11 : Comparative Study on EJB specifications

IT4206 – Enterprise Application Development

Level II - Semester 4

Overview

- This topic will discuss the life cycles of Stateful, Stateless and Singleton Session Beans and implement and deploy a simple Stateless Session Bean on WildFly server.

Intended Learning Outcomes

- At the end of this lesson, you will be able to;
  - Discuss the Stateless, Stateful, and Singleton Session beans.
  - Implement and Deploy Session Beans on WildFly server.
  - Explain the life cycles of Session Beans.

List of sub topics

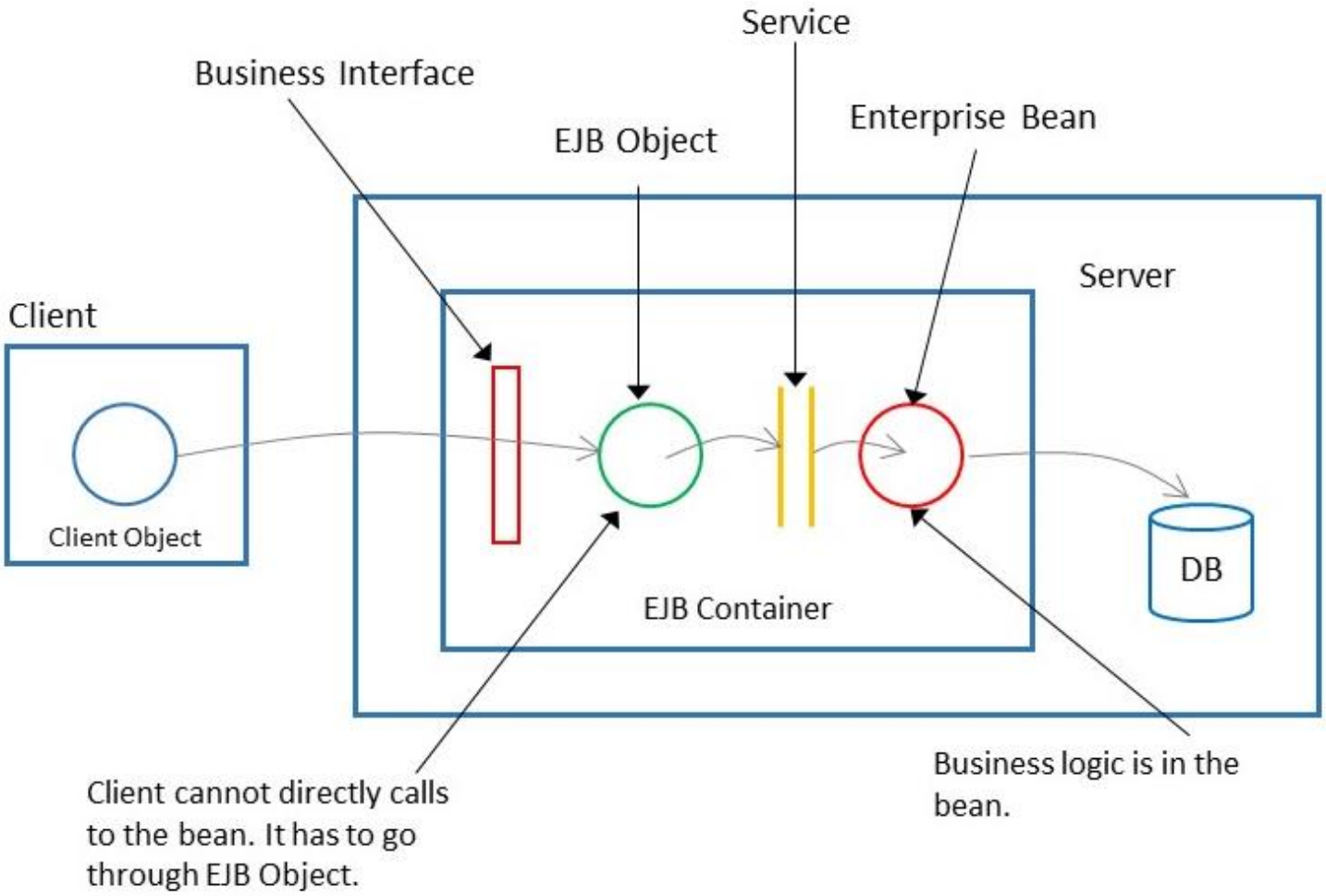
11.1 Session beans

11.2 Enterprise JavaBean life cycles

Remember EJB?

- By using enterprise JavaBeans, EJB Component can be developed.
- EJB Components can be assemble or reassemble into different applications.
  - Consider an Employee bean (component) as an example.
  - Employee bean represents an employee in the database.
  - It can be used in any application which need to represent an employee.
- With the component based development, the developers can use the components in their applications without even touching the code.

How EJB works



Remember types of Beans?

- Entity Beans
- Message driven beans
- Session beans
- Singleton session beans

Session Beans

- Session beans are have two types
  - Stateless
  - Stateful
- Stateful bean can remember the conversational state (client specific state) between method calls.
- Stateless want remember anything about the client between method invocations.
  - This is can be used for services that don't require continued conversation between the client and the service. Client can call the service again and again but the client can't depend on the bean remembering anything about the previous method call.

Create a Simple Session Bean in Eclipse (Stateless)

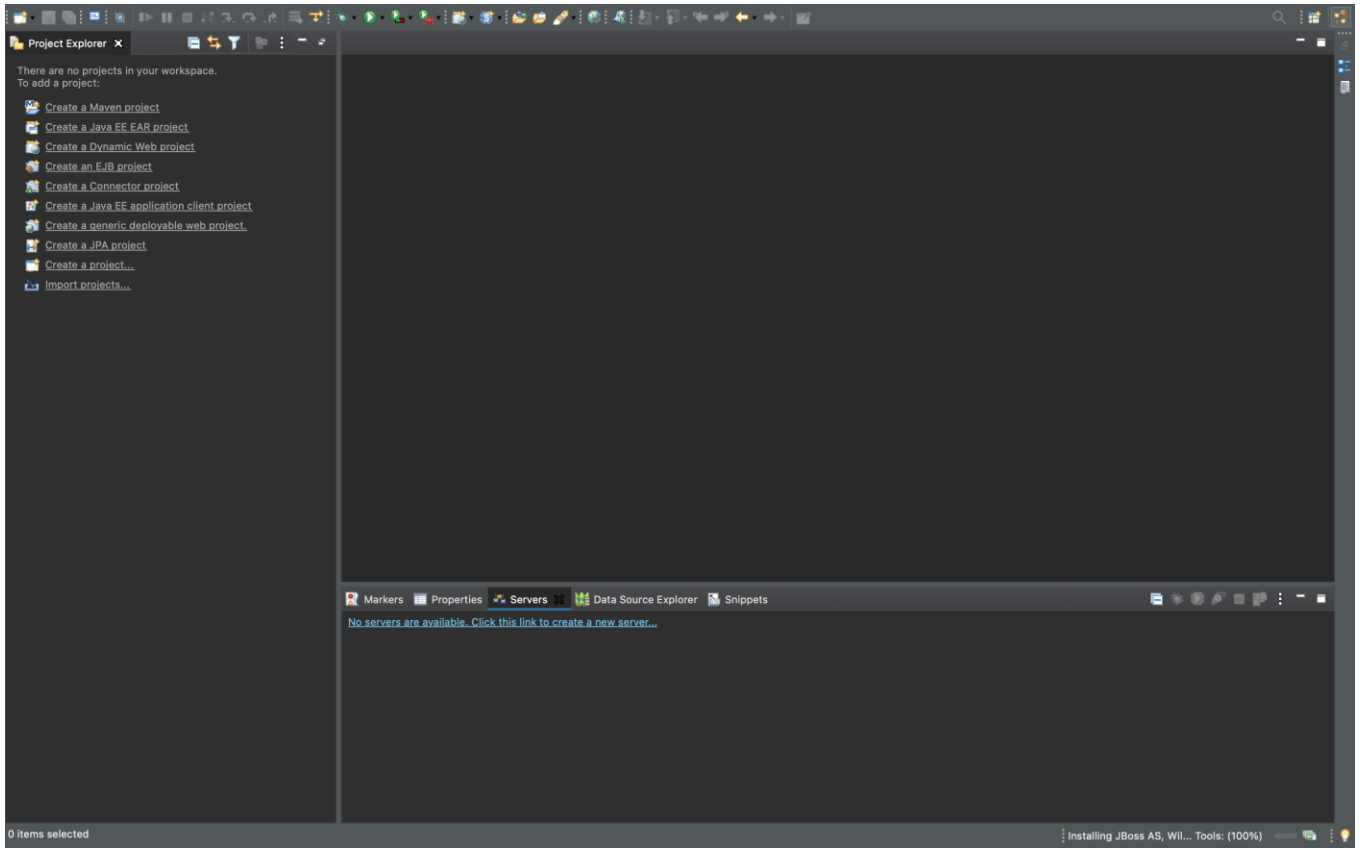
- Lets use some advanced tools:
  - Eclipse IDE
    - Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. ~Wikipedia
  - WildFly
    - WildFly, formerly known as JBoss AS, or simply JBoss, is an application server authored by JBoss, now developed by Red Hat. WildFly is written in Java and implements the Java Platform, Enterprise Edition specification. ~Wikipedia

