Stateless Session Bean

Elkin Prada
Sebastian Urrea
Brayan Rojas
Alvaro Herrera

Corporación Universitaria Minuto de Dios

Arquitectura de Software

04 octubre 2017.

Características

Stateless Session Bean

- Es una clase java normal para implementar procesos de una lógica de negocio.
- Estos Beans son gestionados por un contenedor EJB que se ocupa de inicializarlos y destruirlos.
- Cada nueva instancia es mantenida en un pool.
- No tienen estado.
- Cuentan con una serie de métodos que realizarán un trabajo determinado.
- Una vez ejecutado un método del bean el contenedor puede eliminarlo del pool o mantenerlo.

Ciclo de Vida

Stateless Session Bean

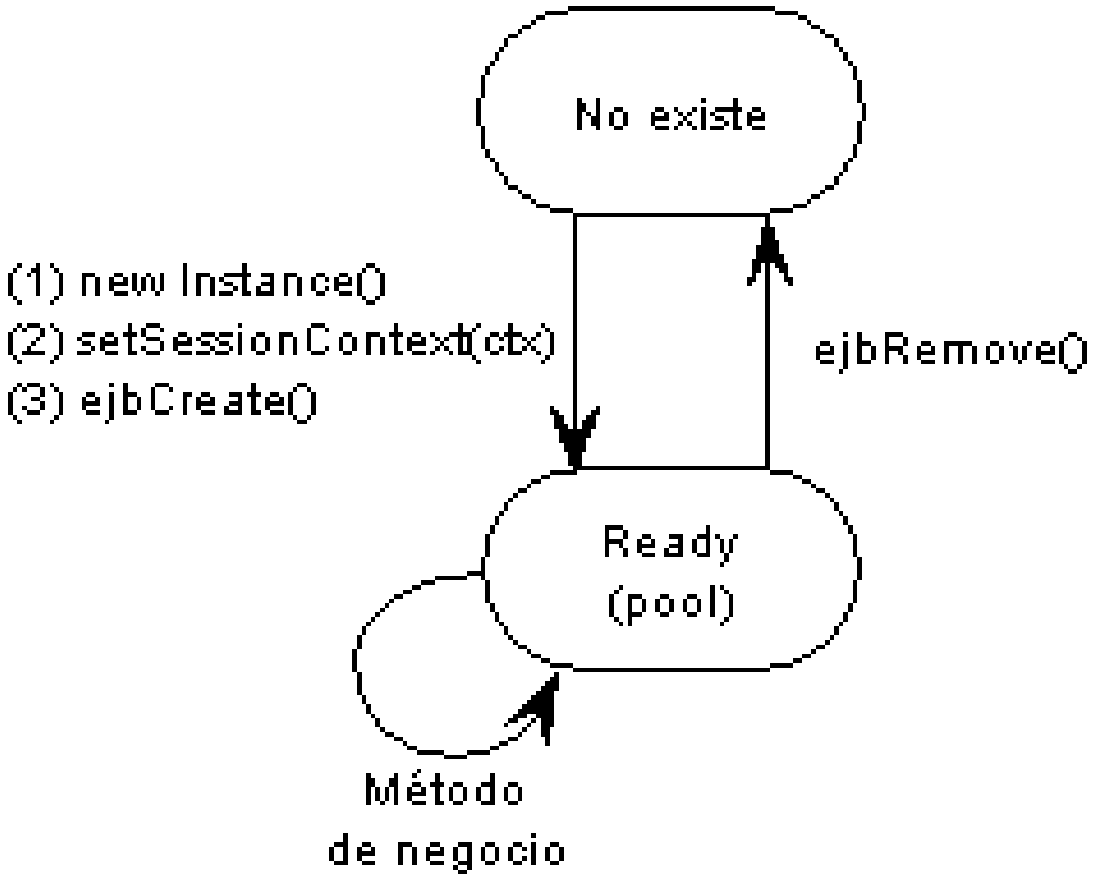


Figura 3.2: Ciclo de vida de stateless session beans

Características

Stateless Session Bean

- Reutilizarlo en otros clientes
- Las acciones a tomar dependen de la implementación del container.
- Un stateless session bean sólo debe contener información que no es específica a un cliente.
- Un método sin parámetros para su creación, llamado ejbCreate(),
- El container no debe permitir la ejecución de múltiples threads sobre una misma instancia.
- Una vez ejecutado un método del bean el contenedor puede eliminarlo del pool o mantenerlo.

