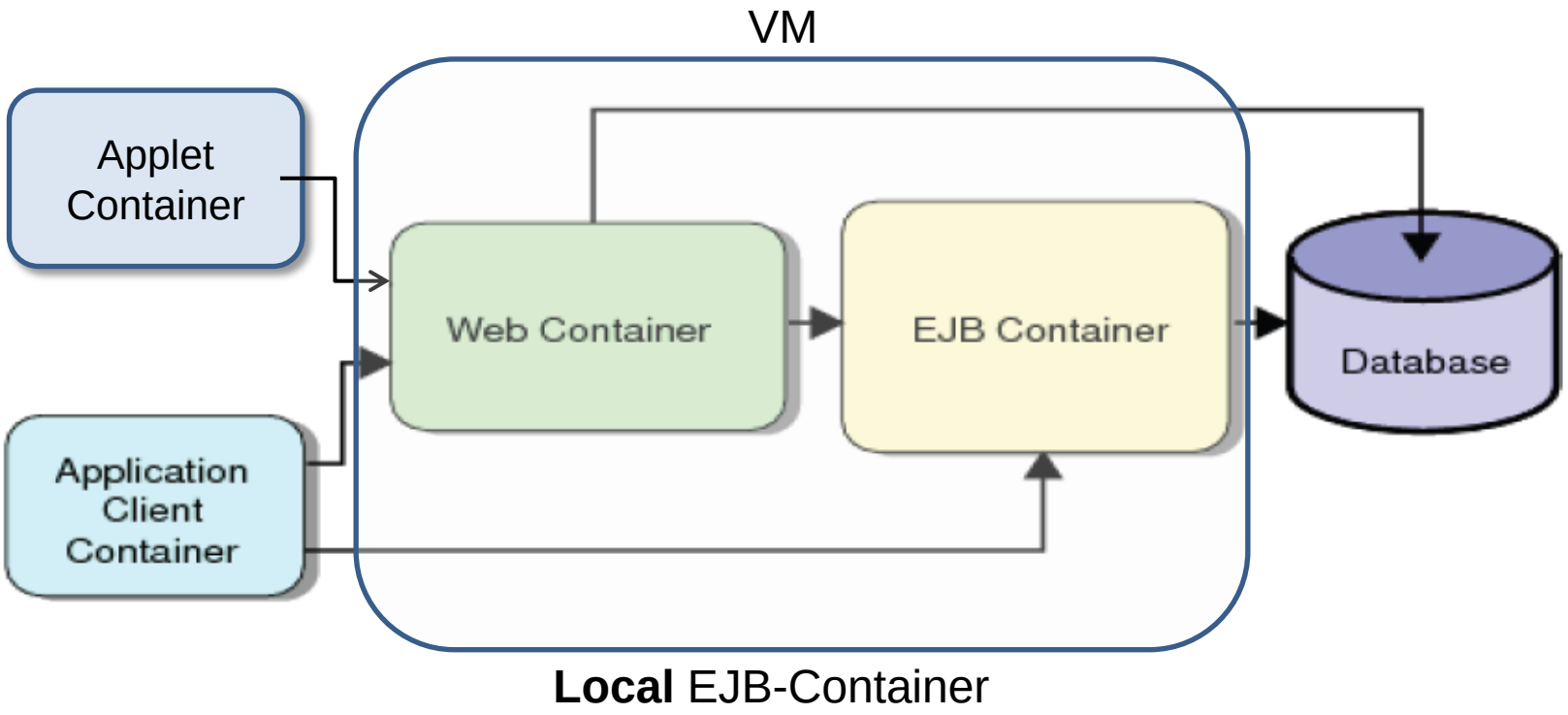
# Modul 8
# Enterprise Java Beans

## EJB

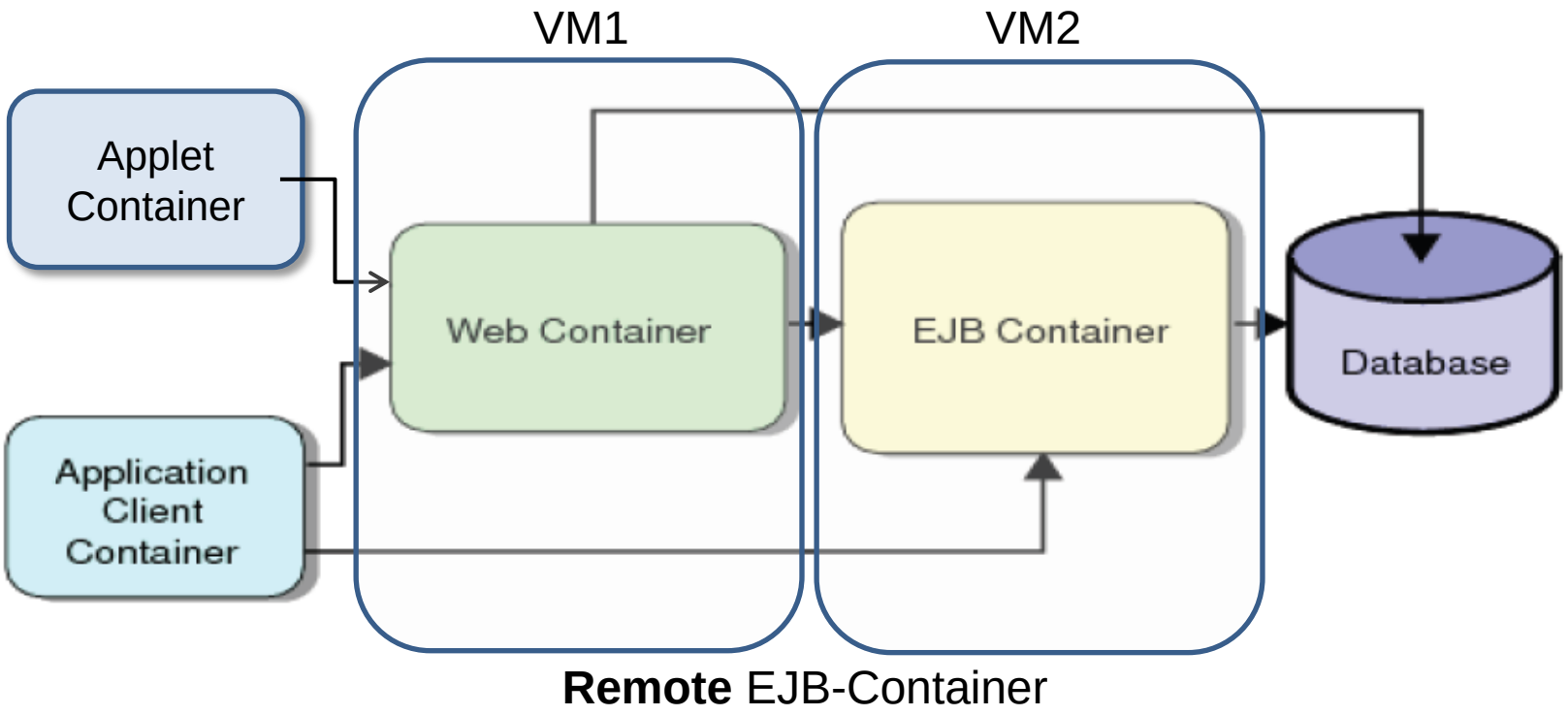
Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Enterprise Java Beans




