

CURSO JAVA EE

EJERCICIO

SGA CON JPA



Experiencia y Conocimiento para tu vida

CURSO JAVA EE

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

- El objetivo del ejercicio agregar persistencia con JPA a nuestro proyecto SGA (Sistema de Gestión de Alumnos). Al finalizar deberemos observar el siguiente resultado:

```
Output ×
GlassFish Server × Test (PersonaServiceTest) ×

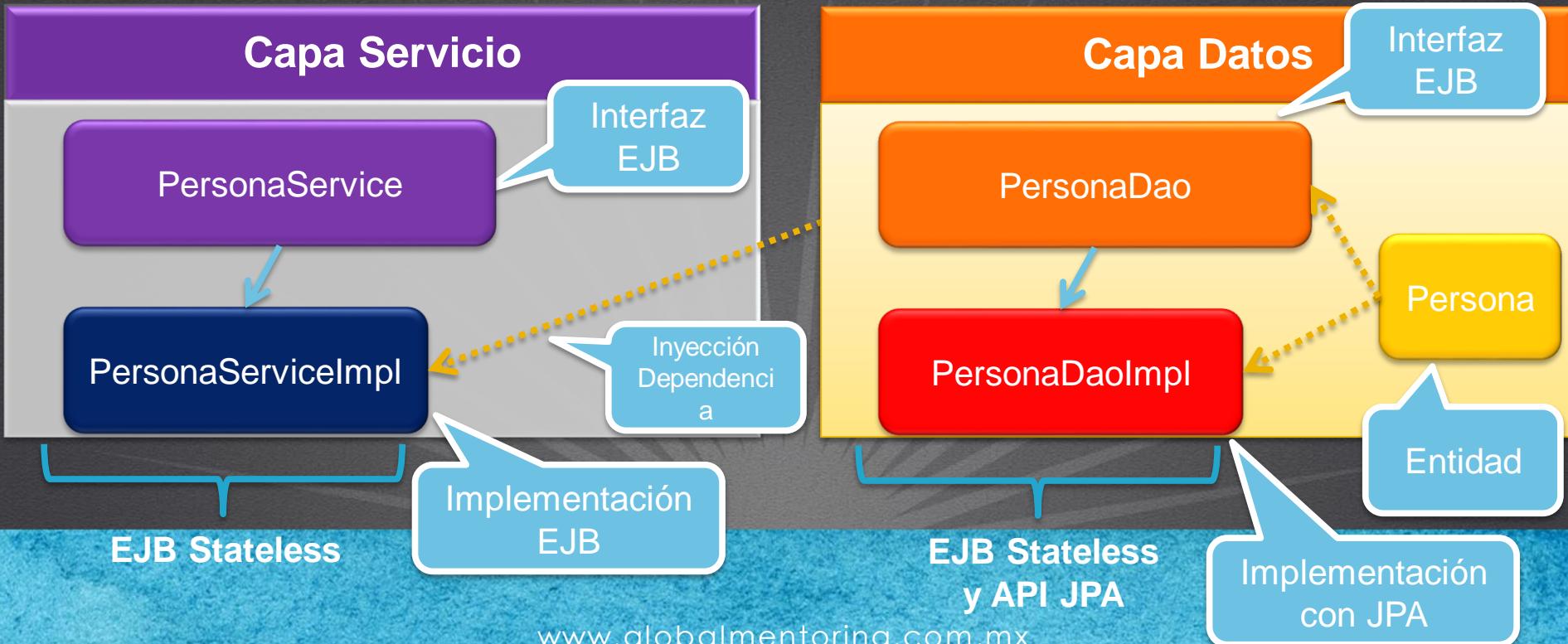
abr 29, 10:42:33 PM com.sun.ejb.containers.BaseContainer initializeHome
INFORMACIÓN: Portable JNDI names for EJB PersonaServiceImpl: [java:global/classes/PersonaServiceImpl!mx.com.gm.sga.servicio.PersonaService, ja
abr 29, 10:42:33 PM com.sun.ejb.containers.BaseContainer initializeHome
INFORMACIÓN: Glassfish-specific (Non-portable) JNDI names for EJB PersonaServiceImpl: [mx.com.gm.sga.servicio.PersonaServiceRemote, mx.com.gm.s
abr 29, 10:42:34 PM org.glassfish.deployment.admin.DeployCommand execute
INFORMACIÓN: classes was successfully deployed in 120,683 milliseconds.
Iniciando test EJB PersonaService
El no. de personas es igual a:2
Persona [idPersona=1, nombre=Oscar, apePaterno=Gomez, apeMaterno=Larios, email=ogomez@mail.com.mx34567, telefono=55780109]
Persona [idPersona=6, nombre=Maria, apePaterno=Gutierrez, apeMaterno=Esparza, email=maria@mail.com.mx, telefono=11113333]
Fin test EJB PersonaService
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 134.737 sec
PlainTextActionReporterSUCCESSNo monitoring data to report.
JdbcRuntimeExtension, getAllSystemRAResourcesAndPools = [GlassFishConfigBean.org.glassfish.jdbc.config.JdbcResource, GlassFishConfigBean.org.g
JdbcRuntimeExtension, getAllSystemRAResourcesAndPools = [GlassFishConfigBean.org.glassfish.jdbc.config.JdbcResource, GlassFishConfigBean.org.g
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2:48.369s
Finished at: Sat Apr 29 22:42:45 CDT
Final Memory: 103M/705M
```

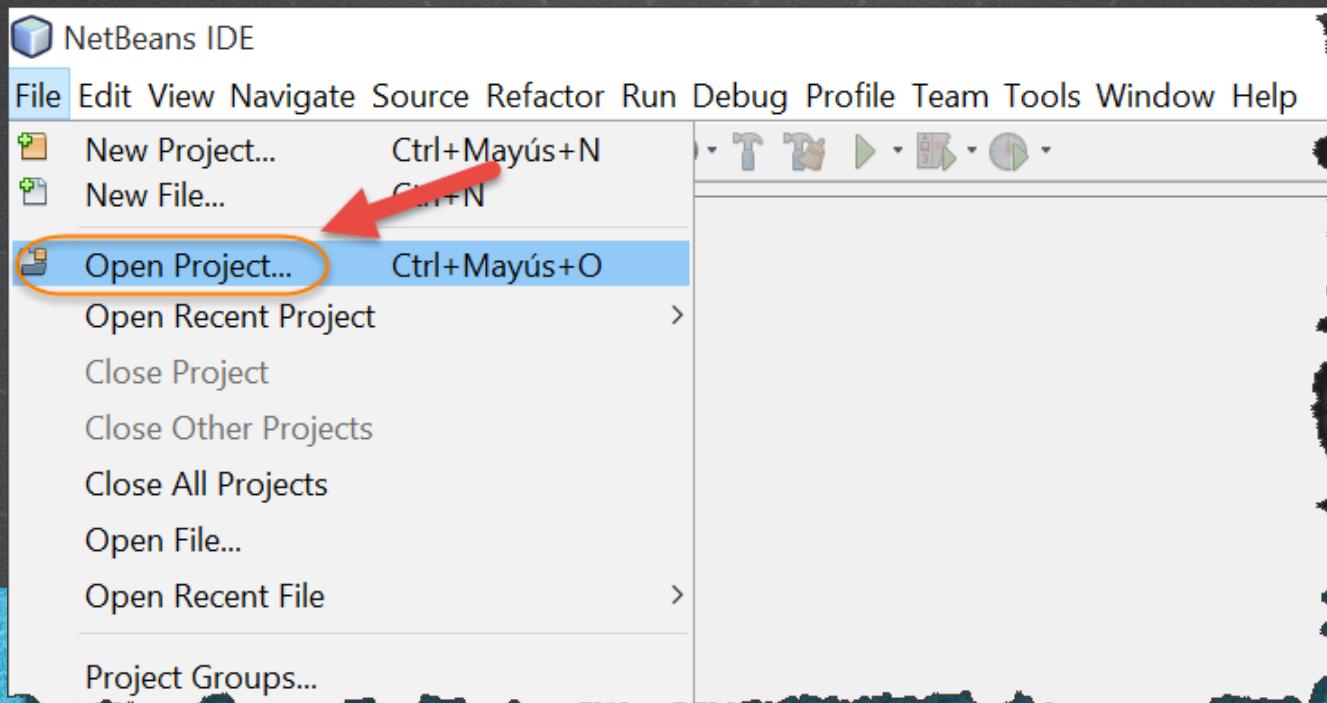
ARQUITECTURA JAVA EE

- Convertiremos nuestra clase Persona en una clase de Entidad, a su vez agregaremos la capa de datos de nuestro Sistema SGA (Sistema de Gestión de Alumnos) con el objetivo de integrar la persistencia con JPA.



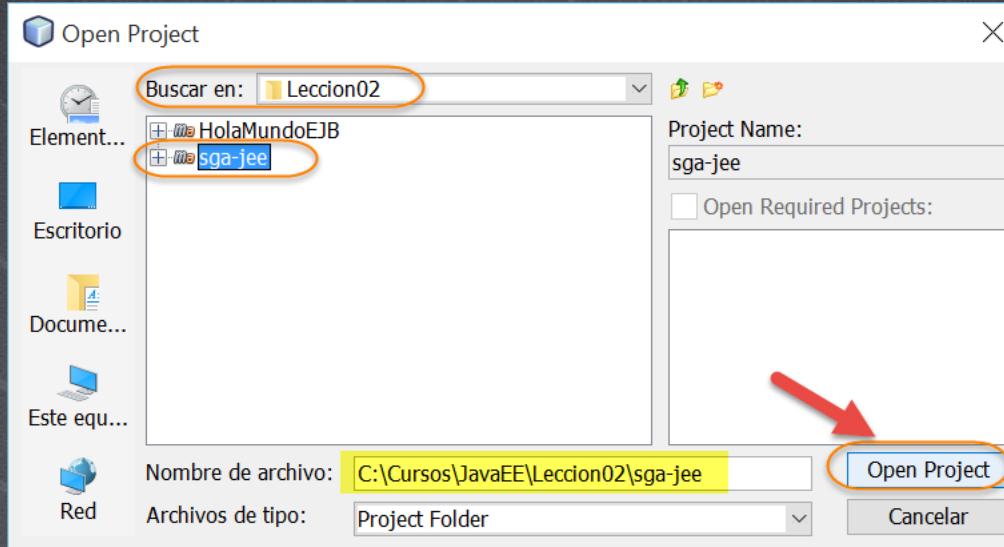
PASO 1. ABRIMOS EL PROYECTO

En caso que no tengamos abierto el proyecto sga-jee lo abrimos:



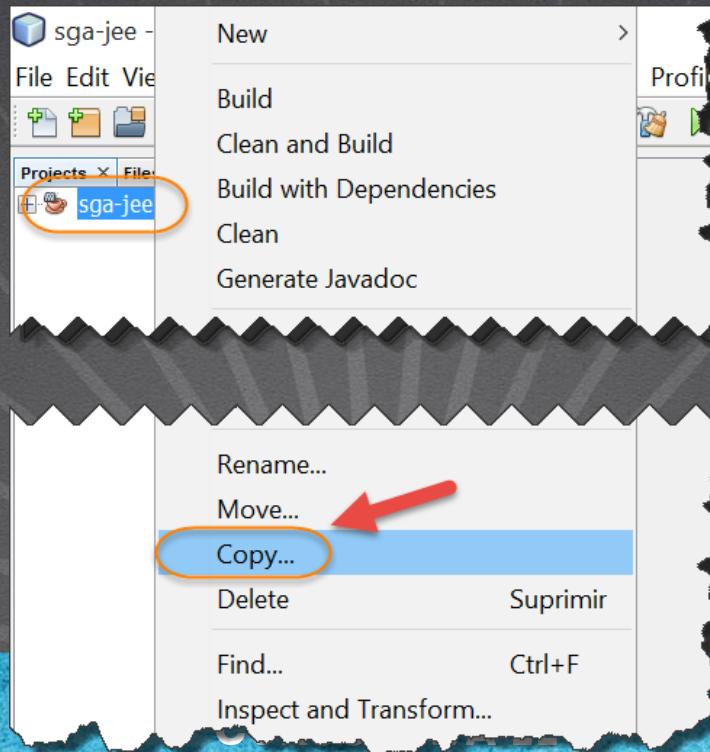
PASO 1. ABRIMOS EL PROYECTO

En caso que no tengamos abierto el proyecto sga-jee lo abrimos:



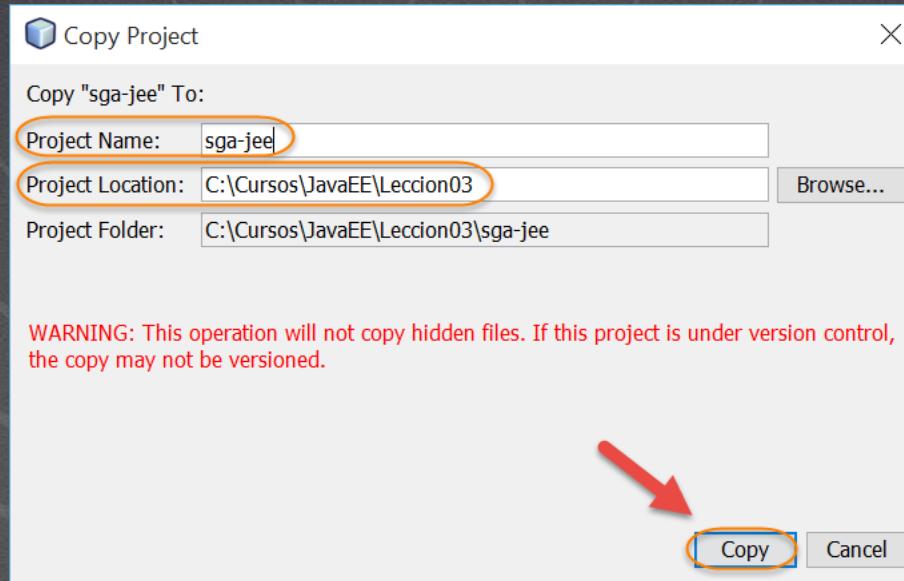
PASO 2. COPIAMOS EL PROYECTO

Copiamos y pegamos el proyecto:



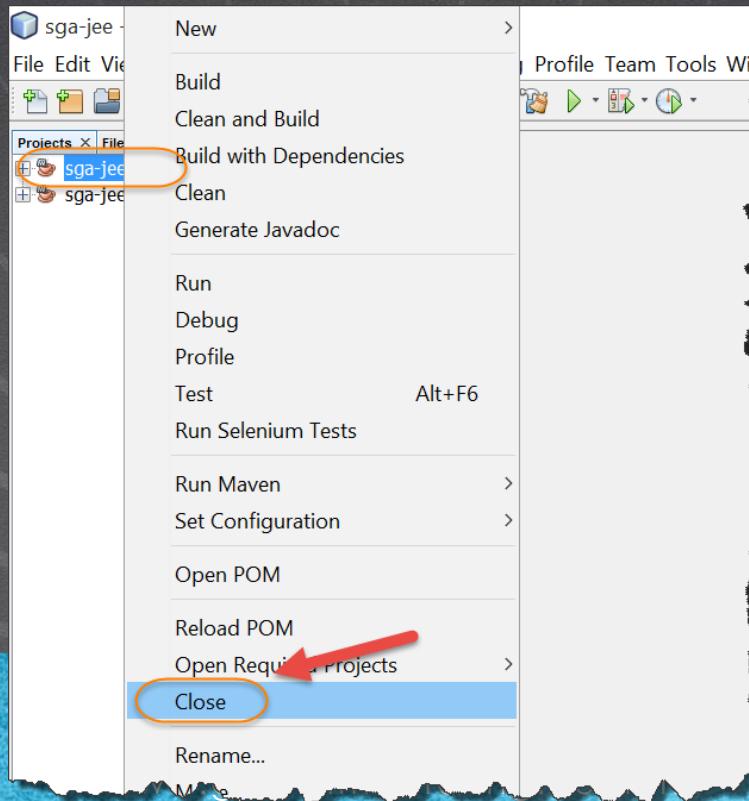
PASO 2. COPIAMOS EL PROYECTO

Copiamos y pegamos el proyecto:



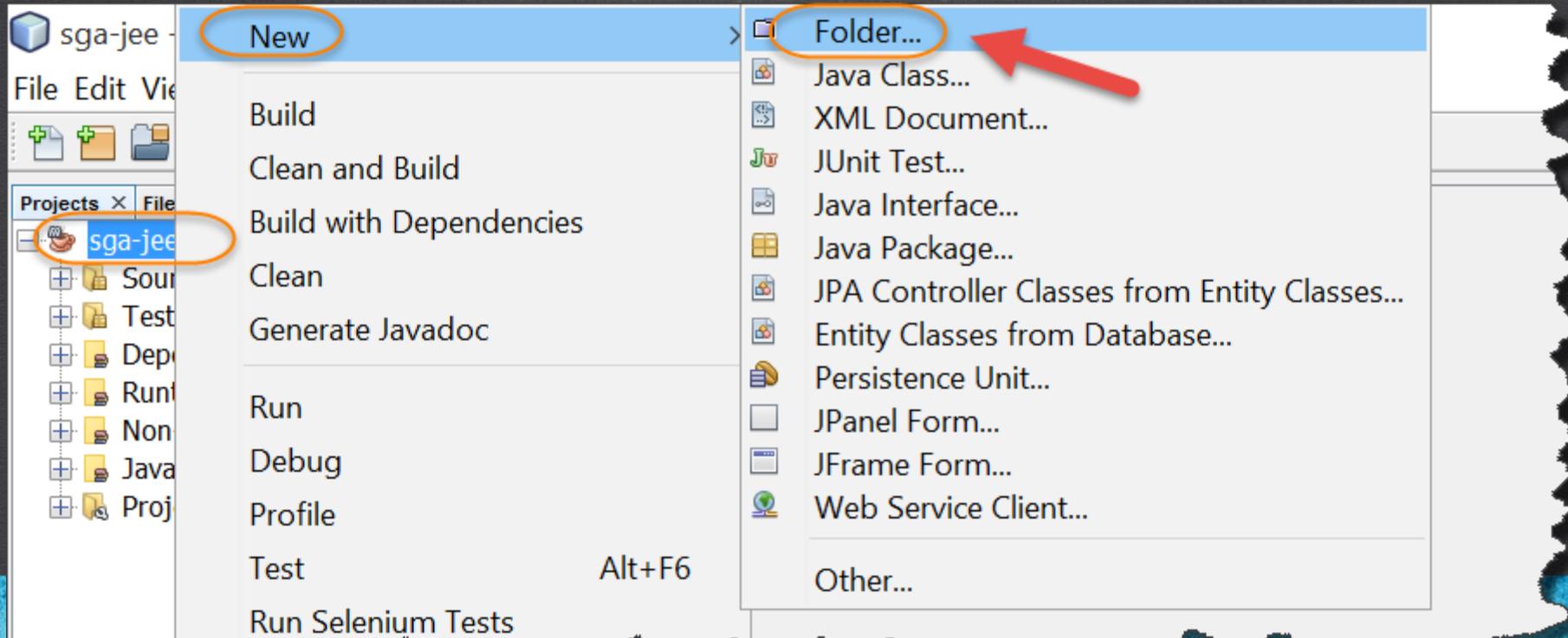
PASO 3. CERRAMOS EL PROYECTO

Cerramos el proyecto anterior y dejamos el nuevo:



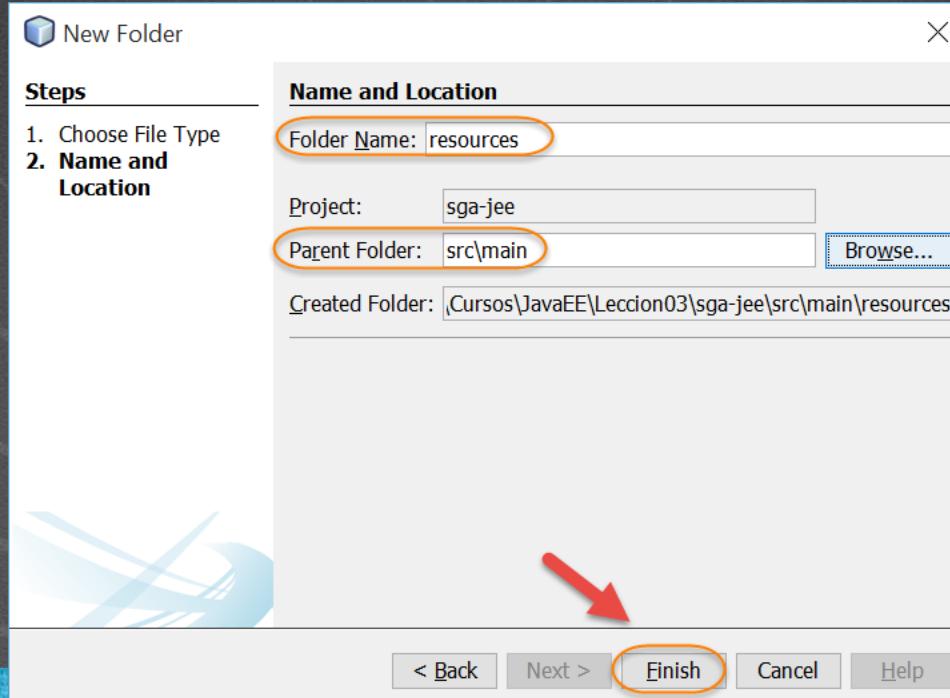
PASO 4. CREAMOS UN FOLDER

Creamos un folder llamado src/main/resources



PASO 4. CREAMOS UN FOLDER

Creamos un folder llamado src/main/resources

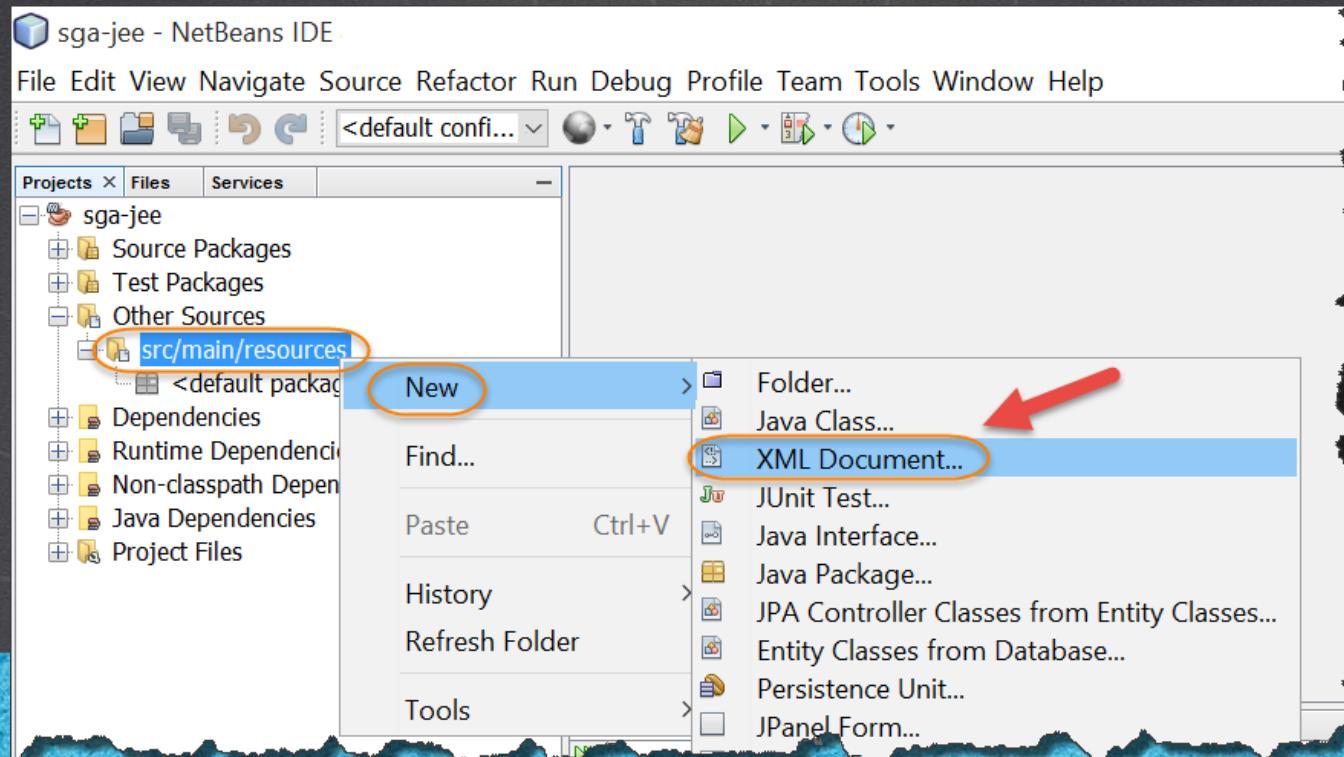


CURSO JAVA EE

www.globalmentoring.com.mx

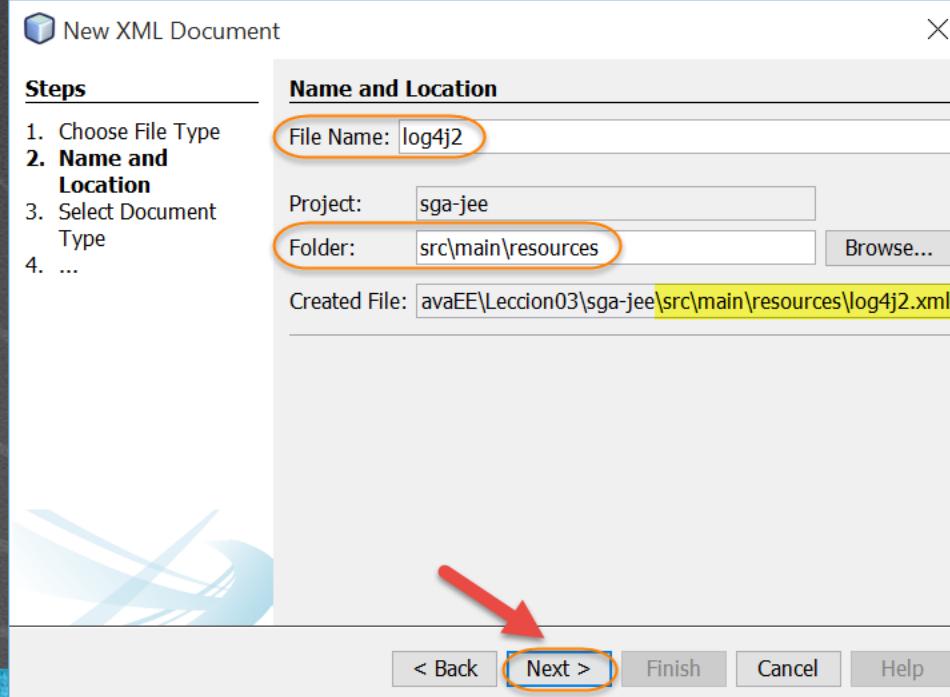
PASO 5. CREAMOS UN ARCHIVO XML

Creamos el archivo log4j2.xml.