Pool de Stateless

Stateless Session Bean

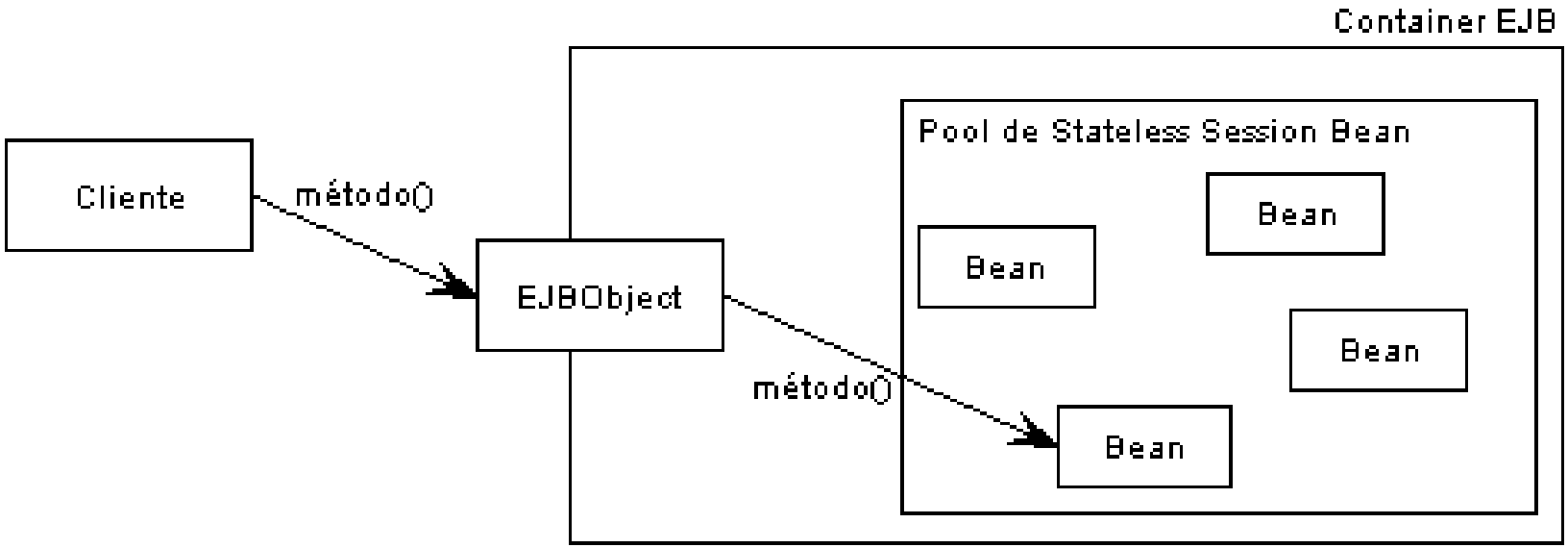
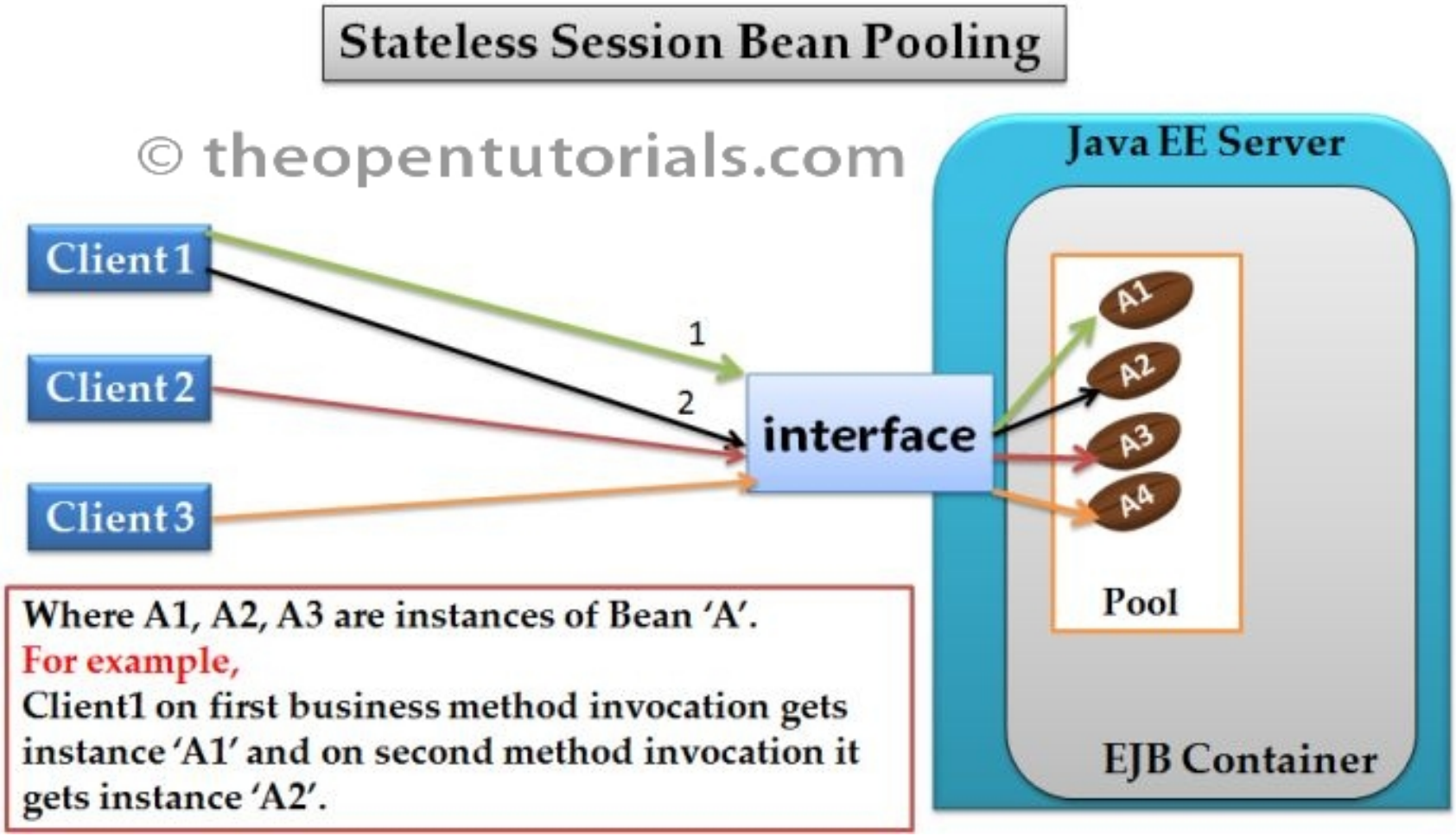