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Enterprise Java Beans




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

EJB components:

- Are managed by the EJB container
- Provide business and messaging functions
- Are scalable, transactional, multi-user, and secure

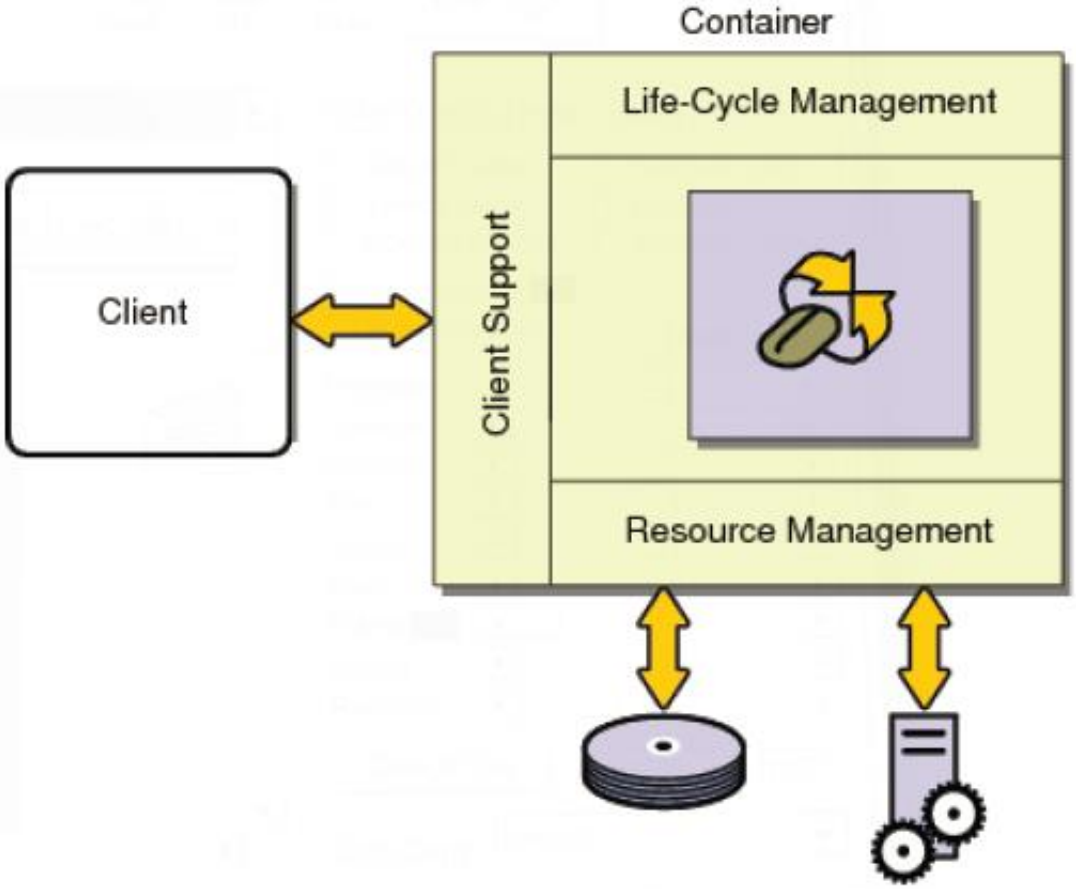
Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# EJB Life Cycle

- An EJB component has a life cycle that is controlled by the EJB container based on the functionality requested by the client and the operational requirements of the system.
- Each type of EJB component has specific life-cycle operations and characteristics.
- Understanding the life cycle of the various EJB component types might allow a component developer to optimize the operation of the bean.

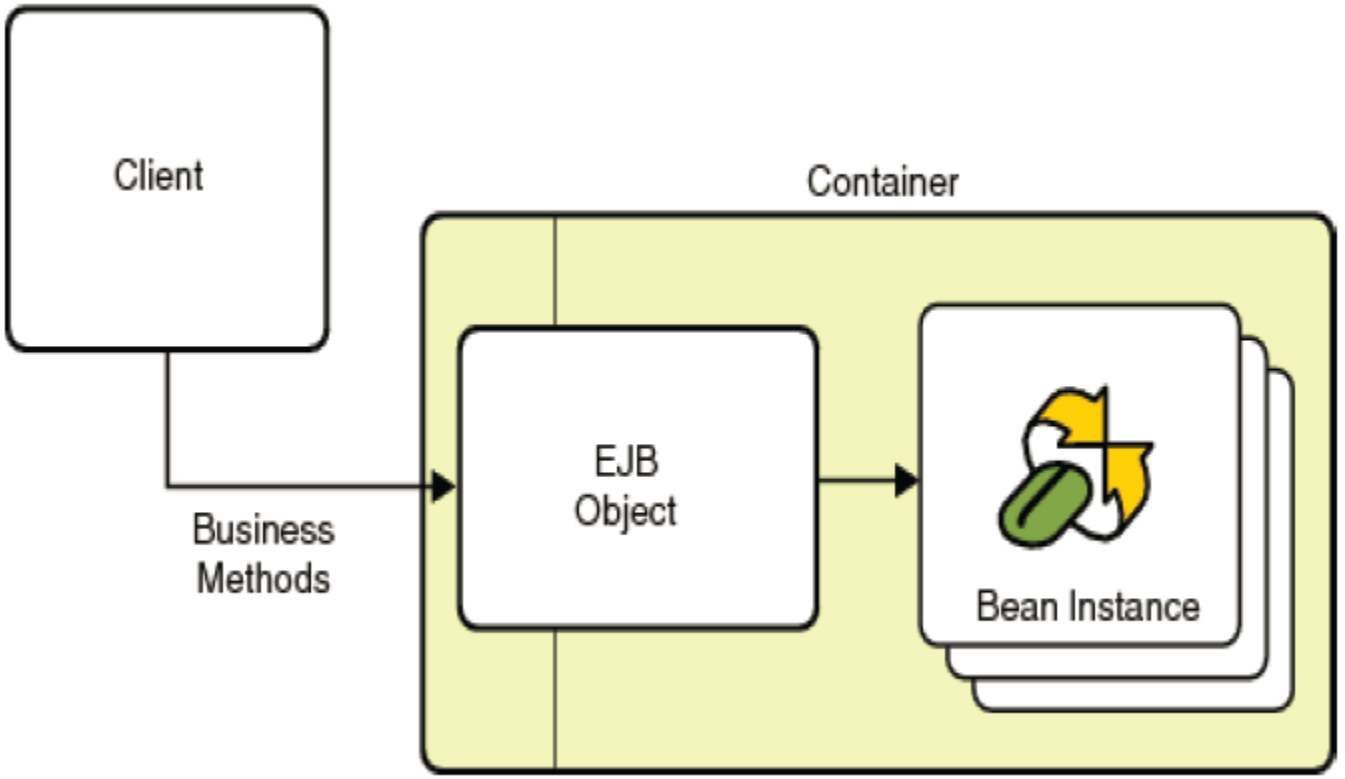
Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Role of the EJB Container



