J2EE

EJB 3.0

Agenda

1 **Session Bean**

2

Objectives

At the end of the module, you will be able to:

- Analyze Session Beans as EJBs
- Develop Stateless Session Beans using **EJB 3.0**

Session Bean

Session Beans

Session Beans are Java components that run in EJB Containers and are typically used to model a particular user task or back-end Process. E.g.: entering customer information, implementing a process that maintains a conversation state with a client application.

Session Beans can hold the business logic for many types of applications.

Session Beans are of two types:

- Stateless
- Stateful

Stateless Session bean does not maintain any client state information.

Stateful beans maintain the state and a particular instance of the bean is associated with a specific client request.

Session Beans

- **Session Bean** : A session bean usually represents a single client. It is not shared across clients. When a client invokes any method of a Session Bean, it is directed through the container to the bean. The session bean is responsible for executing the business logic for the client.

- **Stateful Session Bean** : A stateful session bean is responsible for maintaining the conversational state of a client for the duration of that particular session. Once the client completes his transaction(in other words finishes interacting with the enterprise bean), the EJB container removes the bean and the session ends and all the state data is discarded.

- **Stateless Session Bean** : A stateless session bean does not maintain conversational state for a client. Each request for a stateless session bean is considered as a request from a new client. Any instance variable state will be lost between invocation to the bean.

Since stateless session beans do not maintain state information, the EJB container can reuse bean instances there by optimizing the performance.

Stateless Session Beans

- An EJB 3.0 Stateless Session bean is a POJO, maintained by an EJB Container with a class level annotation of ***@Stateless***
- The functionality of a session bean is defined by its business interface(service interface), which is actaully nothing but a plain old Java interface
- Through this interface, the session bean client retrieves an object(which is known as stub) from the server's JNDI
- The stub object implements bean's business interface

Source: Ratzinger, 2005.

Stateless Session Beans (Contd.).

- The client makes calls to the bean interface method on the stub object
- The stub object is responsible for passing the client calls to the actual bean instance in the container. These bean instances will have the implementations of these methods and will be responsible for carrying out the actual job
- The stub object is automatically generated and sent to the client by the container. It is responsible for routing the method calls to the container. The developer does not have to provide implementation for the stub object

Business Interface

Stateless Session Beans require Business interfaces. But it is not mandatory to define a business interface. It can be automatically generated for you, in case you fail to provide one. This is done through the use of annotation.

A business interface can be local or remote. The type of business interface that is generated automatically, depends on the annotation used. It is either @Remote or @Local. If there is no annotation, then the interface will be designated as local.

All the methods of the bean class will find a corresponding method declaration in the generated business interface.

It is perfectly alright to have a session bean implement multiple interfaces, each interface targetting different types of client (remote or local).

Home Interface and the Bean Class

Home interfaces

Home interfaces are not needed for Stateless session beans.The client acquires a reference to a stateless session bean by means of an injection or annotation.

BeanClass

A Stateless session bean must be annotated with the annotation @Stateless. You can also use the deployment descriptor to denote this bean as a stateless session bean.

The point to remember here is that the bean class need not implement the interface javax.ejb.SessionBean

