Session: 3

Session Beans

Objectives

- Describe Session beans
- Describe the characteristics of Session bean
- List the different types of Session beans
- Explain Stateless Session beans
- Describe the lifecycle of Stateless Session beans
- Explain the process for developing a Stateless Session bean in NetBeans IDE
- Describe the types of communication with Session beans
- Explain asynchronous communication in Stateless Session bean

Introduction

- Session beans represent business processes that are used to handle business logics for the application.
- A business logic can be:
  - Performing addition on two numbers
  - Connecting to a database
  - Performing transaction on a bank account
  - Invoking other Session beans or Message-driven beans

Session Beans 1-3

- Are reusable Java components.
- Execute on the server-side in an EJB container.
- Are deployed on Java-enabled application servers.
- Provide services to their clients which can access them programmatically.
- Are conversational.

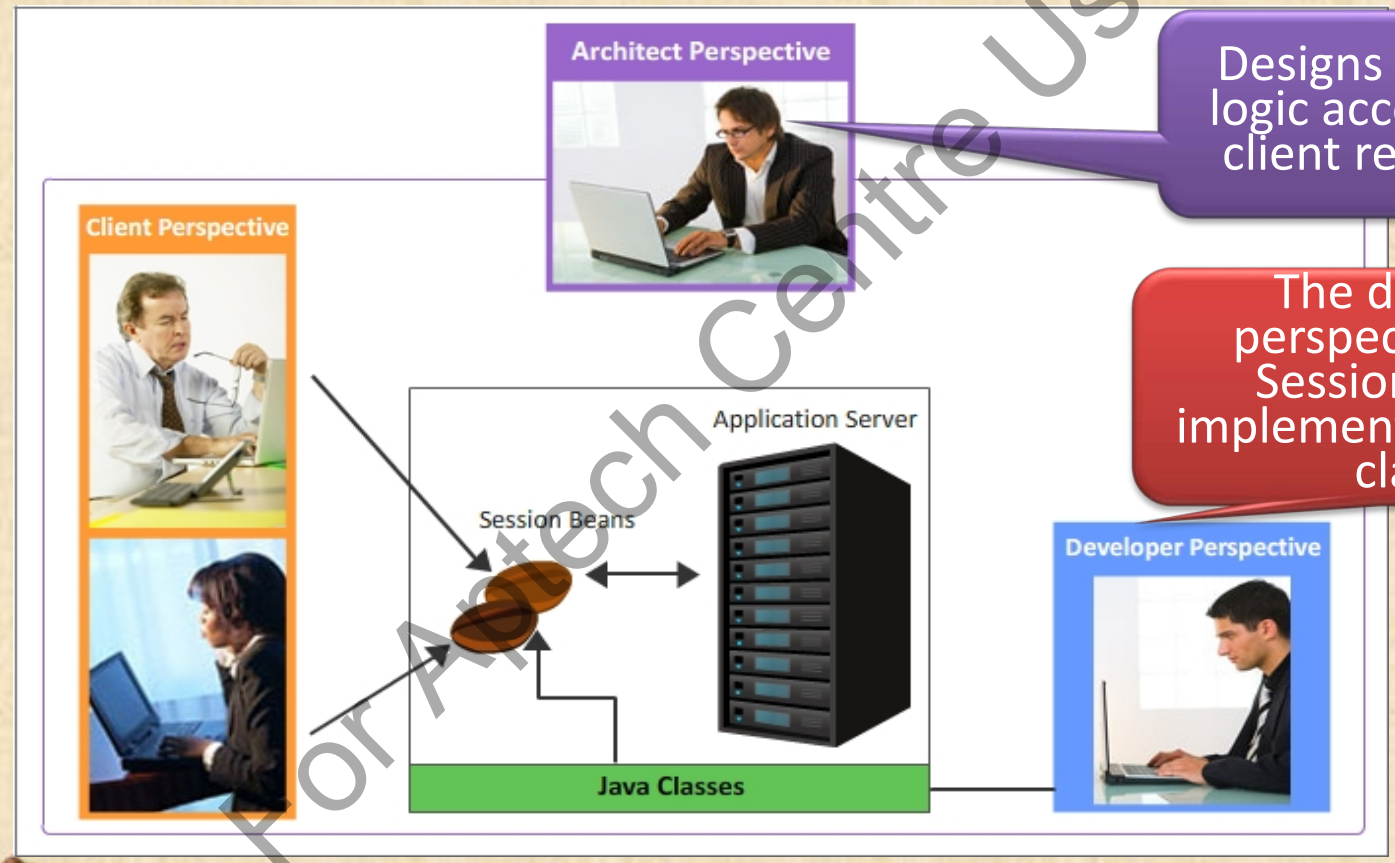
A conversation is an interaction between a bean and a client and it is composed of a number of method calls between them.

Session Bean 2-3

- Session bean components implement `javax.ejb.SessionBean` interface.
- Session beans are not persistent across system failures.
- Lifetime of Session bean:
  - An instance of a Session bean is alive as long as the client is present.
  - Hence, the lifetime of a Session bean is equivalent to the lifetime of a client.

Session Bean 3-3

- Following figure shows the different perspectives for Session beans based on who is accessing it:



Types of Session Beans 1-4

- Based on the lifetime of the Session bean, there are three variants of Session beans:
  - Stateless Session beans
  - Stateful Session beans
  - Singleton Session beans

Types of Session Beans 2-4

❑ Stateless Session beans

- Does not maintain the client state on the server during conversation which means it does not stores the value of the instance variables associated with the client conversation.
- After the method execution is completed, the values are not retained, that is, they are deleted by the container.
- Some of the situations which do not require the client's data to be maintained and are suitable for Stateless Session bean development are:
  - SalaryCalculator
  - MoneyConverter
  - CardVerification

Types of Session Beans 3-4

❑ Stateful Session beans

- Maintain the state of the clients on the server which means they store information obtained from the specific client between the method invocations.
- Some of the situations where it is necessary to store the client's information to maintain its identity and are suitable for Stateful Session bean are:
  - Shopping cart
  - Online banking
  - Product order processing system

Types of Session Beans 4-4

Singleton Session bean

- Are similar to Stateless Session bean.
- Are instantiated once for the entire application.
- A single instance of the Singleton bean is share by all the clients.
- Some of the scenarios where Singleton Session bean can be used for:
  - Executing initialization
  - Clean tasks of the application when it shuts down.

Characteristics of Session Beans 1-2

- Regardless of the type of Session Beans developed in the enterprise application, they possess the following characteristics:
  - Session beans are short-lived objects.
  - Each Session bean instance is associated with a single client.
  - Session bean instance cannot be shared by multiple clients.
  - Session beans represent the state of communication between the client and bean.
  - Session beans are instantiated and managed by the EJB container.
  - Session beans support synchronous and asynchronous communication.
  - Session beans are non-persistent.
  - Session beans can implement transaction boundaries and security mechanisms.

Characteristics of Session Beans 2-2

When should the Session beans be used in enterprise applications?

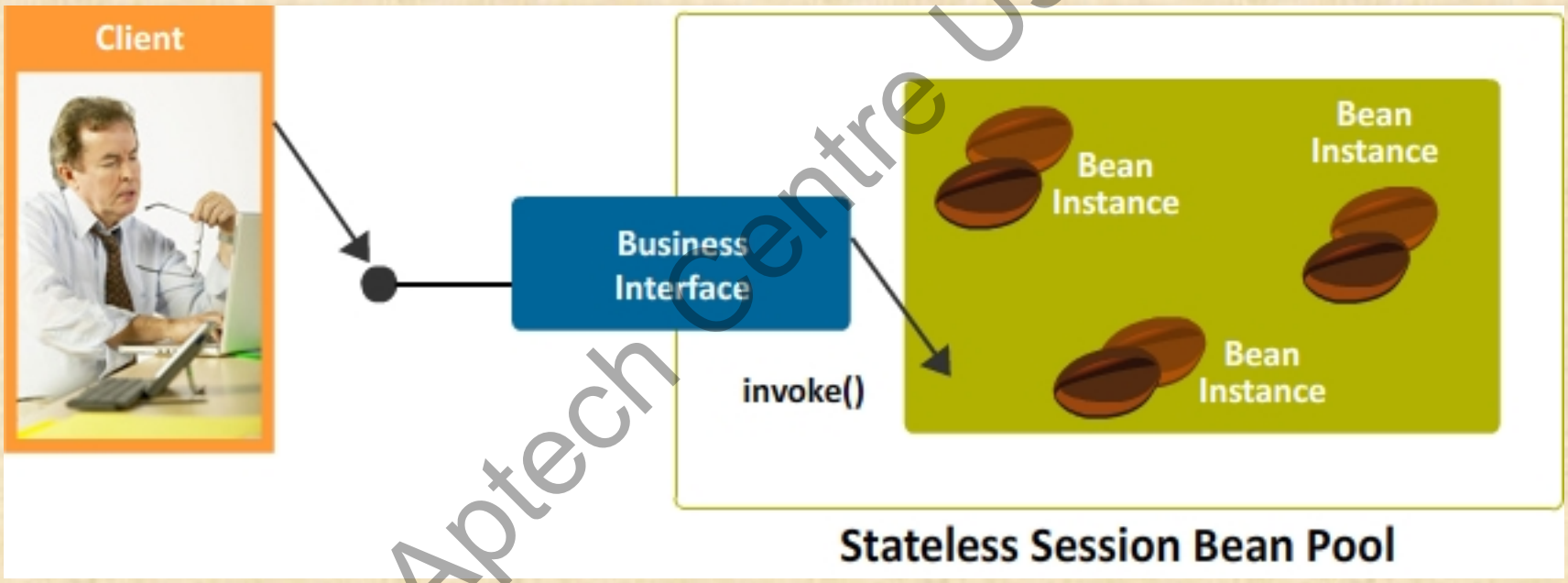
- When the application has to retain the state across multiple method invocations.
- When there is a requirement of communication between the client and other components of the application.