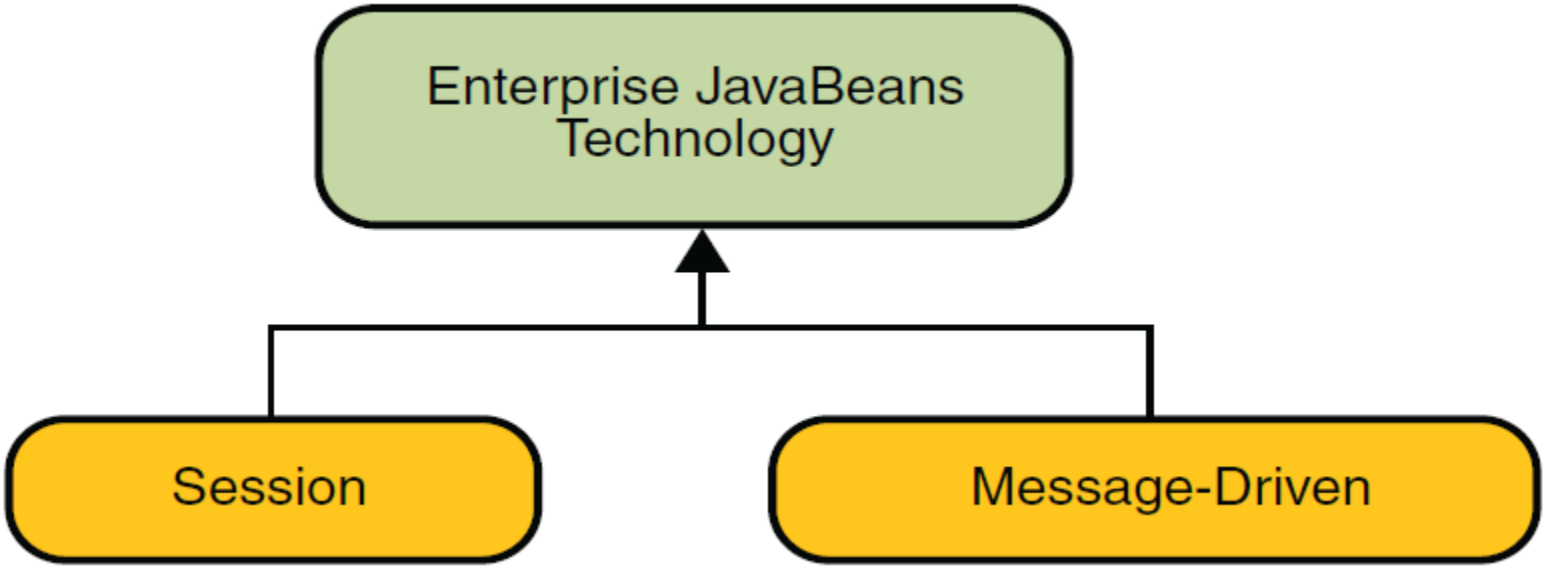

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# EJB Proxies or Stubs




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# EJB Component Types




- Stateful
- Stateless
- Singleton

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

**Stateful Session Beans**

The state of an object consists of the values of its instance variables. In a **stateful session bean**, the instance variables represent the state of a unique client/bean session.
(Shopping carts, user profiles, etc. Often attached to **session object**.)

**Stateless Session Beans**

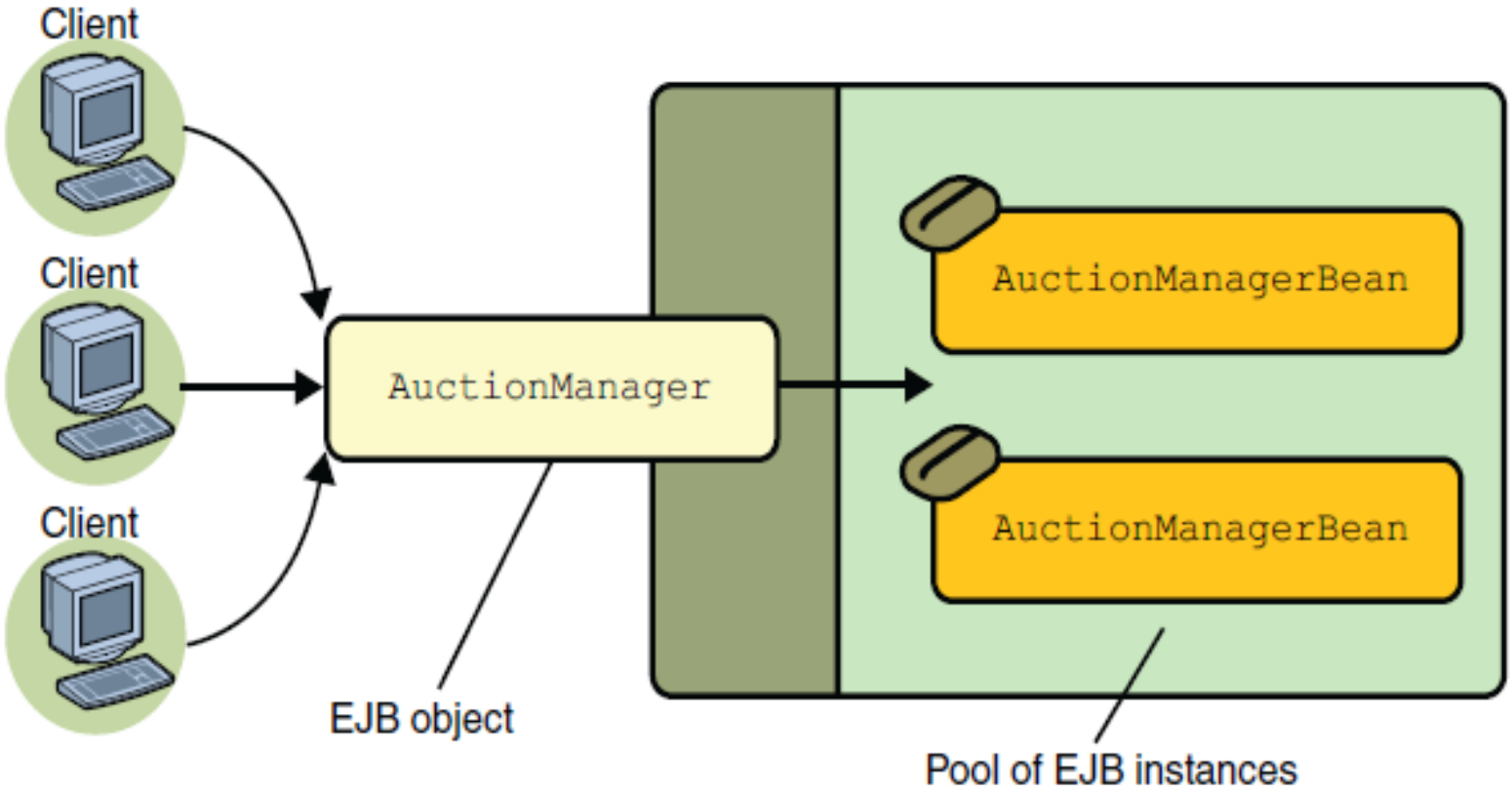
A **stateless session bean** does not maintain a conversational state with the client. Pool of stateful session beans, where instances are re-used.
(Computation tasks, DB-calls, etc. Normally not attached.)

