

**Tema**

ENTERPRISE JAVA BEANS

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

Fecha

15/07/2023

ÍNDICE

[1. OBJETIVO 4](#_Toc125444631)

[2. VIDEO TUTORIAL 4](#_Toc125444632)

[3. MARCO TEÓRICO 8](#_Toc125444633)

[3.1 ENTERPRISE JAVA BEANS. 8](#_Toc125444634)

[3.2 ENTERPRISE JAVA BEANS Y EL SOFTWARE MODULAR. 9](#_Toc125444635)

[3.3 SESSION BEAN, FORMA DE CREARLOS Y UTILIZARLOS EN JAVA. 10](#_Toc125444636)

[3.4 STATELESS SESSION 11](#_Toc125444637)

[3.5 STATEFUL SESSION 12](#_Toc125444638)

[3.6 SINGLETON SESSION 13](#_Toc125444639)

[3.7 JAVA PERSISTENCE API (JPA) 13](#_Toc125444640)

[3.8 JAVA WEB START 14](#_Toc125444641)

[3.9 REMOTE PROCEDURE CALL (RPC) 14](#_Toc125444642)

[3.10 REMOTE METHOD INVOCATION (RMI) 15](#_Toc125444643)

[3.11 RMI REGISTRY 16](#_Toc125444644)

[3.12 TÉRMINOS 16](#_Toc125444645)

[4. DESARROLLO 17](#_Toc125444646)

[4.1 CREACIÓN DEL ARCHIVO ENTERPRISE APPLICATION 17](#_Toc125444647)

[4.2 CREACIÓN DEL ARCHIVO DE LIBRERÍA DE CLASES 19](#_Toc125444648)

[4.3 CREACIÓN DEL ARCHIVO DEL CLIENTE 20](#_Toc125444649)

[4.4 CREACIÓN DEL SESSION BEAN 26](#_Toc125444650)

[5. EJECUCIÓN DEL PROYECTO 31](#_Toc125444651)

[6. CONCLUSIONES 31](#_Toc125444652)

[7. RECOMENDACIONES 32](#_Toc125444653)

[8. BIBLIOGRAFÍA 32](#_Toc125444654)

**ÍNDICE DE IMÁGENES**

[Figura 1. Componentes de Enterprise Java Beans 4](#_Toc125444694)

[Figura 2. Estructura de Enterprise Java Beans 5](#_Toc125444695)

[Figura 3. Tipos de Session Beans 6](#_Toc125444696)

[Figura 4. Componentes de Session Bean 7](#_Toc125444697)

[Figura 5. Pagina Principal de Java Web Start 7](#_Toc125444698)

[Figura 6. Estructura EJB 9](#_Toc125444699)

[Figura 7. Ejemplo formato Session Bean 11](#_Toc125444700)

[Figura 8. Ciclo de vida Stateless Session 12](#_Toc125444701)

[Figura 9. Estructura JPA 14](#_Toc125444702)

[Figura 10. Logo java web start 14](#_Toc125444703)

[Figura 11. Estructura de proceso RMI 15](#_Toc125444704)

[Figura 12. Logotipo identificativo de Netbeans. 17](#_Toc125444705)

[Figura 13. Selección del tipo de aplicación a desarrollar 17](#_Toc125444706)

[Figura 14. Selección del nombre y directorio donde guardar el proyecto. 18](#_Toc125444707)

[Figura 15. Selección del Java EE y creación del EJB Module. 18](#_Toc125444708)

[Figura 16. Estructura del Proyecto 19](#_Toc125444709)

[Figura 17. Selección del proyecto Java Class Library 19](#_Toc125444710)

[Figura 18. Selección del proyecto Java Class Library 19](#_Toc125444711)

[Figura 19. Estructura del proyecto 20](#_Toc125444712)

[Figura 20. Estructura del Proyecto 20](#_Toc125444713)

[Figura 21. Estructura del Proyecto 21](#_Toc125444714)

[Figura 22. Selección del Enterprise Application 21](#_Toc125444715)

[Figura 23. Estructura del proyecto 21](#_Toc125444716)

[Figura 24. Agregación de la libreria 22](#_Toc125444717)

[Figura 25. Selección de la libreria 22](#_Toc125444718)

[Figura 26. Estructura de la librería 23](#_Toc125444719)

[Figura 27. Selección del archivo Main 23](#_Toc125444720)

[Figura 28. Selección del archivo Repetir 24](#_Toc125444721)

[Figura 29. Código del mensaje del archivo Main. 24](#_Toc125444722)

[Figura 30. Creación del Session Bean 26](#_Toc125444723)

[Figura 31. Selección del tipo de archivo Session Bean 26](#_Toc125444724)

[Figura 32. Selección del proyecto Java Class Library 27](#_Toc125444725)

[Figura 33. Estructura de los archivos 27](#_Toc125444726)

[Figura 34. Agregación de Business Method 28](#_Toc125444727)

[Figura 35. Agregación de parámetros 28](#_Toc125444728)

[Figura 36. Estructura del Archivo Repetir 29](#_Toc125444729)

[Figura 37. Ejecución del proyecto. 31](#_Toc125444730)

[Figura 38. Ventana del mensaje 31](#_Toc125444731)

**ÍNDICE DE TABLAS**

[Tabla 1. Términos 16](#_Toc125444732)

[Tabla 2. Codificación archivo Main 25](file:///C:\1erParcial_Morales_B\03.%20Sockets\ENTERPRISE_JAVABEANS_GRUPO6\DOCUMENTACIÓN\ENTERPRISE%20JAVA%20BEANS%20_GRUPO6.docx#_Toc125444733)

[Tabla 3. Codificación del archivo Repetir 30](file:///C:\1erParcial_Morales_B\03.%20Sockets\ENTERPRISE_JAVABEANS_GRUPO6\DOCUMENTACIÓN\ENTERPRISE%20JAVA%20BEANS%20_GRUPO6.docx#_Toc125444734)

# OBJETIVO

El objetivo de la presente práctica de laboratorio es utilizar las funcionalidades de Enterprise Java Beans que nos ayudara definir la renderización de elementos de páginas web desde el servidor y hacerlo de forma más ágil.

