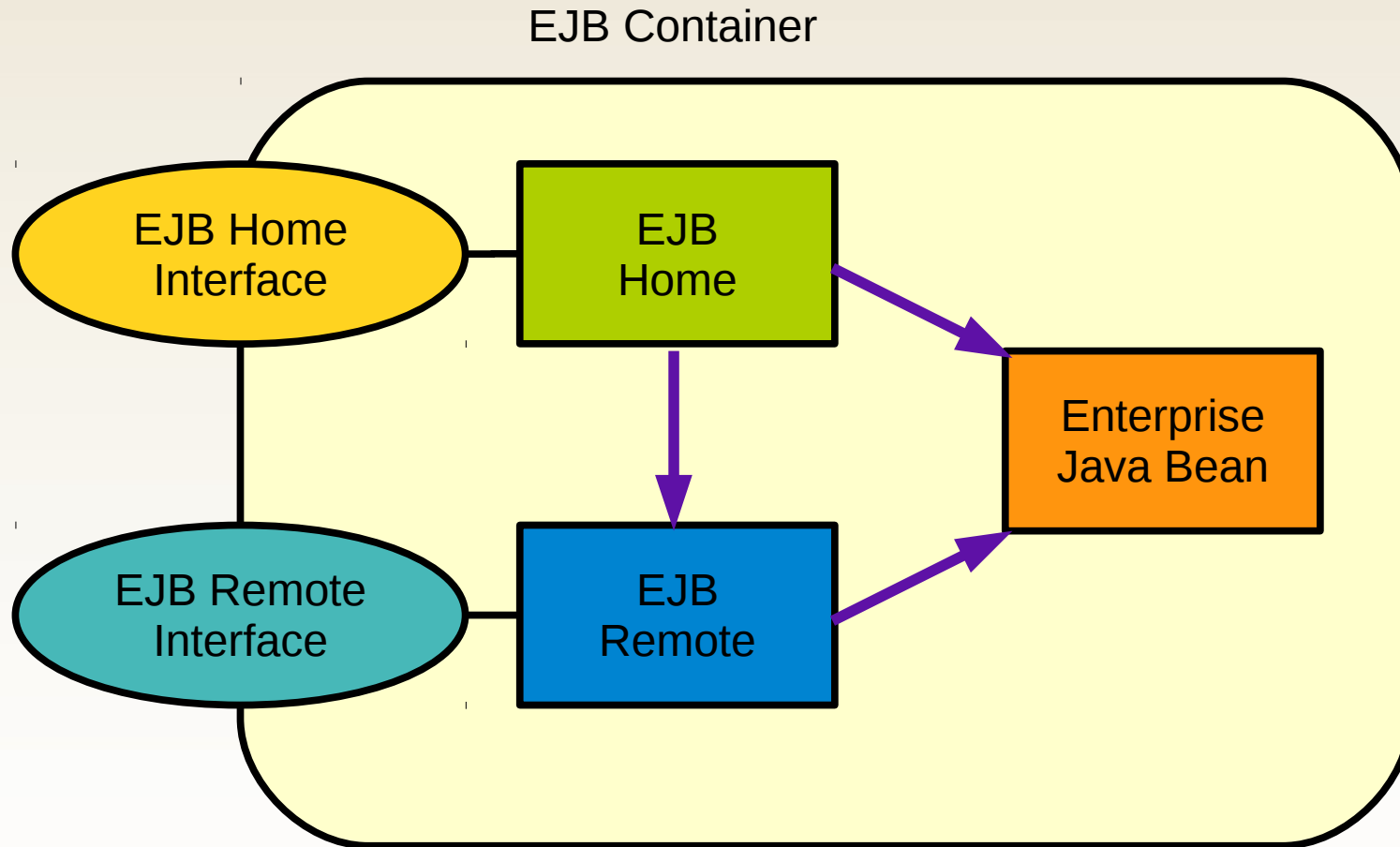
Client öncelikle JNDI üzerinden
Home instance'a erişir
Home instance vasıtası ile Remote
Instance yaratılır ve kullanılır



Remote interface EJB'nin client tarafından
Erişilebilecek metotlarını tanımlar

Remote interface'de tanımlı metotları ve
Bacı callback metotlarını implement eder

EJB Mimarisi



EJB Home ve EJB Remote sınıfları EJB Container tarafından dinamik olarak Oluşturulmaktadır

Client ve EJB arasındaki iletişim RMI over IIOP üzerinden gerçekleşmektedir

EJB Tipleri

- **Stateless Session Bean**

- İş mantığının implement edildiği bileşenlerdir
- İki request arasında herhangi bir **state bilgisi saklamazlar**
- **State yönetimi gerektirmeyen** iş mantığının implement edilmesinde kullanılırlar

- **Stateful Session Bean**

- İş mantığının implement edildiği bileşenlerdir
- Metot çağrıları arasında **state bilgisi saklarlar**
- Herhangi bir **iş akışının olduğu senaryolarda** kullanılırlar