**Singleton Session Beans**

A **singleton session bean** is instantiated once per application and exists for the lifecycle of the application.
(Global state, e.g. user-counter, etc. Normally not attached.)

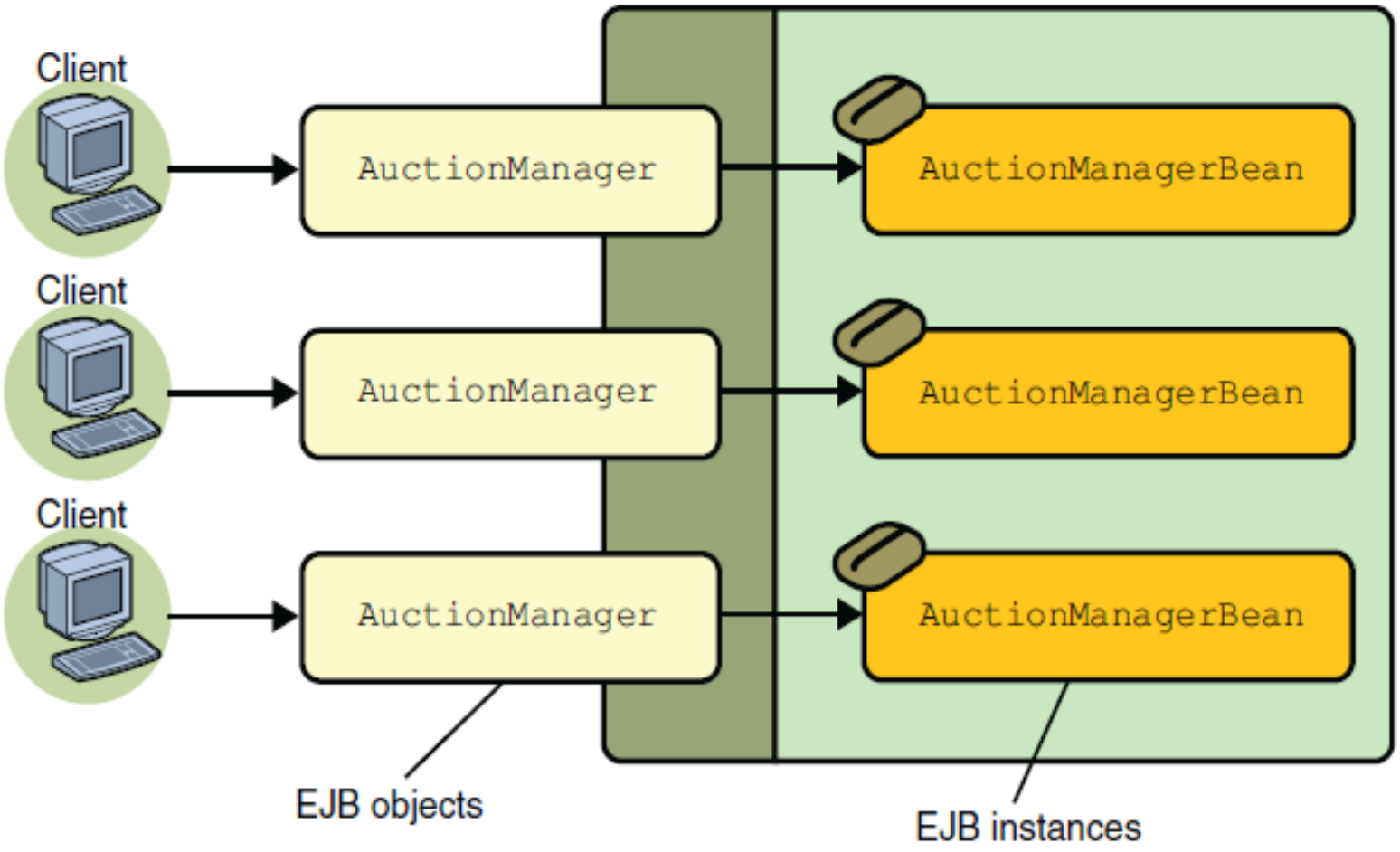
Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Stateless Session Bean