# VIDEO TUTORIAL

Vídeo tutorial número 13 del curso de Java enterprise edition en el que se centra en un tema clave los Enterprise Java Beans que desde java enterprise edition 6 y la aparición de la JPA la Api de persistencia de java que ya conocemos, que prácticamente se limitan a la sesión beans, los beans de sesión importantísimos para entender cómo una aplicación empresarial java negocia con el servidor la principal ventaja que presenta java enterprise edition desde sus orígenes es su construcción sobre componentes de software modulares. En definitiva, es como un director cuya orquesta en lugar de por instrumentos está formada por Apis y java Enterprise Edition es la encargada de definir como coordinarlas. Estas aves pueden ser comunes y compartidas con Java Standard Edition o con otras plataformas de programación, aunque tienen que cumplir con ciertos requisitos de conformidad para que se consideren conformes a Java Standard Edition, que por eso se la considera aunque formalmente como un estándar. Pero otras de estas Apis que forman parte de la orquesta están especificadas como componentes únicos para java enterprise edition y una de ellas eso justo de la que vamos a hablar de EJB de Enterprise Java Beans.

Imagen que contiene Diagrama

Descripción generada automáticamente

Figura . Componentes de Enterprise Java Beans

La especificación de Enterprise Java beans es la que nos detalla cómo los servidores de una aplicación proveen los objetos que son precisamente los que EJB, como decíamos hace un momento desde la versión 6 de Java Enterprise Edition, con la aparición de la JPA (Java Persistense Api, esto Enterprise Java Beans quedan prácticamente limitados a los Sesión Beans que será a los que dedicaremos este vídeo tutorial.

Diagrama

Descripción generada automáticamente

Figura . Estructura de Enterprise Java Beans

Los Beans de sesión dentro del funcionamiento del software modular de Java Enterprise Edition son los encargados de encapsular la lógica del negocio de las aplicaciones empresariales, esto permite separar de modo claro para el desarrollo esta lógica del negocio de otros requerimientos como pueden ser las cavidades escalabilidad, la seguridad, la apariencia, las transacciones etc. Esto se puede aplicar tanto a un equipo que lleva a cabo un desarrollo complejo como también puede servir para distribuir de forma racional el trabajo en proyectos individuales. Los objetos EJB por tanto residen siempre en el servidor y también siempre se va a necesitar un cliente que interactúe con ellos y dentro de este obligatorio diálogo - interacción que se tiene que crear entre el cliente y el servidor a través de los de EJB se diferencian dos tipos de bits de sesión, unos son los que se denominan stateful sessión o las sesiones con estado y otros que se denominan stateless sesión o sesiones sin estado. La diferencia es que en los stateful sesión esa interacción no se cierra, se mantiene abierta entre invocación de invocación a los métodos que interactúan entre cliente y el servidor, mientras que en lo stateless no es así, se cierra la sesión después de la invocación de un método y hay que volver a abrirla cuando vuelva a invocar se otro método que necesita llevar a cabo esa interacción con el servidor, así que es la diferencia entre la sesión stateful ininterrumpida y sesión stateles interrumpidas.

Diagrama

Descripción generada automáticamente

Figura . Tipos de Session Beans

Pues si tenemos esto claro ya estamos preparados para empezar a crear session beans en el Netbeans y para empezar decir que en Netbeans podemos crear sesión vienes en tres tipos de proyectos podemos crearlo en proyectos Enterprise Application, en proyectos EJB Module y en proyectos web application. Los proyectos a EJB Module sólo pueden contener EJBs, mientras que los proyectos de Enterprise Aplication pueden contener tanto los se dejó tales como sus clientes y estos clientes pueden ser tanto aplicaciones web para lo cual usaríamos un proyecto web application, pero también pueden ser aplicaciones java independientes. Esta posibilidad de añadir otra vez a aplicaciones web es una nueva característica que se introduce también en java Enterprise Edition 6 y que facilita el empaquetado y el despliegue de aplicaciones web que usen estos Enterprise Java Beans en este caso los Sessions Beans, y ahora es posible empaquetar el código de la aplicación web y el código EJB en un único archivo WAR con anteriores versiones de Java EE tenemos que crear un archivo Enterprise Aplication pero recuerda lo vamos a ver en un momento como además de utilizar una aplicación web como cliente para negociar con las Session Beans.

Diagrama

Descripción generada automáticamente

Figura . Componentes de Session Bean

Podemos también utilizar clientes independientes aplicaciones clientes independientes que luego estarán disponibles a través de la Java Web Start tiene como aquí explica nos permite descargar y ejecutar aplicaciones Java desde la Web Start es la página en castellano de java.com donde tienes más información sobre Java Web Start.

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

Figura . Página Principal de Java Web Start

# MARCO TEÓRICO

## 3.1 ENTERPRISE JAVA BEANS.

Enterprise Java Beans (EJB) se refiere a [1] un conjunto de especificaciones de Java para desarrollar aplicaciones empresariales escalables y seguras. Es una tecnología desarrollada por inicialmente Sun Microsystems (pero que sin embargo actualmente se encuentra bajo el control de ORACLE) para facilitar el desarrollo de aplicaciones empresariales en Java. EJB es parte del Java Enterprise Edition (Java EE) y se basa en el uso de componentes de software reutilizables y administrados por un contenedor de EJB.