EJB Tipleri

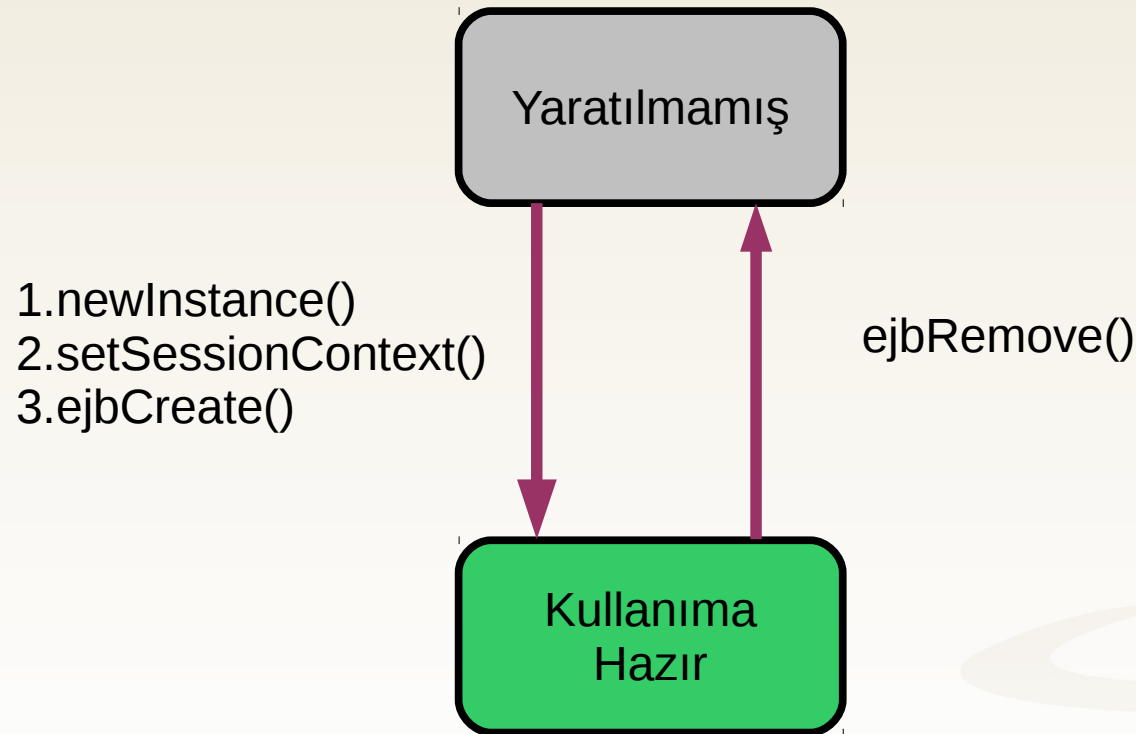
■ Entity Bean

- **Domain modele** karşılık gelen bileşenlerdir
- Veritabanındaki **tablolarla eşleşirler**
- **Verinin persist edilmesinde** kullanılırlar
- Her entity bean **instance'ının veritabanındaki bir kayda karşılık gelmesi** söz konusudur