Stateless Session Bean 1-3

- Is invoked by the client when it requires service of an application server.
- Does not maintain the conversational state of the session with the client.
- Are maintained in a pool by the container.

Stateless Session Bean 2-3

- Following figure demonstrates a pool of beans instantiated and managed by the container on the application server:



Stateless Session Bean 3-3

A pool of Stateless Session Beans is instantiated into the container during application deployment

Client has to access the Session bean, it uses the business interface of the Stateless Session bean

Session Bean provides required service

Session Bean is returned back to the pool of bean instances after client request is serviced

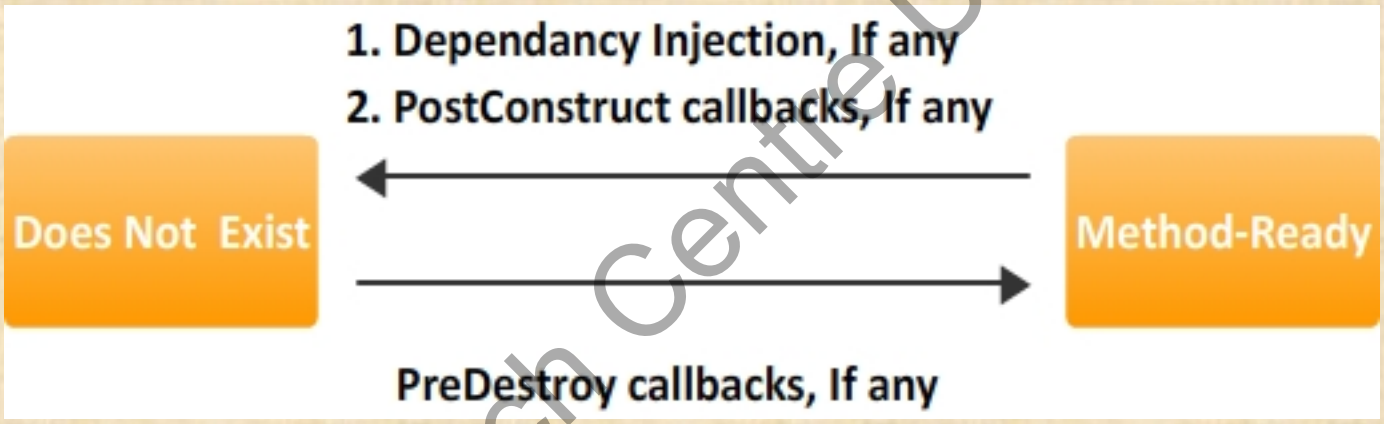
Callback Methods for Stateless Session Bean

Callback methods are the methods used by the container to manage the lifecycle of a Session bean

- PostConstruct callback methods
- PreDestroy callback methods

Lifecycle of a Stateless Session Bean 1-2

- Following figure shows different states in the lifecycle of a Stateless Session bean:



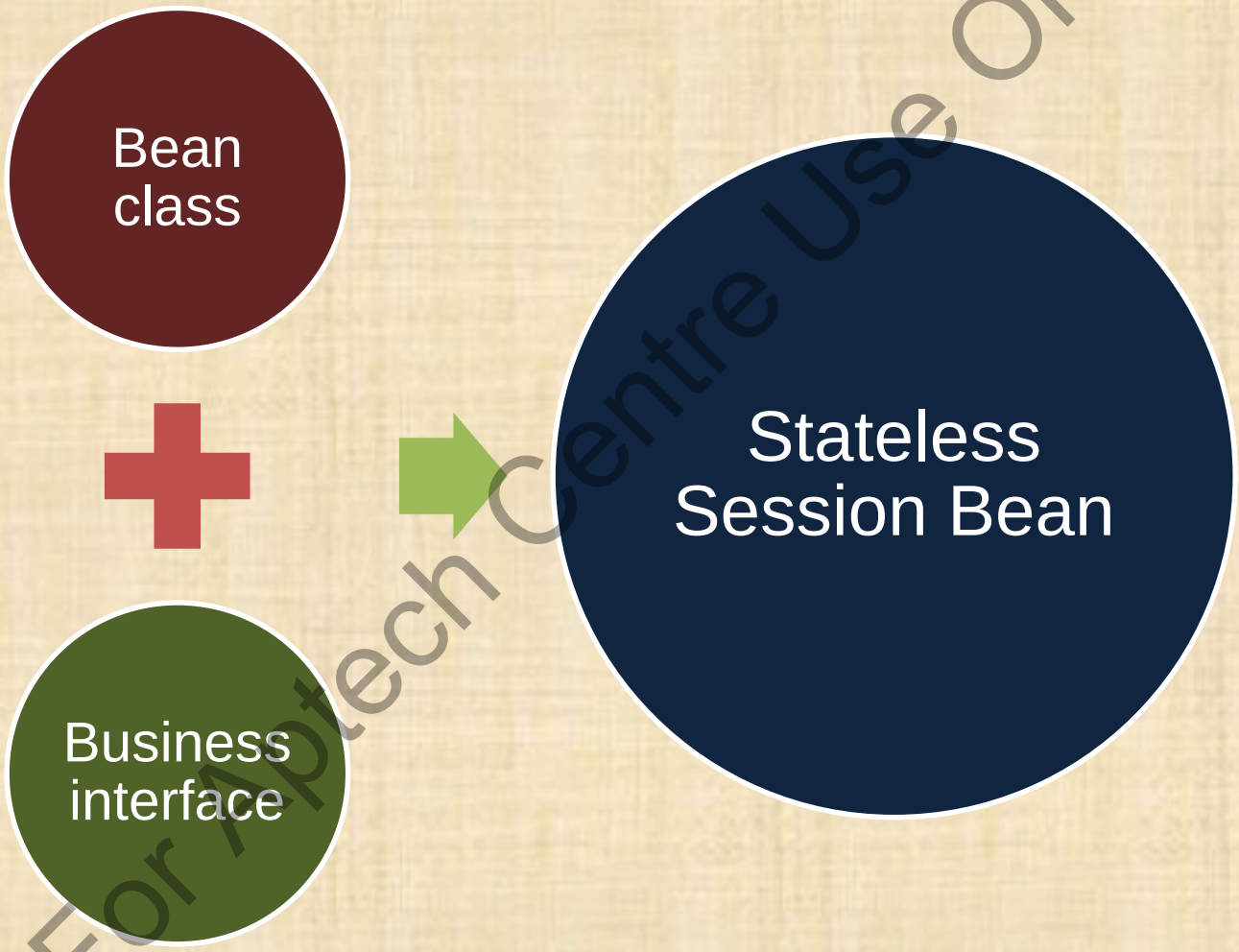
Lifecycle of a Stateless Session Bean 2-2

- Initially the bean is in a '**Does Not Exist**' State.
- Bean is instantiated by the container when there is a client request or during application start up.
- An instance of Session bean is created through **`newInstance()`** method.
- After instantiating all Session beans are placed in method-ready pool.
- In this state, beans are ready to take requests from the client request and respond accordingly.
- Execution of **`PostConstruct`** and **`PreDestroy`** methods is optional.