Enterprise Java Beans tiene una serie de características que lo hacen diferente de otras especificaciones Java, entre las que destacan que contiene propiedades adecuadas para:

* Administración de transacciones automática: EJB proporciona un mecanismo automatizado para el control de transacciones en aplicaciones empresariales, lo que permite a los desarrolladores manejar transacciones complejas sin tener que escribir código específico para ello.
* Seguridad: EJB proporciona mecanicas de seguridad integrados para proteger las aplicaciones empresariales.
* Escalabilidad: EJB es escalable, lo que permite a las aplicaciones manejar un gran número de usuarios simultáneos sin comprometer el rendimiento.
* Portabilidad: EJB es independiente de la plataforma, lo que permite a las aplicaciones desarrolladas con EJB ser ejecutadas en diferentes plataformas y sistemas operativos.

Cabe aclarar que no existe un único tipo de Enterprise Java Beans, ya que existen diferentes tipos de los mismos, entre los cuales se tiene:

* Session Beans: Este tipo de EJB se utiliza para manejar tareas relacionadas con la sesión, como el acceso a la base de datos y la lógica de negocio.
* Entity Beans: Este tipo de EJB se utiliza para representar entidades de negocio, como clientes o productos, y se utiliza para el acceso a la base de datos.
* Message-Driven Beans: Este tipo de EJB se utiliza para manejar mensajes asíncronos enviados a una aplicación.

Un ejemplo de uso de EJB es en una aplicación de compras en línea. Un Session Bean podría ser utilizado para manejar el proceso de compra, incluyendo la validación de la información de pago, la creación de una orden y la gestión de inventario. Un Entity Bean podría ser utilizado para representar productos y para manejar operaciones de acceso a la base de datos relacionadas con los productos. Un Message-Driven Bean podría ser utilizado para manejar notificaciones de envío y actualizaciones del estado de la orden. El contenedor de EJB administraría automáticamente las transacciones y la seguridad en todas estas tareas.

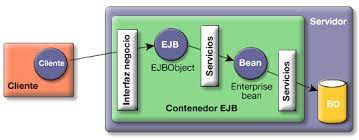


Figura 6. Estructura EJB

Como se puede observar en la imagen anterior, existe una arquitectura cliente servidor a la cual un cliente envía solicitudes que son recibidas por un servidor que contiene Enterprises JavaBeans, esos al recibir la información tienen la capacidad de controlar las solicitudes y manejar el proceso que se encuentre encargado.

## 3.2 ENTERPRISE JAVA BEANS Y EL SOFTWARE MODULAR.

Como se indicó anteriormente Enterprise Java Beans es una tecnología de Java que se utiliza para desarrollar aplicaciones empresariales en una arquitectura de componentes. Los EJBs son un tipo de componente de software modular que se utilizan en aplicaciones Java.

Por otro lado, cuando se habla de modularidad de software [2] se refiere a la técnica de dividir un sistema en módulos o componentes independientes que pueden ser desarrollados, probados y mantenidos de manera independiente. Cada módulo o componente tiene una función específica y se comunica con otros módulos a través de una interfaz estandarizada. Esto permite una mayor flexibilidad y escalabilidad en el desarrollo de aplicaciones, así como una mayor facilidad de mantenimiento y reutilización de código.

La relación existente entre ambos es que los EJBs son un ejemplo de componentes de software modular, ya que proporcionan un conjunto de servicios predefinidos para aplicaciones empresariales. Estos servicios incluyen la gestión de transacciones, la seguridad y el manejo de sesiones. Los desarrolladores pueden utilizar estos servicios en sus aplicaciones sin tener que escribir código adicional para proporcionar estas funcionalidades. Esto ayuda a los desarrolladores a crear aplicaciones empresariales de forma más fácil y rápida.

Además, los EJBs permiten crear aplicaciones escalables y distribuidas, lo que les permite ejecutarse en una variedad de plataformas. Cada EJB se ejecuta en un contenedor EJB, que es una aplicación que proporciona servicios adicionales, como la gestión de transacciones y la seguridad. Esto permite a los desarrolladores enfocarse en la lógica de negocio de sus aplicaciones, en lugar de preocuparse por la infraestructura subyacente.

En palabras más sencillas, la relación entre Enterprise Java Beans y el software modular es que EJB es una tecnología que proporciona un conjunto de componentes de software modulares para desarrollar aplicaciones empresariales en Java. Los EJBs proporcionan servicios predefinidos y una arquitectura escalable y distribuida que ayudan a los desarrolladores a crear aplicaciones de manera más fácil y rápida, y al mismo tiempo ayudan a mantener el software en un buen estado.

## 3.3 SESSION BEAN, FORMA DE CREARLOS Y UTILIZARLOS EN JAVA.

Los session beans de manera más detallada frente a lo explicado con anterioridad. Son utilizados para representar la lógica de negocio de una aplicación y proporcionan una interfaz remota para los clientes.

Existen dos tipos de session beans: los stateless y los stateful. Los beans stateless no tienen estado, es decir, no almacenan información entre las diferentes invocaciones del cliente. Por otro lado, los beans stateful sí almacenan información entre las invocaciones del cliente.