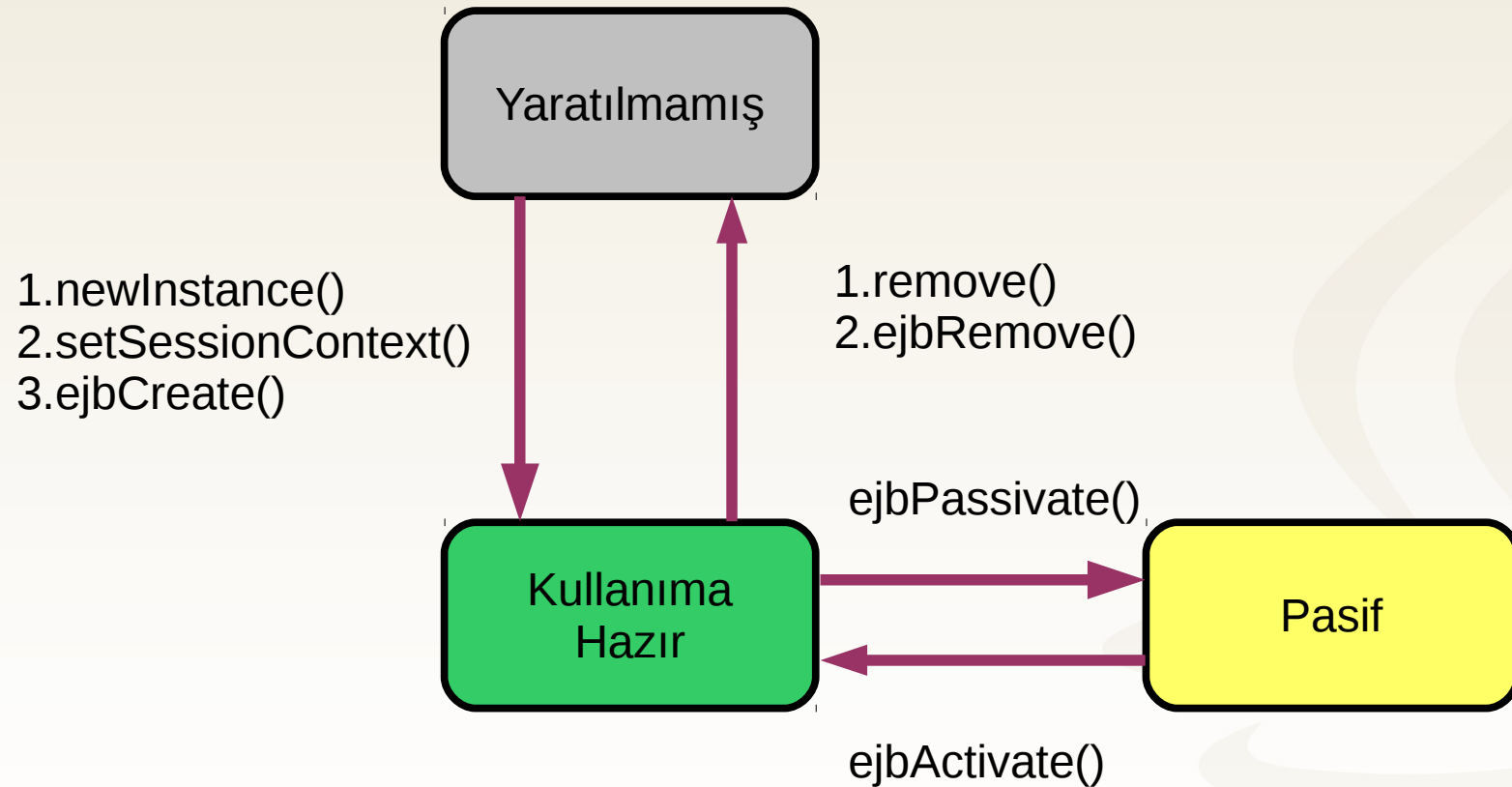
■ Message Driven Bean

- **Mesajlaşma işlemlerinde** kullanılan bileşenlerdir
- **JMS mesajlarını** işleyebilirler
- **Asenkron iletişim** imkanı sağlarlar

