SE for Distributed Systems

Lab 5

Cairo University, Faculty of
Computers and Artificial Intelligence

Lab Objectives:

1. Practice on an RMI example
2. Practice on an EJBs example

Part 1

Design an application that will make bank account objects available to connecting clients, which may then manipulate these remote objects by invoking their methods. Only four account objects will be created and should be accessible through one bank object. Implement your application using Java RMI.

Part 2

What is an Enterprise Java Beans

Simply put, an Enterprise Java Bean is a Java class with one or more annotations from the EJB spec which grant the class special powers when running inside of an EJB container. In the following sections, we'll discuss what these powers are and how to leverage them in your programs.

Types of Enterprise Beans

Let's now go a bit deeper into the specifics of Enterprise beans:

- Session Beans
- Message Driven Beans

Session Beans

A session bean encapsulates business logic that can be invoked programmatically by a client. The invocation can be done locally by another class in the same JVM or remotely over the network from another JVM. The bean performs the task for the client, abstracting its complexity similar to a web service, for example.

The lifecycle of a session bean instance is, naturally, managed by the EJB container. Depending on how they're managed, sessions beans can be in either of the following states:

Cairo University, Faculty of
Computers and Artificial Intelligence

- Stateless
- Stateful
- Singleton

As the name suggests, **Stateless beans** don't have any state. As such, they are shared by multiple clients. They can be singletons but in most implementations, containers create an instance pool of stateless EJB. And, since there is no state to maintain, they're fast and easily managed by the container.

As a downside, owing to the shared nature of the bean, developers are responsible to ensure that they are thread safe.

Stateful beans are unique to each client, they represent a client's state. Because the client interacts ("talks") with its bean, this state is often called the conversational state. Just like stateless beans, instance lifecycle is managed by the container; they're also destroyed when the client terminates.

A **Singleton** session bean is instantiated once per application and exists for the lifecycle of the application. Singleton session beans are designed for circumstances in which state must be shared across all clients. Similar to Stateless beans, developers must ensure that singletons thread safe. However, concurrency control is different between these different types of beans, as we'll discuss further on.

Now, let's get practical and write some code. Here, we're going to create a Maven project with a packaging type of ejb, with a dependency on javaee-api:

```
<project ...>

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.stackify</groupId>

    <artifactId>ejb-demo</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>ejb</packaging>

    <dependencies>

        <dependency>

            <groupId>javax</groupId>
```

Cairo University, Faculty of
Computers and Artificial Intelligence

```
            <artifactId>javaee-api</artifactId>

            <version>8.0</version>

        </dependency>

    </dependencies>

</project>
```

SE for Distributed Systems

Lab 5

Cairo University, Faculty of
Computers and Artificial Intelligence

Modern-day EJB is easy to configure, hence writing an EJB class is just a matter of adding annotations i.e., @Stateless, @Stateful or @Singleton. These annotations come from the javax.ejb package:

```
@Stateless

public class TestStatelessEjb {

    public String sayHello(String name) {

        return "Hello, " + name + "!";

    }

}
```

Or:

```
@Stateful

public class TestStatefulEjb {

}
```

Finally:

```
@Singleton

public class TestSingletonEjb {

}
```

SE for Distributed Systems

Lab 5

Cairo University, Faculty of
Computers and Artificial Intelligence

Accessing Enterprise Beans

To invoke the methods of an EJB locally, the bean can be injected in any managed class running in the container – say a Servlet:

```
public class TestServlet extends HttpServlet {

    @EJB

    TestStatelessEjb testStatelessEjb;

    public void doGet(HttpServletRequest request,

      HttpServletResponse response) {

        testStatelessEjb.sayHello("Stackify Reader");

    }

}
```

Task:

Implement using EJBs a client-server architecture, where the client sends a string for example (*:2:5) and the server should split on regex ":", so the server should receive *, 2 and 5. The server should, then, do the operation sent by the client to the two numbers, for this case (2*5) and send 10 as a result.

References:

1. https://stackify.com/enterprise-java-beans/
2. https://www.javatpoint.com/stateless-session-bean