https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers

https://www.wildfly.org/

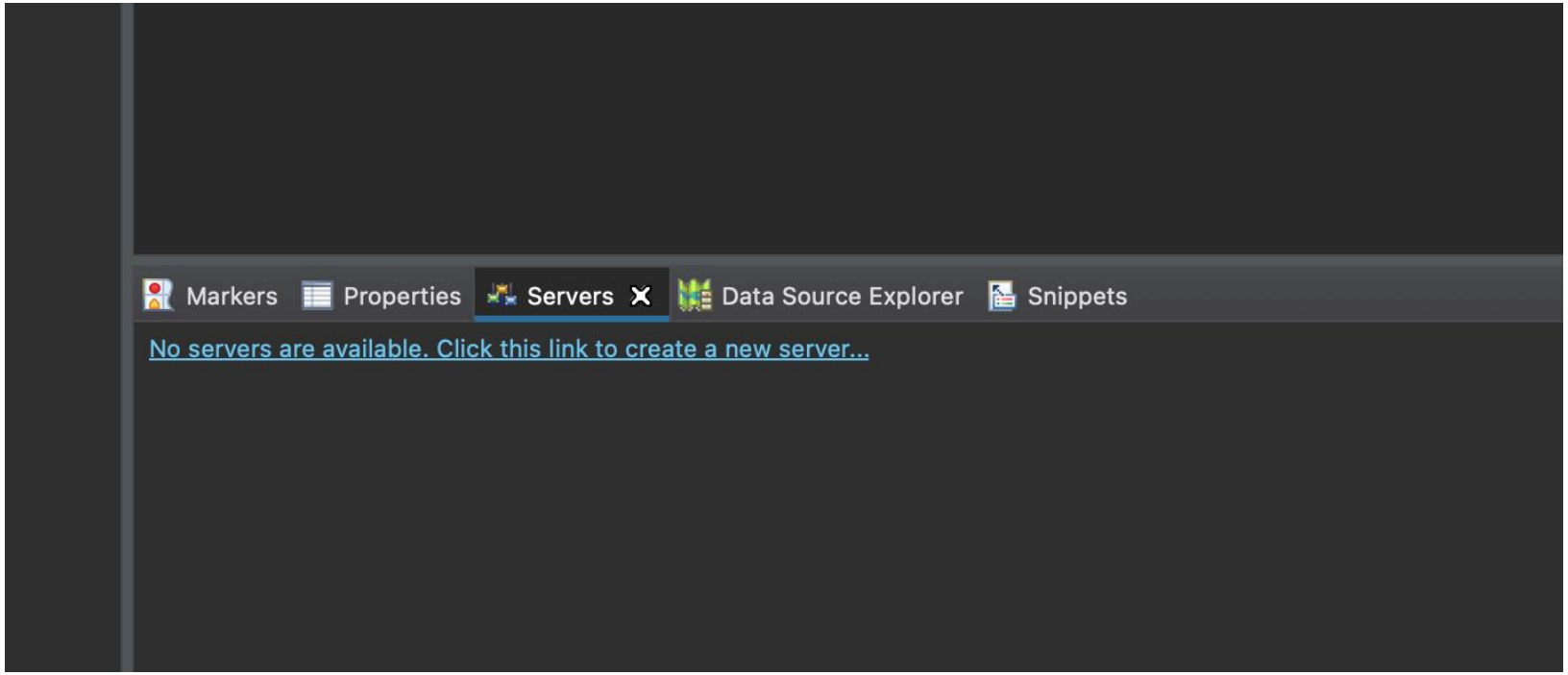
Eclipse

- Download and extract Eclipse IDE and WildFly sever.
- Start the Eclipse IDE.



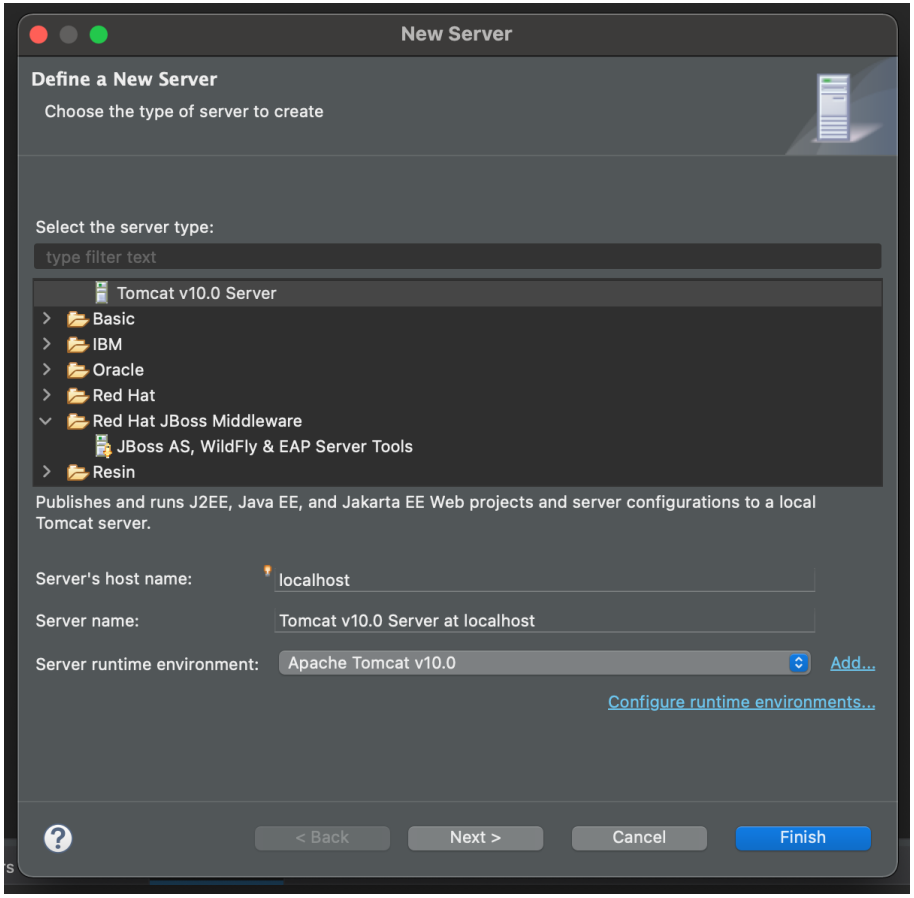
Setup WildFly with Eclipse

- Click the Servers tab and click the link to create a new server.



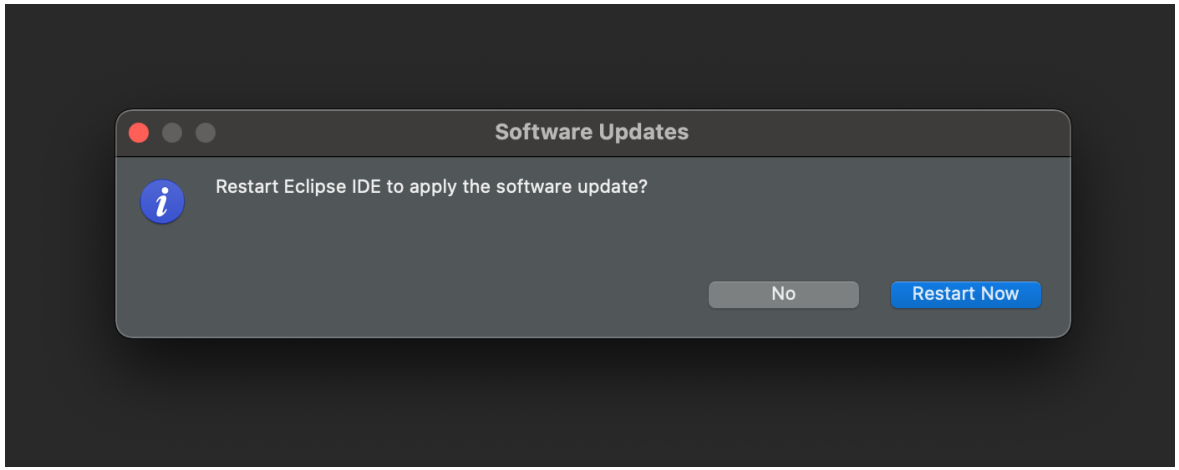
Setup WildFly with Eclipse

- Select JBoss AS, Wildfly… from Red Hat JBoss Middleware and click next.