Figura 3.3: Pool de stateless session beans

Pool

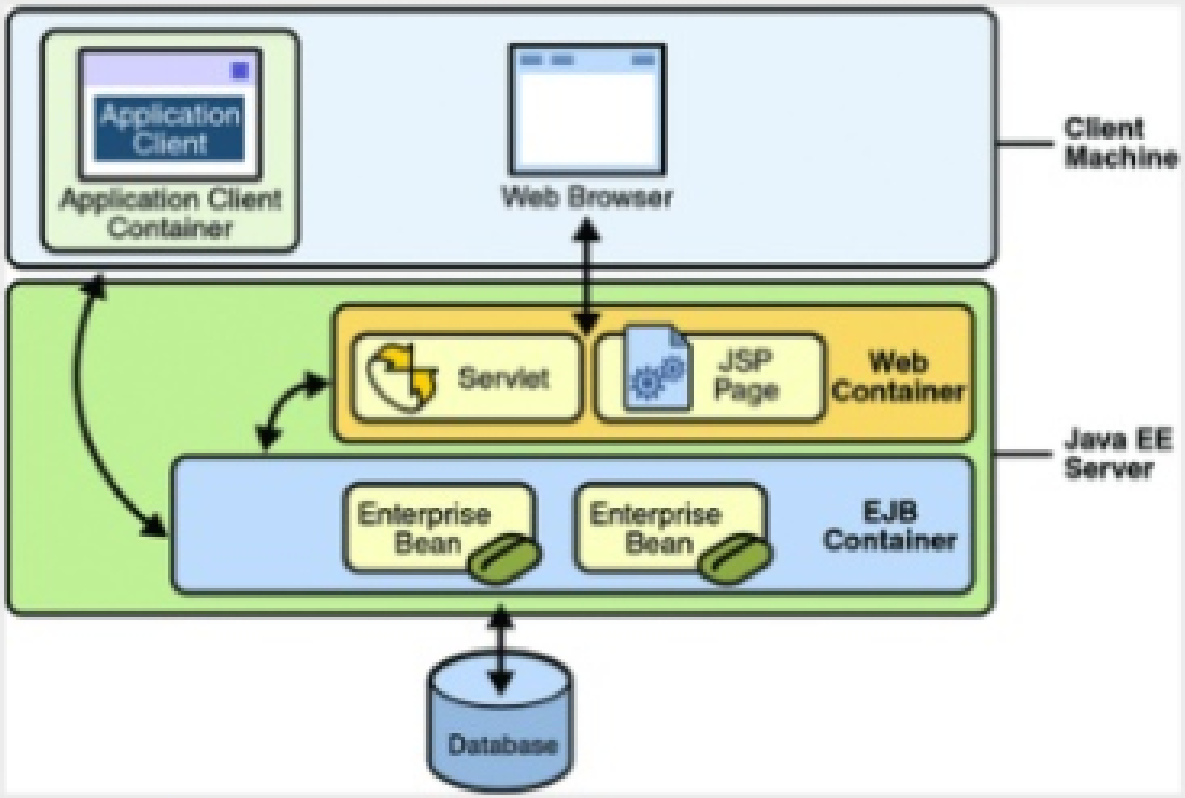
Stateless Session Bean

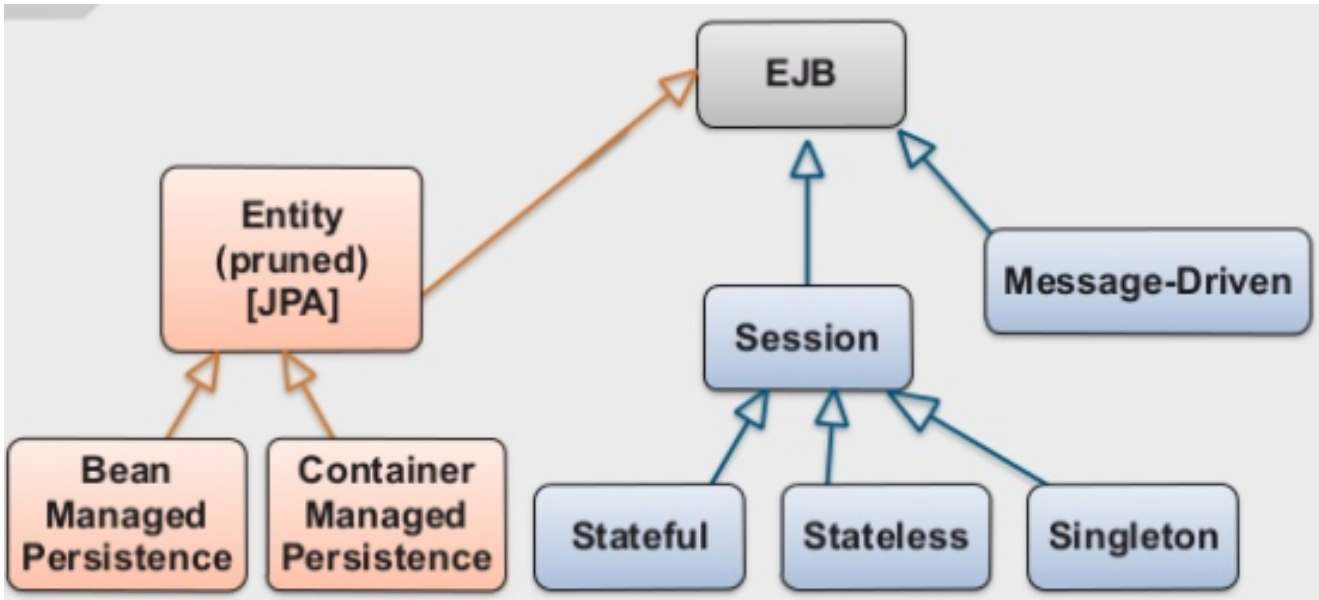


Esquema Containers

Stateless Session Bean

- **The runtime portion of a Java EE product. A Java EE server provides EJB and web containers.**