Para crear un session bean en Java, es necesario seguir los siguientes pasos:

* Crear una interfaz remota que extienda la interfaz javax.ejb.EJBObject o javax.ejb.EJBLocalObject, según si se quiere una interfaz remota o local respectivamente.
* Crear una clase que implemente la interfaz creada en el paso anterior y que extienda javax.ejb.SessionBean o javax.ejb.StatelessSessionBean según el tipo de bean stateless o stateful.
* Crear un archivo de configuración de despliegue (por ejemplo, ejb-jar.xml) para definir la información de configuración del bean, como el nombre de la interfaz remota y el nombre de la clase del bean.
* Desplegar el bean en un contenedor de EJB, por ejemplo, en un servidor de payara y utilizar la interfaz remota para acceder al bean desde un cliente.

Una vez creado el session bean se puede utilizar en una aplicación mediante una inyección de dependencia o mediante una búsqueda en el JNDI.

A continuación, se presenta un ejemplo sencillo sobre cómo utilizar la sesión vean junto como una explicación de los mismos.

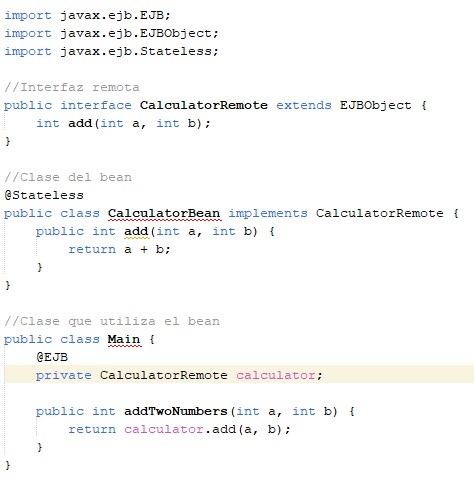


Figura 7. Ejemplo formato Session Bean

Este código muestra el uso de un Enterprise Java Bean (EJB) para crear una interfaz remota de una calculadora. La interfaz "CalculatorRemote" extiende la interfaz "EJBObject" y tiene un método "add" que toma dos parámetros enteros y devuelve un entero. La clase "CalculatorBean" implementa esta interfaz y define una implementación del método "add" que simplemente devuelve la suma de sus dos parámetros. La clase "Main" tiene una variable de instancia "calculator" que es anotada con @EJB, lo que significa que es una referencia a una instancia de CalculatorBean. La clase "Main" tiene un método "addTwoNumbers" que utiliza el bean para sumar dos números y devolver el resultado.

## 3.4 STATELESS SESSION

Stateless session es una sesión en la que el servidor no guarda ninguna información sobre el estado de la sesión. Cada petición de un cliente se trata de manera independiente, sin tener en cuenta las peticiones anteriores.

No mantiene un estado conversacional con el cliente. Cuando un cliente invoca los métodos de un bean sin estado, las variables de instancia del bean pueden contener un estado específico para ese cliente, pero solo durante la invocación. Cuando finaliza el método, no se debe conservar el estado específico del cliente. Sin embargo, los clientes pueden cambiar el estado de las variables de instancia en beans sin estado agrupados, y este estado se mantiene hasta la siguiente invocación del bean sin estado agrupado. Excepto durante la invocación del método, todas las instancias de un bean sin estado son equivalentes, lo que permite que el contenedor EJB asigne una instancia a cualquier cliente. Es decir, el estado de un bean de sesión sin estado debe aplicarse a todos los clientes.

Debido a que pueden admitir múltiples clientes, los beans de sesión sin estado pueden ofrecer una mejor escalabilidad para aplicaciones que requieren una gran cantidad de clientes. Por lo general, una aplicación requiere menos beans de sesión sin estado que beans de sesión con estado para admitir la misma cantidad de clientes. Puede ser accedido por un cliente a la vez. En caso de acceso simultáneo, el contenedor EJB enruta cada solicitud a una instancia diferente.

Escala de tiempo

Descripción generada automáticamente

Figura 8. Ciclo de vida Stateless Session

## 3.5 STATEFUL SESSION

Una sesión stateful es aquella en la que el servidor guarda información sobre el estado de la sesión, permitiendo que las peticiones subsiguientes tengan en cuenta las peticiones anteriores. En general, las sesiones stateless son más simples de implementar y escalar, pero las stateful ofrecen un mayor nivel de personalización y características avanzadas.

El estado de un objeto consta de los valores de sus variables de instancia. En un bean de sesión con estado, las variables de instancia representan el estado de una sesión única de cliente/bean. Debido a que el cliente interactúa ("habla") con su bean, este estado a menudo se denomina estado conversacional.

Como sugiere su nombre, un bean de sesión es similar a una sesión interactiva. Un bean de sesión no se comparte; solo puede tener un cliente, de la misma forma que una sesión interactiva puede tener solo un usuario. Cuando el cliente termina, su bean de sesión parece terminar y ya no está asociado con el cliente.

El estado se conserva durante la sesión de cliente/bean. Si el cliente elimina el bean, la sesión finaliza y el estado desaparece. Sin embargo, esta naturaleza transitoria del estado no es un problema porque cuando finaliza la conversación entre el cliente y el bean, no hay necesidad de retener el estado

## 3.6 SINGLETON SESSION

Se crea una instancia de un bean de sesión singleton una vez por aplicación y existe durante el ciclo de vida de la aplicación. Los beans de sesión Singleton están diseñados para circunstancias en las que los clientes comparten una sola instancia de bean empresarial y acceden al mismo tiempo.

Los beans de sesión singleton ofrecen una funcionalidad similar a los beans de sesión sin estado, pero se diferencian de ellos en que solo hay un bean de sesión singleton por aplicación, a diferencia de un grupo de beans de sesión sin estado, cualquiera de los cuales puede responder a una solicitud de cliente. Al igual que los beans de sesión sin estado, los beans de sesión singleton pueden implementar puntos finales de servicios web.

Los beans de sesión Singleton mantienen su estado entre las invocaciones de los clientes, pero no es necesario que lo mantengan durante los bloqueos o apagados del servidor.