Setup WildFly with Eclipse

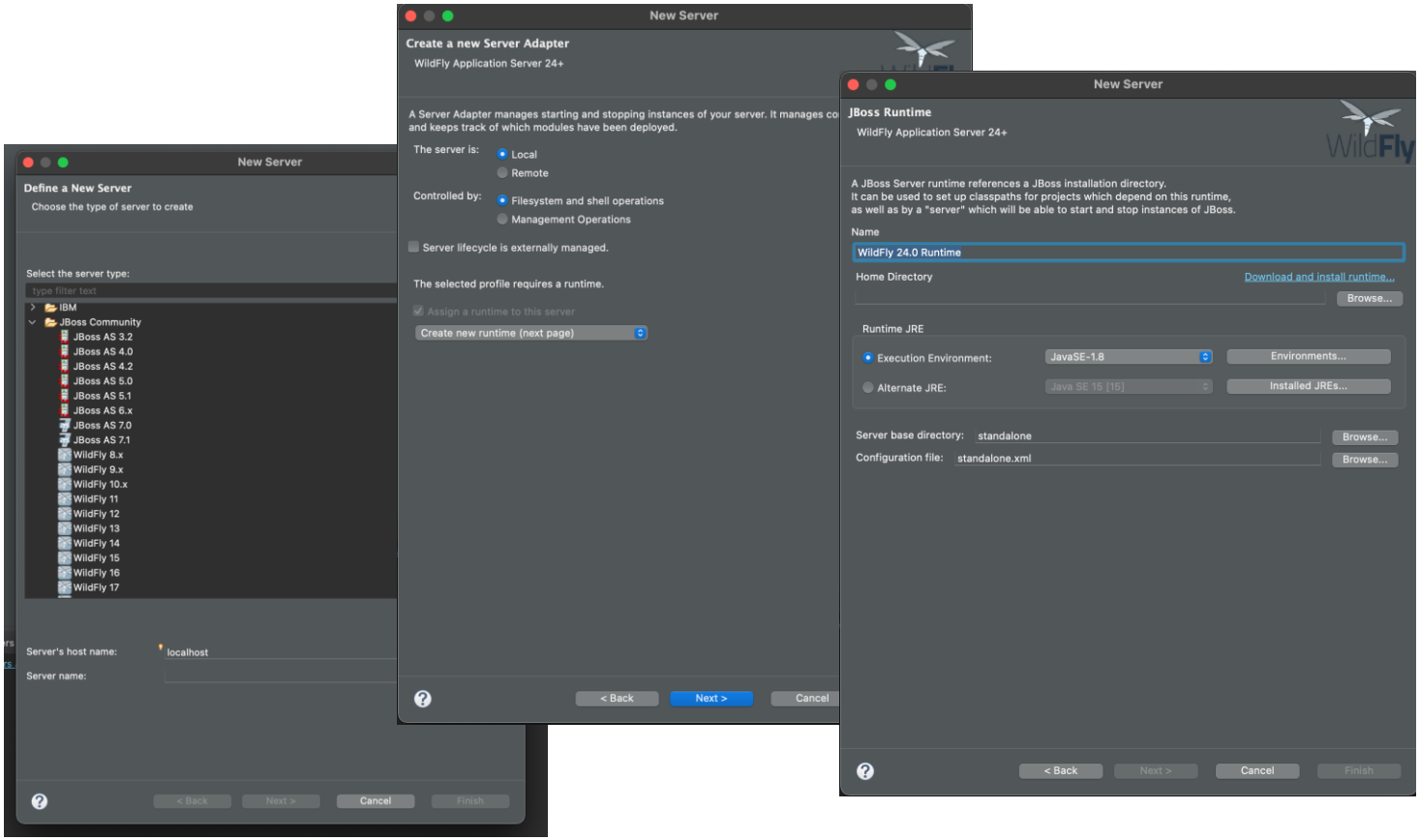
- Restart the Eclipse.



- After restarting, again go to the New server wizard to add WildFly server.

Setup WildFly with Eclipse

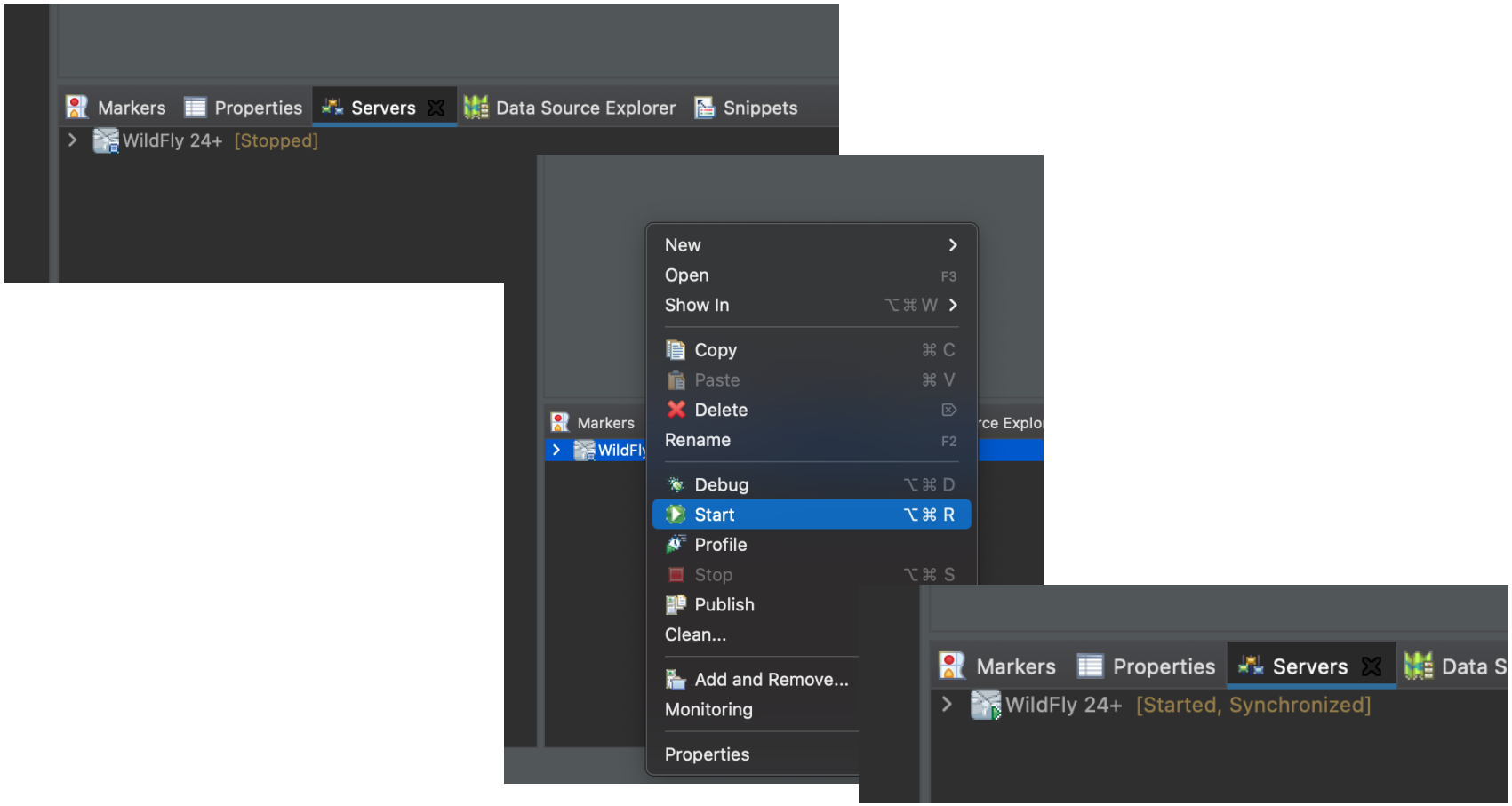
- Select the correct version of WildFly under the JBoss Community (Select a stable version)