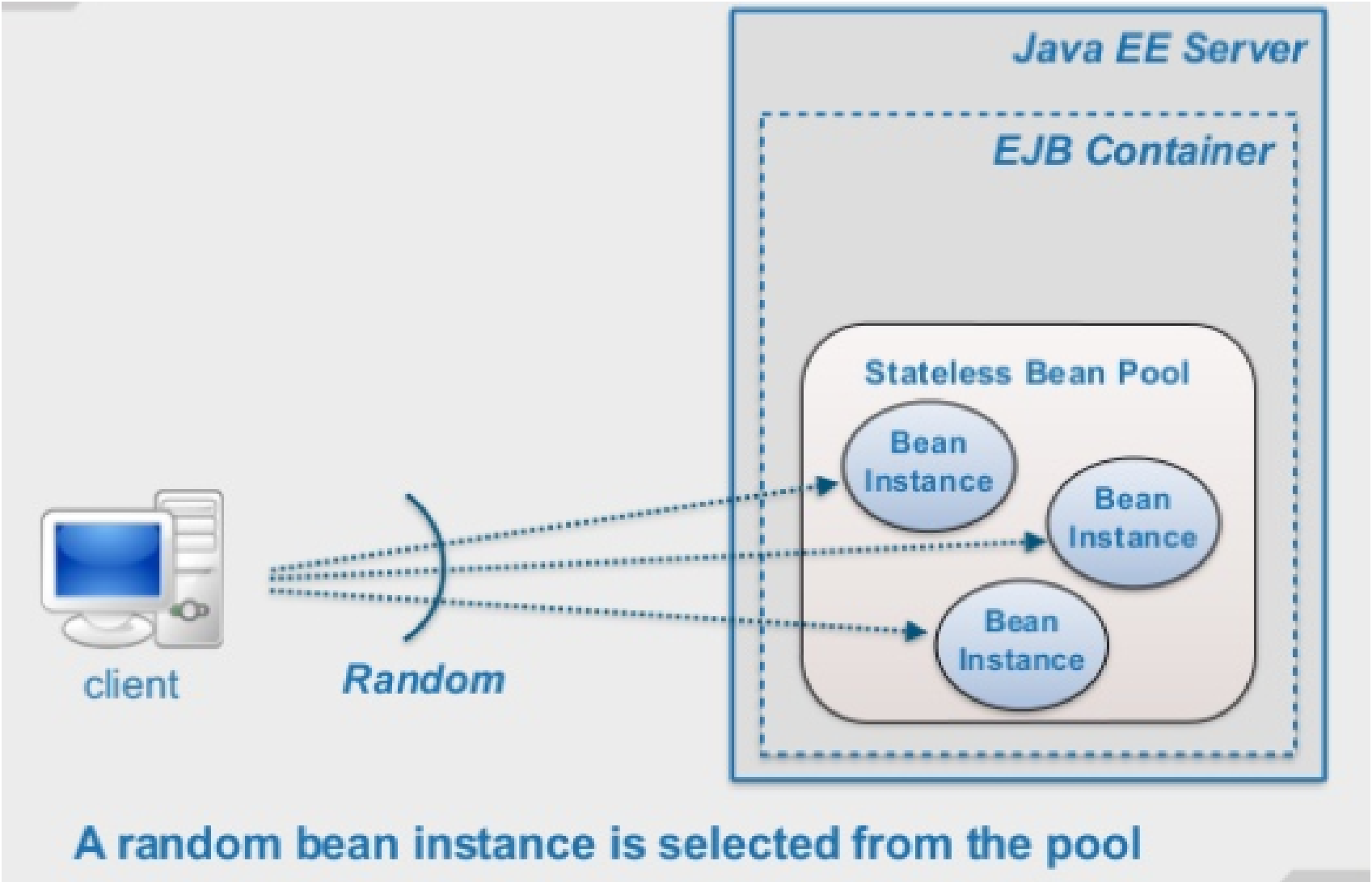




Bean Type	*Annotation*
Session Bean	*Performs a task for a client; optionally, may implement a web service*
Message-driven Bean	*Acts as a listener for a particular messaging type, such as the Java Message Service API*