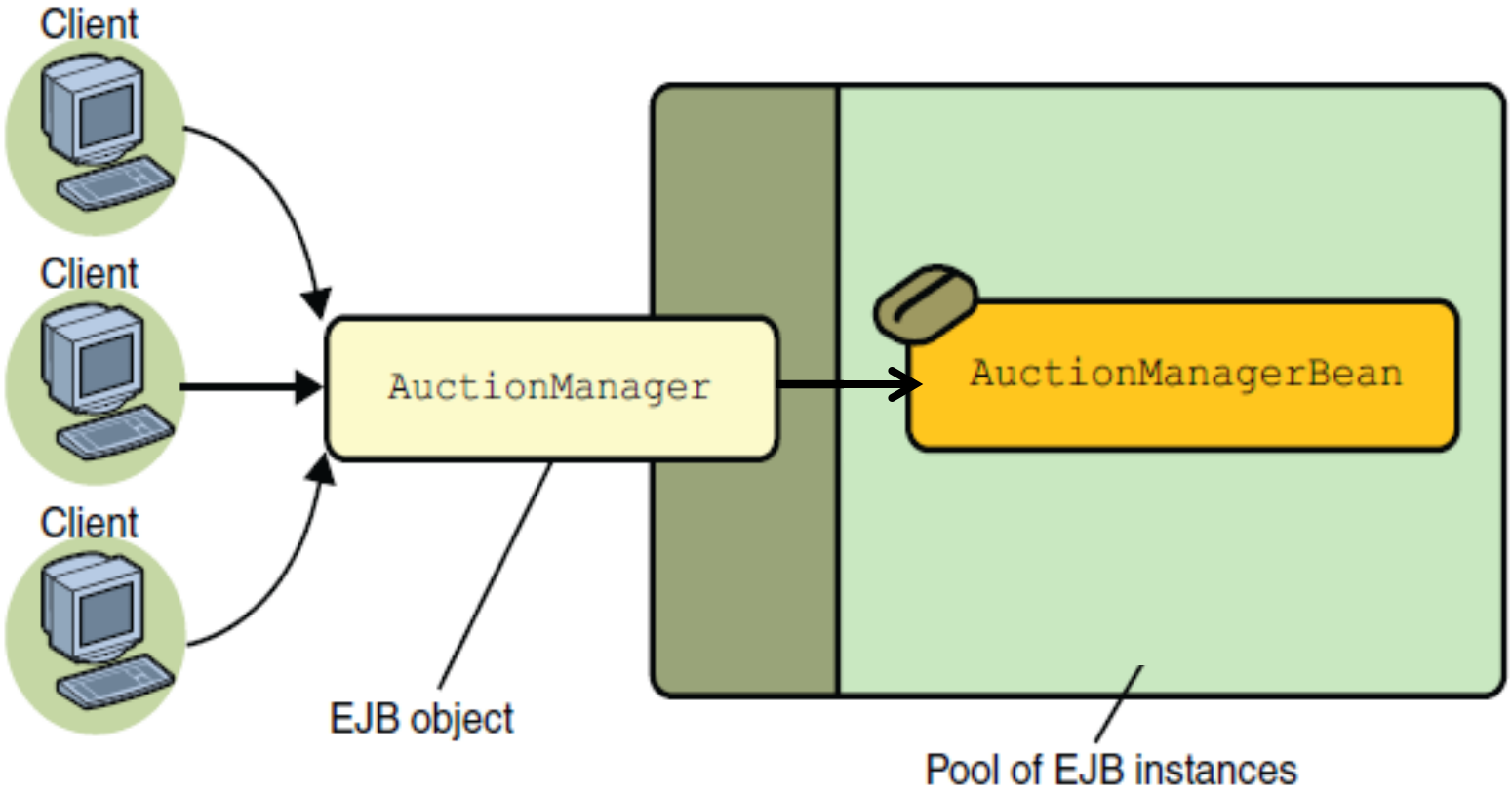

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Stateful Session Bean




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Singleton Session Bean




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

m.socrative.com
Room: 790303

Sie möchten mittels einer EJB einen globalen Zähler (Counter) erstellen, der die Anzahl aller Seitenaufrufe Ihrer Anwendung zählt.

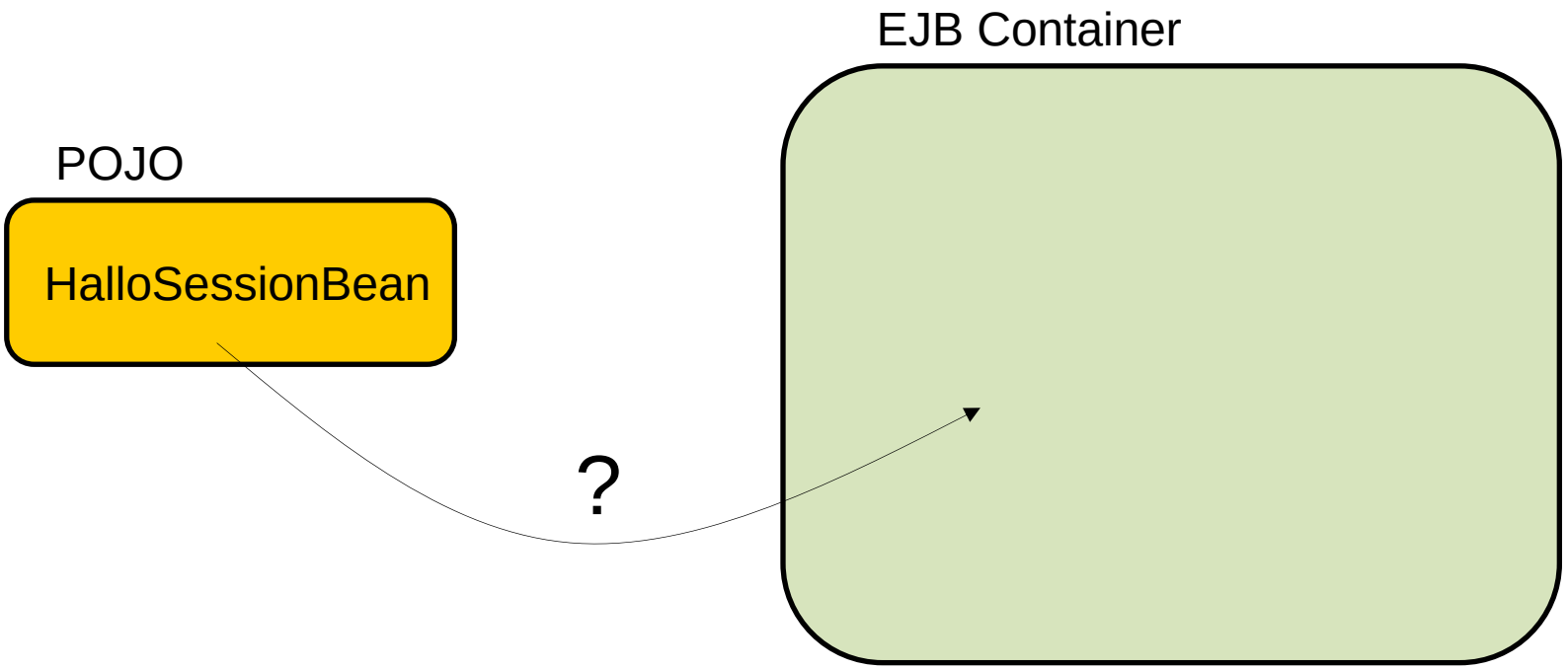
Welche Session Bean verwenden Sie?

A) Stateful Session Bean

B) Stateless Session Bean

C) Singleton Session Bean

# Modul 8
# Creating Session Beans




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

Which kind of Session Bean?

Local or Remote, Stateless or Stateful or Singleton

Execute three steps

1. Declare a business **interface** for the session bean.
2. Create the session bean **class** that implements the business interface.
3. Configure the session bean by either **annotating** the session bean class or providing a deployment descriptor (DD).

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Creating Session Beans: Example