PASO 5. CREAMOS UN ARCHIVO XML

Creamos el archivo log4j2.xml

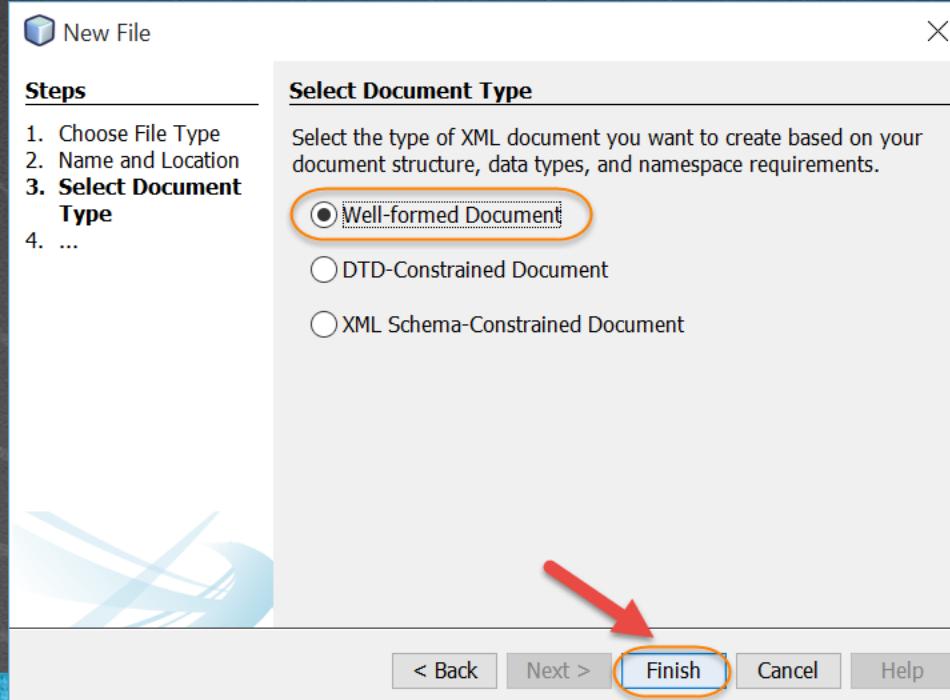


CURSO JAVA EE

www.globalmentoring.com.mx

PASO 5. CREAMOS UN ARCHIVO XML

Creamos el archivo log4j2.xml

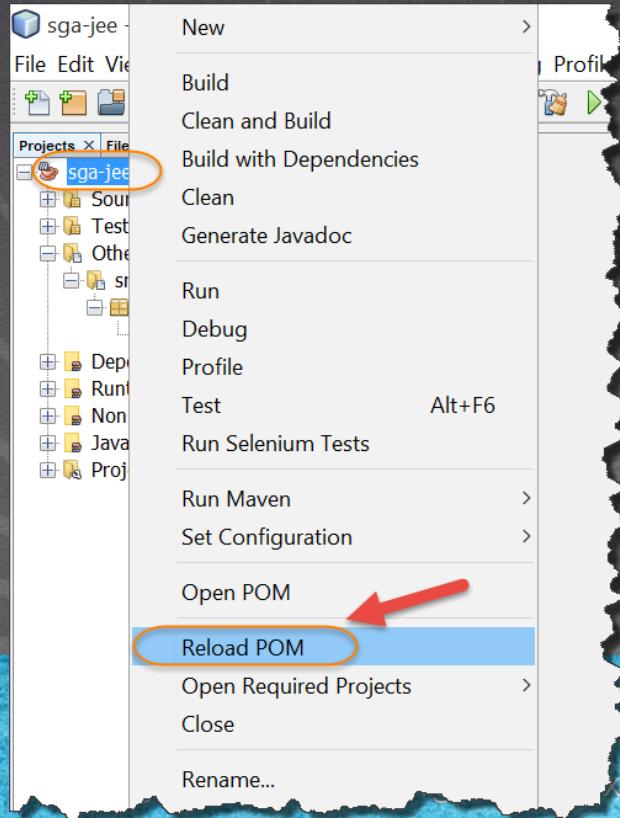


CURSO JAVA EE

www.globalmentoring.com.mx

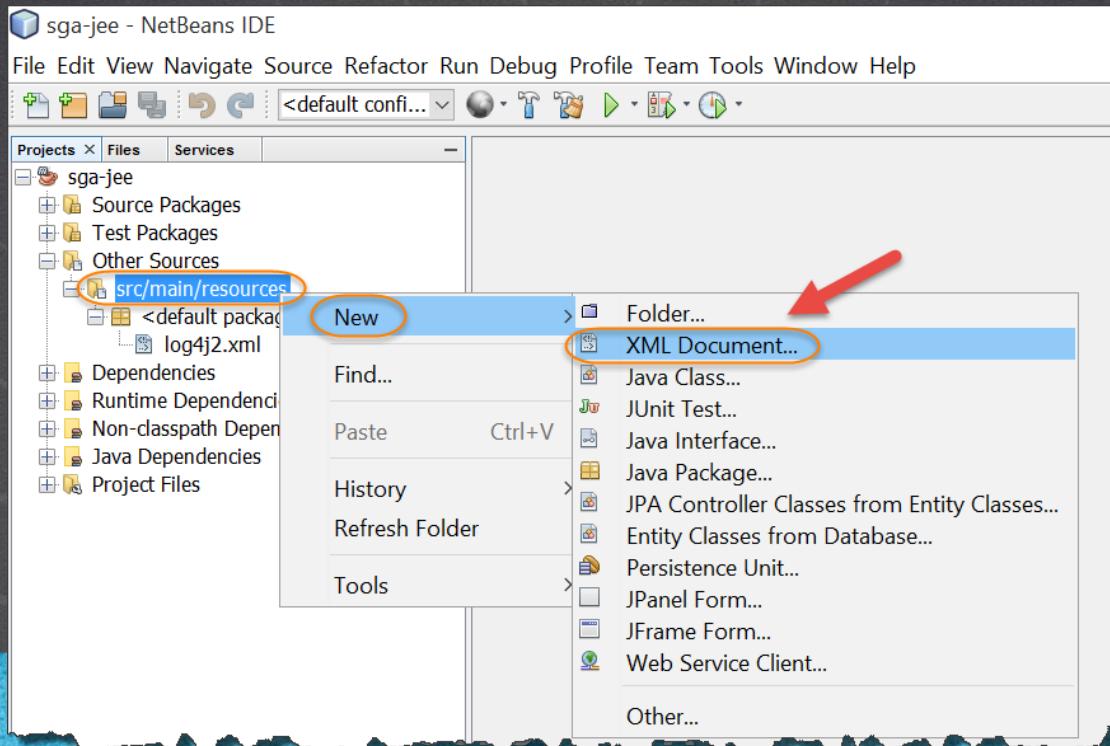
PASO 6. RECARGAMOS EL PROYECTO

Recargamos el proyecto para ver la nueva carpeta en caso que no se visualice:



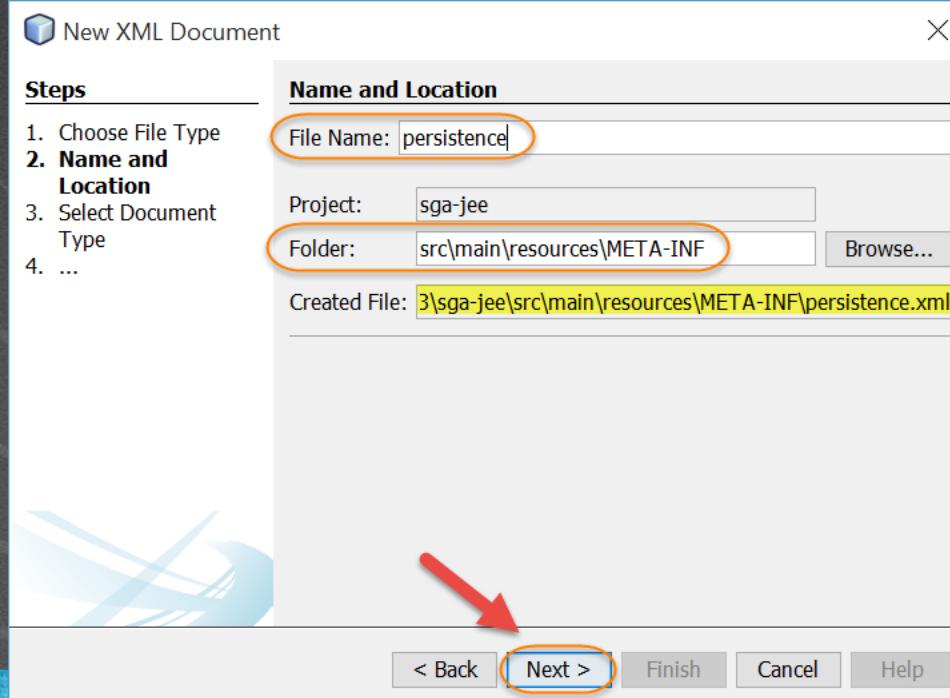
PASO 7. CREAR UN ARCHIVO XML

Creamos el archivo persistence.xml



PASO 7. CREAR UN ARCHIVO XML

Creamos el archivo persistence.xml

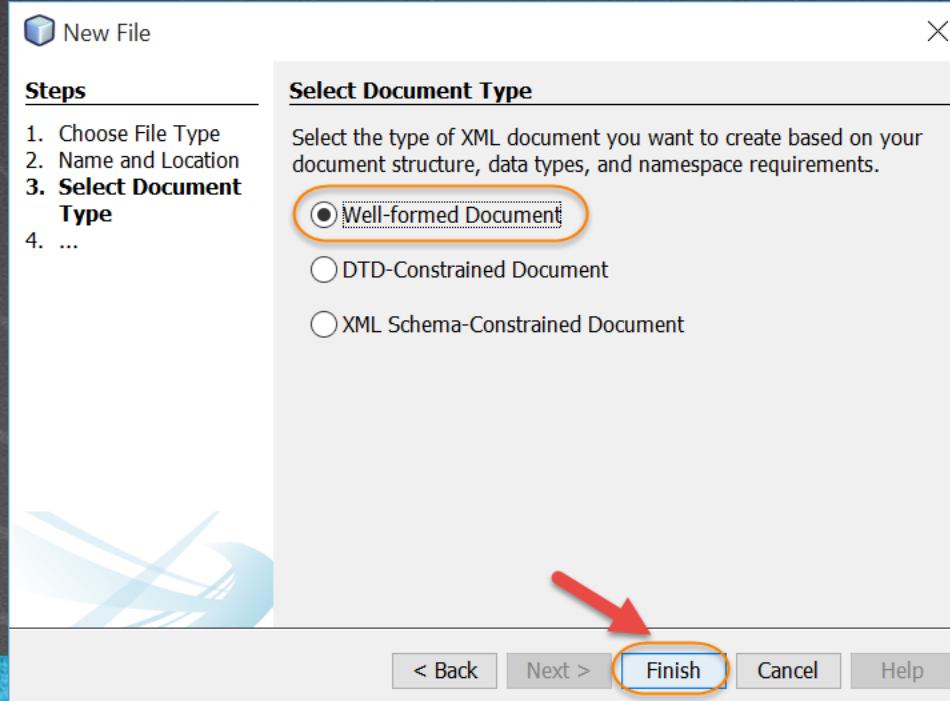


CURSO JAVA EE

www.globalmentoring.com.mx

PASO 7. CREAR UN ARCHIVO XML

Creamos el archivo persistence.xml

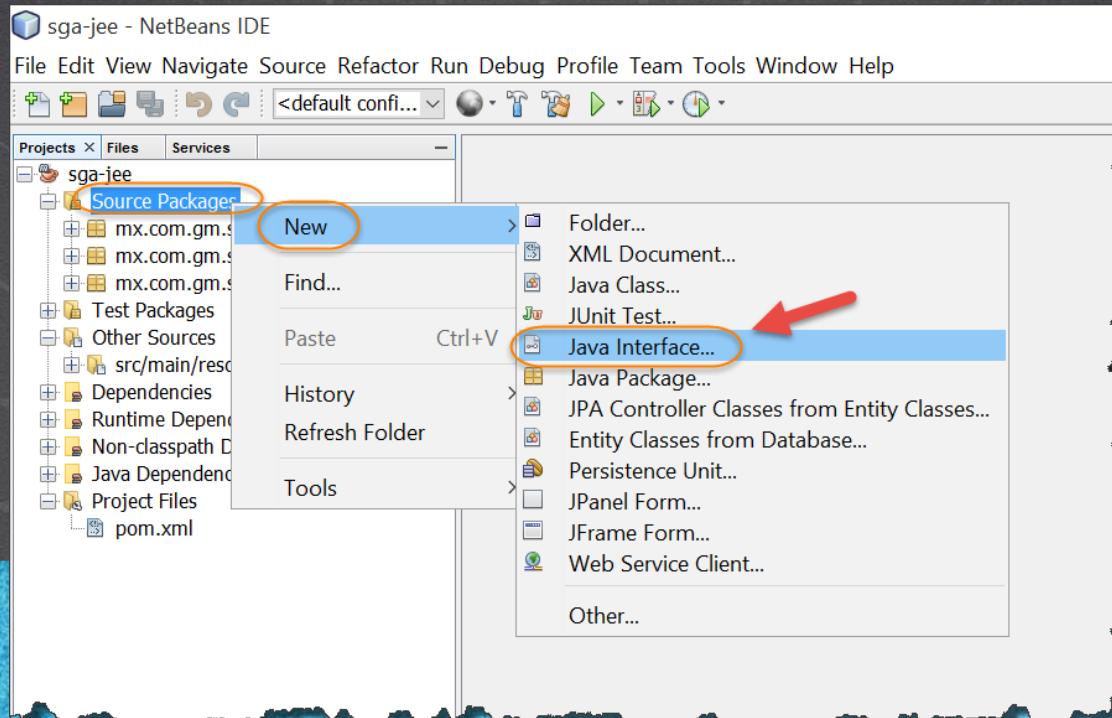


CURSO JAVA EE

www.globalmentoring.com.mx

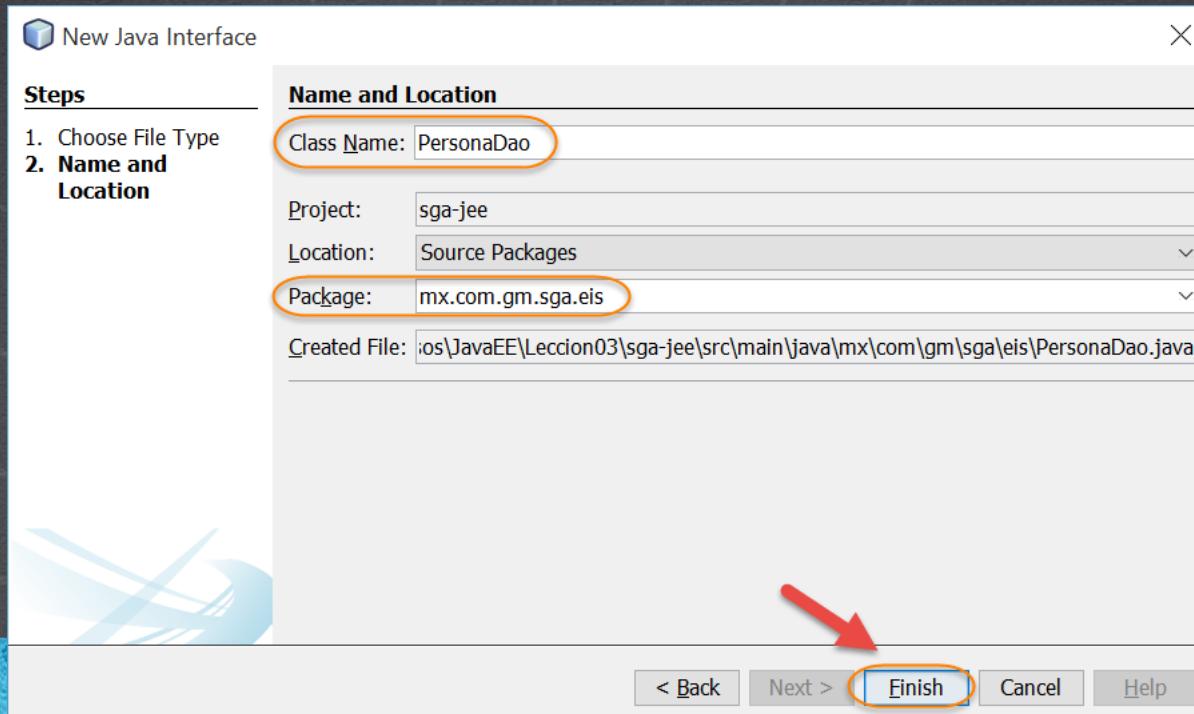
PASO 8. CREAR UNA CLASE

Creamos una interfaz PersonaDao. El paquete EIS significa Enterprise Information System y básicamente es la información de la empresa, sin embargo este paquete representa nuestra capa de datos:



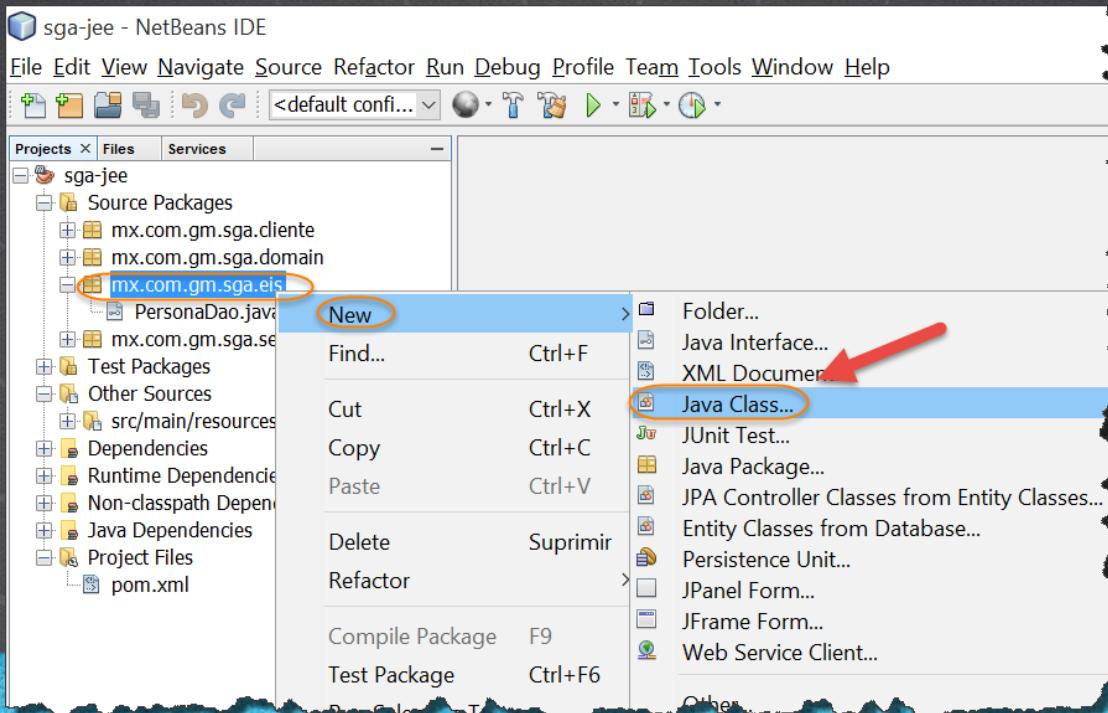
PASO 8. CREAR UNA CLASE

Creamos una interfaz PersonaDao:



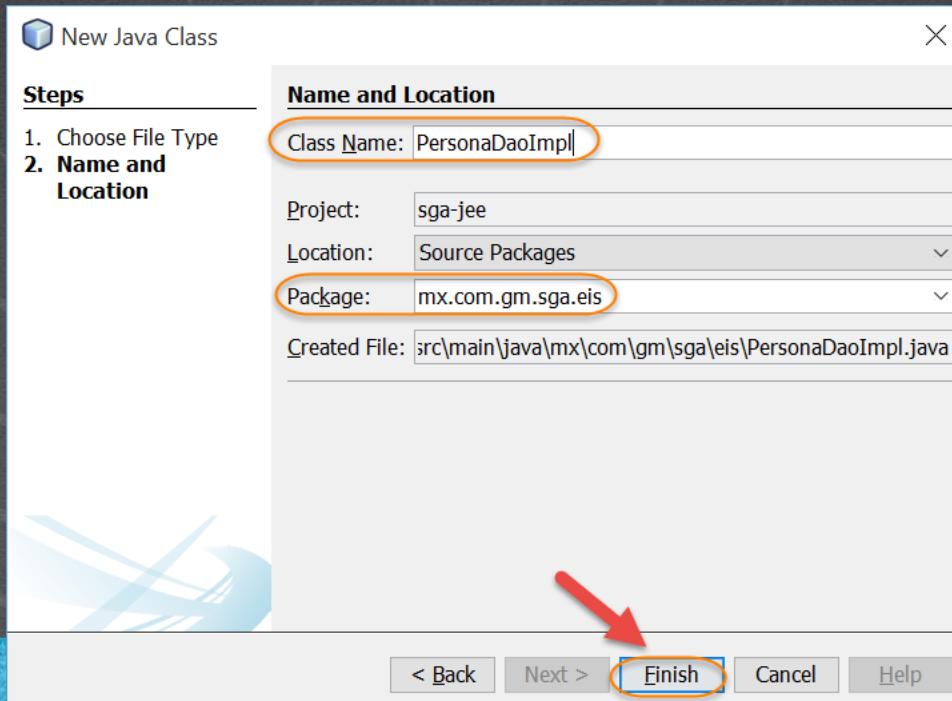
PASO 9. CREAR UNA CLASE

Creamos la clase PersonaDaoImpl.java:



PASO 9. CREAR UNA CLASE

Creamos la clase PersonaDaoImpl.java:



PASO 10. MODIFICAMOS EL CÓDIGO

Archivo log4j2.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss} [%t] %-5level %logger{36} - %msg%n" />
        </Console>
    </Appenders>
    <Loggers>
        <Root level="debug">
            <AppenderRef ref="Console" />
        </Root>
    </Loggers>
</Configuration>
```

PASO 11. MODIFICAMOS EL CÓDIGO

Archivo persistence.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="PersonaPU" transaction-type="JTA">
      <jta-data-source>jdbc/PersonaDb</jta-data-source>
    </persistence-unit>
</persistence>
```

PASO 12. MODIFICAMOS EL CÓDIGO

Archivo pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>mx.com.gm.sga</groupId>
    <artifactId>sga-jee</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <glassfish.embedded-static-shell.jar>
            C:\AppServers\glassfish4\glassfish\lib\embedded\glassfish-embedded-static-shell.jar
        </glassfish.embedded-static-shell.jar>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.glassfish.extras</groupId>
            <artifactId>glassfish-embedded-static-shell</artifactId>
            <version>4.1.2</version>
            <scope>system</scope>
            <systemPath>${glassfish.embedded-static-shell.jar}</systemPath>
        </dependency>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-api</artifactId>
            <version>7.0</version>
            <scope>provided</scope>
        </dependency>
```

PASO 12. MODIFICAMOS EL CÓDIGO

Archivo pom.xml:

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
<dependency>
    <groupId>org.glassfish.main.appclient</groupId>
    <artifactId>gf-client</artifactId>
    <version>4.1.2</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.41</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.8.2</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.8.2</version>
</dependency>
</dependencies>
</project>
```

PASO 13. MODIFICAMOS EL CÓDIGO

Archivo Persona.java:

Click para descargar el código

```
package mx.com.gm.sga.domain;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.Table;

@Entity
@NamedQueries({
    @NamedQuery(name = "Persona.findAll", query = "SELECT p FROM Persona p ORDER BY p.idPersona"))
@Table(name = "persona")
public class Persona implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_persona")
    private int idPersona;

//Click en el link para descargar el código completo
```

PASO 14. MODIFICAMOS EL CÓDIGO

Archivo PersonaDao.java:

```
package mx.com.gm.sga.eis;

import java.util.List;
import mx.com.gm.sga.domain.Persona;

public interface PersonaDao {

    public List<Persona> findAllPersonas();

    public Persona findPersonaById(Persona persona);

    public Persona findPersonaByEmail(Persona persona);

    public void insertPersona(Persona persona);

    public void updatePersona(Persona persona);

    public void deletePersona(Persona persona);
}
```

PASO 15. MODIFICAMOS EL CÓDIGO

Archivo PersonaDaoImpl.java:

Click para descargar el código

```
package mx.com.gm.sga.eis;

import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import mx.com.gm.sga.domain.Persona;

@Stateless
public class PersonaDaoImpl implements PersonaDao {

    @PersistenceContext(unitName = "PersonaPU")
    EntityManager em;

    @Override
    public List<Persona> findAllPersonas() {
        return em.createNamedQuery("Persona.findAll").getResultList();
    }

    @Override
    public Persona findPersonaById(Persona persona) {
        return em.find(Persona.class, persona.getIdPersona());
    }

//Click en el link para descargar el código completo
```

PASO 16. MODIFICAMOS EL CÓDIGO

Archivo PersonaServiceImpl.java:

Click para descargar el código

```
package mx.com.gm.sga.servicio;

import java.util.List;
import javax.inject.Inject;
import javax.ejb.Stateless;
import mx.com.gm.sga.domain.Persona;
import mx.com.gm.sga.eis.PersonaDao;

@Stateless
public class PersonaServiceImpl implements PersonaServiceRemote, PersonaService {

    @Inject
    private PersonaDao personaDao;

    @Override
    public List<Persona> listarPersonas() {
        return personaDao.findAllPersonas();
    }

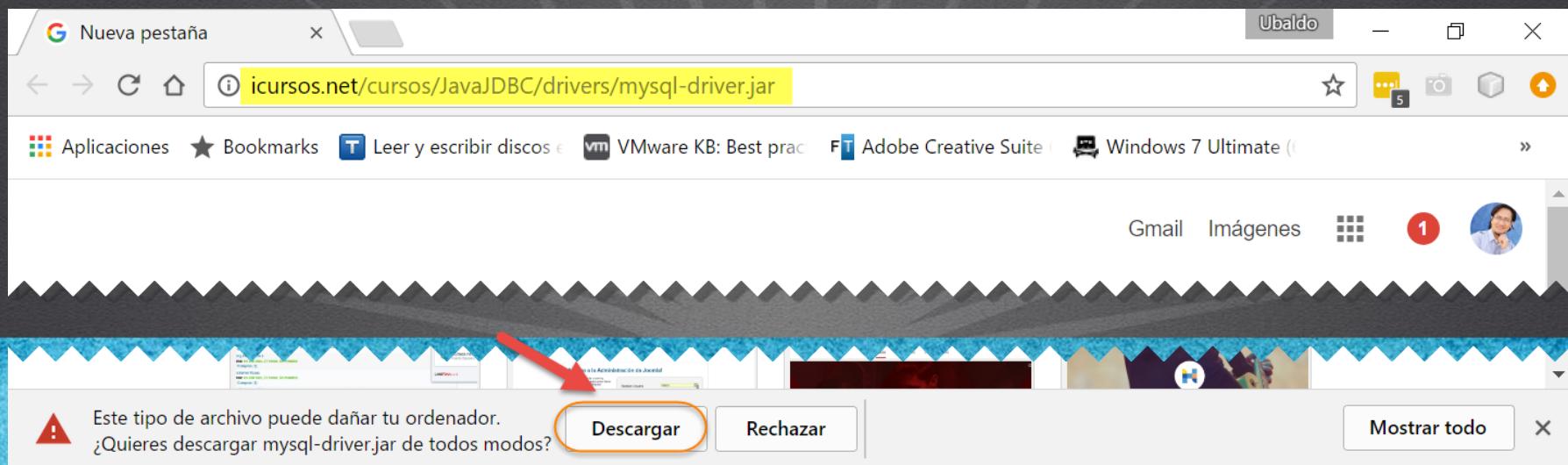
    @Override
    public Persona encontrarPersonaPorId(Persona persona) {
        return personaDao.findPersonaById(persona);
    }

//Click en el link para descargar el código completo
```

PASO 17. CONFIGURACION CONEXIÓN JTA

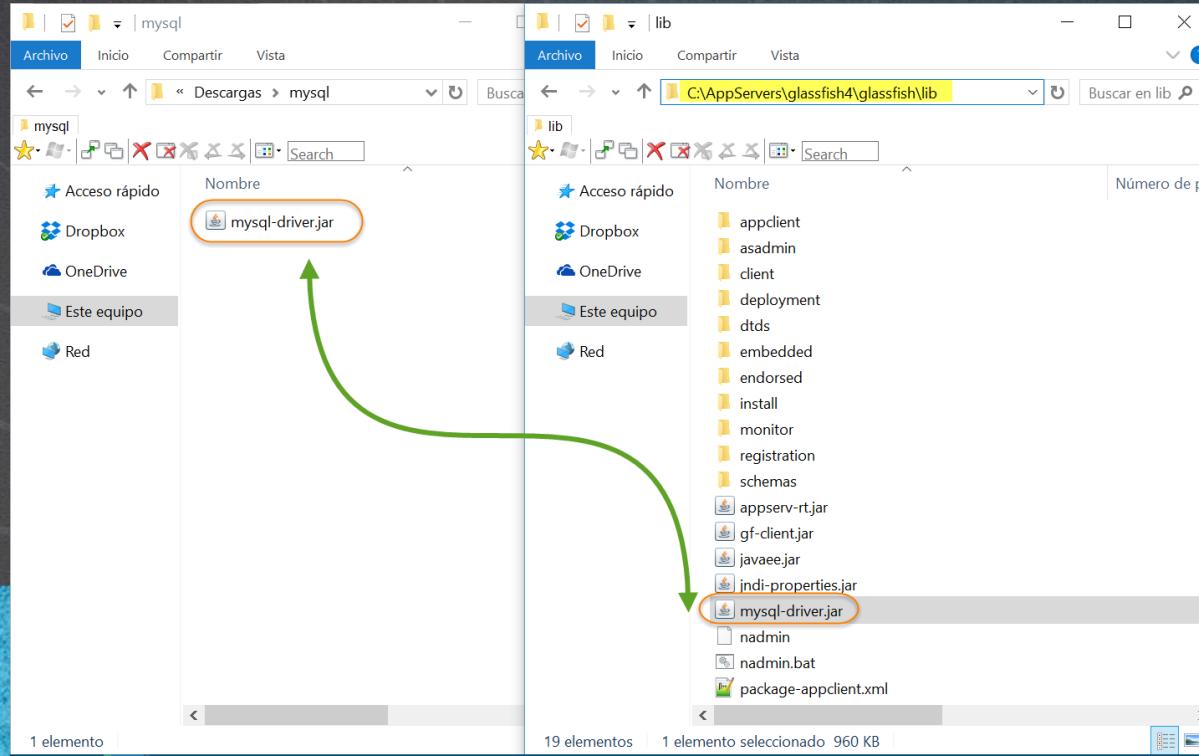
Configuramos la conexión de JTA en Glassfish. Agregamos el driver de mysql a Glassfish. Descargamos el .jar de mysql:

<http://icursos.net/cursos/JavaJDBC/drivers/mysql-driver.jar>



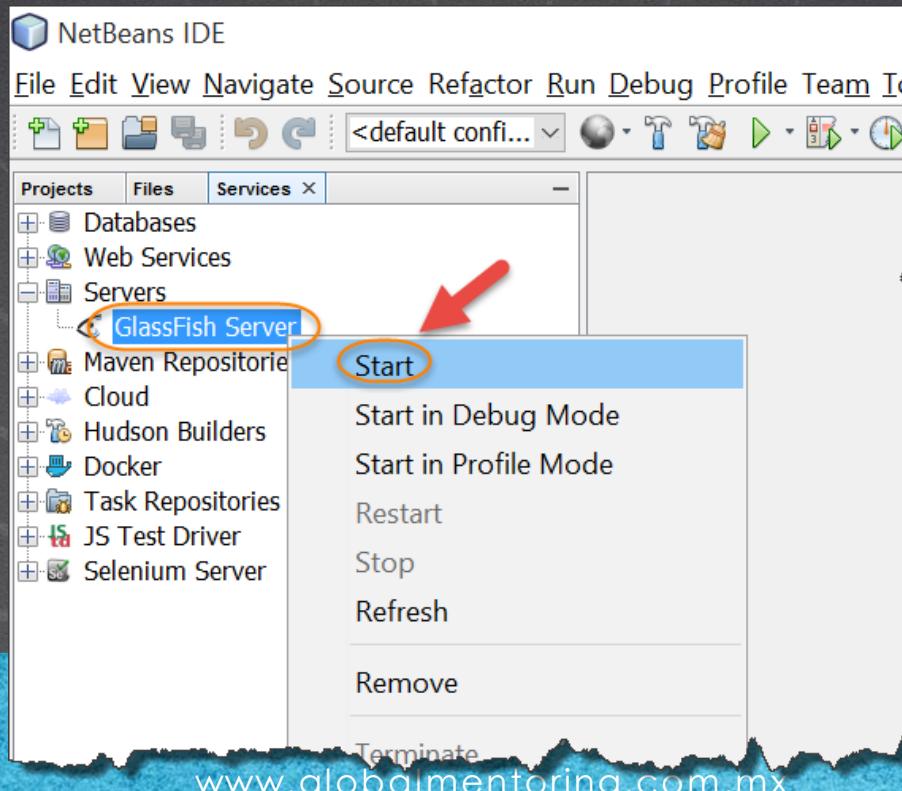
PASO 17. CONFIGURACION CONEXIÓN JTA

Copiamos el archivo recién descargado en la ruta de instalación de Glassfish. Ej. C:\AppServers\glassfish4\glassfish\lib



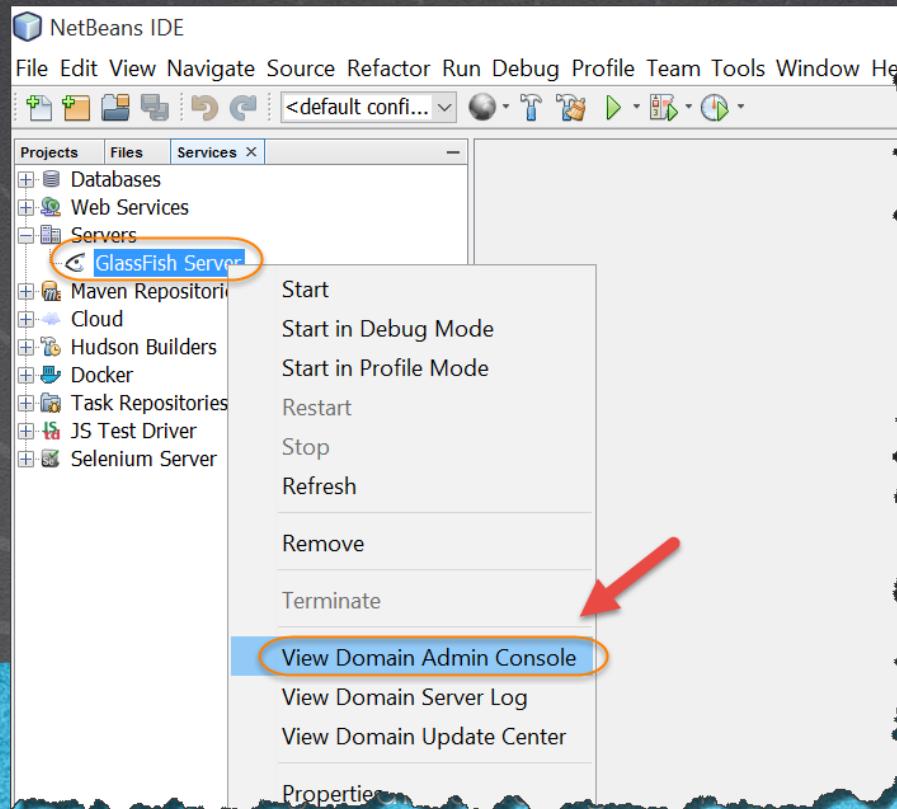
PASO 17. CONFIGURACION CONEXIÓN JTA

Configuramos la conexión de JTA en Glassfish. Levantamos Glassfish:



PASO 17. CONFIGURACION CONEXIÓN JTA

Entramos a la consola de administración de Glassfish:



PASO 17. CONFIGURACION CONEXIÓN JTA

Creamos un nuevo pool de conexiones:

The screenshot shows the GlassFish Admin Console interface. The title bar reads "JDBC Connection Pools" and the URL is "localhost:4848/common/index.jsf". The top navigation bar includes "Home", "About...", "User: admin", "Domain: domain1", "Server: localhost", and "Help". The main content area is titled "JDBC Connection Pools" with a sub-instruction: "To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection." Below this is a table titled "Pools (3)" with columns: Select, Pool Name, Resource Type, Classname, and Description. The table lists three existing pools: DerbyPool (javax.sql.DataSource, org.apache.derby.jdbc.ClientDataSource), SamplePool (javax.sql.DataSource, org.apache.derby.jdbc.ClientDataSource), and TimerPool (javax.sql.XADatasource, org.apache.derby.jdbc.EmbeddedXADataSource). A red arrow points from step 4 to the "New..." button in the toolbar above the table. The left sidebar, titled "Tree", shows the following structure with numbered steps: 1. Resources (highlighted with a red circle), 2. JDBC (highlighted with a red circle), 3. JDBC Connection Pools (highlighted with a red circle and selected), and 4. New... (the target of the red arrow).