*tested in WildFly 10.x

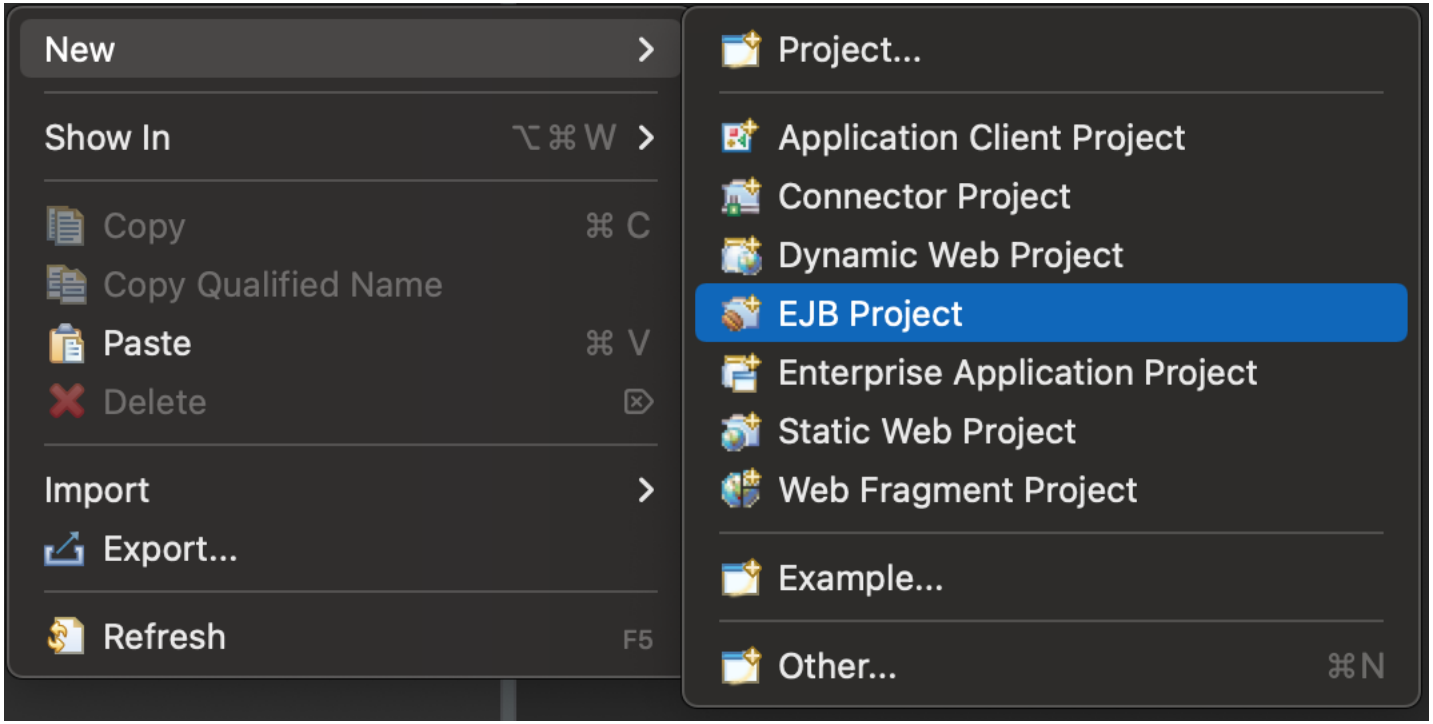
Setup WildFly with Eclipse

- Start the WildFly server



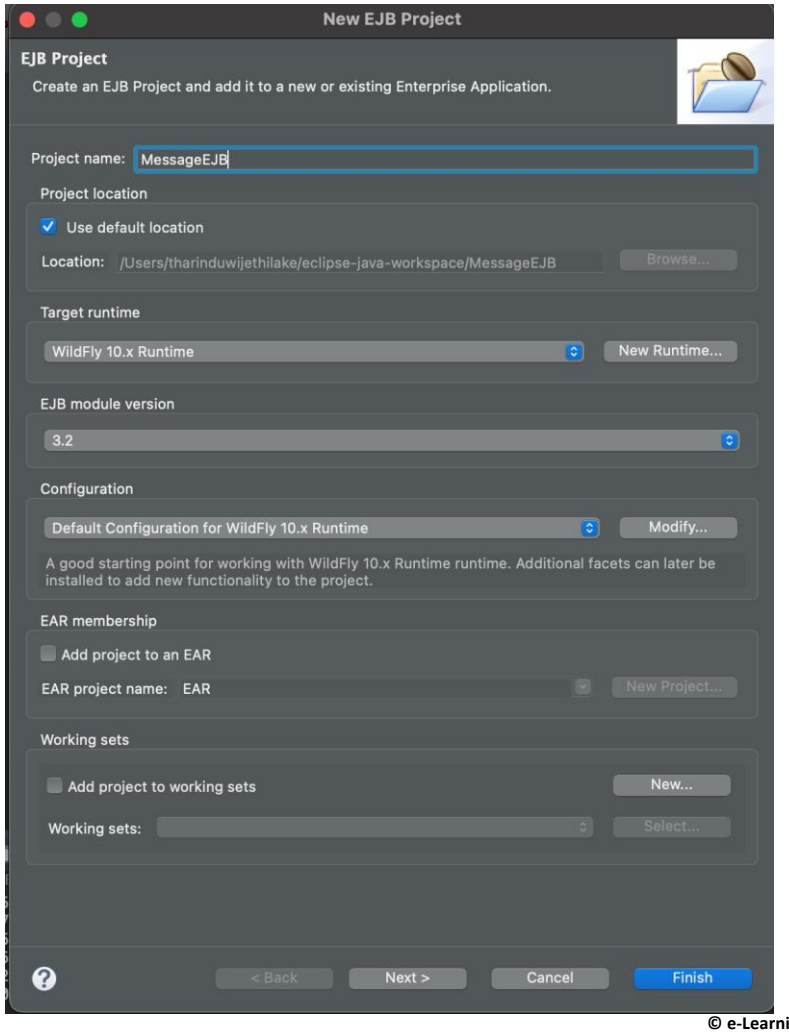
Create a Simple Session Bean in Eclipse (Stateless)

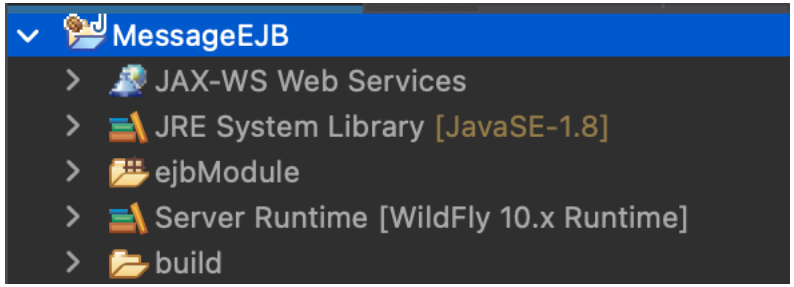
- Create a new EJB project



Create a Simple Session Bean in Eclipse (Stateless)

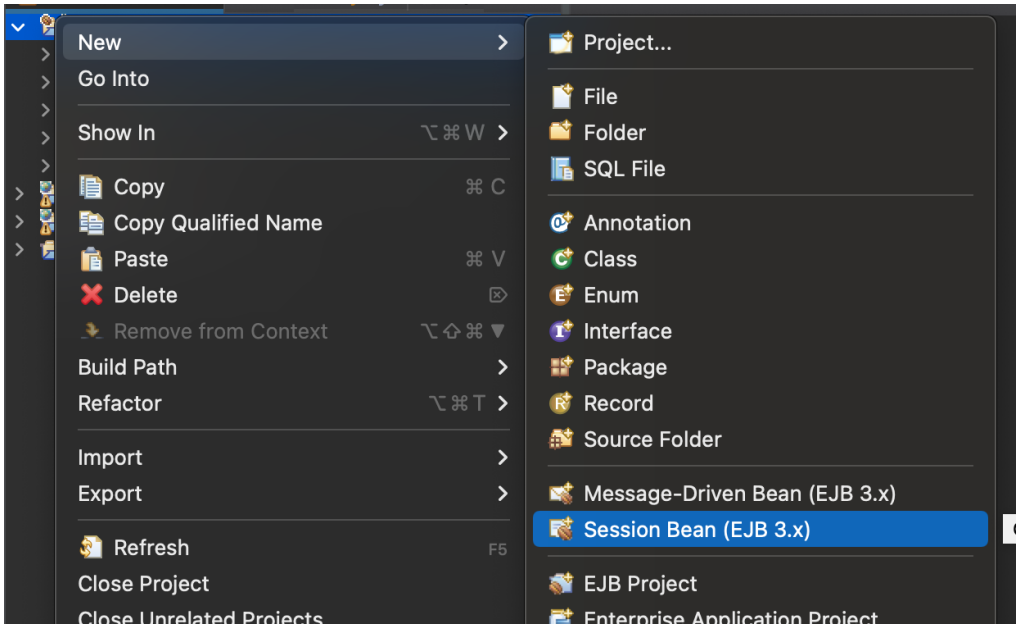
- Insert a Project name and click finish





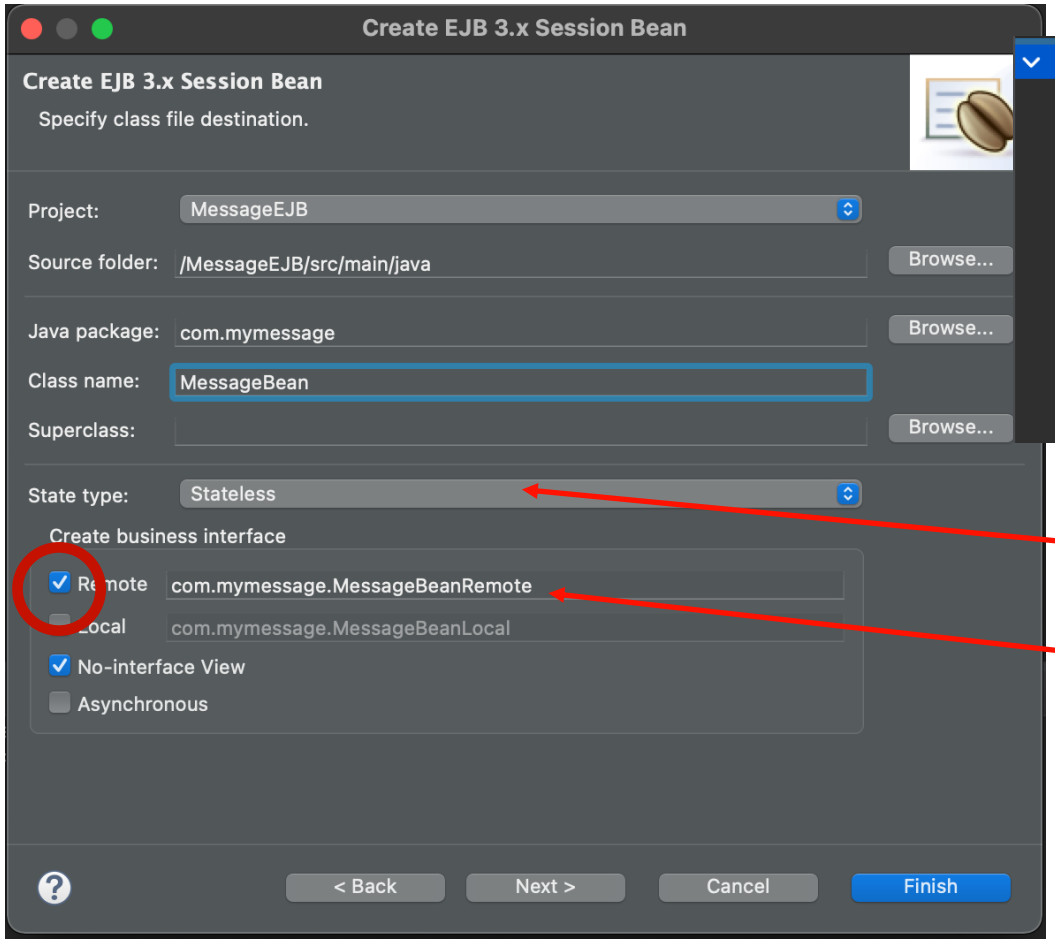
Create a Simple Session Bean in Eclipse (Stateless)