Annotations in Stateless Session Bean

- Annotations are metadata which are used by the container during application deployment.
- A Stateless Session bean can be created by annotating with **`@Stateless.`**
- Interfaces of the Session bean can be annotated with **`@Local`** and **`@Remote`** interfaces.
- Life cycle callback methods of the Stateless Session bean can be annotated with **`@PostConstruct`** and **`@PreDestroy.`**

Developing Stateless Session Bean 1-10



Developing Stateless Session Bean 2-10

- Following are some of the rules to be followed while developing a Stateless Session bean:
  - There should be at least one business interface for the Session bean.
  - Session bean class cannot be declared as **`final`** or **`abstract`**.
  - Session bean class should implement a no argument constructor.
  - Session bean class can be a subclass of another Session bean or a POJO class.
  - Business and life cycle callback methods are defined in the Session bean class or **`super`** class.
  - Business method names present in the business interface and in the Session bean class must not begin with **ejb**.
  - Business methods should be declared as public and cannot be declared as **`final`** or **`static.`**

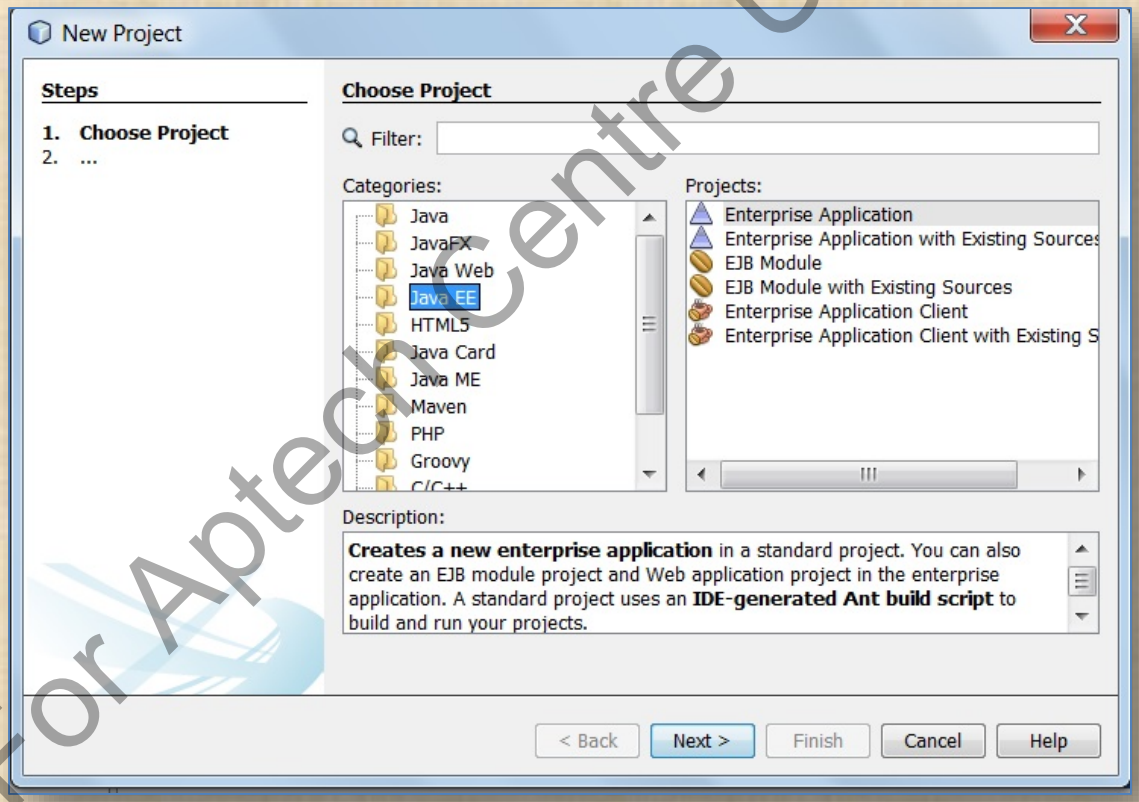
Developing Stateless Session Bean 3-11

To create a Stateless Session bean in NetBeans IDE, perform the following steps:

- Open NetBeans IDE.
- Select **File → New Project** to create an enterprise application. Choose New Project from the File menu.

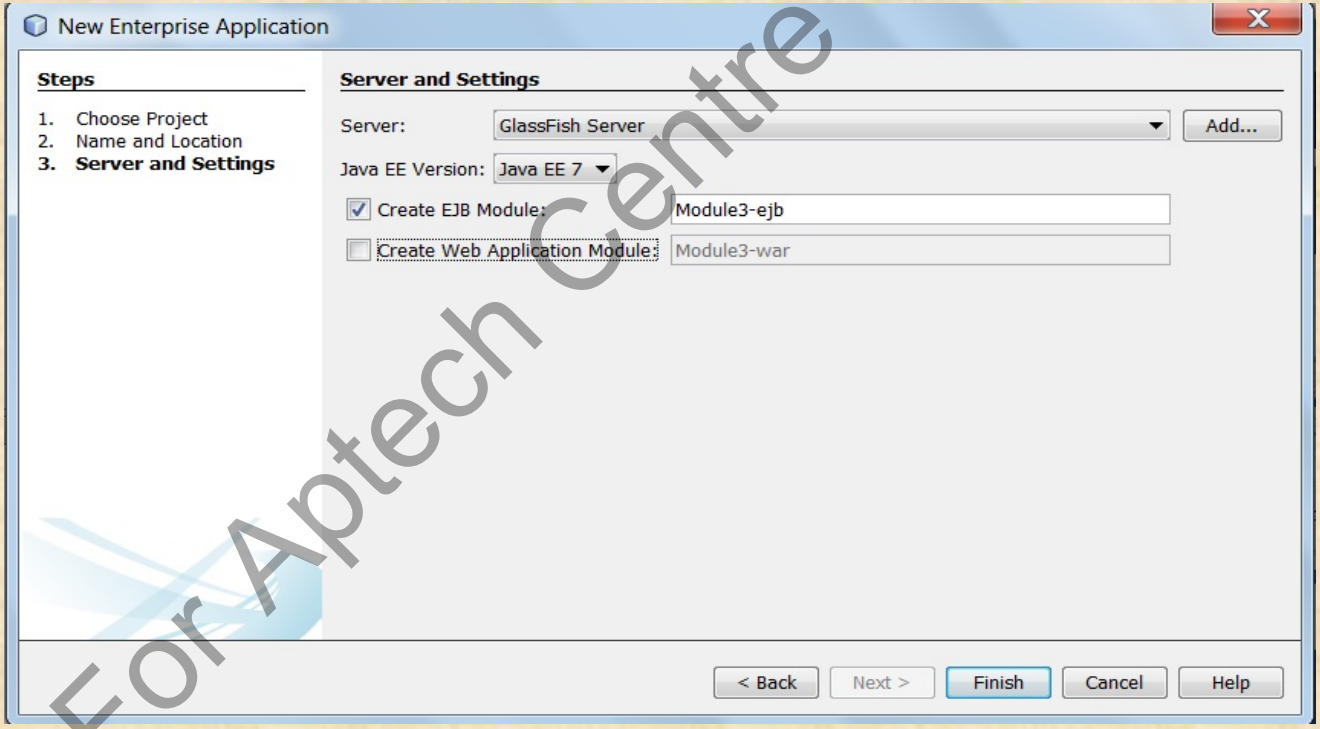
Developing Stateless Session Bean 4-11

- Following figure shows the screen which prompts the developer for the enterprise application name, select the appropriate name and click **Next**:



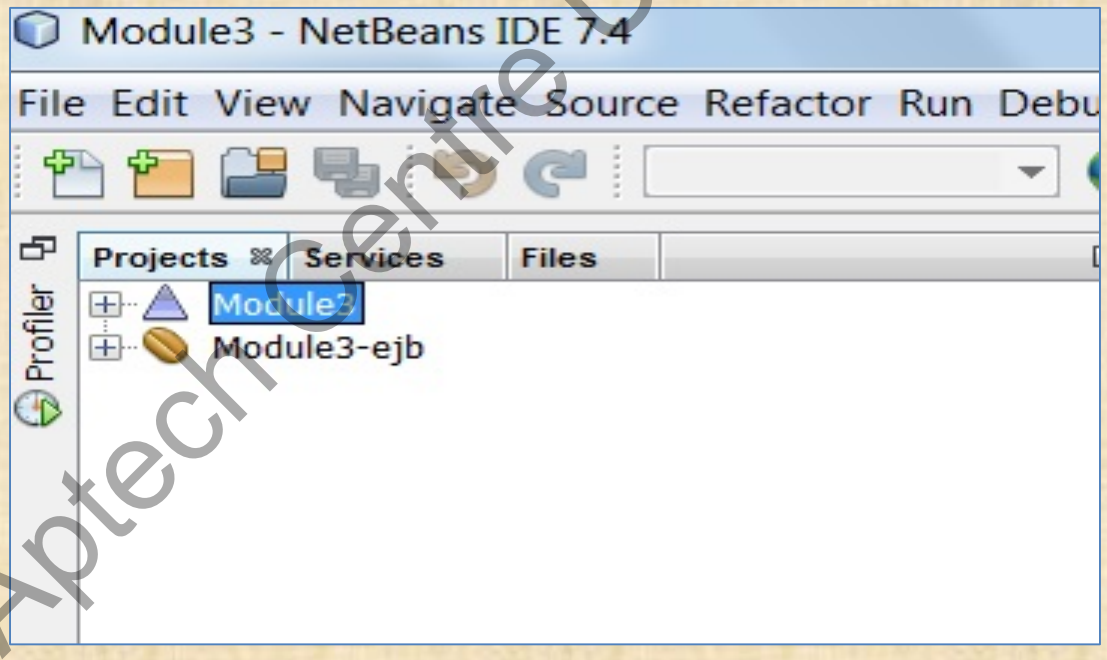
Developing Stateless Session Bean 5-11

- Following figure shows the screen where the developer has to select the application server onto which the application has to be deployed:



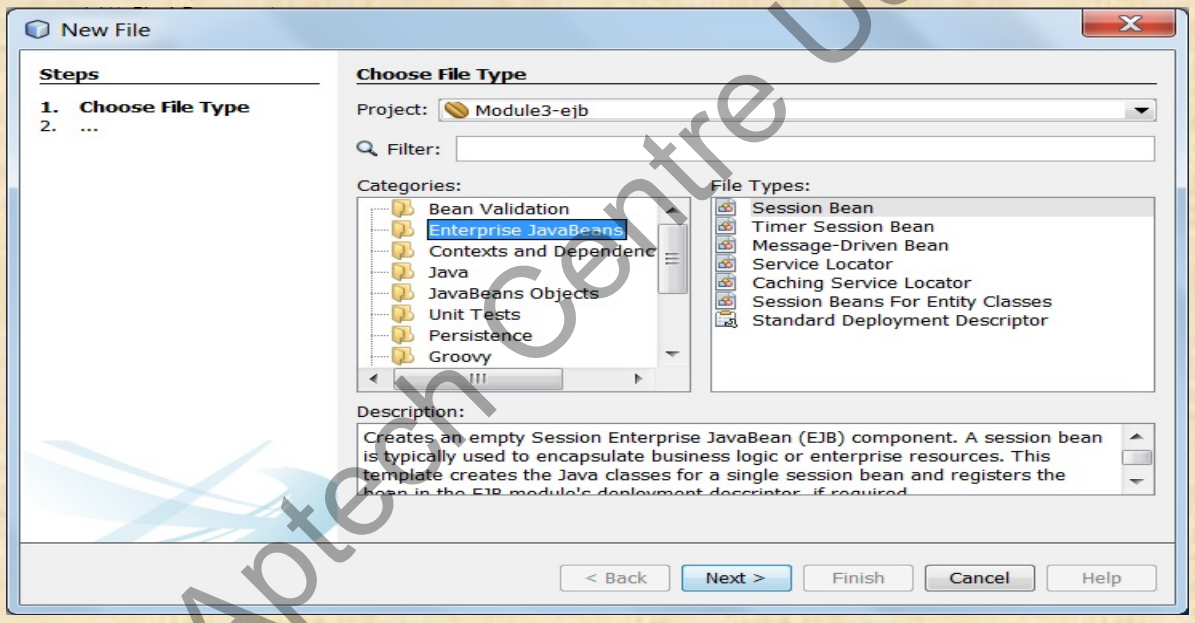
Developing Stateless Session Bean 6-11

- Following figure shows the Project in the **Project** window after it is created by the IDE:



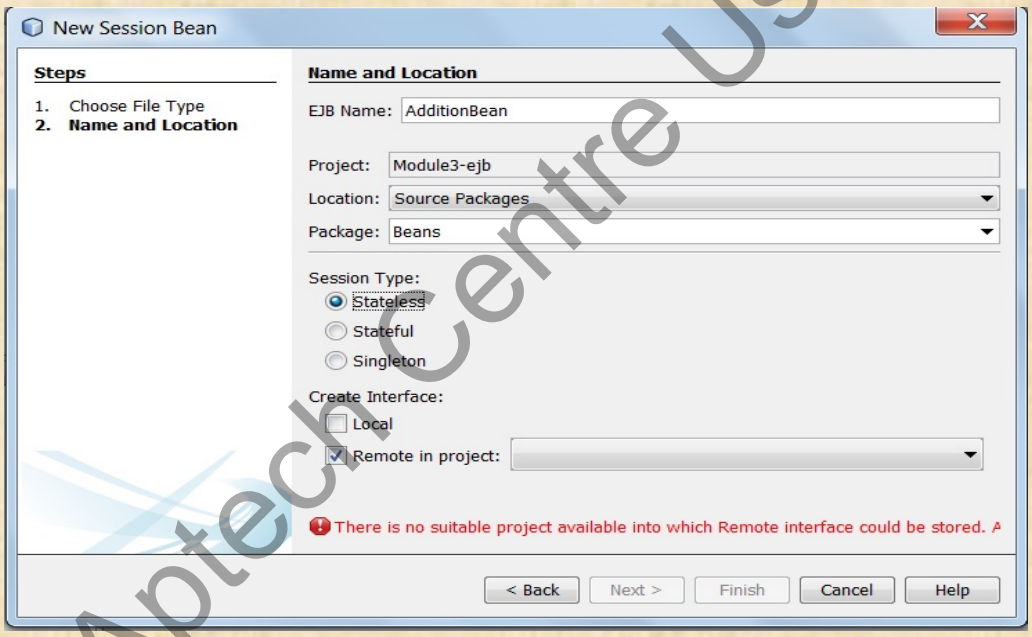
Developing Stateless Session Bean 7-11

- Following figure shows how to create a Session bean after the Project has been created in the IDE:



Developing Stateless Session Bean 8-11

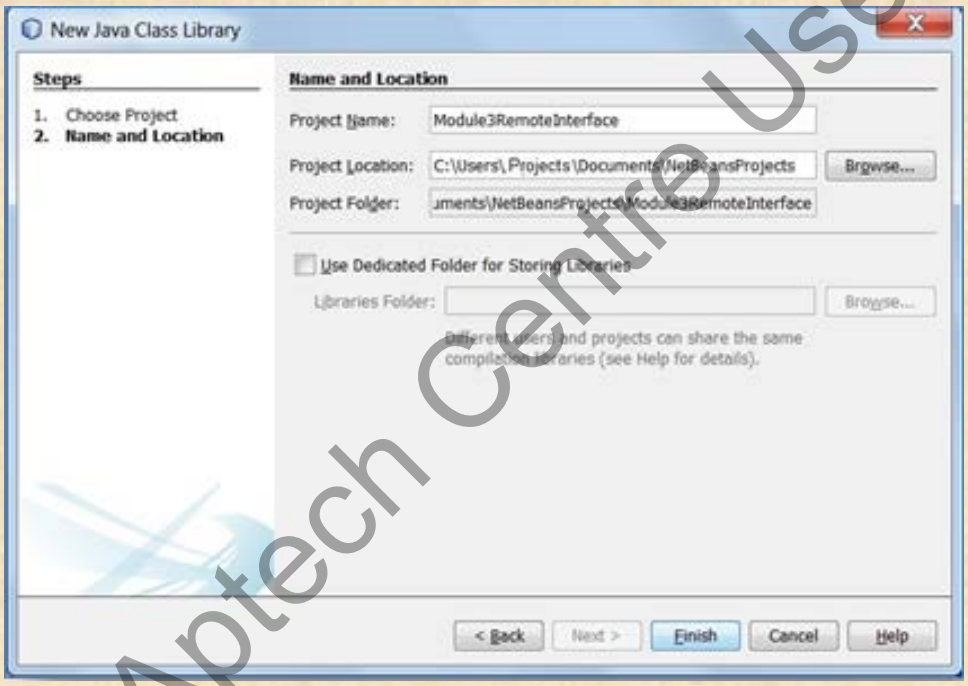
- Following figure shows the wizard to select options for the Stateless Session bean:



- A Stateless Session bean is created with remote interface to access it. The wizard prompts for creating a remote interface.