Las aplicaciones que utilizan un bean de sesión singleton pueden especificar que se debe crear una instancia del singleton al iniciar la aplicación, lo que permite que el singleton realice tareas de inicialización para la aplicación. El singleton también puede realizar tareas de limpieza al cerrar la aplicación, porque el singleton funcionará durante todo el ciclo de vida de la aplicación.

## 3.7 JAVA PERSISTENCE API (JPA)

JPA es una API Object Relational Mapping (ORM), que permite interactuar con la base de datos por medio de objetos, de esta forma, JPA es el encargado de convertir los objetos Java en instrucciones para el Manejador de Base de Datos (MDB) y sustituye a las, es decir automatiza la asignación de objetos java a tabla de objetos de base de datos relacionales [3].

Las características que forman parte del trabajo del la JPA son:

* **Metadatos:** JPA Asigna objetos a una base de datos mediante el uso de metadatos
* **Entidades y metadatos:**
  + Entidades tienen asociados metadatos que describen el mapeado
  + Permiten reconocer las asignaciones
* **Formatos:**
  + Anotaciones descritas en javax.persistence.
  + Descriptores XML
* **Configuración por excepción**:
  + JPA cuenta con reglas del mapeado por defecto
  + Solo se necesita metadatos para excepciones.

Diagrama

Descripción generada automáticamente con confianza baja

Figura . Estructura JPA

Al trabajar con bases de datos en Java lo primero que nos enseñan es a utiliza el API de JDBC el cual nos permite realizar consultas directas a la base de datos a través de consultas SQL nativas. JDBC por mucho tiempo fue la única forma de interactuar con las bases de datos, pero representaba un gran problema y es que Java es un lenguaje orientado a objetos y se tenía que convertir los atributos de las clases en una consulta SQL como SELECT, INSERT, UPDATE, DELTE, etc. lo que ocasionaba un gran esfuerzo de trabajo y un provocaba muchos errores en tiempo de ejecución, debido principalmente a que las consultas SQL se tenían que generar frecuentemente al vuelo. Los componentes que forman parte de las JPA son ORM, EntityManager.

## 3.8 JAVA WEB START

Java Web Start es una tecnología que permite a los usuarios iniciar aplicaciones Java desde un navegador web. Una vez iniciado, la aplicación se ejecuta en el entorno de tiempo de ejecución de Java (JRE) instalado en el equipo del usuario. Java Web Start también permite a los desarrolladores especificar requisitos de seguridad para su aplicación, lo que permite restringir el acceso a ciertas funciones de la aplicación. Sin embargo, con el tiempo Java Web Start fue desplazado por otras tecnologias como HTML5 y las aplicaciones móviles.



Figura 10. Logo java web start

## 3.9 REMOTE PROCEDURE CALL (RPC)

RPC (Remote Procedure Call) es un protocolo de comunicación que permite a un programa ejecutar un procedimiento en una computadora remota, como si fuera un procedimiento local. Es similar en funcionamiento a una llamada a una función o método en un programa, pero el procedimiento se encuentra en una computadora diferente. El protocolo RPC se encarga de manejar la comunicación entre el programa cliente y el servidor, de manera que el programador pueda concentrarse en el diseño del procedimiento en sí, en lugar de preocuparse por los detalles de la comunicación.

Para que se entienda mejor, el llamado remoto de procesamiento es, procesar de manera remota algún segmento de código (utilizando una CPU, Memoria, Stack, etc, remotos), necesario localmente, sin salir del procesamiento local.

Este tema es bastante más complicado de lo que parece, ya que hay que tener en cuenta una serie de detalles y problemas, como por ejemplo:

* Marshalling y Unmarshalling: Estandarizar el orden de los datos
* Inestabilidad de la Red: Como retornar a la programación
* Seguridad: No cualquiera pueda interrumpir mi flujo de procesamiento.
* Representación de Datos: Por Ejemplo, UTF8 v/s ASCII
* Distribución de Hardware: Distinta zona de memoria, stack, clock de la CPU, etc

## 3.10 REMOTE METHOD INVOCATION (RMI)

RMI (Remote Method Invocation) es un paquete de Java que permite a los desarrolladores invocar métodos en objetos remotos. Utiliza la arquitectura cliente-servidor, donde el cliente invoca métodos en un objeto remoto que se ejecuta en el servidor.

Para la comunicación entre el servidor y el cliente, se trabaja con interfaces, que deben ser implementadas por el servidor y/o cliente, para que los STUBs puedan realizar la transparencia para ambos. Además, esto evita que deba existir una definición local real de la clase remota, vale decir, en el cliente solo debe estar definida la interface, no la clase remota.

Diagrama

Descripción generada automáticamente

Figura . Estructura de proceso RMI

Para utilizar RMI, los desarrolladores deben definir una interfaz que especifica los métodos que pueden ser invocados de forma remota. Luego, deben crear una clase que implementa esa interfaz y la registra con un registro RMI. El cliente obtiene una referencia al objeto remoto a través del registro y puede invocar sus métodos como si estuvieran ejecutándose localmente.

RMI utiliza el protocolo IIOP (Internet Inter-ORB Protocol) para comunicarse entre el cliente y el servidor. También utiliza la serialización de Java para convertir los objetos en una secuencia de bytes que pueden ser transmitidos a través de la red. Además, permite a los desarrolladores de Java crear aplicaciones distribuidas fácilmente mediante la invocación de métodos remotos en objetos registrados en un servidor.

## 3.11 RMI REGISTRY

Otro punto importante en RMI, es el como se produce la conectividad entre el cliente y servidor. Para esto se ocupa una herramienta de JAVA, llamada RMI Registry.