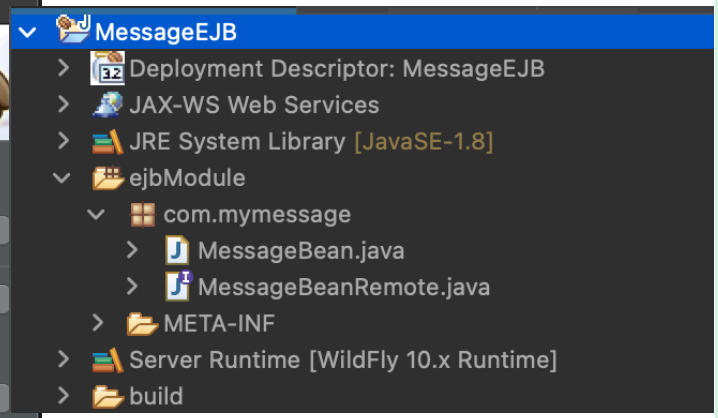
- Let's create a Session Bean.
- In this example we are going to create a stateless session bean.
- We will create the session bean, export it to a jar (EJB), and use it in another application using interfaces as discussed previously.



Create a Simple Session Bean in Eclipse (Stateless)

- Lets create a Session Bean (Tick the Remote, to create the interface)





Select the bean type, stateless, stateful, etc

Clients access a session bean through the bean's interfaces

Interfaces enable communication between the client and the bean. A bean can have local interfaces or remote interfaces. Local - Bean must be in the same VM. Remote - work for any client.

Create a Simple Session Bean in Eclipse (Stateless)

- Edit the MessageBeanRemote.java
- Lets create a method called “myMessage”
- In the interface only include the method signature and fields.

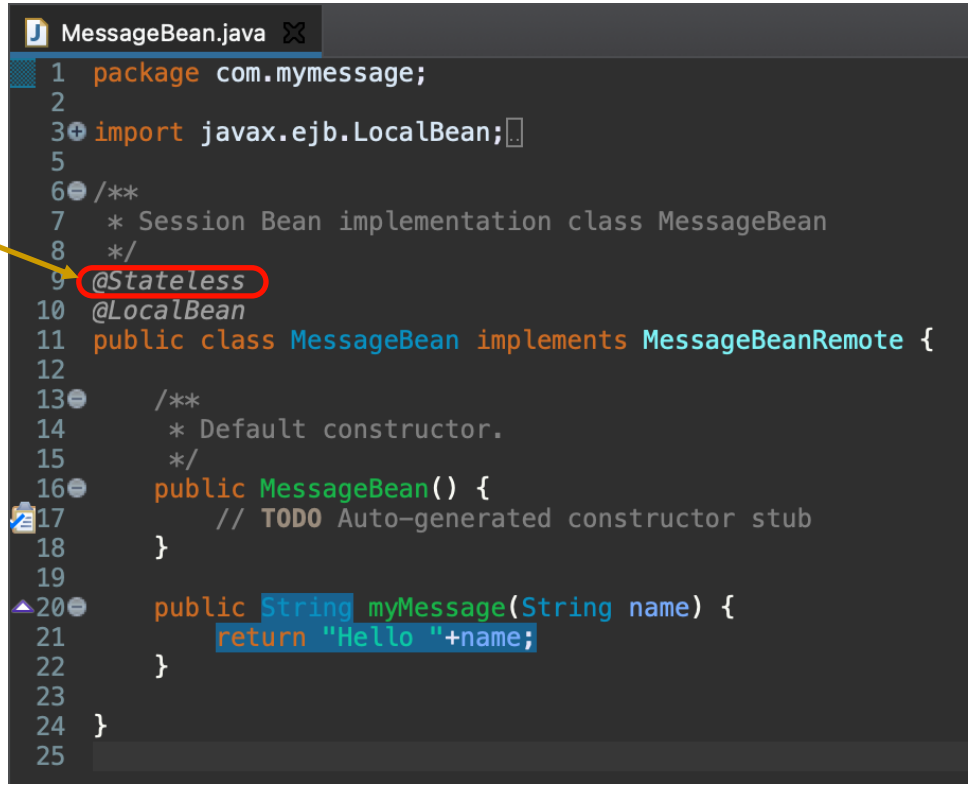
MessageBeanRemote.java

```
1  package com.mymessage;
2
3  import javax.ejb.Remote;
4
5  @Remote
6  public interface MessageBeanRemote {
7      public String myMessage(String name);
8  }
9
```

Create a Simple Session Bean in Eclipse (Stateless)

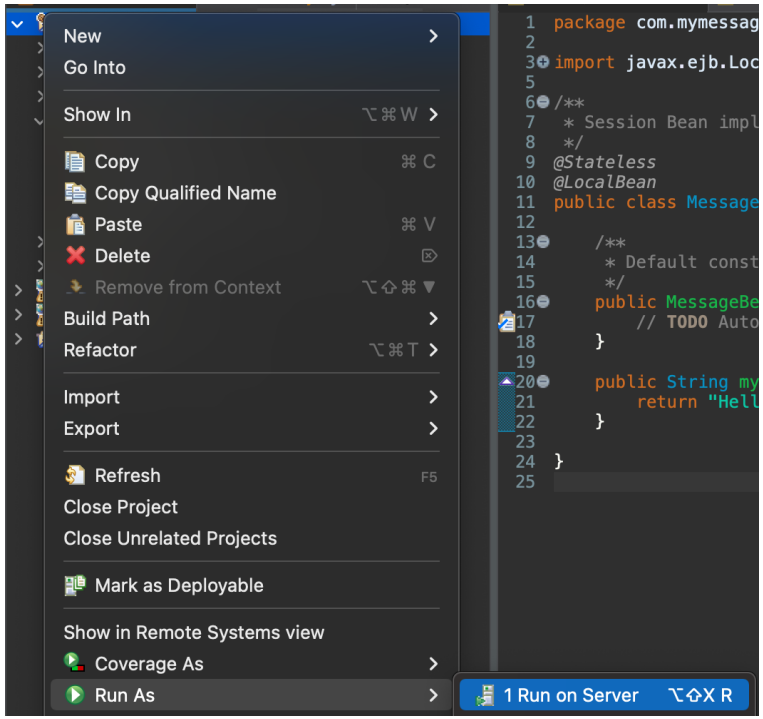
- Edit the MessageBean.java
- Implement the interface in the MessageBean class.

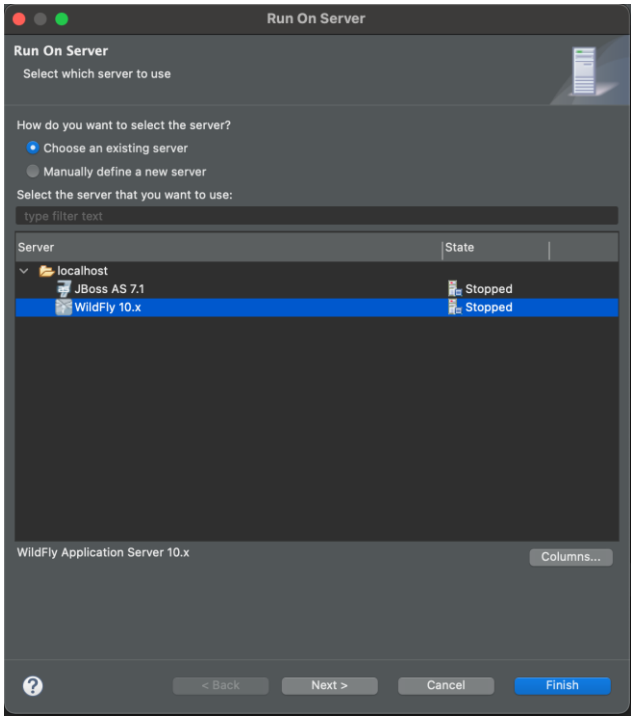
Annotation is used to decorate the bean as stateless session bean



Create a Simple Session Bean in Eclipse (Stateless)