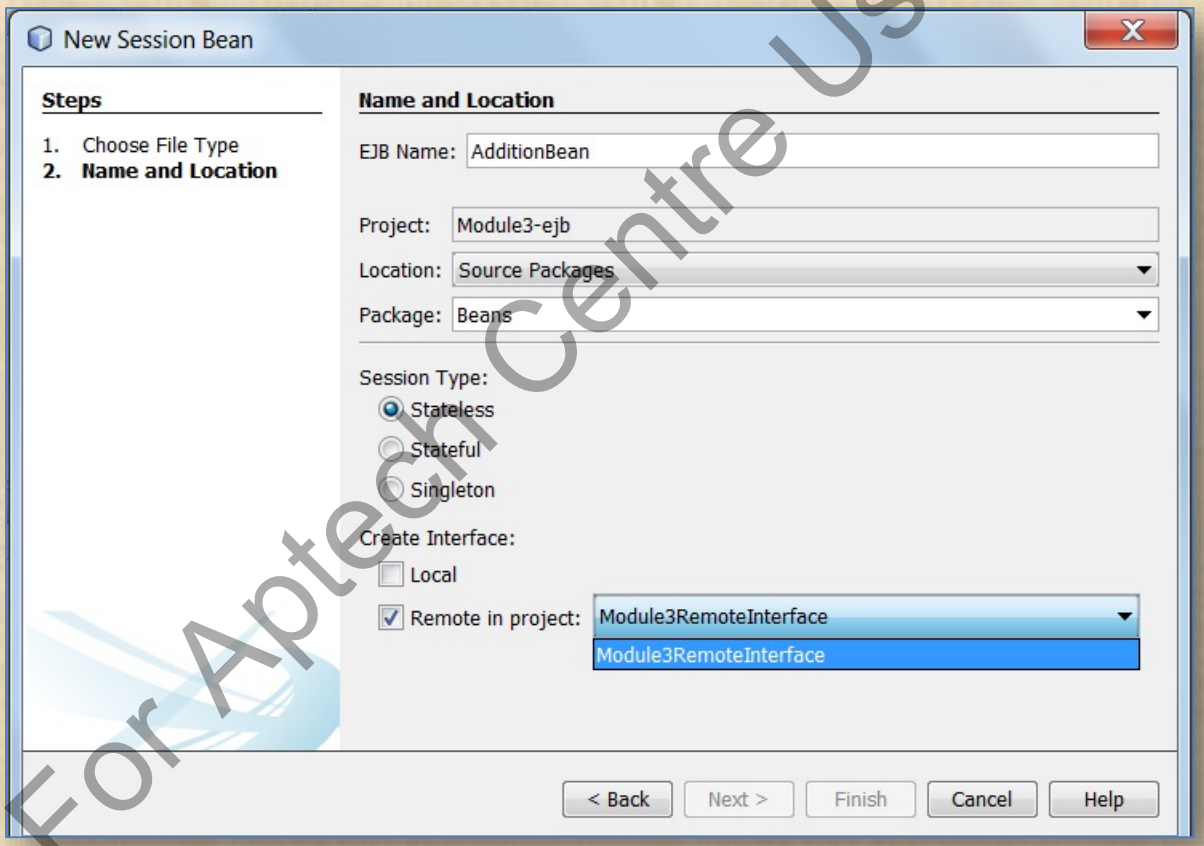
Developing Stateless Session Bean 9-11

- Following figure shows how to create a remote interface:



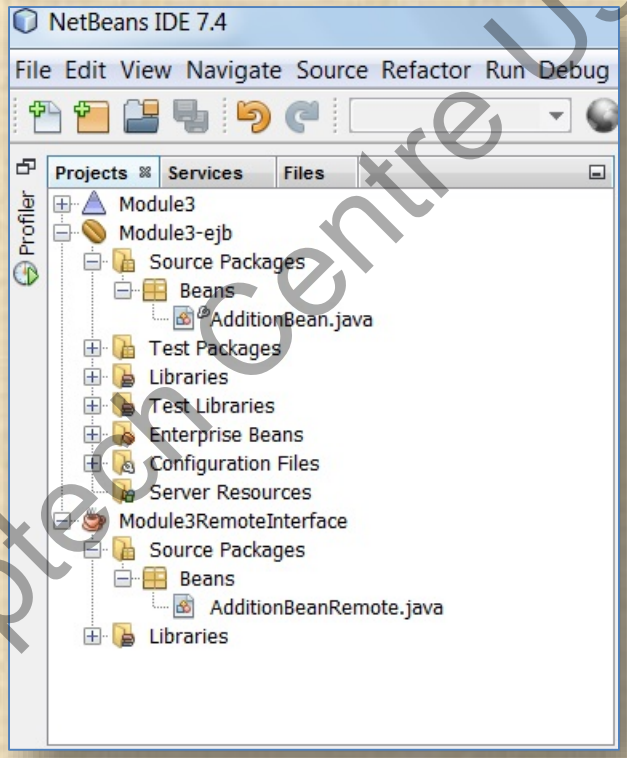
Developing Stateless Session Bean 10-11

- Following figure shows the bean creation process by choosing remote interface:



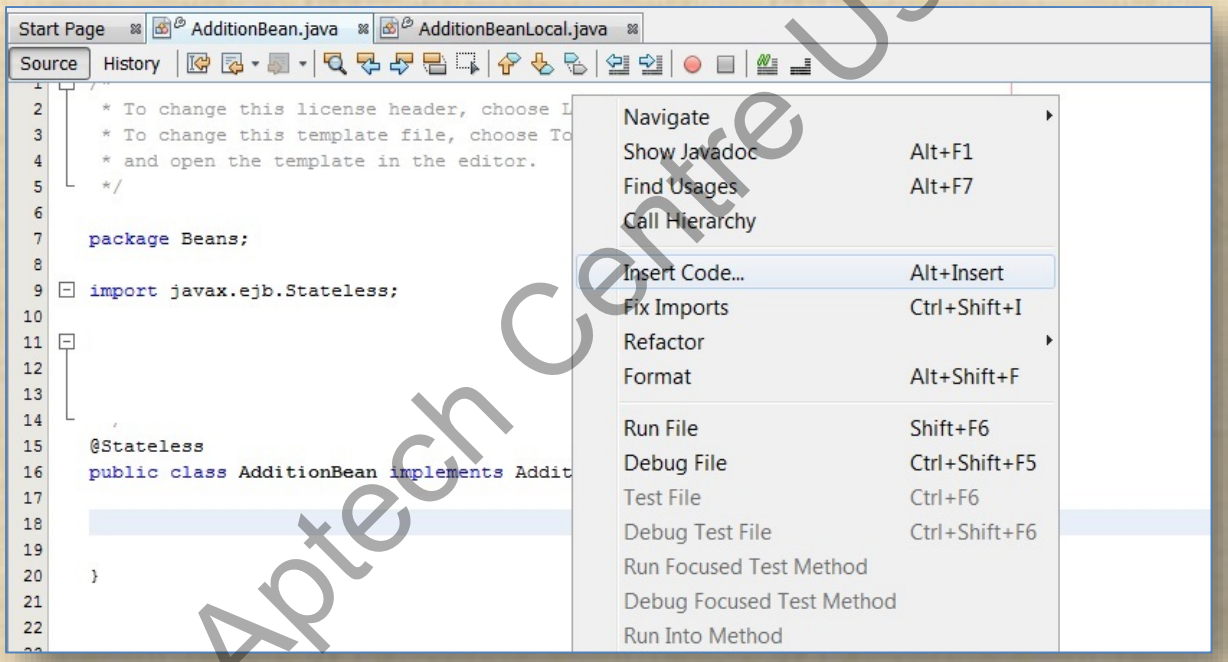
Developing Stateless Session Bean 11-11

- Following figure shows the directories created after creating the Stateless Session bean:



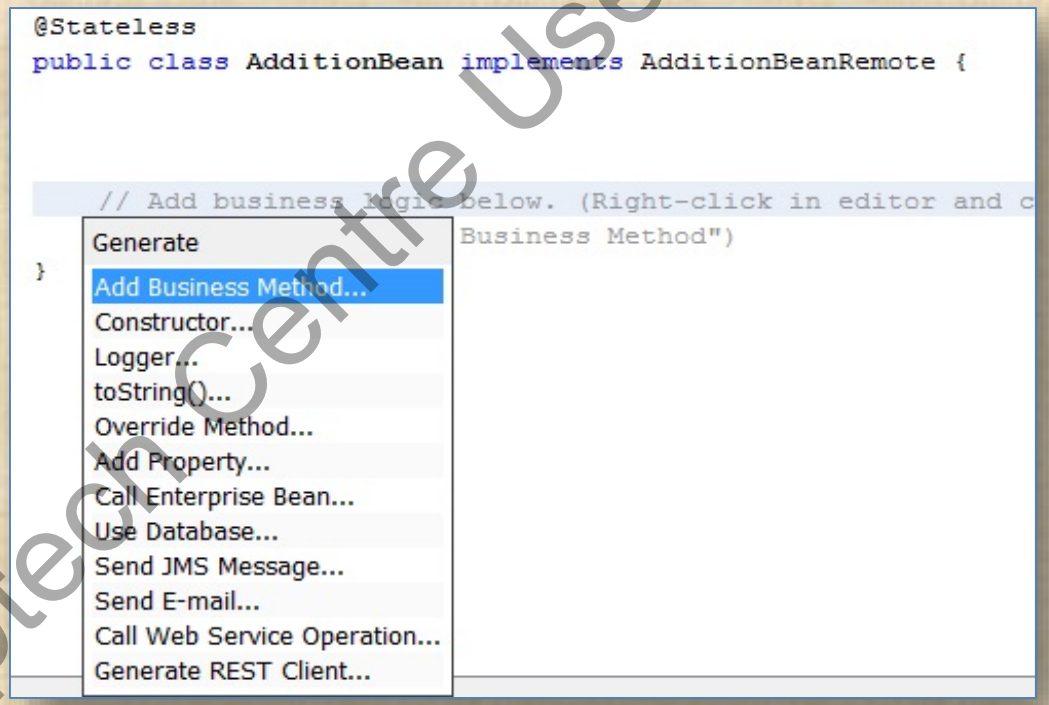
Creating Business Methods 1-6

- Following figure shows how to create business method in the Session bean **`AdditionBean.java`:**



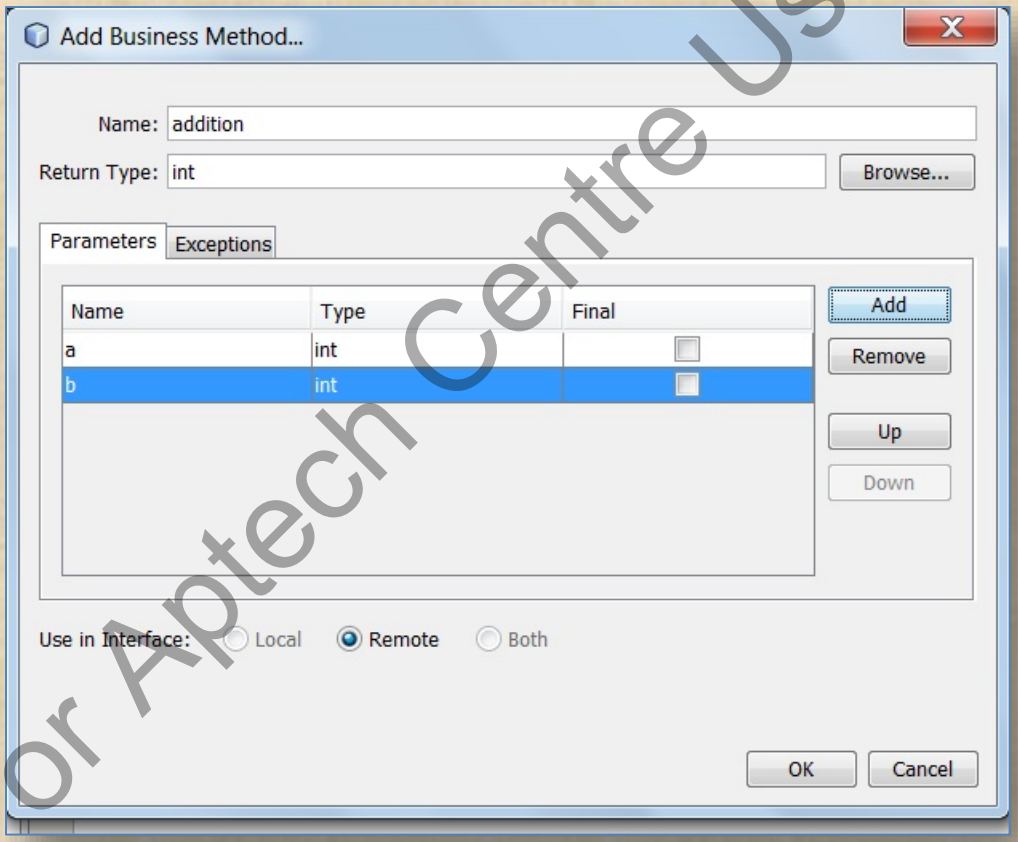
Creating Business Methods 2-6

- Following figure shows how to add business method in Session bean:



Creating Business Methods 3-6

- Following figure shows how to configure the business method through the wizard provided by NetBeans IDE:



Creating Business Methods 4-6

- Following figure shows the code created by the wizard:

```
package Beans;

import javax.ejb.Stateless;


@Stateless
public class AdditionBean implements AdditionBeanRemote {

    @Override
    public int addition(int a, int b) {
        return 0;
    }
}
```

Creating Business Methods 5-6

- Following figure shows the modifications to be made to the remote interface of the Session bean:

```
package Beans;

import javax.ejb.Remote;

@Remote
public interface AdditionBeanRemote {

    int addition(int a, int b);


}
```

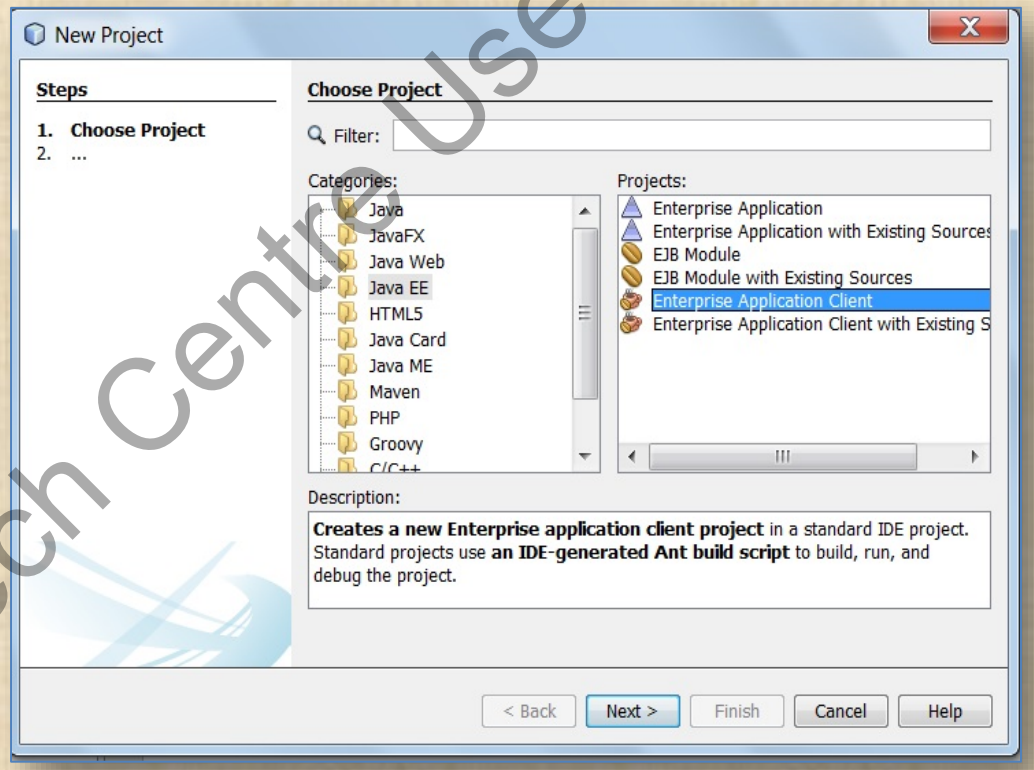

Creating Business Methods 6-6

- Following code snippet shows the code for the business method:

```
package Beans;
import javax.ejb.Stateless;
@Stateless
public class AdditionBean implements
AdditionBeanRemote {
  @Override
 public int addition(int a, int b) {
   return a+b;
}

}
```