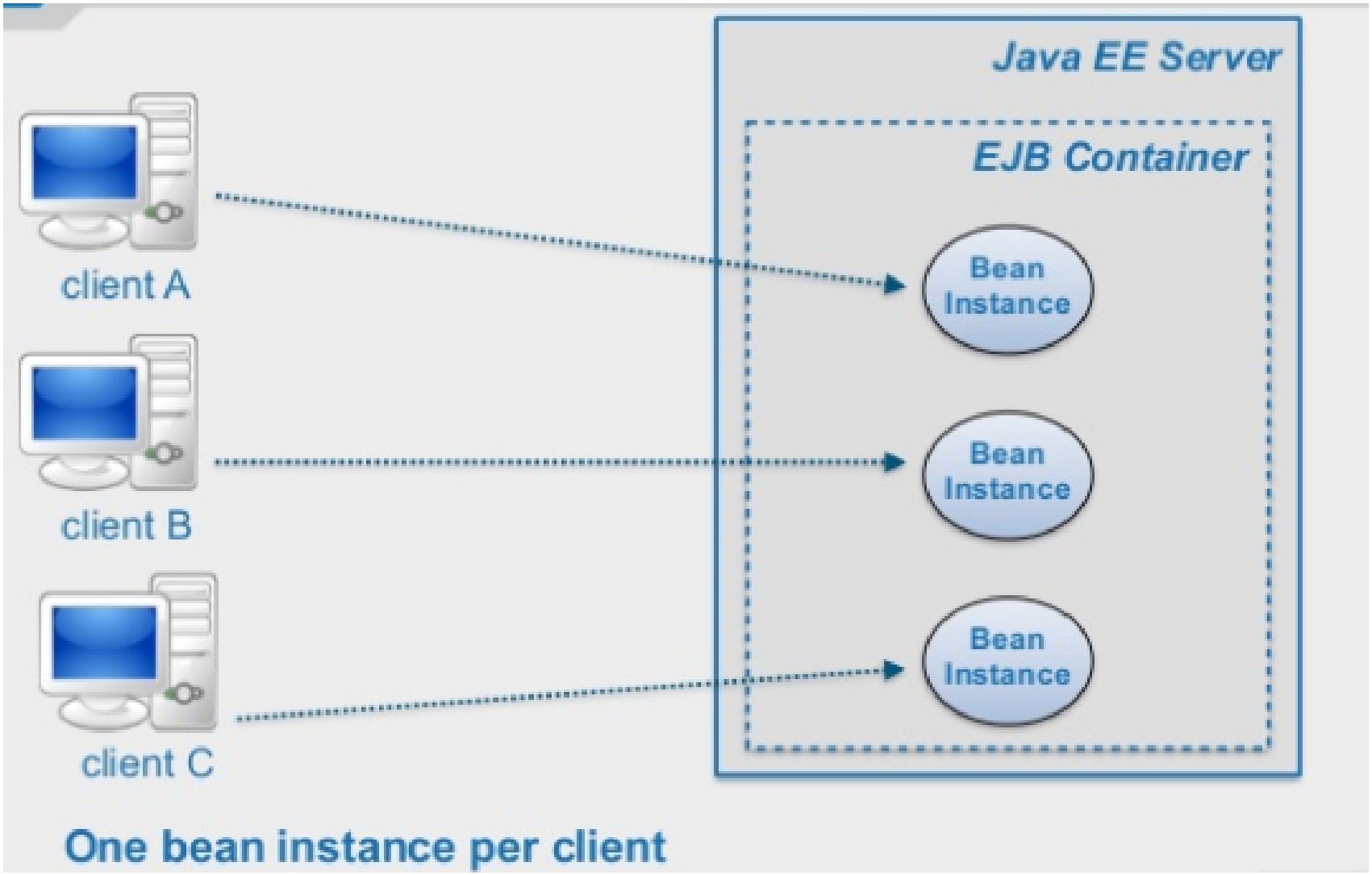
Instancia random

Stateless Session Bean



Instancia

Stateless Session Bean



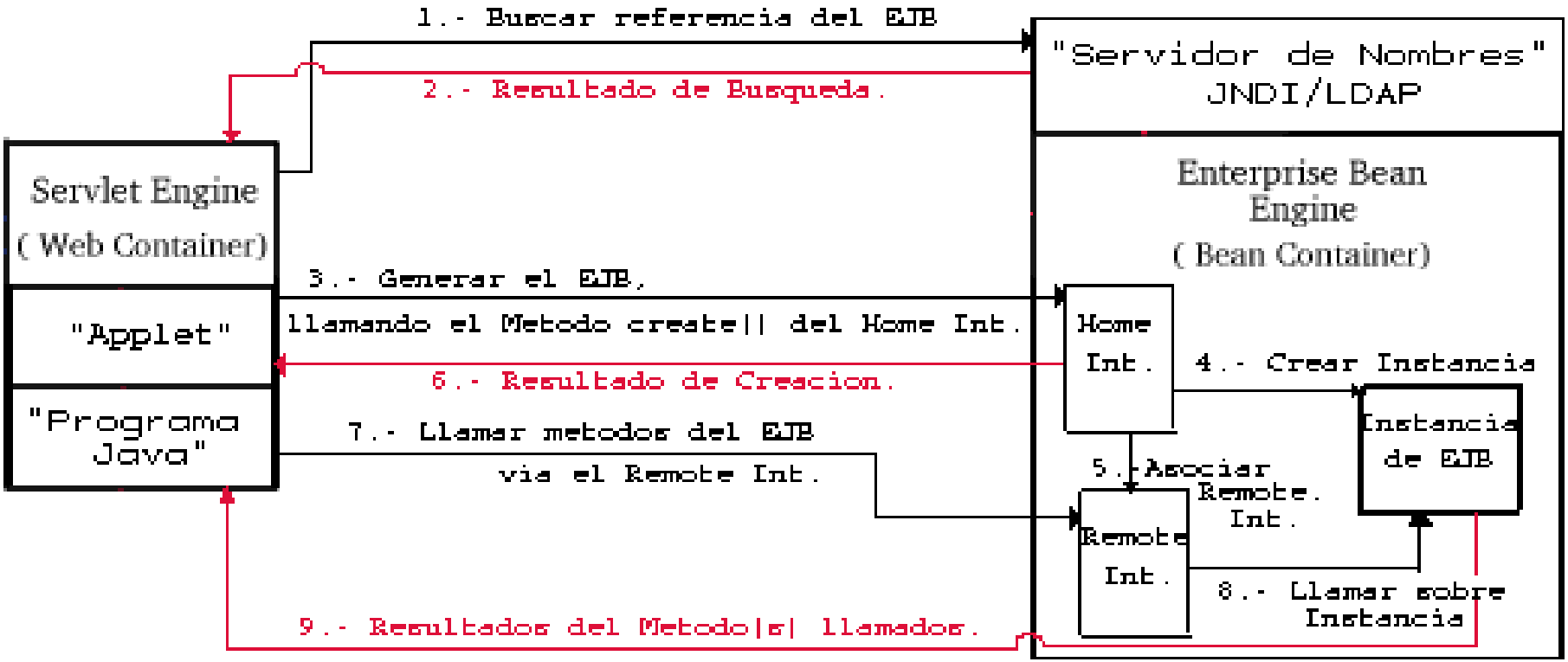
Ejemplos

Stateless Session Bean

- Implementar un carro de compras donde el cliente selecciona variados ítemes que deben ser agregados a él. Para realizar este proceso es necesario que una componente deba mantenerse ligada a un cliente en particular entre distintos requests, actualizando su estado cuando corresponda.
- Un método que realice un cálculo matemático con los valores entregados y retorne el resultado, o un método que verifique la validez de un número de tarjeta de crédito son posibles métodos implementables por stateless session beans.

Ejemplos

Stateless Session Bean



Ejemplos

Stateless Session Bean

- **Local Interface**

```java
@Local
public interface MyStatelessBeanLocal {
    String sayHello(String name);
}
```

- **Remote Interface**

```java
@Remote
public interface MyStatelessBeanRemote {
    String sayHello(String name);
}
```

Ejemplos

Stateless Session Bean

```
@Stateless
@Local(MyStatelessBeanLocal.class)
@Remote(MyStatelessBeanRemote.class)
public class MyStatelessBean implements MyStatelessBeanRemote,
        MyStatelessBeanLocal {
    public MyStatelessBean() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public String sayHello(String name) {
        return "hello, " + name;
    }

}
```

Ejemplos

Stateless Session Bean

- **Dependency Injection**

```
@EJB
MyStatelessBeanLocal myDIBeanLocal;