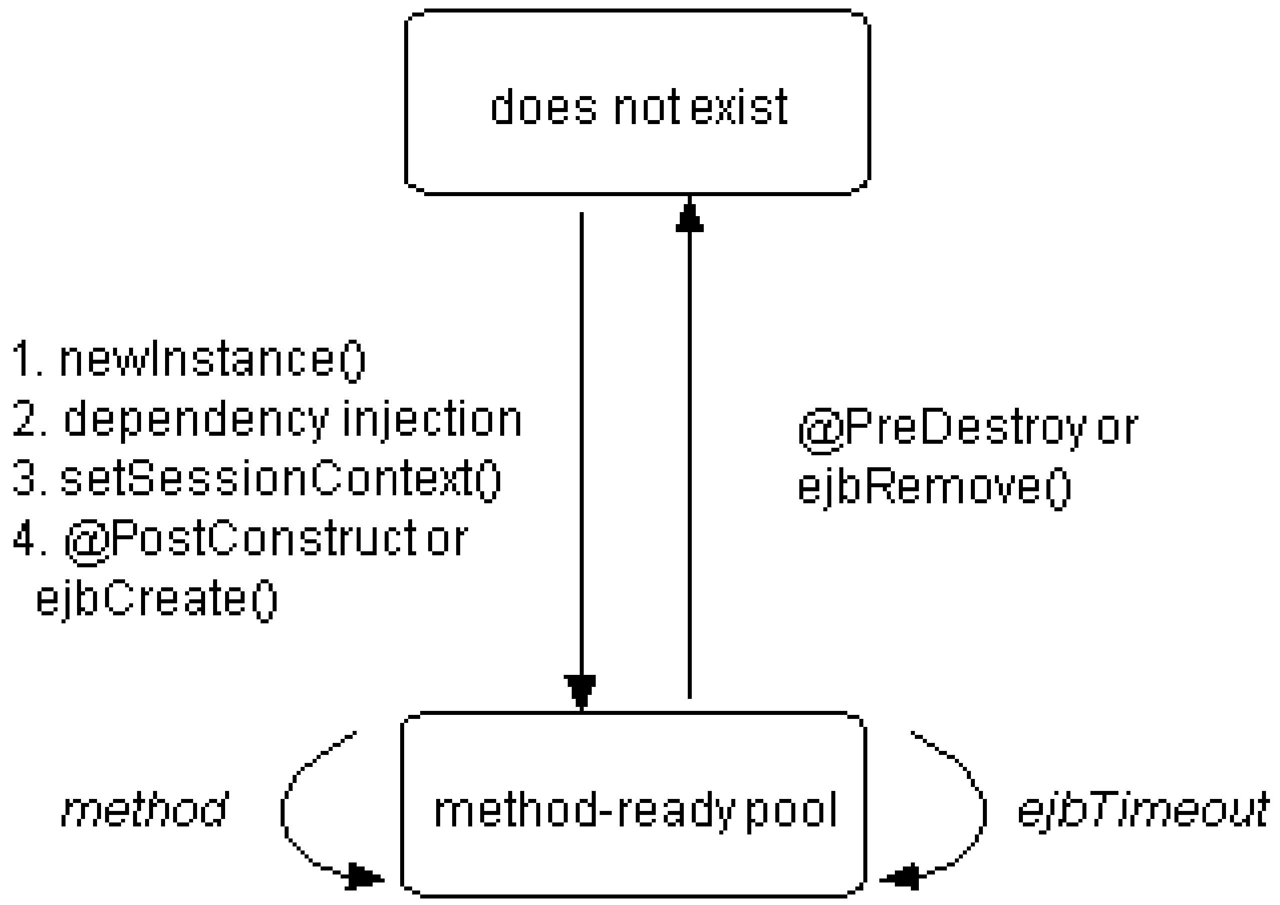
Sample Stateless Session Bean

```
 import  javax.ejb.*;

@Stateless
@Remote

 public class HelloWorldBean
{
  public String sayHello() {
        return “Hello!!”;
   }
}
```

Callbacks in Stateless Session Beans

- Following are the life cycle event callbacks, which are supported for Stateless session beans :
- **PostConstruct**
- This occurs after any dependency injection has been performed by the container and before the first business method is invoked.
- **PreDestroy**
- This occurs at the time the bean instance is destroyed.

Life Cycle of Stateless Session Bean



Source: Purich, 2007.

The Client

The stub object is created when the session bean is deployed into the EJB container. At this point, it gets registered in the server’s JNDI registry.

The client program accesses the stub of the bean using the class name of the interface in the JNDI registry. The client can make method calls on this stub object and this call is delegated to the bean instance.

When a client is looking for the session bean stub through the remote interface, the container returns a stub object which is in serialized form. This serialized stub object implements the remote interface.

The Client (Contd.).

- A client application that accesses session beans can be one of three types:
  - Remote
  - Local
  - Web Service
- **Remote Client** runs in a separate JVM from the session beans they access.
  - A remote client can be another EJB, a Java client program or a Java Servlet/JSP, which are location independent.
  - A remote client accesses a session bean through the bean’s **remote** business interface.
- **Local Client** runs in the same JVM
  - A local client can be another EJB or a web application, which are location dependent.
  - A local client accesses the session bean through the local business interface.

The Client (Contd.).

Web service Client can be any type of client application that can send and receive SOAP messages. Since the stateless session bean is published as a Web Service, to access it, programmatic interfaces should be generated from the WSDL document.

When the cllient uses the remote interfaces, the stub is serialized and de-serialized. All the calls to the bean object are made over the network. Naturally this approach is less efficient as compared to the local interface. This is the precise reason why you should not try to use a remote interface when the client is local.

Match the following :

(a) EJB Container	(a) Type of a Session Bean
(b) Transaction Management	(b) Runs in the same JVM as Server
© Business Interface	© A Runtime environment for managing EJBs
(d) Stateful	(d) POJI
(e) PostConstruct	(e) One of the several services provided by EJB Container
(f) Local Client	(f) Life Cycle Event Callback

Summary

In this module, you were able to:

- Analyze Session Beans as EJBs
- Develop Stateless Session Beans using **EJB 3.0**

Thank You