- Deploy the Session bean

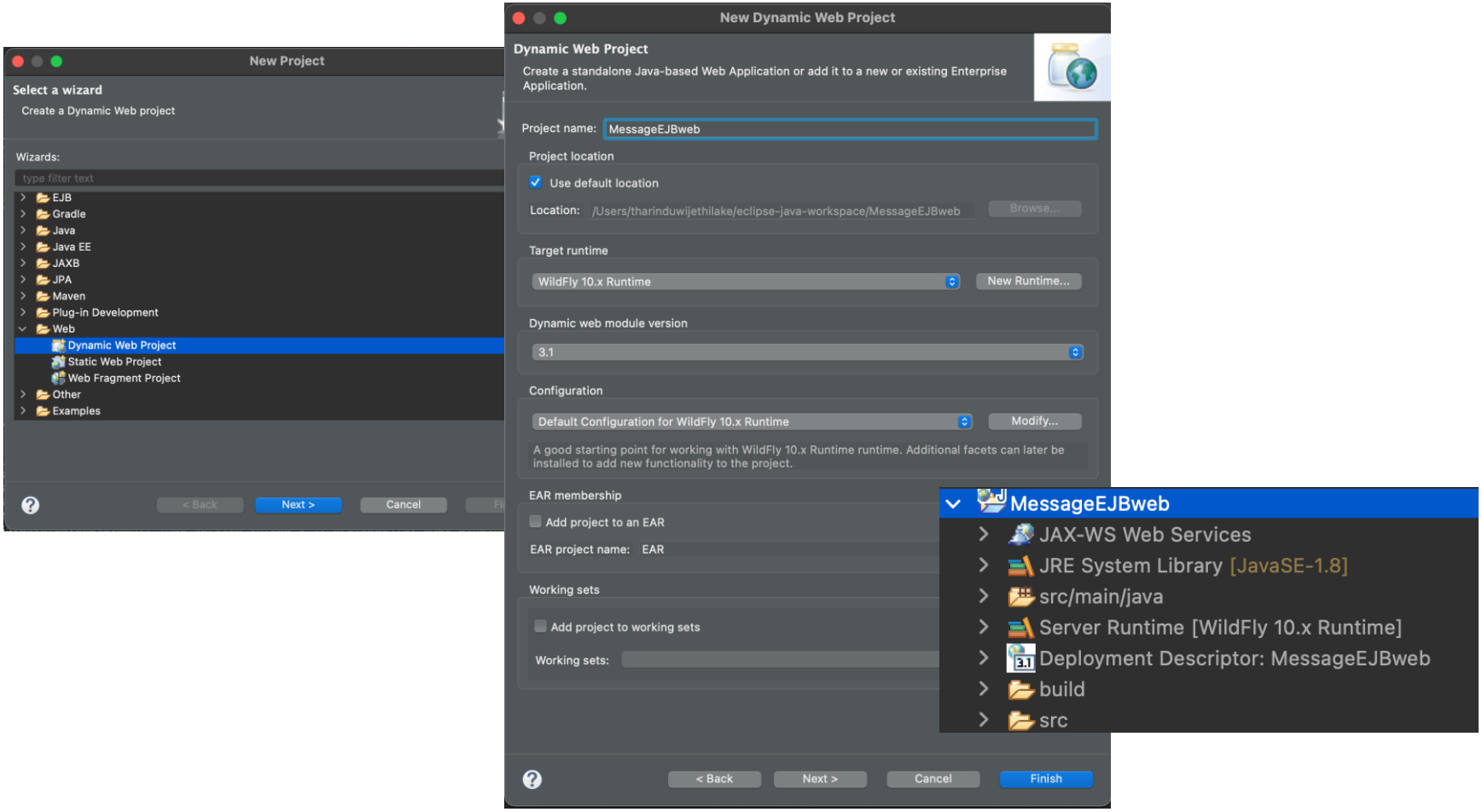





```
java:global/MessageEJB/MessageBean!com.mymessage.MessageBeanRemote
java:app/MessageEJB/MessageBean!com.mymessage.MessageBeanRemote
java:module/MessageBean!com.mymessage.MessageBeanRemote
java:jboss/exported/MessageEJB/MessageBean!com.mymessage.MessageBeanRemote
java:global/MessageEJB/MessageBean!com.mymessage.MessageBean
java:app/MessageEJB/MessageBean!com.mymessage.MessageBean
java:module/MessageBean!com.mymessage.MessageBean
```

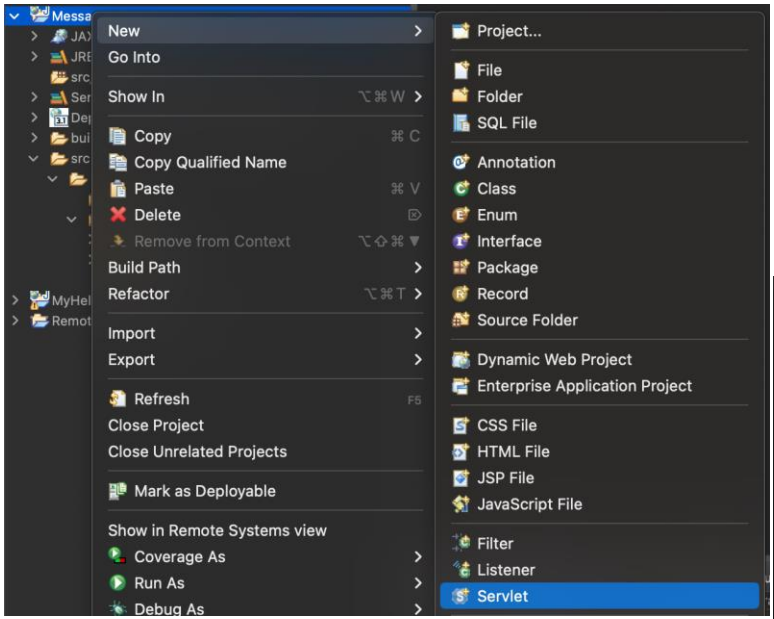
Create a Simple Session Bean in Eclipse (Stateless)

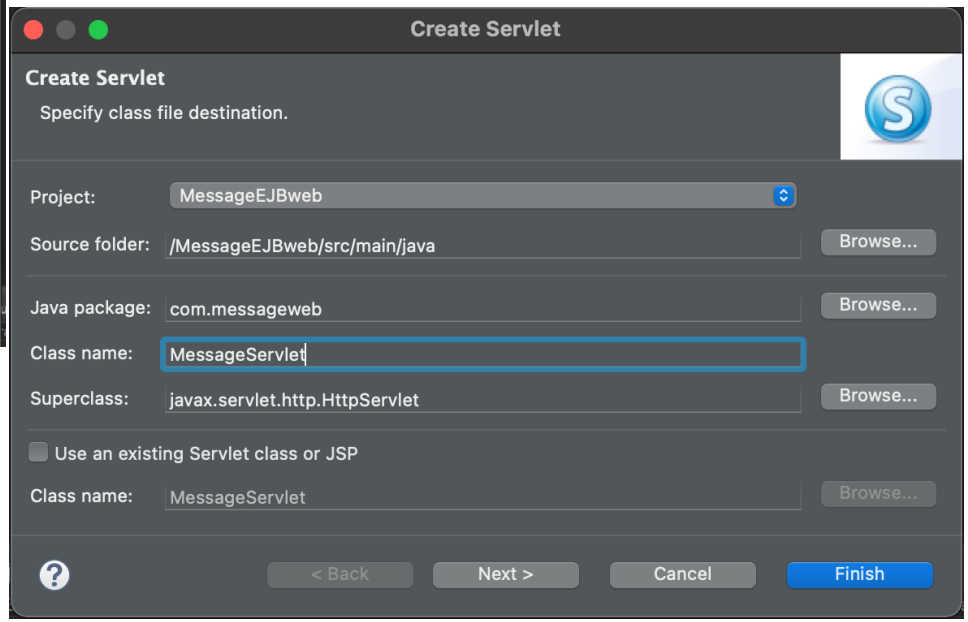
- Let's create the web app to invoke the session bean. (Separate project)



Create a Simple Session Bean in Eclipse (Stateless)