El RMI Registry puede estar localizado en un lugar distinto al servidor, y se encarga de registrar un determinado objeto y asignarle un servidor que se encargará de procesar dicho objeto.

El funcionamiento general es:

1. Se ejecuta el RMI Registry, en algún lugar de la red.
2. El servidor que desea manejar un objeto se registra en dicho servidor,
3. El RMI Registry registra el par: OBJETO/SERVIDOR
4. El cliente que necesita utilizar un determinado objeto hace una consulta al RMI Registry, quien devuelve el STUB listo para la comunicación

## 3.12 TÉRMINOS

A continuación, se presentará en la siguiente tabla términos acerca de las funcionalidades dentro del proyecto

Tabla . Términos

|  |  |
| --- | --- |
| @Remote | Si el cliente EJB se encuentra en un entorno diferente en el que se implementará el bean de sesión EJB. |
| @Local | Si el cliente EJB está en el mismo entorno donde se implementará el bean de sesión EJB. |
| @Stateless | Indica que es un bean sin estado. EJB Container crea automáticamente las configuraciones o interfaces relevantes requeridas al leer esta anotación durante la implementación. |
| @Override | Es una anotación estándar de Java que se introdujo por primera vez en Java 1.5. La anotación @Override indica que el método de la clase secundaria anula el método de la clase base. |

# 4. DESARROLLO

## 4.1 CREACIÓN DEL ARCHIVO ENTERPRISE APPLICATION

Abra el IDE de desarrollo Netbeans.

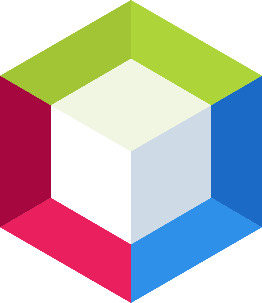


Figura . Logotipo identificativo de Netbeans.

Primero dentro del disco C y cree una carpeta llamada DISTRIBUIDAS allí crea otra subcarpeta llamada ENTERPRISE JAVABEANS, dentro de esta crea otra subcarpeta con el nombre de ENTERPRISE JAVABEANS\_GRUPO# a su vez dentro de esta última crea 2 carpetas, una con el nombre de “aplicativo” y otra con el nombre de “documentación”. Una vez haya creado las carpetas abra apache netbeans y seleccionar crear nuevo proyecto, en el tipo de aplicación buscamos en la carpeta Java EE y se selecciona Enterprise Application.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Selección del tipo de aplicación a desarrollar

Se da clic en siguiente y ponemos el nombre de ENTERPRISE JAVA BEANS\_GRUPO6, y pulsamos en siguiente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Selección del nombre y directorio donde guardar el proyecto.

En esta pantalla se elige la versión de Java EE 6 y se verifica que el casillero del check este únicamente para la creación del EJB Module ya que no se utilizara el restante, seguidamente pulsamos en terminar

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

Figura . Selección del Java EE y creación del EJB Module.

Se mostrará los dos proyectos creados.

Texto

Descripción generada automáticamente

Figura . Estructura del Proyecto

## 4.2 CREACIÓN DEL ARCHIVO DE LIBRERÍA DE CLASES

Para crear una librería de clase de Java se pulsa en nuevo proyecto y se selecciona dentro de Java el proyecto Java Class Library, y damos clic en siguiente.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Selección del proyecto Java Class Library

Se le da en nombre de ENTERPRISE JAVA BEANS\_CLIENTE\_LIB\_GRUPO# y seguidamente se pulsa en seleccionar.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Selección del proyecto Java Class Library

Se mostrará el proyecto creado.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura . Estructura del proyecto

## 4.3 CREACIÓN DEL ARCHIVO DEL CLIENTE

Al no usar la aplicación Web se necesita crear un tercer proyecto que es el módulo de la aplicación cliente. Procedemos a crear un nuevo proyecto y dentro de Java Enterprise se elige el tipo de proyecto Enterprise Appliaction Client y se pulsa en siguiente.

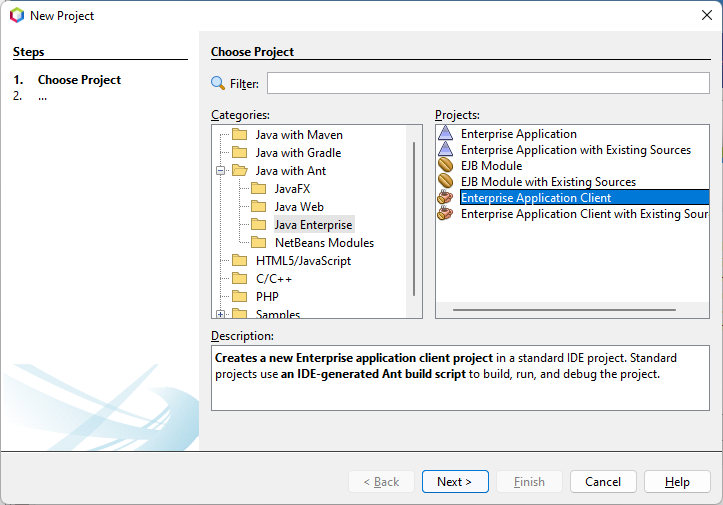


Figura . Estructura del Proyecto

Le llamamos con el nombre ENTERPRISE JAVA BEANS\_CLIENTE\_GRUPO6 y se da clic en siguiente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Estructura del Proyecto

Se añade la Enterprise Application creada anteriormente con el nombre de ENTERPRISE JAVA BEANS\_GRUPO6 y de igual forma seleccionamos la versión 6 de Java EE, damos clic en terminar.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Selección del Enterprise Application

Y finalmente se creará la aplicación del cliente.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Estructura del proyecto

Ahora se deberá agregar la librería creada del cliente, por lo que damos clic derecho en la carpeta de librería.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Agregación de la libreria

Seleccionamos la librería creada anteriormente.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura . Selección de la libreria

Se muestra la librearía agregada dentro de la carpeta.