1. Declare a business **interface** for the session bean.

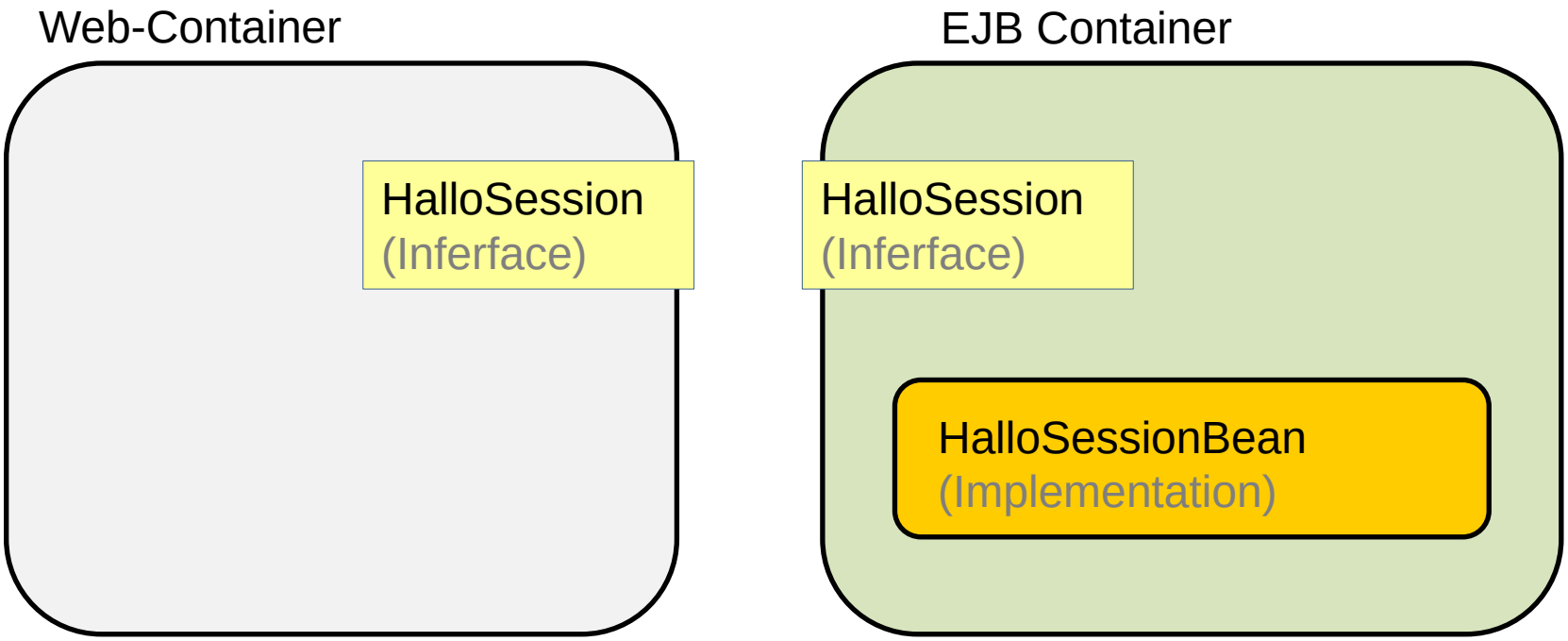
```
import javax.ejb.*;

@Remote // or @Local
public interface HelloSession {
   public java.lang.String hello();
}
```

# Creating Session Beans: Example

2. Create the session bean **class** that implements the business interface.
3. Configure the session bean by either **annotating**

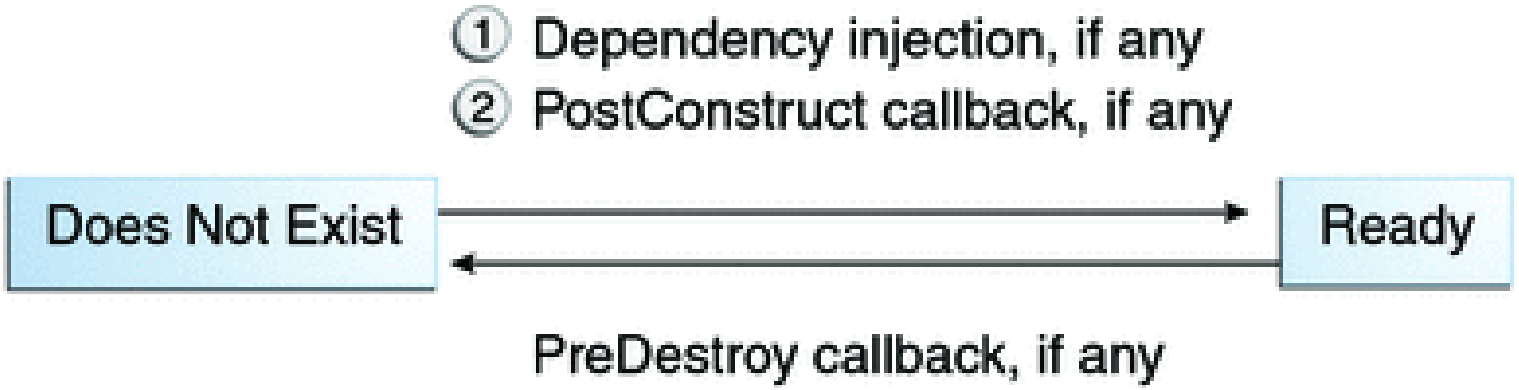
```
import javax.ejb.*;

@Stateless //or @Stateful or @Singelton
public class HelloSessionBean implements HelloSession {

public java.lang.String hello() {
    return "Hello World!";
  }
}
```

# Modul 8
# Creating Session Beans




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

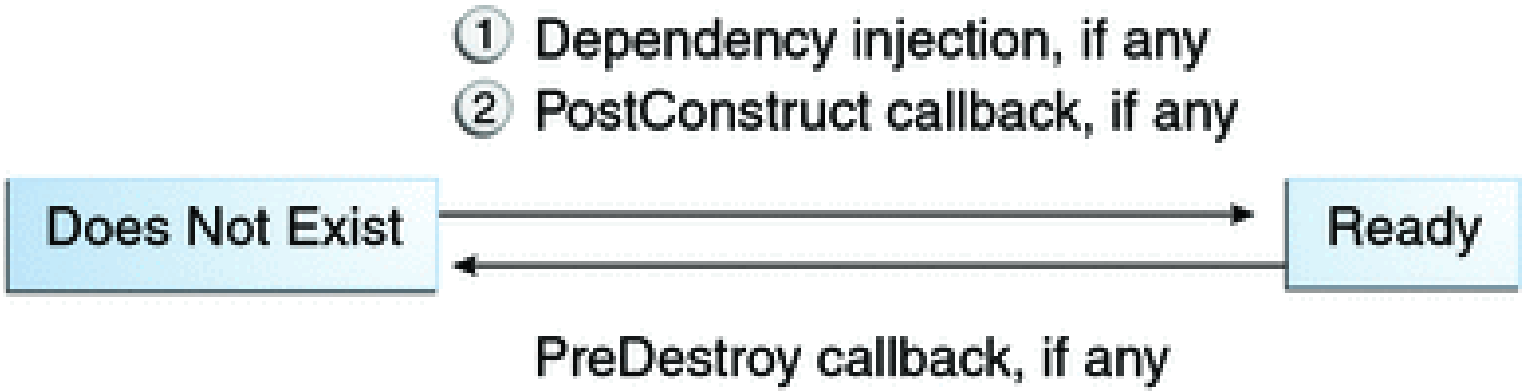
# The Lifecycle of a Stateless Session Bean