- Let's create a servlet to handle the request

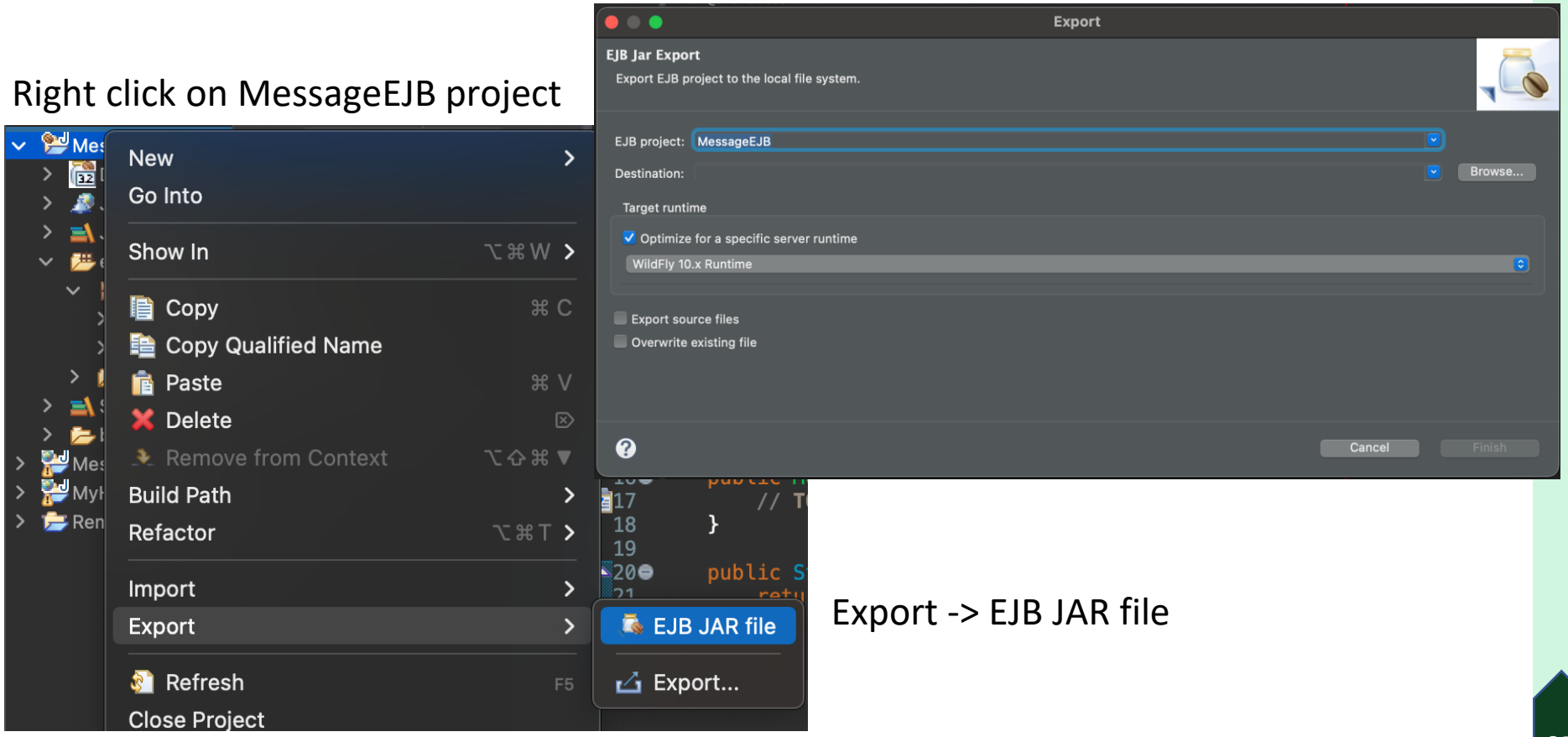




Create a Simple Session Bean in Eclipse (Stateless)

- Now we need to export the EJB from the MessageEJB project to a EJB JAR file and include it in the MessageEJBweb project.

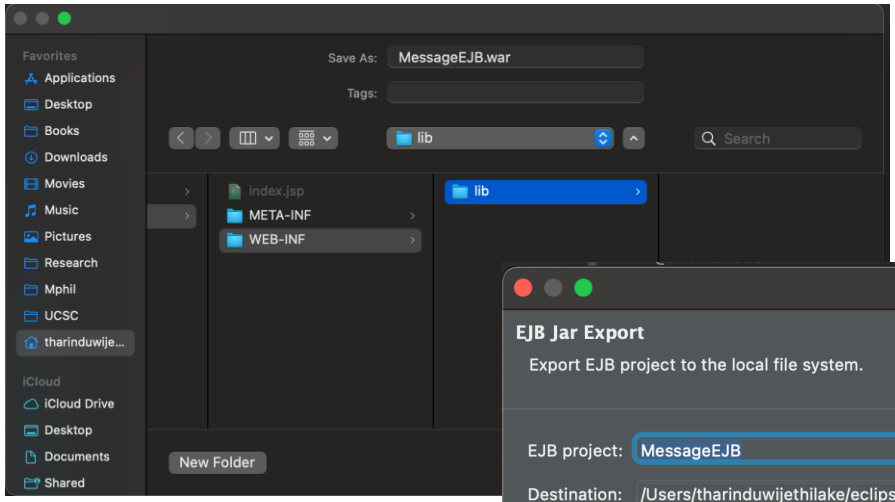
Right click on MessageEJB project



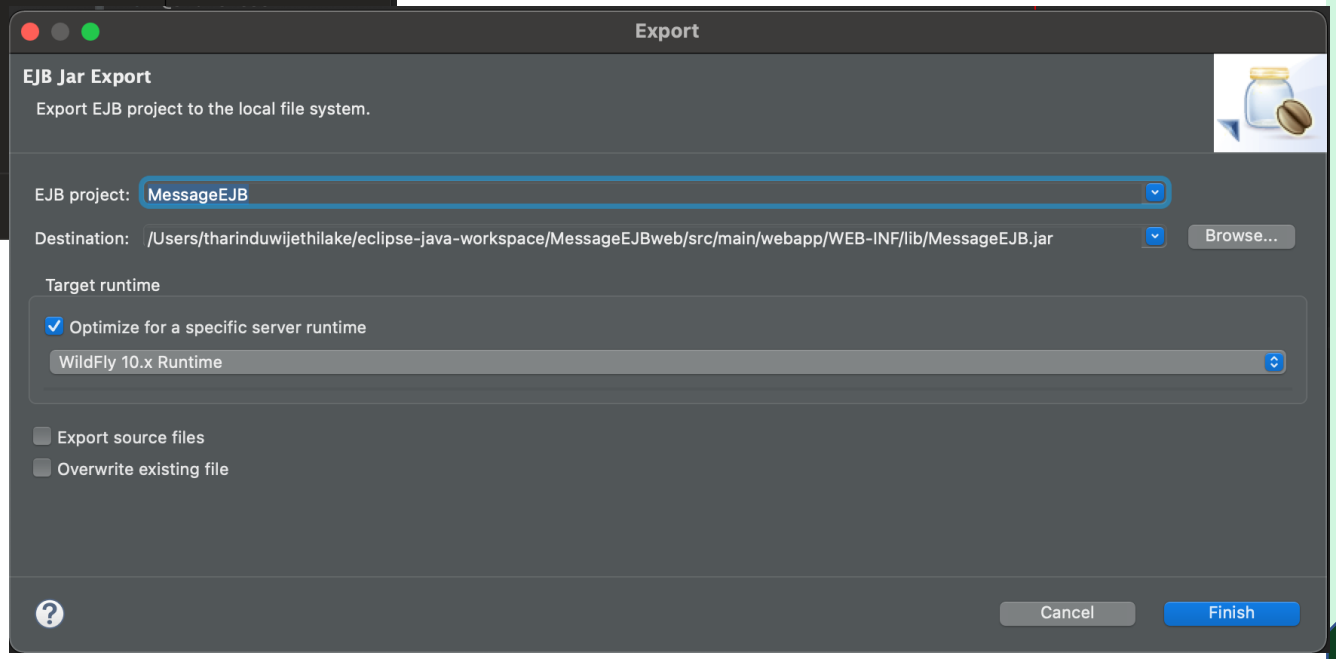
Export -> EJB JAR file

Create a Simple Session Bean in Eclipse (Stateless)

- Select the export location as /MessageEJBweb/src/main/webapp/WEB-INF/lib/

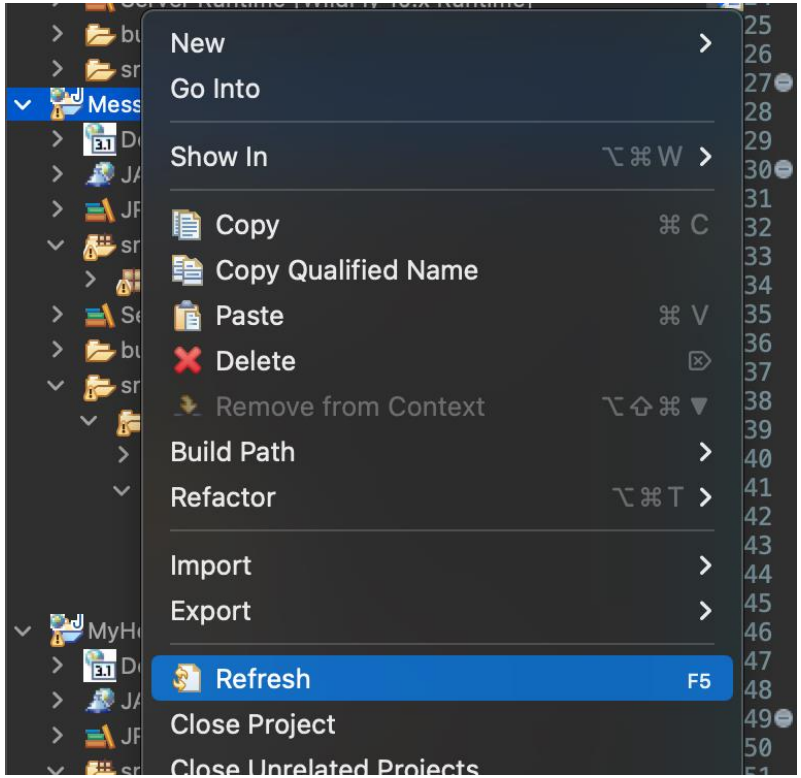


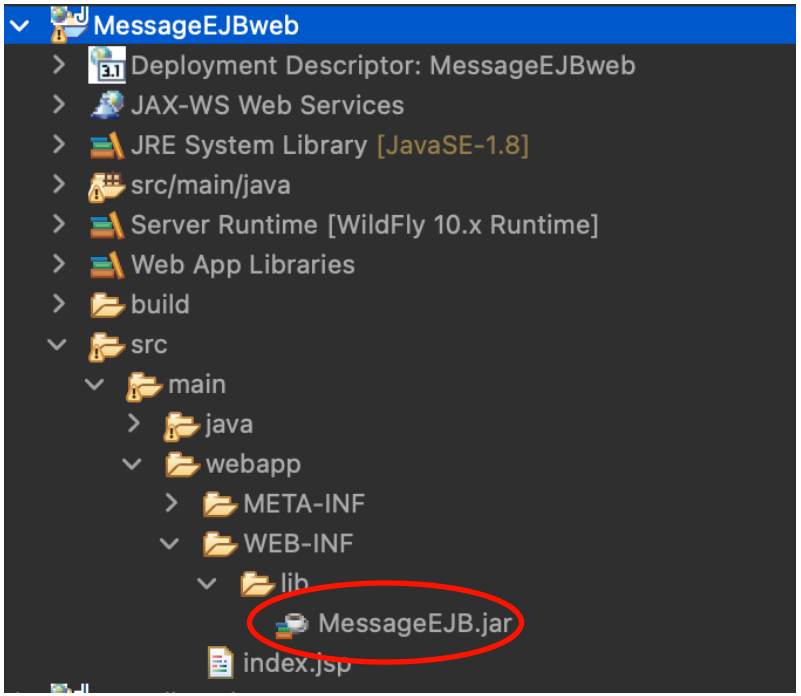
It is not compulsory to export to this location. You can export the JAR to any location and later move to the lib of the project you need to use it.