```

- **JNDI Lookup**

```
Hashtable jndiProperties = new Hashtable();
jndiProperties.put(Context.URL_PKG_PREFIXES,
    "org.jboss.ejb.client.naming");
Context context = new InitialContext(jndiProperties);
MyStatelessBeanLocal myJNDIBeanLocal = (MyStatelessBeanLocal)
context.lookup("java:module/MyStatelessBean!
com.ece.jee.MyStatelessBeanLocal");
```

- **Clients do not use the new operator to obtain a new instance**

Ejemplos

Stateless Session Bean

- **Dependency Injection**

```
@EJB
MyStatelessBeanRemote myDIBeanRemote;
```

- **JNDI Lookup**

```
Hashtable jndiProperties = new Hashtable();
jndiProperties.put(Context.URL_PKG_PREFIXES,
    "org.jboss.ejb.client.naming");
Context context = new InitialContext(jndiProperties);
MyStatelessBeanLocal myJNDIBeanLocal = (MyStatelessBeanLocal)
context.lookup("java:module/MyStatelessBean!
com.ece.jee.MyStatelessBeanLocal");
```

- **Clients do not use the new operator to obtain a new instance**

Ejemplos

Stateless Session Bean

```
package com.osmosislatina.ejb.intereses;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Intereses extends EJBObject {

   public double calcularInteres(double capital, double tasa, double plazo)
       throws RemoteException;

}
```

```
package com.osmosislatina.ejb.intereses;

import javax.ejb.EJBHome;
import javax.ejb.CreateException;
import java.rmi.RemoteException;

public interface InteresesHome extends EJBHome {

   Intereses create() throws RemoteException, CreateException;

}
```