- **`@PostConstruct`** callback
- **`@PreDestroy`** callback

Quellehttp://docs.oracle.com/javaee/6/tutorial/doc/

# The Lifecycle of a Singleton Session Bean

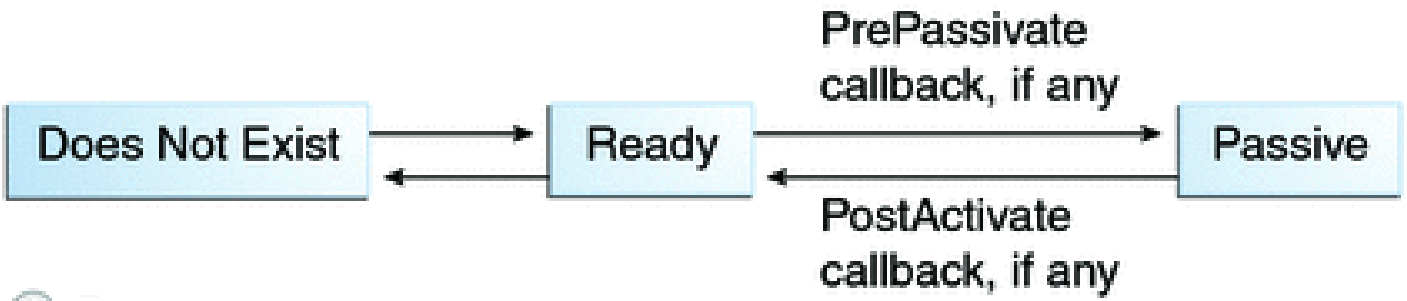



- **`@Startup`** annotaion for eager initialization
- **`@PostConstruct`** callback
- **`@PreDestroy`** callback

Quellehttp://docs.oracle.com/javaee/6/tutorial/doc/

# Modul 8
# The Lifecycle of a Singleton Session Bean

① Create
② Dependency injection, if any
③ PostConstruct callback, if any
④ Init method, or ejbCreate<METHOD>, if any




① Remove
② PreDestroy callback, if any

- `@PostConstruct` callback
- `@PreDestroy` callback

- `@PrePassivate` callback
- `@PostActivate` callback
- `@PostConstruct` callback
- `@PreDestroy` callback

# Modul 8
# Example of a Callback Method in a Bean Class

```
import javax.ejb.*;
import java.util.*;

@Stateful
 public class ShoppingCartBean implements ShoppingCart {
   private float total;
   private List productCodes;
   public int someShoppingMethod() {...};
   //...
   @PreDestroy private void endShoppingCart() {...};
}
```

Quellehttp://docs.oracle.com/javaee/6/tutorial/doc/

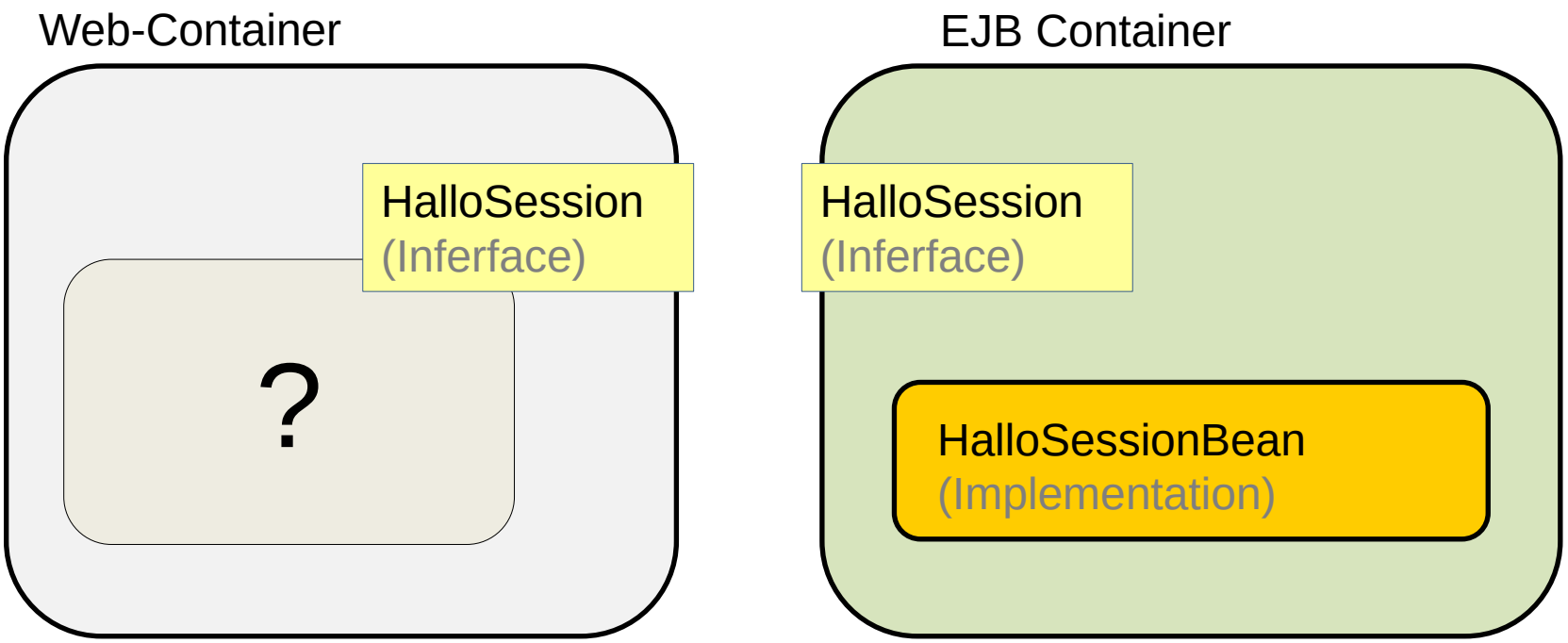
m.socrative.com
Room: 790303

Sie erstellen eine Stateful Session Bean, um einen Einkaufswagen für eine Web-Anwendung zu implementieren.

Wer ist typischerweise der „Client“ dieser Stateful Session Bean?

A) Ein Browser
B) Ein Servlet
C) Eine Desktop-Anwendung

# Modul 8
# Creating Session Bean Client




Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Creating a client using Dependency Injection (local only):

```
import javax.ejb.*;

public class InternalClient {

@EJB
private static HalloSession halloEjb;

 public static void main (String args[]) {
   System.out.println("Out:" + halloEjb.hallo() );
 }

}
```

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

## Creating a client using JNDI-LookUp (local & remote):

```
…
Properties props = new Properties();
props.setProperty("java.naming.factory.initial",
        "com.sun.enterprise.naming.SerialInitContextFactory");
props.setProperty("java.naming.provider.url",
        "iiop://192.168.178.34:3700");

InitialContext ctx = new InitialContext(props);

HalloSession halloEjb;

halloEjb = (HalloSession) ctx.lookup("java:global/M8_Test/
        HalloSessionBean");

System.out.println("Out:" + halloEjb.hallo() );
…
```

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Remarks: Stateful Session Bean Client

**`@Remove`** Method

The container will remove the enterprise bean after a **`@Remove`** method completes, either normally or abnormally.

```
@Remove
public void remove() {
      contents = null;
}
```

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

# Modul 8
# Remarks: Singelton Session Bean Client & Concurracncy

@Lock(LockType.READ): concurrent access
@Lock(LockType.WRITE): exclusive access

```
@ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)
@Singleton
public class ExampleSingletonBean {
   private String state;

   @Lock(LockType.READ)
   public String getState() { return state; }

   @Lock(LockType.WRITE)
   public void setState(String newState) {
      state = newState;
   }
}
```