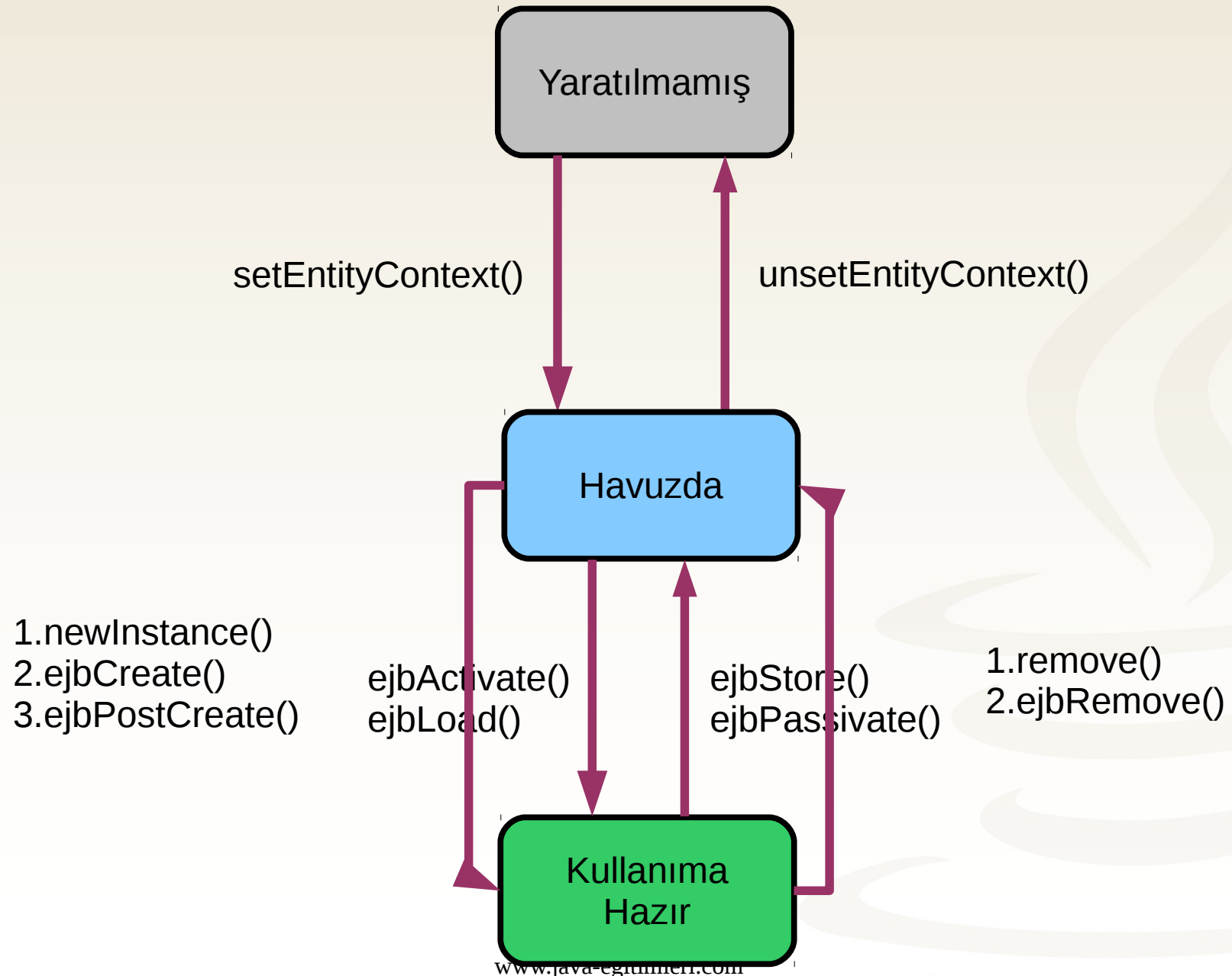
Stateless Session Bean Yaşam Döngüsü



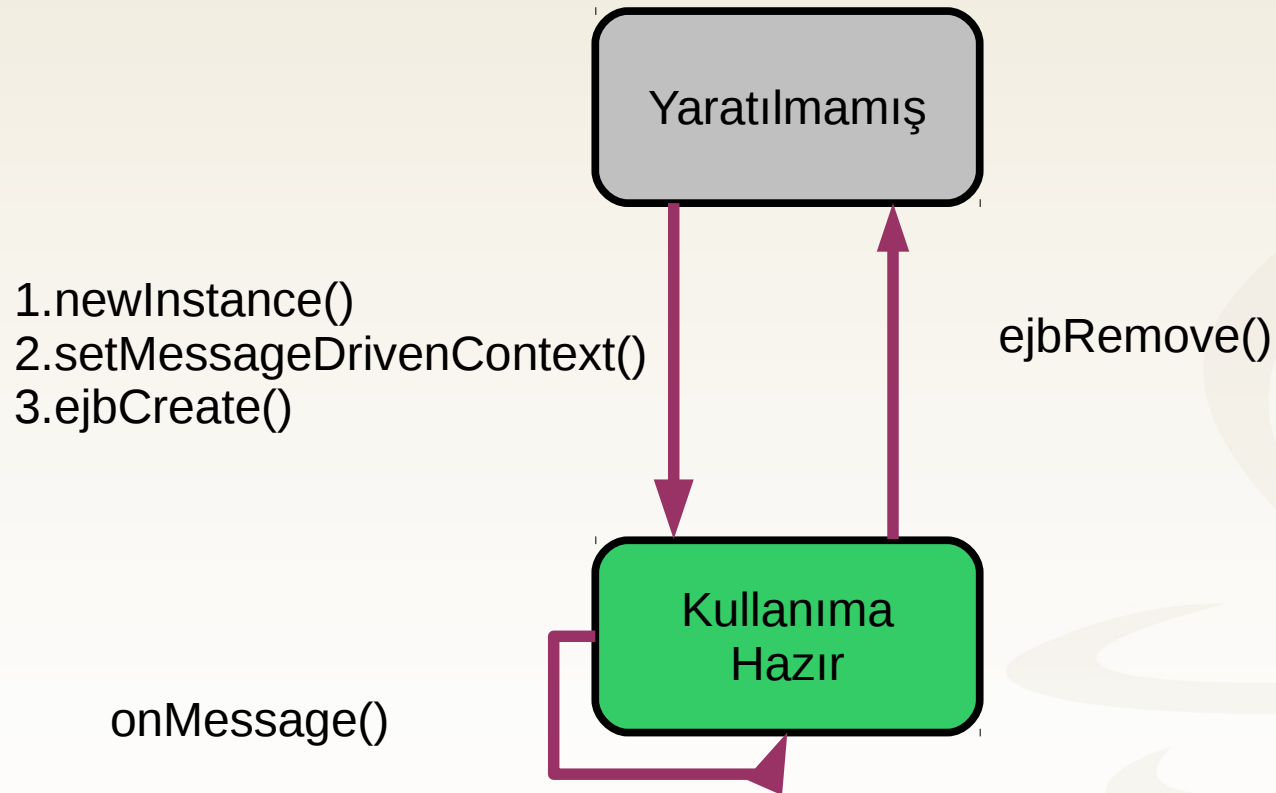
Stateful Session Bean Yaşam Döngüsü



Entity Bean Yaşam Döngüsü



Message Bean Yaşam Döngüsü



Local Interface

- EJB'ler **distributed mimariler için** tasarlanmıştır
- Bu durumda EJB bileşenlerine **farklı JVM'den erişim** söz konusudur
- Ancak çoğu zaman EJB bileşenlerine **aynı JVM içerisinden erişim** de olabilmektedir
- Aynı JVM içerisinden erişimlerde, **performans artışı sağlamak** amacı ile local interface'ler vardır

Local Interface

- Remote Interface ve Home Interface'lerin birde **lokal karşılıkları** tanımlanır
- **EJBObject**'in local karşılığı **EJBLocalObject**
- **EJBHome**'un local karşılığı **EJBLocalHome**