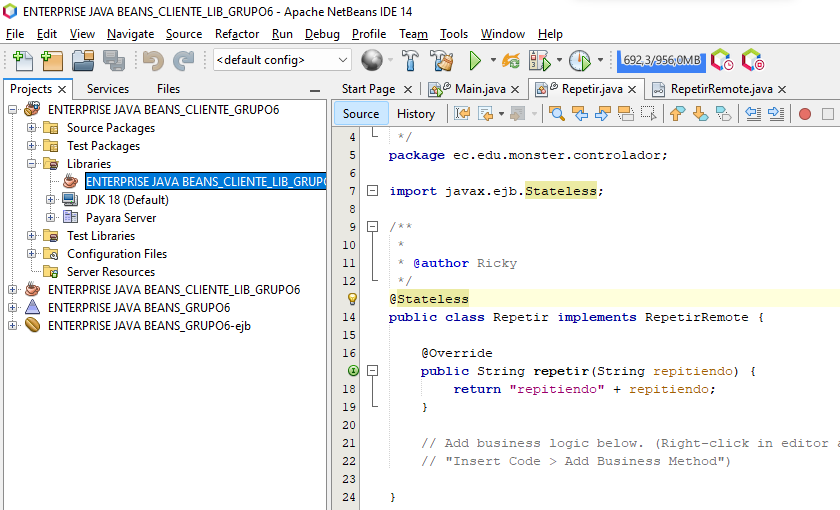


Figura . Estructura de la librería

Seguidamente dentro del archivo main, se pulsa clic derecho dentro el la clase y se elige Insert code -> Call Enterprise Bean.

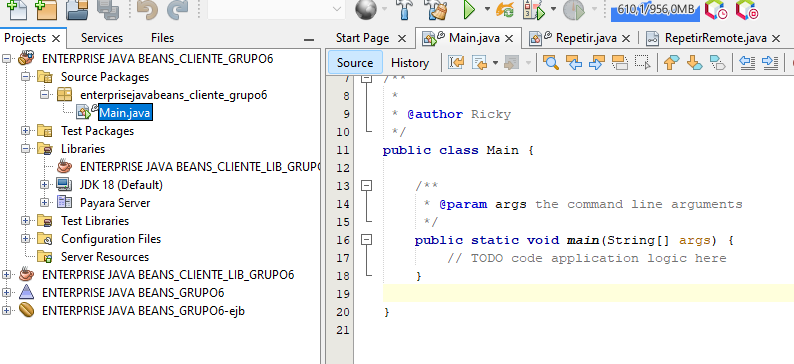


Figura . Selección del archivo Main

En la ventana de Call Enterprise Bean se selecciona el archivo Repetir y se pulsa en el botón OK

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Figura . Selección del archivo Repetir

Y por último se ingresa el mensaje que muestra la ejecución del proyecto.

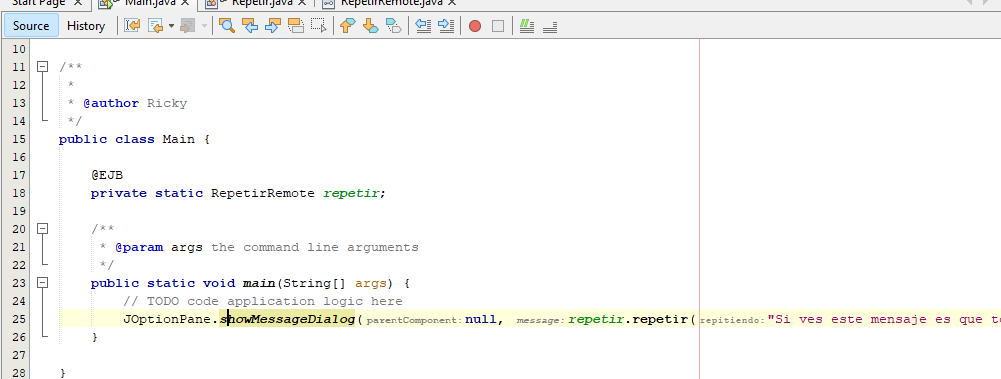


Figura . Código del mensaje del archivo Main.

Tabla . Codificación archivo Main

package sessionbeanprimeroappclient;

import ec.edu.monster.modelo.RepetirRemote;

import javax.ejb.EJB;

import javax.swing.JOptionPane;

public class Main {

@EJB

private static RepetirRemote repetir;

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

JOptionPane.showMessageDialog(null, repetir.repetir("si ves este mensaje es que todo esta funcionando bien"));

}

}

## 4.4 CREACIÓN DEL SESSION BEAN

Se da clic derecho en el proyecto de tipo EJB creado anteriormente llamado ENTERPRISE JAVA BEANS\_GRUPO# y se selecciona New -> Otro.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Creación del Session Bean

A continuación, dentro de categorías Enterprise JavaBeans se elige el archivo de tipo Session Bean y se da clic en siguiente.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura . Selección del tipo de archivo Session Bean

Se selecciona el tipo de Session que en este caso será Stateless y se especifica la creación de la interface remota dando clic en el check de Remote, una ves seleccionado se elige la librería cliente creada anteriormente llamado ENTERPRISE JAVA BEANS\_CLIENTE\_LIB\_GRUPO#.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Selección del proyecto Java Class Library

Los archivos serán creados y se podrá visualizar dentro del proyecto.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figura . Estructura de los archivos

A continuación, se añade dentro del archivo Repetir el método Business dando clic derecho en la clase y se pulsa en insert code, se elige Add Business Method.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura . Agregación de Business Method

Llamamos al método repetir y se incluye el retorno de tipo String, además se agrega un nuevo parámetro llamado repitiendo, se verifica que la interface sea remota y finalizamos.