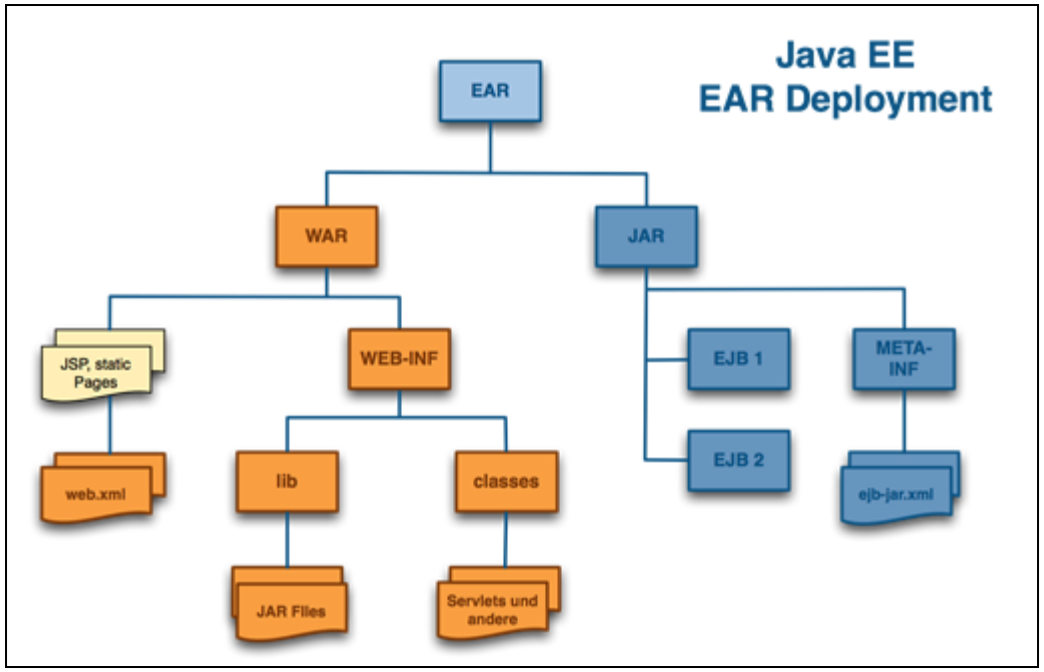
Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

## Timer service in the enterprise beans

```
...
    public void setTimer(long intervalDuration) {
        logger.info("Setting a programmatic timeout for " +
                intervalDuration + " milliseconds from now.");
        Timer timer =
                timerService.createTimer(intervalDuration,
                        "Created new programmatic timer");
    }

    @Timeout
    public void programmaticTimeout(Timer timer) {
        logger.info("Programmatic timeout occurred.");
    }
...
```

Quellehttp://docs.oracle.com/javaee/6/tutorial/doc/

# Modul 8
# Remarks: Session Beans & Deployment




Deployment structure with business interface.
Supported by Full-Profile sever (not by WebProfile server)
Netbeans: New Project -> JavaEE -> Enterprise Application

# Modul 8
# Remarks: *No-interface* view of Session Beans

**Local** EJBs only, single WAR-File:

No need for an interface anymore (EJB 3.1)

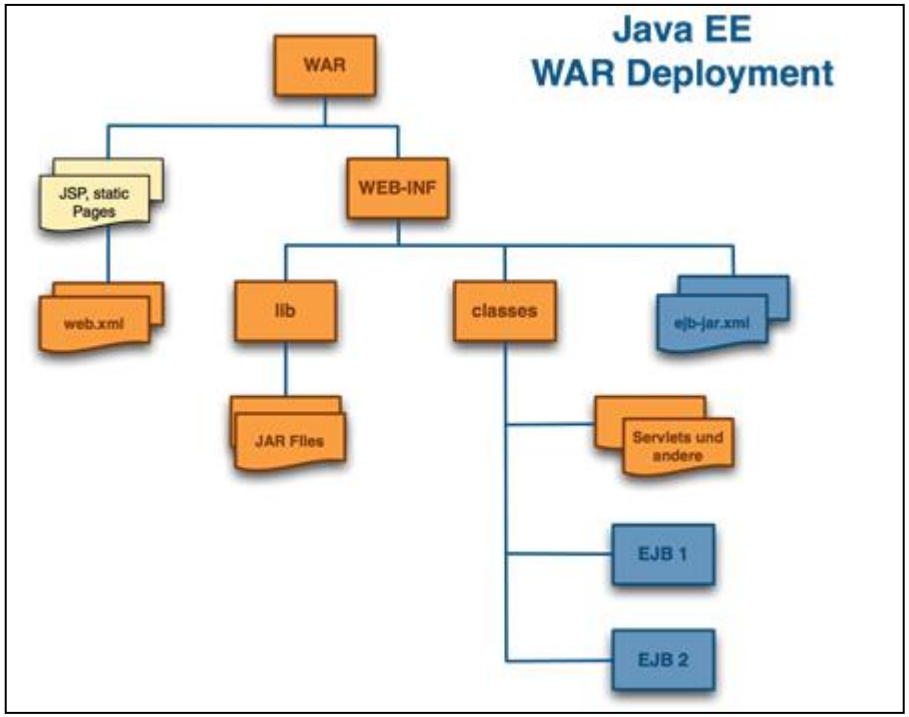
```
@Stateless //or @Stateful or @Singelton
public class HelloSessionBean {

public java.lang.String hello() {
    return "Hello World!";
  }
}
```

EJB-Container

```
public class InternalClient {

@EJB
private static HalloSessionBean halloEjb;

 public static void main (String args[]) {
   System.out.println("Out:" + halloEjb.hallo() );
 }
}
```

EJB-Client

# Modul 8
# Remarks: *No-interface* view of Session Beans




Deployment structure **without** business interface (*no-interface* view).
Supported by Full-Profile- and by WebProfile-servers (EJBLite).
Netbeans: New Project -> Java Web -> Web Application

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

1. Do I need remote EJBs?

   Yes, distributed computing is required,…
   ⇒ Interface view
   ⇒ Deploy EJB-Module “alone” or EAR-Deployment
   ⇒ Server must support Remote EJBs (full profile)

   No, I only work on my local side, …
   ⇒ *No-Interface* view is sufficient
   ⇒ WAR-Deployment is sufficient
   ⇒ Server with Web-Profile is sufficient

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

2. Create the EJBs

   Remote EJBs:
   - Create Interface
   - Create Implementation
   - Add Annotation (@stateless, @stateful, @singleton)

   Local EJBs:
   - Create Implementation
   - Add Annotation (@stateless, @stateful, @singleton)

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

3. Create the EJB-Client

   Remote EJBs:
   - JNDI-LookUp

   Local EJBs:
   - JNDI-LookUp or Dependancy Injection (**`@EJB`**)

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle

4. Deploy EJB-Component and Client

   Remote EJBs:
   - EAR-Deployment or deploy EJB-Module without client

   Local EJBs:
   - WAR-Deployment

Quelle: Developing Applications for the Java™ EE Platform , FJ-310-EE5, oracle