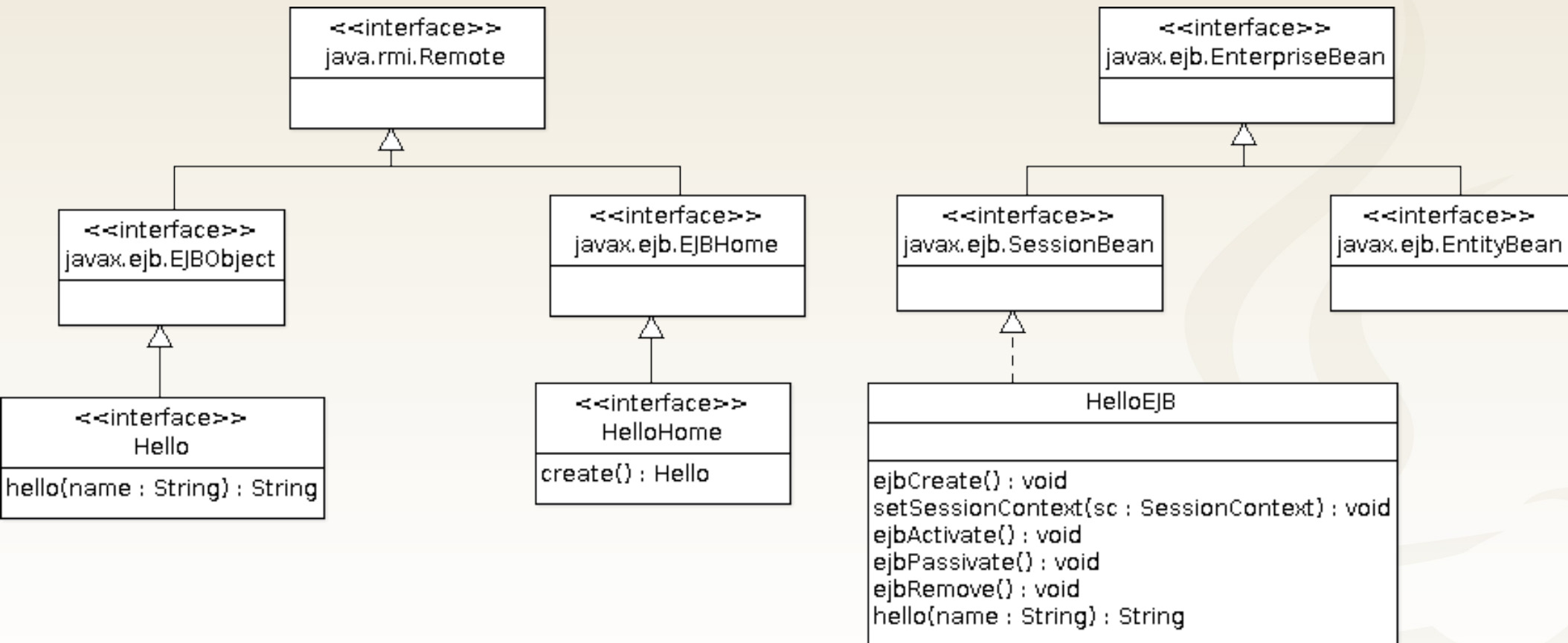
Ne Zaman Local Ne Zaman Remote Interface Kullanılmalı?

- Eğer EJB farklı JVM'deki bir client tarafından erişilecek ise **remote interface**
- Eğer metod parametrelerinin **mutlaka pass by value** olarak taşınması isteniyor ise **remote interface**
- **Entity bean'ler** çoğunlukla bir Session Bean üzerinden işlem görürler, bu nedenle **local interface**
- **Container managed relationship** için **local interface**

Ne Zaman Local Ne Zaman Remote Interface Kullanılmalı?

- Eğer bir EJB sadece diğer EJB'lerden erişiliyor ise **local interface**
- Eğer bir EJB sadece aynı application server içindeki web bileşenlerinden erişiliyor ise **local interface**

Session Bean Örneği



Session Bean Örneği

- Home interface içinde **bir veya daha fazla create(..)** metodu tanımlanır
- Her bir create metodunun EJB sınıfında **ejbCreate karşılığı** olmalıdır
- Home ve Remote interface'de tanımlı bütün metotlar java.rmi.**RemoteException** fırlatmalıdır
- Home iterface'deki **create** ve EJB'deki **ejbCreate** metodu javax.ejb.**CreateException** fırlatmalıdır
- EJB sınıfındaki diğer **callback metotları** javax.ejb.**EJBException** fırlatmalıdır

EJB Deployment

- EJB interface ve sınıfları **bir jar dosyası** içerisinde bir araya getirilir
- Buna **ejb jar** adı verilir
- Bu jar da /META-INF/**ejb-jar.xml** içerisinde EJB konfigürasyonu yapılır
- Her application server'ın **kendine has** bir **deployment descriptor** dosyası da META-INF altında bulunabilir
- **Sunucuya özel tanımlar** da bu dosyada yapılır

ejb-jar.xml Örneği

```
<ejb-jar>
  <display-name>MyEJB</display-name>
  <enterprise-beans>
    <session>
      <ejb-name>Hello</ejb-name>
      <home>example.HelloHome</home>
      <remote>example.Hello</remote>
      <ejb-class>example.HelloBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Enterprise Java Archive Dosyası (EAR)

- **Web, EJB modüllerinin ve diğer yardımcı kütüphanelerin bir araya getirilmesiyle oluşur**
- Bir ear içerisinde **birden fazla** web ve ejb modülü olabilir
- **Ear uzantısına** sahip bir **JAR formatında** dosyadır
- İçerisinde META-INF/**application.xml** isimli bir deployment descriptor dosyası bulunur
- Ayrıca META-INF dizini altında **server'a özel deployment descriptor dosyaları** da bulunabilir

Servlet İçinden EJB Erişimi

```
InitialContext context = new InitialContext();  
HelloHome home = (HelloHome) context.lookup("java:comp/env/ejb/Hello");  
Hello hello = home.create();  
response.getWriter().write(hello.hello("kenan"));
```

EJB application server tarafından JNDI ağacında bir yere bind edilir, Bu yer her server için Farklı olabilir. Web.xml'deki <ejb-ref> tanımı ile bu binding'e java:comp/env node'u altından Referans verilmiş olunur. Bir nevi pointer'a benzetilebilir.

Bu sayede EJB home'a Servlet içinden standart bir JNDI name ile lookup yapmak mümkün Hale gelir