Create a Simple Session Bean in Eclipse (Stateless)

- Refresh the MessageEJBweb project





You can put any bean (jar) in the lib directory and use it in the application

Create a Simple Session Bean in Eclipse (Stateless)

- Lets edit the MessageServlet.java servlet file

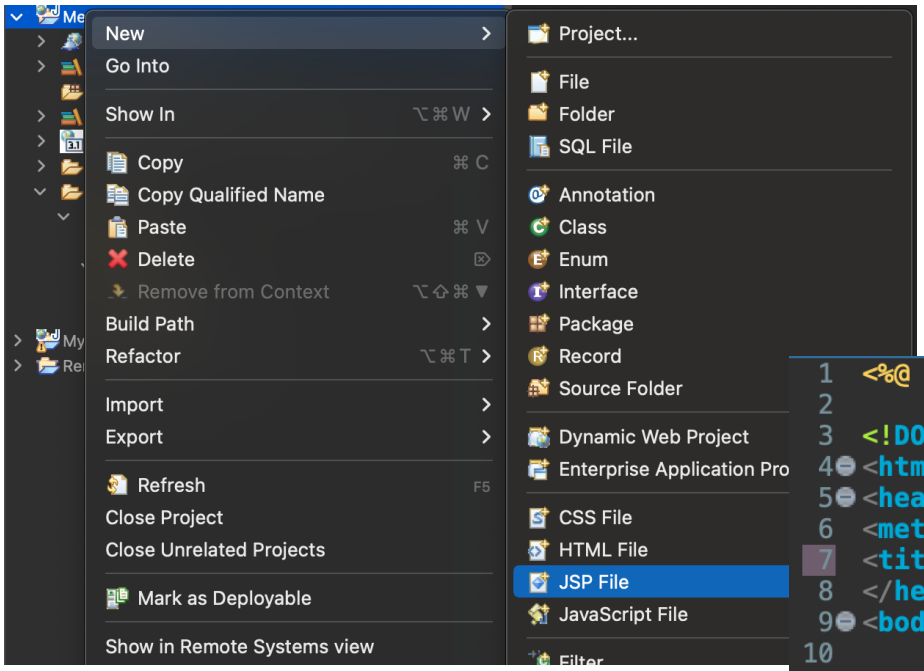
```
import javax.ejb.EJB;

/**
 * Servlet implementation class MessageServlet
 */
@WebServlet("/MessageServlet")
public class MessageServlet extends HttpServlet {

    @EJB
     private MessageBeanRemote MessageBean;


    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        String name=request.getParameter("name");
        String text=MessageBean.myMessage(name);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Message Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>" + text + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
```

Import javax.ejb.EJB package to use EJBs in the project

You can use the interface of EJB as MessageBean

Create a Simple Session Bean in Eclipse (Stateless)

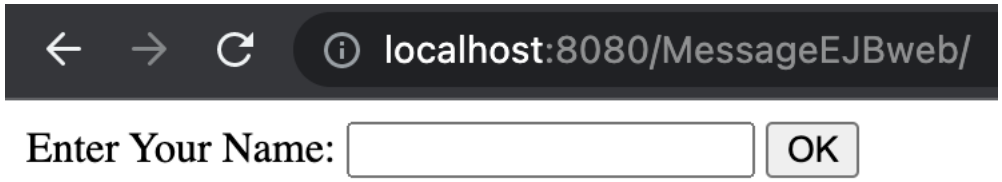
- Lets create a JSP file as index.jsp

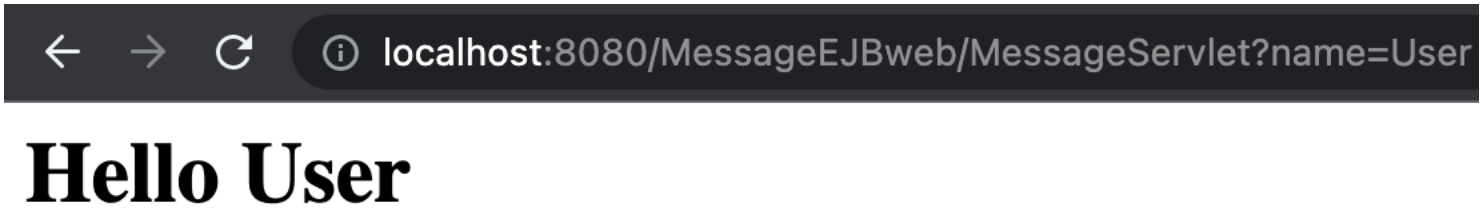



```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Message EJB</title>
</head>
<body>

<form  method="get" action="MessageServlet">
Enter Your Name: <input type="text" name="name">
<input type="submit" value="OK">
</form>

</body>
</html>
```

Create a Simple Session Bean in Eclipse (Stateless)

- Run the MessageEJBweb on Server

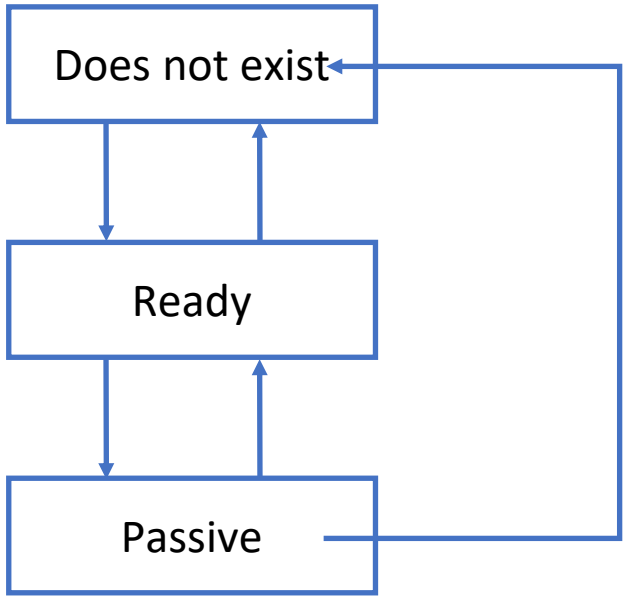




Life cycle of Stateful Session Bean

- Stateful session bean life cycle contains three states
  - Does Not Exist
    - Before a stateful session bean is deployed, it is in the Does Not Exist state
    - If an instance of a stateful session bean hasn't been accessed for a period of time, the EJB container will set the bean to the Does Not Exist state.
  - Ready
    - After successful deployment, the EJB container does any required dependency injection on the bean and it goes into the Ready state. Now the bean is ready to have its methods called by a client application.
  - Passive
    - EJB container may decide to move Bean from the main memory to secondary storage, when this happens the bean goes into the Passive state.

Life cycle of Stateful Session Bean

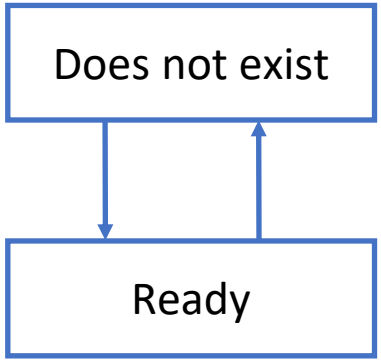


Life cycle of Stateful Session Bean

- Decorate any methods in stateful session beans with following annotations,
  - @PostActivate - invoked just after the stateful session bean has been activated
  - @PrePassivate - invoked just before the stateful session bean is passivated
  - @PreDestroy - invoke when a Bean in Ready state times out and is sent to the Does not Exist state
  - @Remove - if a method with @Romove executed, any methods decorated with the @PreDestroy annotation are executed and the bean is marked for garbage collection.

Life cycle of Stateless and Singleton Session Bean

- Stateless session bean life cycle contains two states
  - Does Not Exist
  - Ready



- Stateless and singleton session beans are never passivated

Life cycle of Stateless and Singleton Session Bean

- A stateless or singleton session bean's methods can be decorated with following annotations,
  - @PostConstruct - method will execute when the session bean goes from the Does Not Exist to the Ready State
  - @PreDestroy - method will execute when a stateless session bean goes from the Ready state to the Does Not Exist state