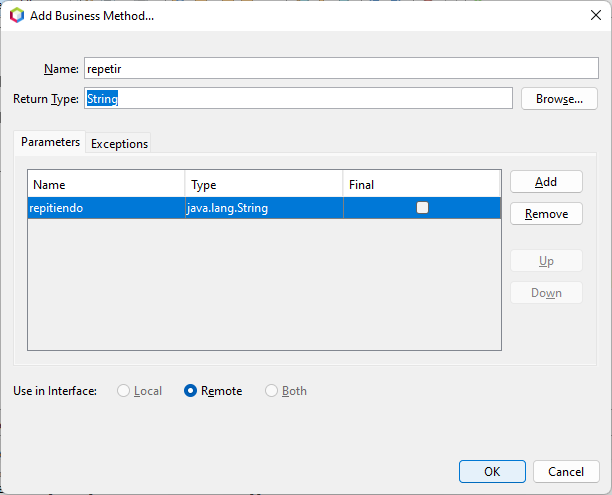


Figura . Agregación de parámetros

Se muestra el método añadido en la interfaz y modificamos para que devuelva el parámetro repitiendo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Estructura del Archivo Repetir

Tabla . Codificación del archivo Repetir

package ec.edu.monster.modelo;

import javax.ejb.Stateless;

@Stateless

public class Repetir implements RepetirRemote {

@Override

public String repetir(String repitiendo) {

return "repitiendo: " + repitiendo;

}

// Add business logic below. (Right-click in editor and choose

// "Insert Code > Add Business Method")

}

# 5. EJECUCIÓN DEL PROYECTO

Se pulsa clic derecho en el proyecto ENTERPRISE JAVA BEANS\_GRUPO# y se selecciona Run para ejecutar el proyecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura . Ejecución del proyecto.

Se muestra el mensaje de notificacion creado para validar que el proyecto funcione correctamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Figura . Ventana del mensaje

# 6. CONCLUSIONES

* Enterprise Java Beans (EJB) permite la creación de aplicaciones empresariales robustas y escalables. EJB proporciona un marco para la gestión de transacciones, seguridad y administración de recursos, lo que permite a los desarrolladores concentrarse en la lógica de negocio de la aplicación en lugar de preocuparse por aspectos técnicos subyacentes. Sin embargo, EJB también puede ser considerado como una tecnología compleja y difícil de aprender para los desarrolladores novatos, y su uso no es siempre necesario en aplicaciones de menor escala.
* Los Session Beans proporcionan una interfaz para acceder a los recursos de la aplicación, como bases de datos o servicios web, y pueden ser utilizados tanto por aplicaciones cliente como por otros EJBs. Uno de los principales beneficios de los Session Beans es que proporcionan una forma estandarizada y segura de acceder a los recursos de la aplicación, ya que el contenedor EJB se encarga de la gestión de transacciones, seguridad y administración de recurso.

# 7. RECOMENDACIONES

* Se recomienda utilizar la versión de jdk numero 11 para abajo debido a que esta es aceptada en el servidor de payara, lo cual al momento de desplegar nuestra aplicación nos facilitará la ejecución ya que este es compatible con el IDE de apache netbeans.
* Se recomienda tener cuidado al momento de realizar el cambio de nombre de alguna función sobre todo en el controlador ya que un cambio en el controlador puede tener la capacidad de afectar a todo el proyecto, asi como al momento de realizar la configuración de las rutas de inicio del proyecto.

# 8. BIBLIOGRAFÍA

|  |  |
| --- | --- |
| [1] | C. Á. Caules, «Enterprise Java Beans y su funcionamiento,» ArquitecturaJava, 29 06 2018. [En línea]. Available: https://tinyurl.com/5cxwzn7e. [Último acceso: 17 01 2023]. |
| [2] | D. Lara, «Modularidad en la programación orientada a objetos,» Styde, 07 07 2015. [En línea]. Available: https://tinyurl.com/2srhuk69. [Último acceso: 18 01 2023]. |
| [3] | O. Blancarte, «OscarBlancarte,» [En línea]. Available: https://www.oscarblancarteblog.com/tutoriales/java-persistence-api-jpa/. [Último acceso: Enero 2023]. |
| [4] | TuProgramacion, «TuProgramacion,» [En línea]. Available: http://www.tuprogramacion.com/glosario/que-es-un-orm/. [Último acceso: Enero 2022]. |
| [5] | Vivek, «Baeldung,» 23 Febrero 2022. [En línea]. Available: baeldung.com/jpa-entities. [Último acceso: Enero 2023]. |
| [6] | U. d. Alicante, «jtech,» 7 Marzo 2013. [En línea]. Available: http://www.jtech.ua.es/j2ee/restringido/jpa/sesion05-apuntes.html. [Último acceso: Enero 2023]. |
| [7] | O. Blancarte, «OscarBlancarteblog,» 10 Octubre 2018. [En línea]. Available: https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/. [Último acceso: Enero 2023]. |
| [8] | Eclipse, «Eclipse,» [En línea]. Available: https://www.eclipse.org/eclipselink/. [Último acceso: Enero 2023]. |
| [9] | «JavaServer Faces(JSF),» Junta de desarrollo de Andalucía, 13 11 2016. [En línea]. Available: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/101. [Último acceso: 06 01 2023]. |