```
<web-app>  
  <ejb-ref>  
    <description>Hello Session Bean</description>  
    <ejb-ref-name>ejb/Hello</ejb-ref-name>  
    <ejb-ref-type>Session</ejb-ref-type>  
    <home>example.HelloHome</home>  
    <remote>example.Hello</remote>  
  </ejb-ref>  
</web-app>
```

EJB – EJB İletişimi

```
GoodbyeHome home = (GoodbyeHome) sessionContext.lookup("Goodbye");
//sessionContext.lookup("java:comp/env/ejb/Goodbye");
Goodbye goodbye = home.create();
String msg = goodbye.goodbye(name);
```

```
<session>
  <ejb-name>Hello</ejb-name>
  ...
  <ejb-ref>
    <description>Goodbye Session Bean</description>
    <ejb-ref-name>ejb/Goodbye</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>example.GoodbyeHome</home>
    <remote>example.Goodbye</remote>
    <ejb-link>Goodbye</ejb-link>
  </ejb-ref>
</session>
```

Kullanımı opsiyoneldir
Ancak kodun içinde
JNDI name'in statik
Olarak yazılmasını
Ortadan kaldırdığı için
Fardalı bir tanımdır

```
<session>
  <ejb-name>Goodbye</ejb-name>
  ...
</session>
```

EJB – DataSource Erişimi

```
InitialContext context = new InitialContext();
DataSource dataSource = (DataSource)
context.lookup("java:comp/env/jdbc/testdb");
Connection c = dataSource.getConnection();
CallableStatement cstmt = c.prepareCall("call curdate()");
cstmt.execute();
ResultSet rs = cstmt.getResultSet();
rs.next();
return "goodbye " + name + " " + rs.getDate(1).toString();
```

<session>

...

<resource-ref>

<res-ref-name>jdbc/testdb</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

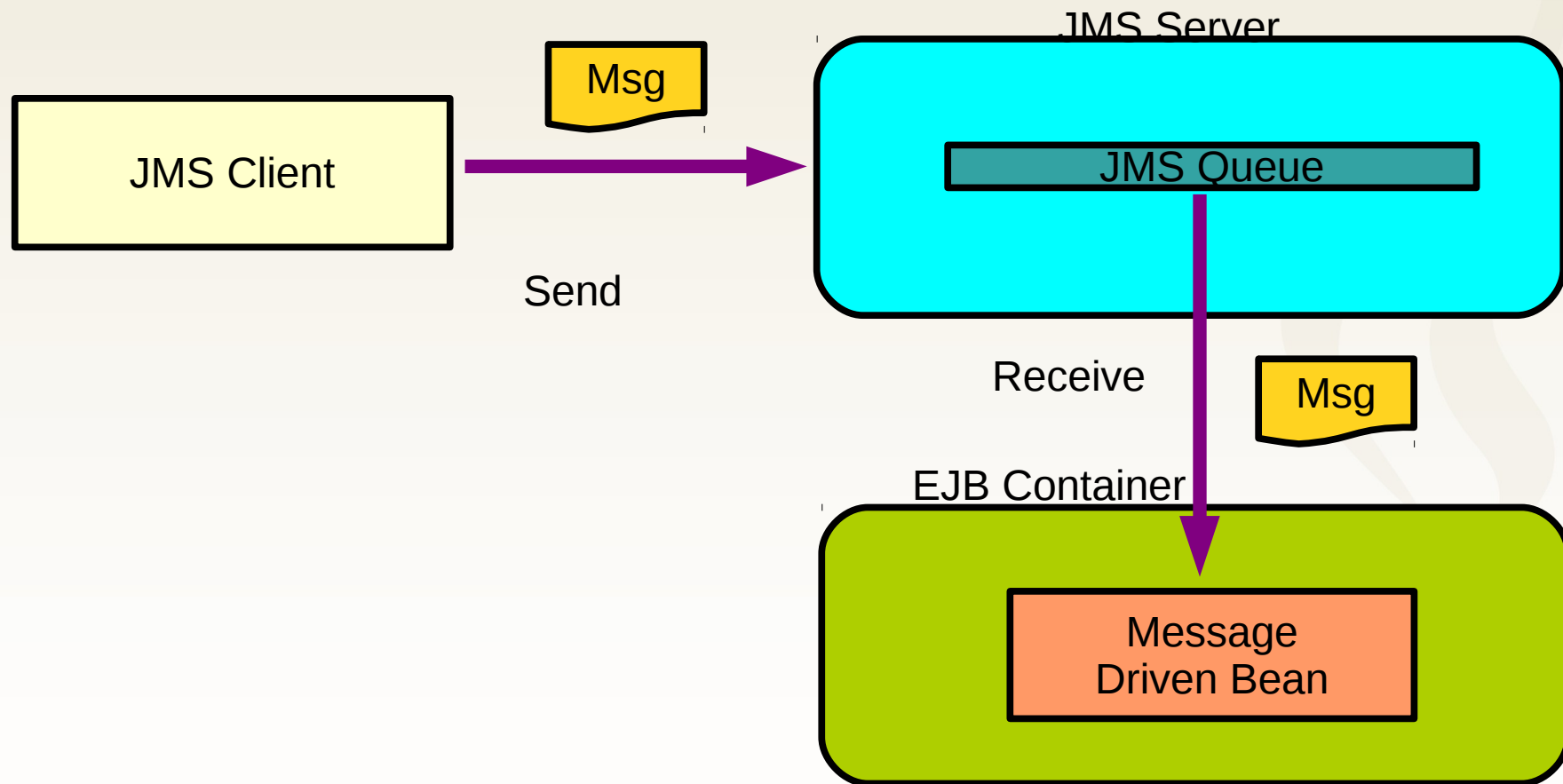
<res-auth>Container</res-auth>

</resource-ref>

</session>

javax.sql.DataSource, javax.mail.Session,
javax.jms.QueueConnectionFactory, javax.jms.TopicConnectionFactory
gibi container tarafından yönetilen resource factory nesnelerine erişim
imkanı sağlar


Message Driven Bean



Message Driven Bean Örneği