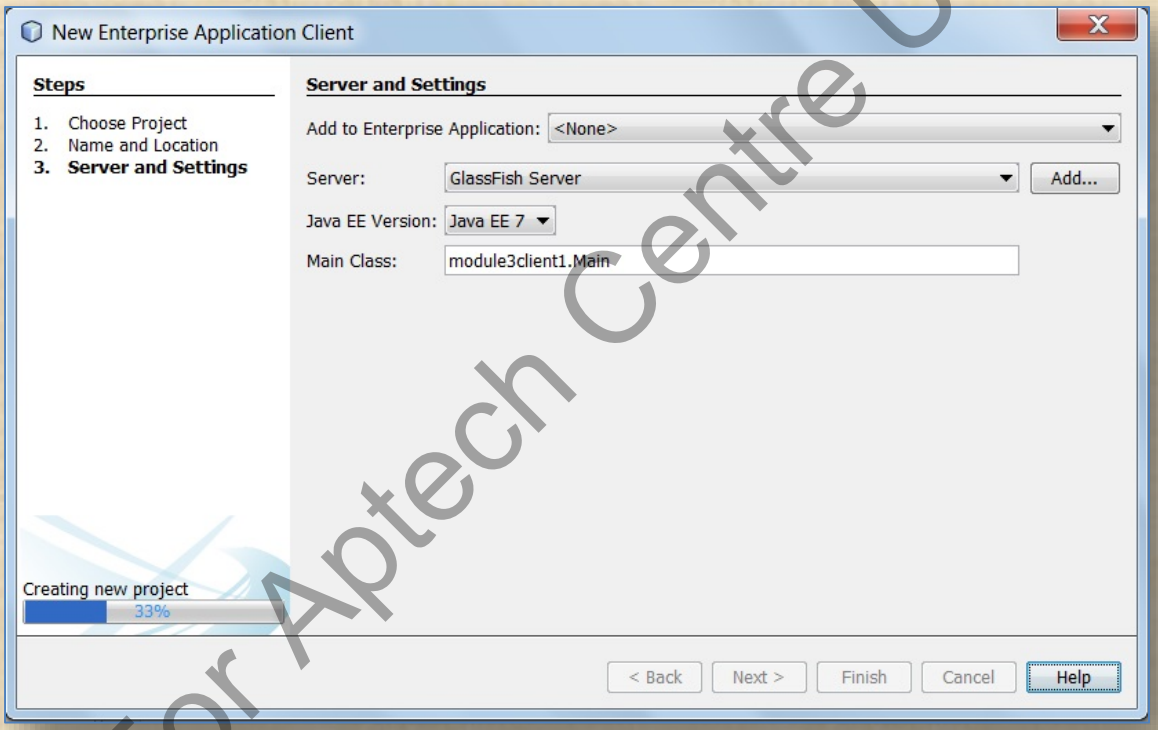
Creating a Client for Session Bean 1-5

- Following figure shows how to create a client to access the Session bean:



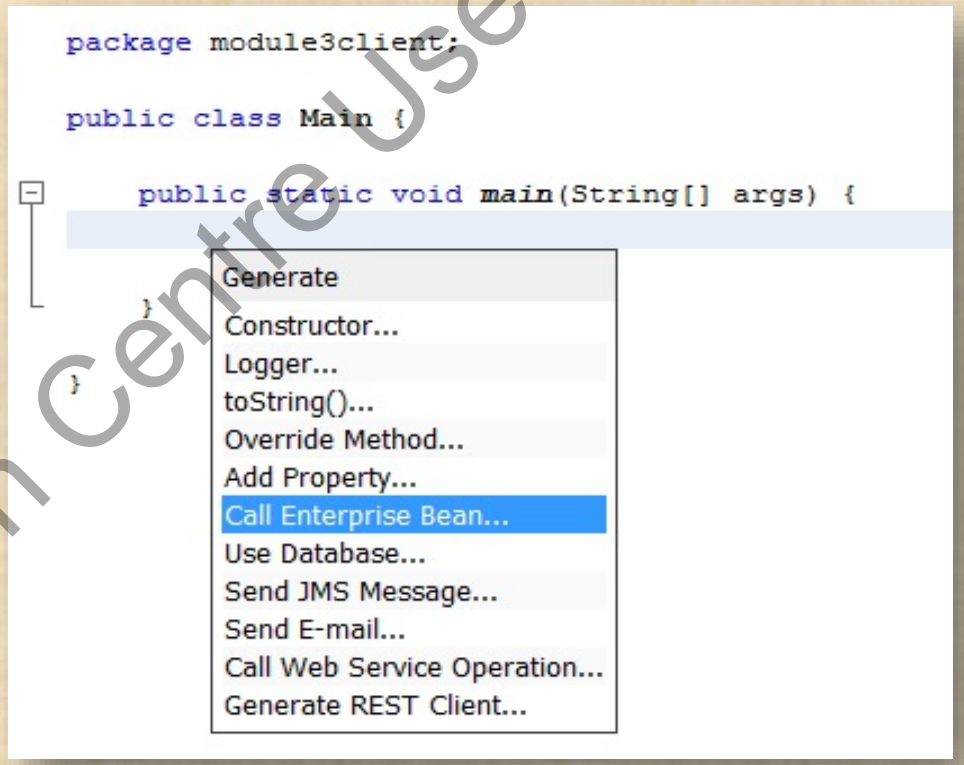
Creating a Client for Session Bean 2-5

- Following figure shows how to select the name for the client application:



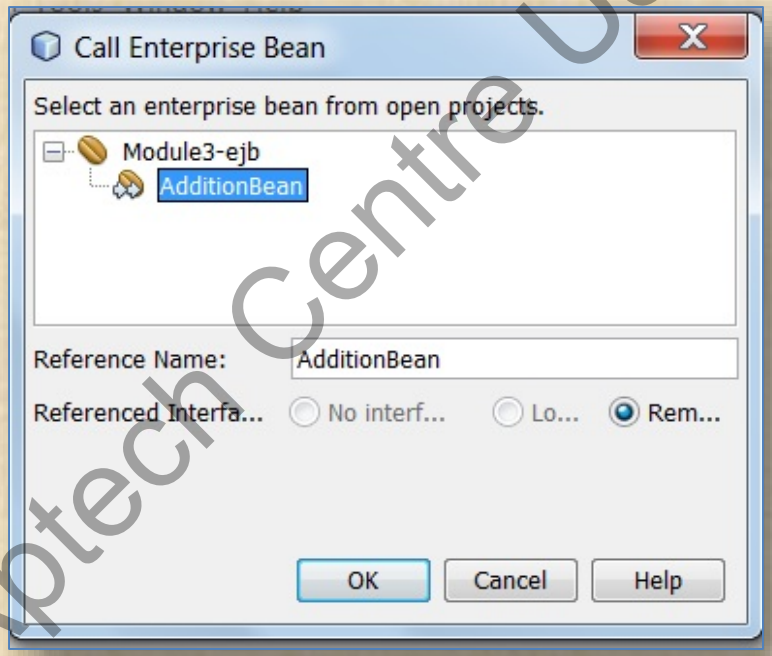
Creating a Client for Session Bean 3-5

- Following figure shows how to invoke a Session bean from the client application:



Creating a Client for Session Bean 4-5

- Following figure shows the wizard which allows choosing the Session bean to be accessed by the client:

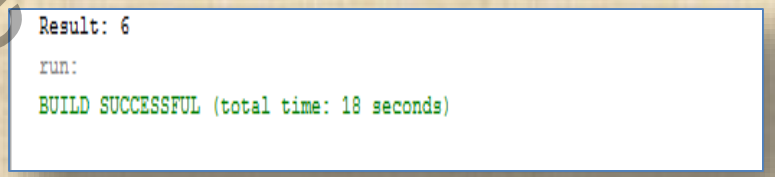


Creating a Client for Session Bean 5-5

- Following code snippet demonstrates the client code through which the session bean is accessed:

```
package module3client;
import Beans.AdditionBeanRemote;
import javax.ejb.EJB;
public class Main {
    @EJB
    private static AdditionBeanRemote
additionBean;
    public static void main(String[]
args) {

      System.out.println("Result:"+
additionBean.addition(4,2));
    }
  }
```

- Following figure shows the output after deploying and running the application client:



Configuring and Deploying Session Beans 1-2

- A Session bean can be configured through deployment descriptor and annotations.
- Deployment descriptor is a declarative XML file.
- **`ejb-jar.xml`** is the deployment descriptor for the enterprise application.
- Deployment descriptor determines how the EJB container manages the bean when deployed.