Ejemplos

Stateless Session Bean

```
package com.osmosislatina.ejb.intereses;

import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class InteresesBean implements SessionBean {

   public double calcularInteres(double capital, double tasa, double plazo) {

   System.out.println("Un Cliente llamo la función para calculo de Interés");
   return capital * Math.pow(1+tasa, plazo) - capital;
   }

   public InteresesBean() {}
   public void ejbCreate() {}
   public void ejbRemove() {}
   public void ejbPassivate() {}
   public void ejbActivate() {}
   public void setSessionContext(SessionContext sc) {}
}
```

Ejemplos

Stateless Session Bean

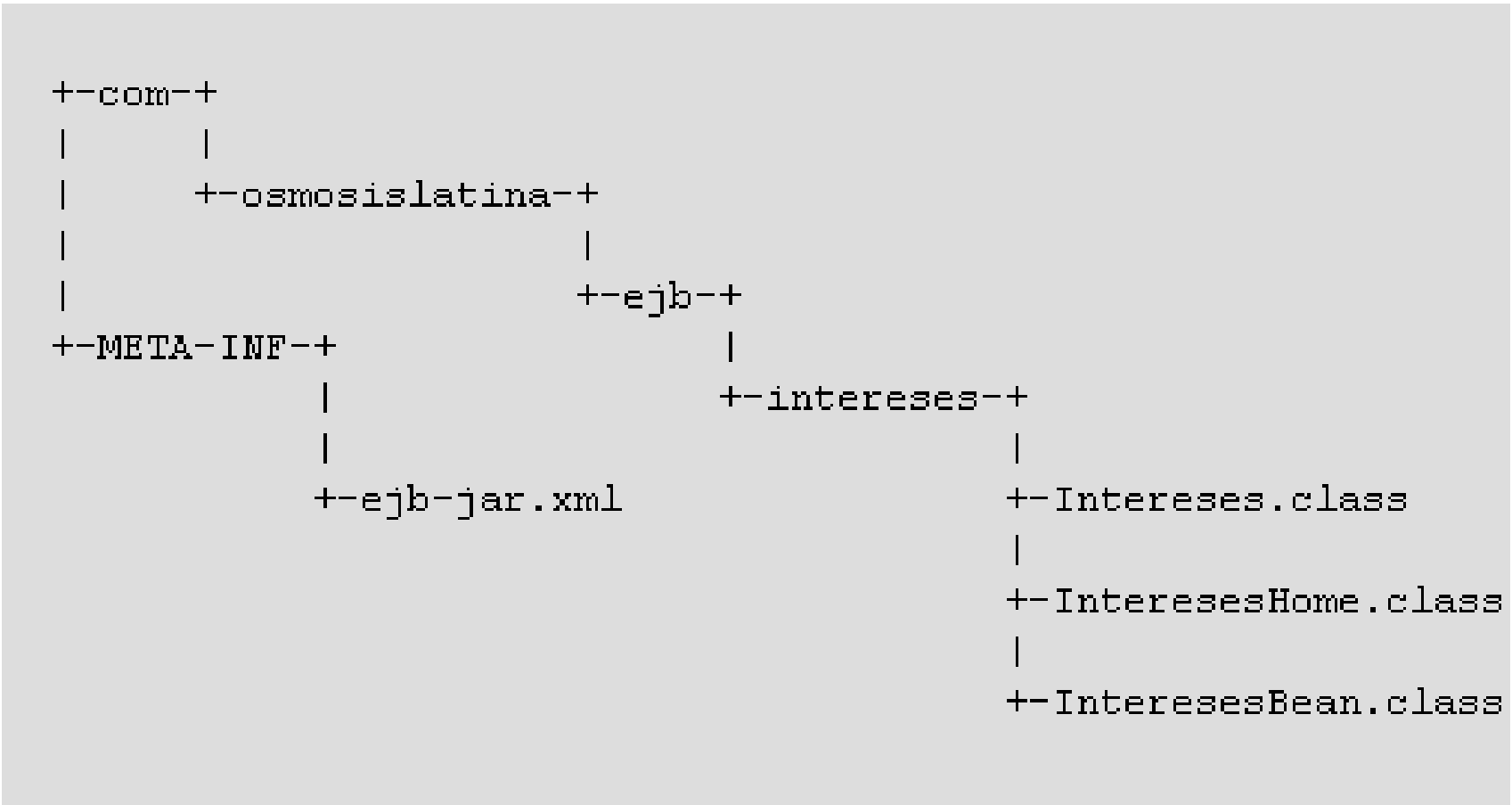
```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE ejb-jar PUBLIC
        "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
        "http://java.sun.com/dtd/ejb-jar_2_0.dtd">

 <ejb-jar>
      <description>Calculo de Intereses</description>
      <display-name>Calculo</display-name>
      <enterprise-beans>
        <session>
          <ejb-name>Calculo</ejb-name>
          <home>com.osmosislatina.ejb.intereses.InteresesHome</home>
          <remote>com.osmosislatina.ejb.intereses.Intereses</remote>
          <ejb-class>com.osmosislatina.ejb.intereses.InteresesBean</ejb-class>
          <session-type>Stateless</session-type>
          <transaction-type>Bean</transaction-type>
        </session>
      </enterprise-beans>
 </ejb-jar>
```

Ejemplos

Stateless Session Bean



Example (Theorem Slide Code)

```
public interface CustomerServiceLocal {

 public void saveCustomer( Customer customer );
 public Customer getCustomer( Long id );
 public Collection getAllCustomers();
 public void deleteCustomer( Long id );

}
```