```
public class HelloMDB implements MessageDrivenBean,  
    MessageListener {  
  
    private MessageDrivenContext context;  
  
    public void ejbCreate() {}  
  
    public void ejbRemove() {}  
  
    public void setMessageDrivenContext(MessageDrivenContext  
context) {  
        this.context = context;  
    }  
  
    public void onMessage(Message message) {  
        TextMessage txtMessage = (TextMessage)message;  
        System.out.println(txtMessage.getText());  
    }  
}
```

Message Driven Bean Örneği

```
<ejb-jar>  ejb-jar.xml
  <enterprise-beans>
    <message-driven>
      <ejb-name>HelloMDB</ejb-name>
      <ejb-class>example.HelloMDB</ejb-class>
      <transaction-type>Container</transaction-type>
      <message-driven-destination>
        <destination-type>javax.jms.Queue</destination-type>
      </message-driven-destination>
    </message-driven>
  </enterprise-beans>
</ejb-jar>
```

```
<?xml version='1.0' encoding='UTF-8' ?>
<jboss>
  <enterprise-beans>
    <message-driven>
      <ejb-name>HelloMDB</ejb-name>
      <destination-jndi-name>queue/hello</destination-jndi-name>
    </message-driven>
  </enterprise-beans>
</jboss>
```

jboss.xml

Queue konfigürasyonu da Server'a özel biçimde yapılır

resource-env-ref Kullanımı

- **resource-ref** DataSource, ConnectionFactory gibi server tarafından yönetilen **resource factory nesnelerine erişim** sağlamaktaydı
- **resource-env-ref** ise server tarafından yönetilen Queue ve Topic gibi **resource nesnelerine erişim** sağlar
- Servlet ve EJB içerisinden bu nesneler, **JNDI lookup** yapılarak kullanılabilir
- **web.xml** veya **ejb-jar.xml** içinde kullanılır

resource-env-ref Kullanımı

```
<session>
...
<resource-ref>
  <description>JMS Queue ConnectionFactory</description>
  <res-ref-name>ConnectionFactory</res-ref-name>
  <res-type>javax.jms.QueueConnectionFactory</res-type>
  <res-auth>Container</res-auth>
</resource-ref>

<resource-env-ref>
  <description>JMS Queue</description>
  <resource-env-ref-name>queue/hello</resource-env-ref-name>
  <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>
</resource-env-ref>
</session>
```

Container Managed Persistence ve Relationship

- Entity nesnenin field'larının **container tarafından persist edilmesidir**
- Entity nesneler arasındaki **ilişkiler de EJB container tarafından yönetilir**
- CMR ile CMP yakından alakalıdır

Container Managed Persistence ve Relationship

- Entity nesnenin **persistent field**'ları bean sınıfı içerisinde tanımlanmaz
- Persistent field'lara karşılık gelen **abstract getter/setter metotları** tanımlanır
- Container entity bean sınıfını **runtime'da extend** ederek bu metotları implement eder

Entity Bean Örneği

```
public interface Owner extends EJBObject {
    public String getFirstName() throws RemoteException;
    public void setFirstName(String firstName) throws
RemoteException;
    public String getLastName() throws RemoteException;
    public void setLastName(String lastName) throws RemoteException;
}
```

```
public interface OwnerHome extends EJBHome {
    public Owner create(String firstName, String lastName) throws
RemoteException, CreateException;
    public Owner findByPrimaryKey(Integer ownerId) throws
RemoteException, FinderException;
    public Owner findByLastName(String lastName) throws
RemoteException, FinderException;
    public Collection findAllOwners() throws RemoteException,
FinderException;
}
```

Entity Bean Örneği

```
public abstract class OwnerBean implements EntityBean {

    protected EntityContext entityContext;

    public abstract String getFirstName();
    public abstract void setFirstName(String firstName);
    public abstract String getLastName();
    public abstract void setLastName(String lastName);

    public Integer ejbCreate(String firstName, String lastName)
throws CreateException {
    setFirstName(firstName);
    setLastName(lastName);
    return null;
}

    public void ejbPostCreate(String firstName, String lastName) {
    }

    ...
}
```


Entity Bean Örneği

```
public abstract class OwnerBean implements EntityBean {
    ...