Configuring and Deploying Session Beans 2-2

- Following code snippet demonstrates the deployment descriptor for the Session bean created earlier:

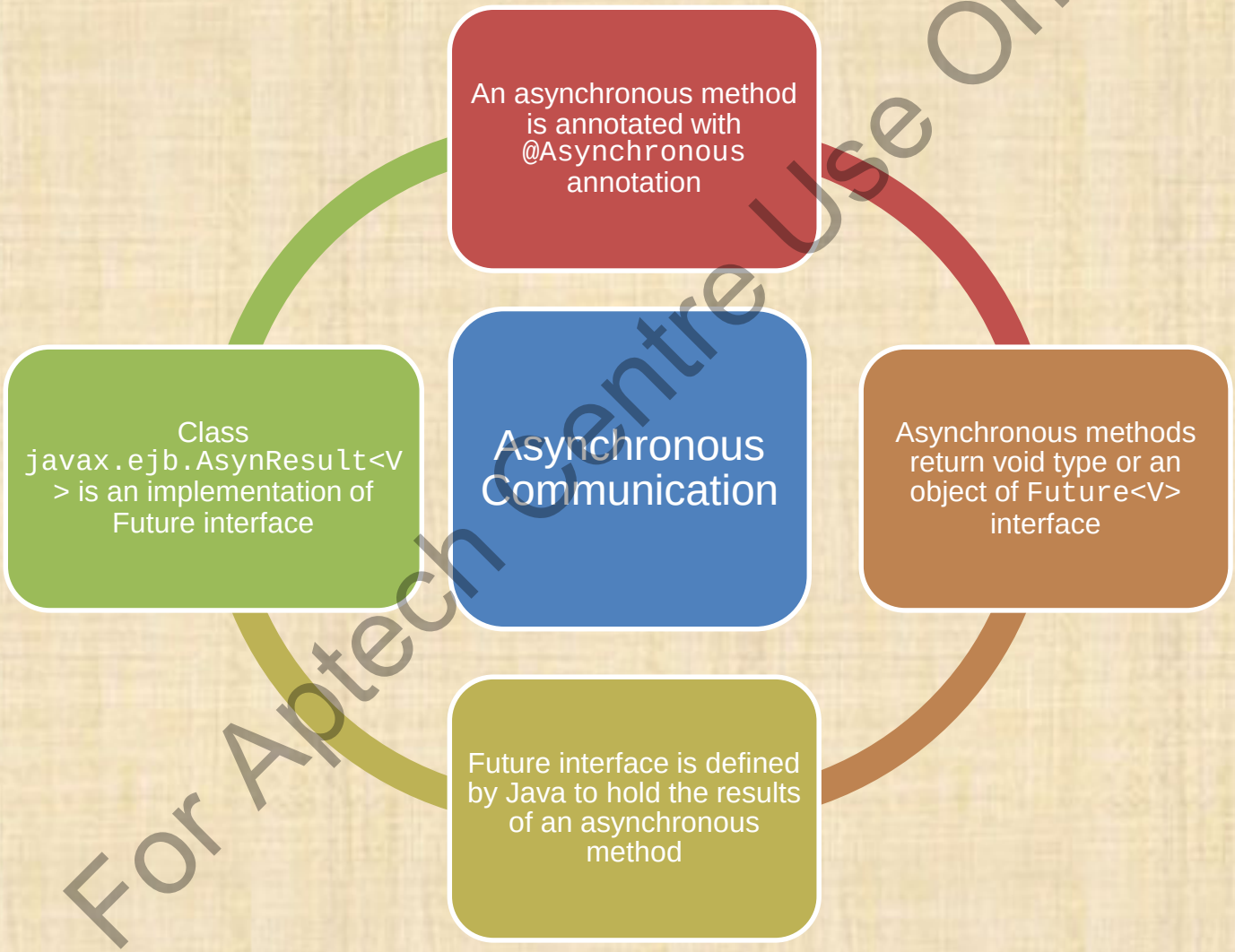
```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC
   '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN'
   'http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd'>
<ejb-jar>
   <enterprise-beans>
      <session>
         <ejb-name>AdditionBean</ejb-name>

         <ejb-class>Beans.AdditionBean</ejb-class>
         <session-type>Stateless</session-type>
         <transaction-type>Container</transaction-type>
      </session>
…..
</ejb-jar>
```


Communication with Session Beans

- Communication of the Session bean with client can be synchronous or asynchronous.

Synchronous communication	Asynchronous communication
• Client and bean are actively involved in the communication. • Reliable mode of communication.	• Client sends a request and does not wait for response. • Used in case of long running operations.

Creating Asynchronous Method in Session Beans 1-3



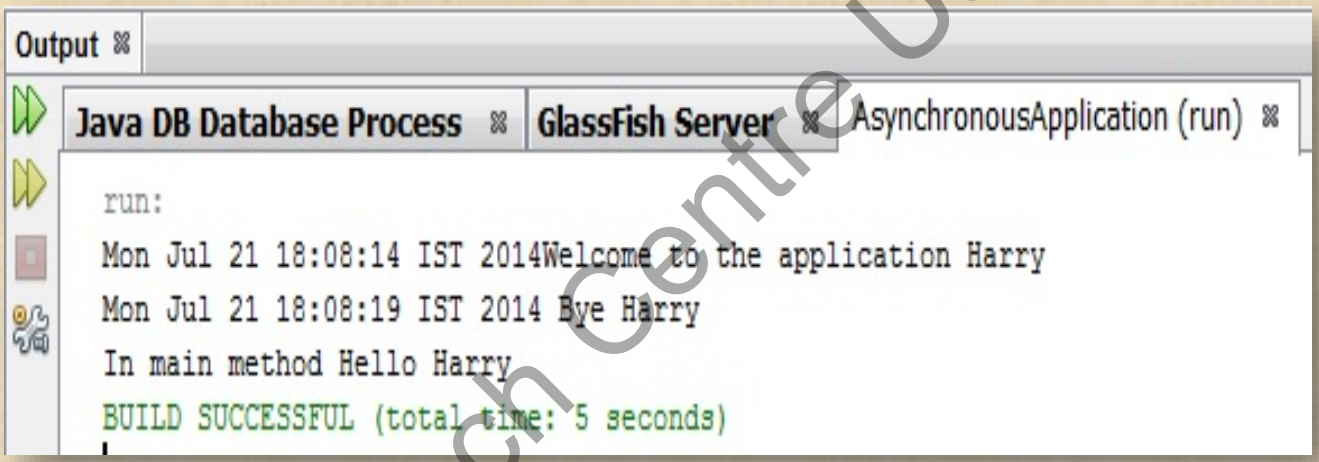
Creating Asynchronous Method in Session Beans 2-3

- Following code snippet demonstrates implementation of asynchronous methods:

```
package beans;
import java.util.Date;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Future;
import javax.ejb.AsyncResult;
import javax.ejb.Asynchronous;
import javax.ejb.Stateless;
@Stateless
public class AsyncMethods {
    @Asynchronous
    public Future<String> sayHello(String name) {
        System.out.println(new Date().toString()+"Welcome to the application "+name );
        try{
           Thread.sleep(5*1000);
        }catch (Exception e){
            e.printStackTrace();
        }
        System.out.println(new Date().toString()+" Bye "+name);
        return new AsyncResult<String>("Hello "+name);

    }
public static void main(String args[]) throws InterruptedException, ExecutionException{
    AsyncMethods AS = new AsyncMethods();
        Future<String> S = AS.sayHello("Harry");
        System.out.println("In main method "+ S.get());
}   }
```

Creating Asynchronous Method in Session Beans 3-3

- Following figure shows the execution of the asynchronous method:



Future Interface

get() – This method waits if the method is not complete, if the method completes it returns an object of result type V.

get(long TimeOut, TimeUnit unit)- This method waits for the result from the asynchronous method for the duration specified in the TimeOut parameter.

cancel() – This method attempts to cancel an asynchronous method.

isCancelled() – This method checks whether a method is cancelled or not and returns a boolean value.

isDone() – This method checks whether a method is complete or not and returns a boolean value.

Summary

- Stateless Session beans are used when the Session bean does not need to maintain the conversational state.
- Each Session bean is associated with one client.
- Container maintains a pool of Stateless Session beans.
- There are two stages in the lifecycle of the application – instantiated and removed.
- The container invokes the required callback methods for the Session bean.
- The enterprise application is deployed and configured through the deployment descriptor and annotations.
- ejb-jar.xml is the deployment descriptor for enterprise applications using EJB specification.
- Bean methods can be asynchronously invoked with the help of **`Future<V>`** interface.