Example (Interface local)

```
@Local
public interface CustomerServiceLocal {

 public void saveCustomer( Customer customer );
 public Customer getCustomer( Long id );
 public Collection getAllCustomers();
 public void deleteCustomer( Customer customer );

}
```

Example (Interface remota)

```
@Remote
public interface CustomerServiceRemote{

 public void saveCustomer( Customer customer );
 public Customer getCustomer( Long id );
 public Collection getAllCustomers();
 public void deleteCustomer( Customer customer );

}
```

Ejemplos

Stateless Session Bean

Example (Interface remota)

```
@Stateless
public class CustomerService implements CustomerServiceLocal
{
 @PersistenceContext()
 private EntityManager em;

 public void saveCustomer( Customer customer ) {
  em.persist( customer );
 }
 public Customer getCustomer( Long id ) {
Query q = em.createQuery( "SELECT c FROM Quiz c WHERE c.id =
            setParameter( "id", id );

        return (Customer)q.getSingleResult();
}
```

Example (Interface remota)

```
public Collection getAllCustomers() {
  return em.createQuery
  ( "SELECT c from Customer c" ).getResultList();
 }

 public void deleteCustomer( Customer customer ) {
  em.remove( customer );
 }

}
```

Ejemplos

Stateless Session Bean

- **POJOs**
  - Instance of the Bean relates to a specific client (in memory while he/she is connected)
  - Expires in case of inactivity (similar to session in Servlet/Jsp)
  - Ordinary Java classes; no special interfaces or parent classes.
- **Local or remote access**
  - Can be accessed either on local app server or remote app server
- **Session Expiry**
  - The Session expires after the method annotated with @Remove is executed
  - Session can also expire in case of a time-out

Métodos

Stateless Session Bean

@PostConstruct

public void initialize() { ... at Bean's initialization ... }

@PreDestroy

- public void destroy() { ... destruction of Bean ... }

@PrePassivate //only for Stateful beans

public void beforeSwap() { ... to do before Bean is passivated ... }

@PostActivate //only for Stateful beans

public void afterSwap() { ... to do after Bean is activated ... }

Ejemplos

Stateless Session Bean

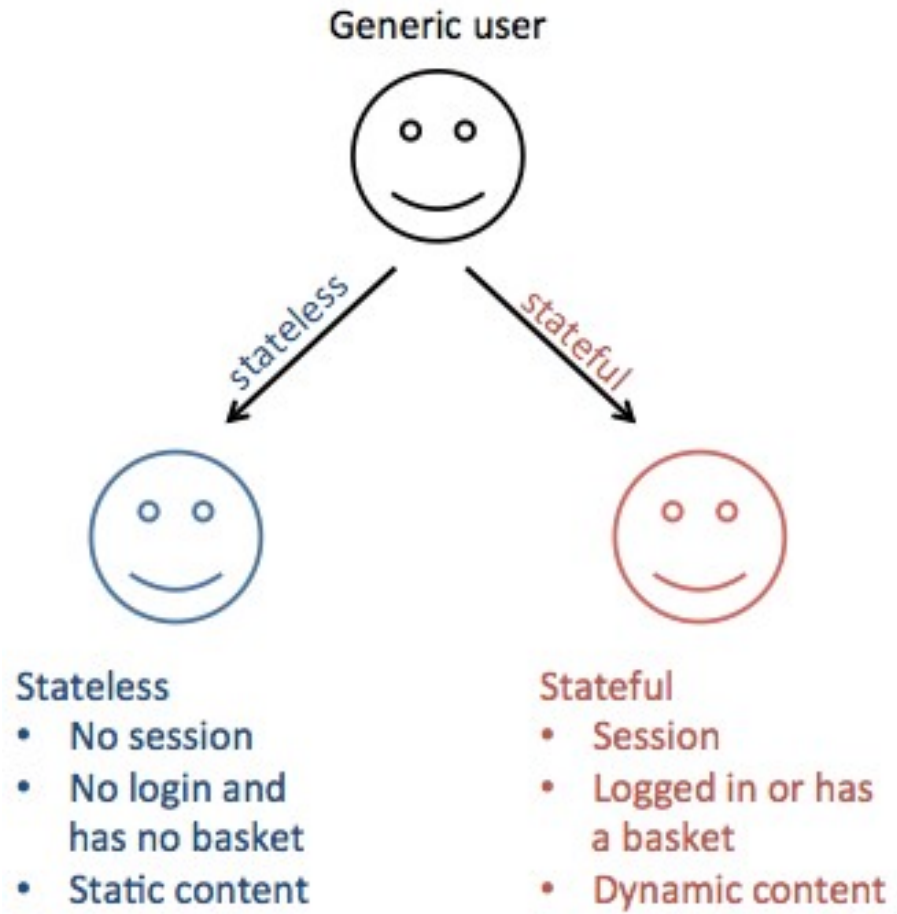
- Container does not manage a pool for Stateful EJBs
- If the instance is not removed it stays in the memory
- A timeout from the server destroys the bean instance from READ or PASSIVE state
- A method with @Remove annotation is used to manage the destruction of instance

```
@Remove
    public void destroy(){

    }
```

Diferencias

Stateless Session Bean



References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

The End