    public void ejbActivate() {
    }
    public void ejbPassivate() {
    }

    public void ejbLoad() {
    }
    public void ejbStore() {
    }

    public void ejbRemove() throws RemoveException {
    }

    public void setEntityContext(EntityContext ctx) {
        entityContext = ctx;
    }
    public void unsetEntityContext() {
        entityContext = null;
    }
}
```

Entity Bean Metadata

```
<ejb-jar>
  <persistence-type>container</persistence-type>

  <cmp-version>2.x</cmp-version>

  <cmp-field>
    <field-name>firstName</field-name>
  </cmp-field>

  <cmp-field>
    <field-name>lastName</field-name>
  </cmp-field>

  <query>
    <query-method>
      <method-name>findAllOwners</method-name>
    </query-method>

    <ejb-ql>SELECT OBJECT(o) FROM owner</ejb-ql>
  </query>
</ejb-jar>
```

EJB QL

- **Object oriented sorgu dilidir, SQL'e benzer**
- Sorgunun **veritabanı platformundan bağımsız** olmasını sağlar
- **Entity bean'lara özeldir**
- EJB QL sorgusunun **döndüğü değer**
 - Nesne
 - Bir grup nesneyi içeren collection
 - CMP alanlarından bir kısmı olabilir
- EJBHome'daki **finder metotlarını** implement etmek için kullanılır

EJB QL Örnekleri

- *SELECT* OBJECT (o) *FROM* Order AS o
IN(o.lineltems) li *WHERE*
li.product.product_type=?1 AND
li.product.price=?2
- *SELECT* OBJECT (emp) *FROM* employee as
emp *WHERE* emp.name LIKE 'CHRIS%'
- *SELECT* li.product *FROM* Order As o,
IN(o.lineltems) li
- *SELECT* o.lineltems *FROM* Order As o

Container Managed Relationship Örneği

```
public abstract class CustomerBean implements EntityBean {  
  
    //container-managed relationship field  
    public abstract Address getAddress();  
  
    public abstract void setAddress(Address addr);  
  
    ...  
}
```



Customer-Address 1:1 relationship

```
public abstract class AddressBean implements EntityBean {  
  
    ...  
}
```

Container Managed Relationship Örneği

```
<ejb-jar>
  <relationships>
    <ejb-relation>
      <ejb-relation-name>Customer-Address has 1:1 relationship</ejb-
relation-name>
      <ejb-relationship-role>
        <ejb-relationship-role-name>Customer-Address</ejb-relationship-
role-name>
        <multiplicity>One</multiplicity>
        <relationship-role-source>
          <ejb-name>Customer</ejb-name>
        </relationship-role-source>
        <cmr-field>
          <cmr-field-name>address</cmr-field-name>
        </cmr-field>
      </ejb-relationship-role>
      <ejb-relationship-role>
        <ejb-relationship-role-name>Address-belongs-to-Customer</ejb-
relationship-role-name>
        <multiplicity>One</multiplicity>
        <cascade-delete />
        <relationship-role-source>
          <ejb-name>Address</ejb-name>
        </relationship-role-source>
      </ejb-relationship-role>
    </ejb-relation>
  </relationships>
</ejb-jar>
```

Bean Managed Persistence

- Entity bean instance'ın **persistence işlemlerinin bean içerisinde yapılmasıdır**
- **ejbLoad()** metodunda entity state'i veritabanından yüklenir
- **ejbStore()** metodunda entity state'i veritabanına yazılır
- **CMP'de abstract** olarak tanımlanan entity bean sınıfı **BMP'de concrete** olarak implement edilir

Container Managed Transaction

- Transaction davranışı sınıf veya metot düzeyinde deployment descriptor **metadata** ile sağlanır
- Default olarak session ve message bean'lerin metotları **transactional**'dir
- **Metot başında** gerekiyorsa transaction başlatılır
- Metot başarılı sonlanmış ve transaction metot girişinde başlatılmış ise **transaction commit** edilir
- Eğer metot içerisinde exception fırlatılırsa **transaction rollback** edilir

Transaction Attribute Tanımları

- REQUIRED, REQUIRES_NEW
- MANDATORY, NEVER
- SUPPORTS, NOT_SUPPORTED
- Bu attribute'ların **hepsi sadece session bean için geçerlidir**
- **Message bean, sadece required ve not_supported attribute'larını destekler**

Transaction Metadata Örneği

```
<ejb-jar>  
...  
<assembly-descriptor>  
  <container-transaction>  
    <method>  
      <ejb-name>Hello</ejb-name>  
  
      <method-name>hello</method-name>  
    </method>  
  
    <trans-attribute>REQUIRED</trans-attribute>  
  </container-transaction>  
</assembly-descriptor>  
</ejb-jar>
```

Bean Managed Transaction

```
UserTransaction tx = null;

try {
    InitialContext ctx = new InitialContext();

    tx = (UserTransaction) ctx.lookup("java:comp/UserTransaction");
    tx = getSessionContext().getUserTransaction();

    tx.begin();

    //transactional işlemler gerçekleştirilir

    tx.commit();
} catch (Exception e) {
    if (tx != null) tx.rollback();
}
```

İletişim



www.harezmi.com.tr

www.java-egitimleri.com



info@harezmi.com.tr

info@java-egitimleri.com



[@HarezmiBilisim](https://twitter.com/HarezmiBilisim)

[@JavaEgitimleri](https://twitter.com/JavaEgitimleri)