Select	Pool Name	Resource Type	Classname	Description
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADataSource	

PASO 17. CONFIGURACION CONEXIÓN JTA

Proporcionamos los siguientes datos para crear el pool de conexiones:

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

* Indicates required field

General Settings

Pool Name: * PersonaPool

Resource Type: javax.sql.ConnectionPoolDataSource

Must be specified if the datasource class implements more than 1 of the interface.

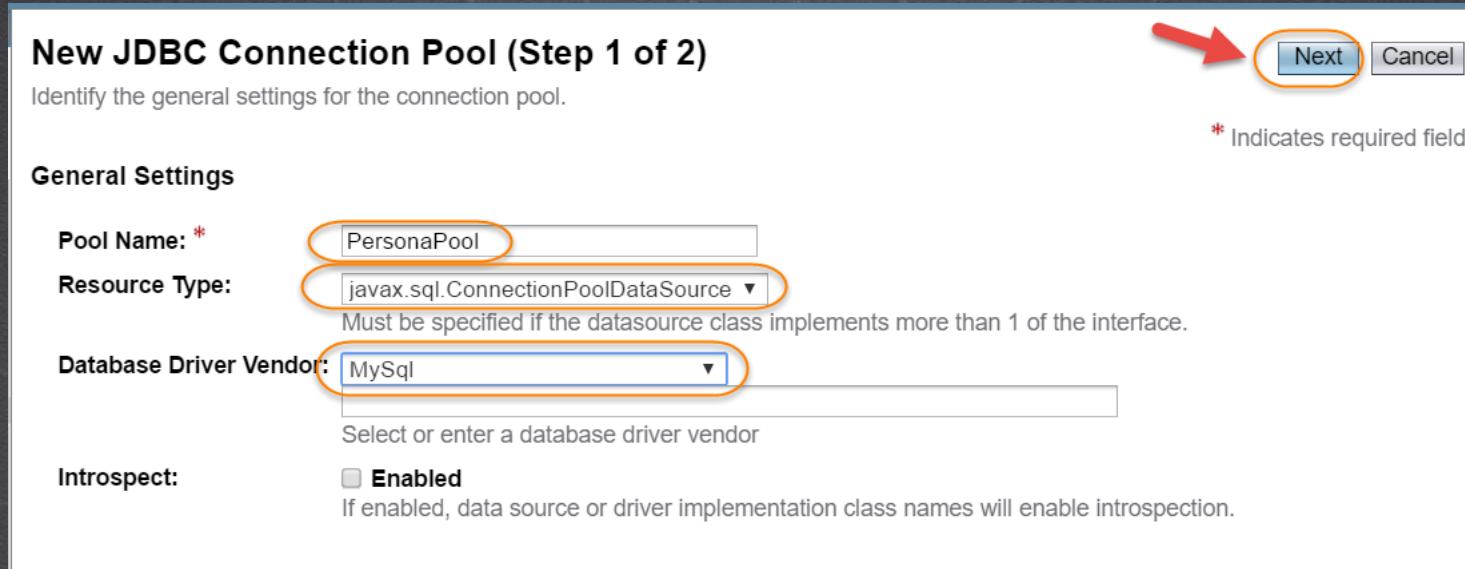
Database Driver Vendor: MySql

Select or enter a database driver vendor

Introspect: Enabled

If enabled, data source or driver implementation class names will enable introspection.

Next **Cancel**



PASO 17. CONFIGURACION CONEXIÓN JTA

Dejamos los valores por default:

New JDBC Connection Pool (Step 2 of 2)

Identify the general settings for the connection pool. Datasource Classname or Driver Classname must be specified for the connection pool.

* Indicates required field

General Settings

Pool Name: PersonaPool

Resource Type: javax.sql.ConnectionPoolDataSource

Database Driver Vendor: MySql

Datasource Classname: com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource ▾

Select or enter vendor-specific classname that implements the DataSource and/or XADatasource APIs

Driver Classname: ▾

Select or enter vendor-specific classname that implements the java.sql.Driver interface.

Ping: Enabled

When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Description:

Previous Finish Cancel

PASO 17. CONFIGURACION CONEXIÓN JTA

Dejamos los valores por default:

Pool Settings

Initial and Minimum Pool Size: Connections
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: Connections
Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity: Connections
Number of connections to be removed when pool idle timeout expires

Idle Timeout: Seconds
Maximum time that connection can remain idle in the pool

Max Wait Time: Milliseconds
Amount of time caller waits before connection timeout is sent

Transaction

Non Transactional Connections: Enabled
Returns non-transactional connections

Transaction Isolation:
If unspecified, use default level for JDBC Driver

Isolation Level: Guaranteed
All connections use same isolation level; requires Transaction Isolation

PASO 17. CONFIGURACION CONEXIÓN JTA

Borramos todas las propiedades:

The screenshot shows a table titled "Additional Properties (227)". The table has columns for "Name", "Value", and "Description". A red circle labeled "1" is on the checkbox column header. A red circle labeled "2" is on the "Delete Properties" button at the top right of the table. A red arrow points from the text above to this button.

	Name	Value	Description
<input checked="" type="checkbox"/>	SelfDestructOnPingSecondsLifetime	0	
<input checked="" type="checkbox"/>	UseUsageAdvisor	false	
<input checked="" type="checkbox"/>	AllowSlaveDownConnections	false	
<input checked="" type="checkbox"/>	LoadBalanceBlacklistTimeout	0	
<input checked="" type="checkbox"/>	QueryTimeoutKillsConnection	false	
<input checked="" type="checkbox"/>	CacheServerConfiguration	false	
<input checked="" type="checkbox"/>	RoundRobinLoadBalance	false	
<input checked="" type="checkbox"/>	ClientCertificateKeyStoreUrl		
<input checked="" type="checkbox"/>	UseCursorFetch	false	
<input checked="" type="checkbox"/>	JdbcCompliantTruncation	true	
<input checked="" type="checkbox"/>	UseOnlyServerErrorMessages	true	
<input checked="" type="checkbox"/>	AllowPublicKeyRetrieval	false	
<input checked="" type="checkbox"/>	DefaultAuthenticationPlugin	com.mysql.jdbc.authentication.MysqlNativePasswor	
<input checked="" type="checkbox"/>	ExceptionInterceptors		
<input checked="" type="checkbox"/>	DontTrackOpenResources	false	
<input checked="" type="checkbox"/>	UseInformationSchema	false	
<input checked="" type="checkbox"/>	UseNoneForElapsedTimo	false	

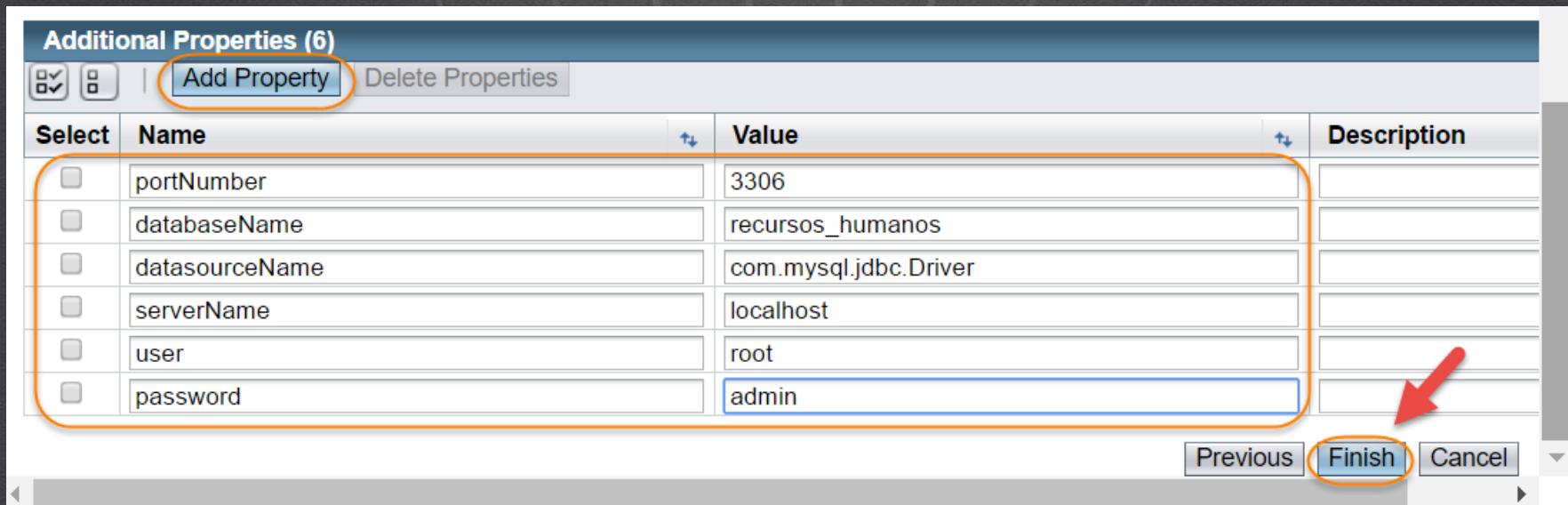
PASO 17. CONFIGURACION CONEXIÓN JTA

Agregamos las siguientes propiedades:

portNumber	3306
databaseName	recursos_humanos
datasourceName	com.mysql.jdbc.Driver
serverName	localhost
user	root
password	admin

PASO 17. CONFIGURACION CONEXIÓN JTA

Agregamos las siguientes propiedades, proporcionando los valores mostrados y después damos click en Finish:



PASO 17. CONFIGURACION CONEXIÓN JTA

Verificamos la conexión a mysql desde Glassfish:

The screenshot shows the GlassFish Admin Console interface. The title bar reads "JDBC Connection Pools". The URL in the address bar is "localhost:4848/common/index.jsf". The page header includes "User: admin | Domain: domain1 | Server: localhost" and "GlassFish™ Server Open Source Edition". On the left, a tree view under "Resources" shows "JDBC Connection Pools" highlighted with an orange oval and a red arrow pointing to it. The main content area is titled "JDBC Connection Pools" with a sub-section "Pools (4)". A table lists four connection pools:

Select	Pool Name	Resource Type	Classname	Description
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input checked="" type="checkbox"/>	PersonaPool	javax.sql.ConnectionPoolDataSource	com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource	
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	_TimerPool	javax.sql.XADatasource	org.apache.derby.jdbc.EmbeddedXADatasource	

PASO 17. CONFIGURACION CONEXIÓN JTA

Verificamos la conexión a mysql desde Glassfish:

The screenshot shows the GlassFish Admin Console interface. The title bar says "Edit JDBC Connection Po" and "localhost:4848/common/index.jsf". The URL in the address bar is "localhost:4848/common/index.jsf". The top menu has "Home", "About...", "User: admin | Domain: domain1 | Server: localhost", and "Help". On the left, a tree view shows "Server (Admin Server)", "Clusters", "Standalone Instances", "Nodes", "Applications", "Lifecycle Modules", "Monitoring Data", "Resources", "Concurrent Resources", "Connectors", "JDBC", "JDBC Resources", "JDBC Connection Pools", "DerbyPool", "PersonaPool" (which is selected and highlighted in blue), "SamplePool", and "TimerPool". The main content area has tabs "General", "Advanced", and "Additional Properties". The "General" tab is selected. The title of the form is "Edit JDBC Connection Pool". The sub-instruction says "Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database." Below are several configuration fields:

- Load Defaults**: A button.
- Flush**: A button.
- Ping**: A button, which is circled in yellow and has a red arrow pointing to it from the left.

* Indicates required field

General Settings

Pool Name: PersonaPool

Resource Type: javax.sql.ConnectionPoolDataSource ▾
Must be specified if the datasource class implements more than 1 of the interface.

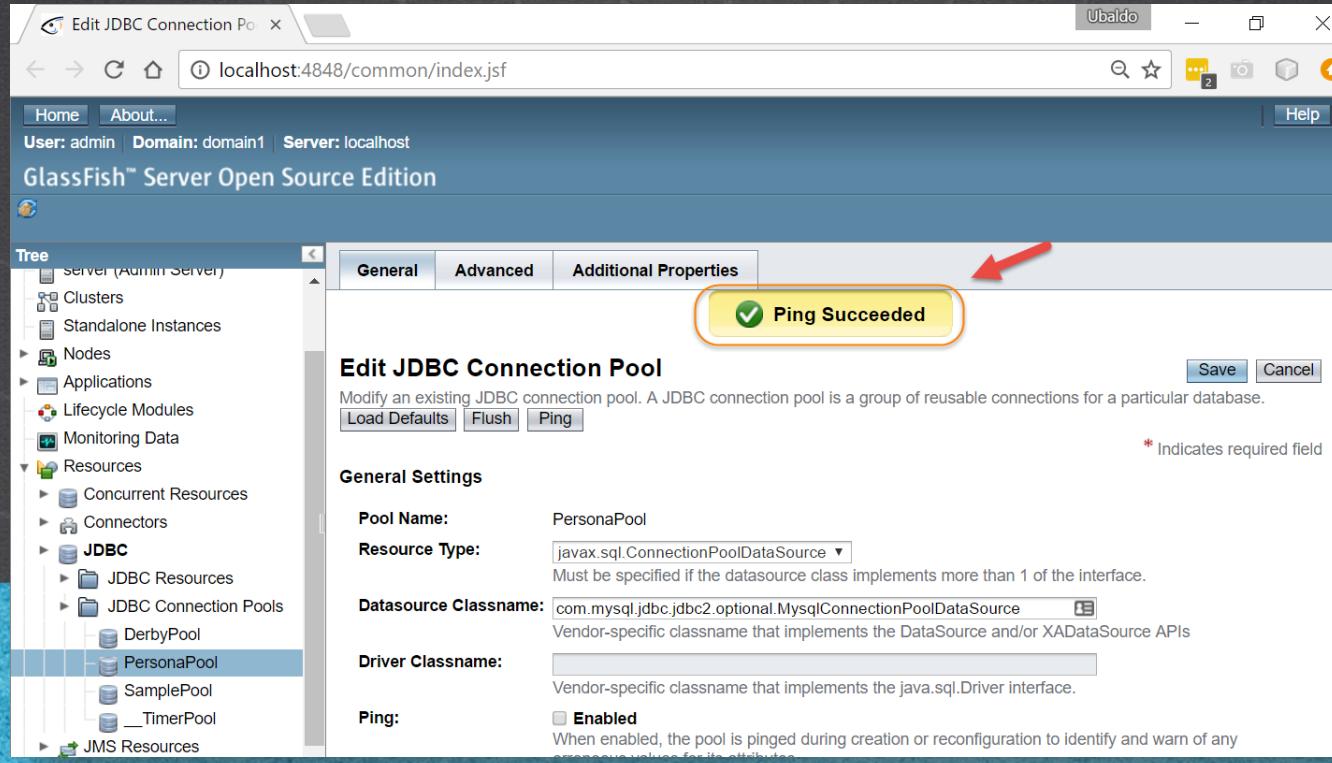
Datasource Classname: com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource ▾
Vendor-specific classname that implements the DataSource and/or XADatasource APIs

Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.

Ping: Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

PASO 17. CONFIGURACION CONEXIÓN JTA

Verificamos la conexión a mysql desde Glassfish. Si hace ping la conexión ha sido exitosa:



PASO 18. CONFIGURACION CONEXIÓN JTA

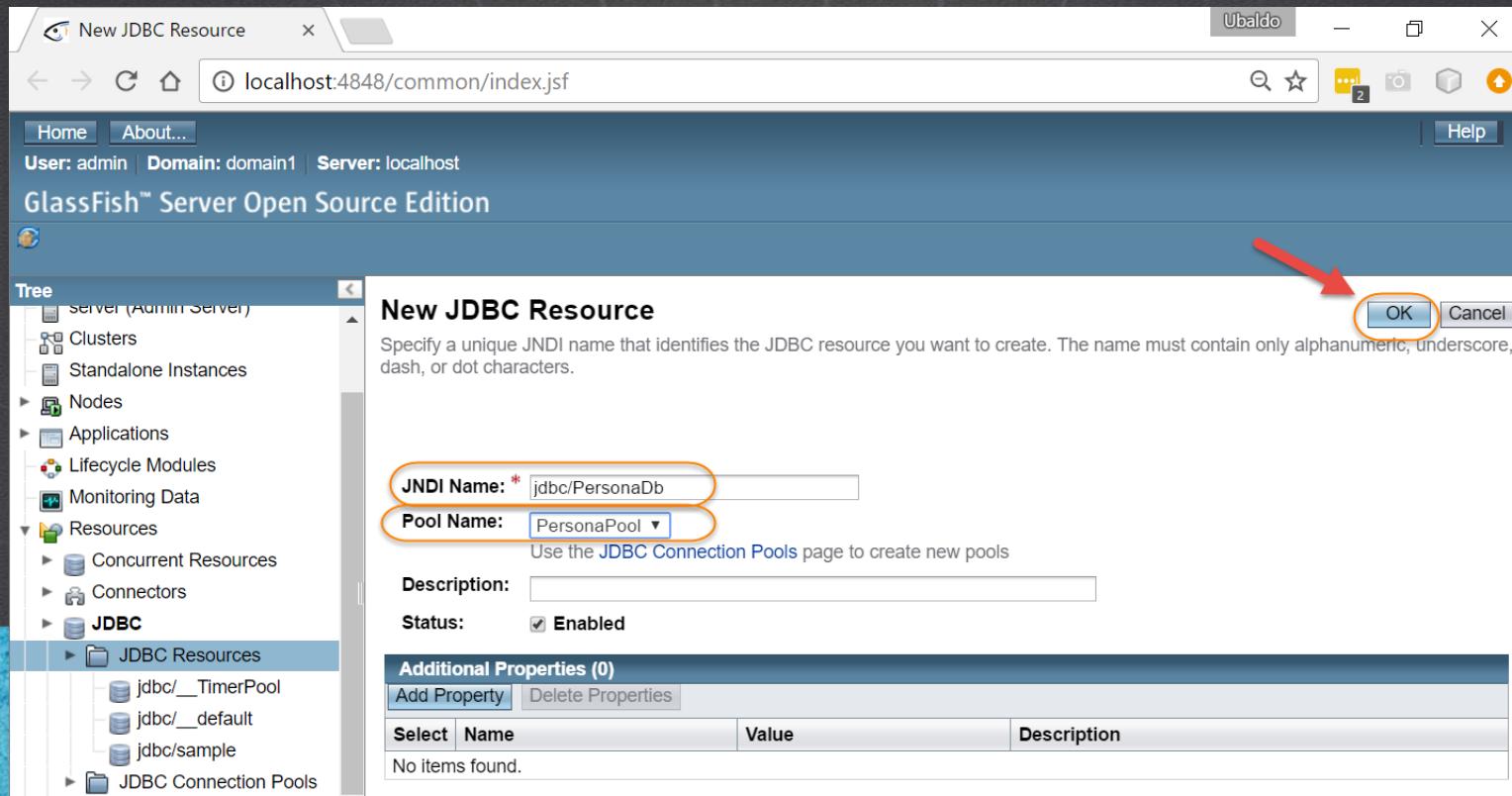
Creamos ahora el recurso de JDBC:

The screenshot shows the GlassFish Admin Console interface. The title bar reads "JDBC Resources". The URL in the address bar is "localhost:4848/common/index.jsf". The top navigation bar includes "Home", "About...", "User: admin | Domain: domain1 | Server: localhost", and "Help". On the left, a tree view under "Tree" shows "server (Admin Server)" and "Resources" expanded, with "JDBC" selected (marked with a red circle labeled 1). Below it, "JDBC Resources" is selected (marked with a red circle labeled 2). The main content area is titled "JDBC Resources" and contains the message "JDBC resources provide applications with a means to connect to a database." A table lists three existing resources: "jdbc/_TimerPool", "jdbc/_default", and "jdbc/sample". The table columns are "Select", "JNDI Name", "Logical JNDI Name", "Enabled", "Connection Pool", and "Description". The "New..." button in the toolbar is highlighted with a red circle labeled 3 and a red arrow pointing to it.

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool	Description
<input type="checkbox"/>	jdbc/_TimerPool		✓	__TimerPool	
<input type="checkbox"/>	jdbc/_default	java:comp/DefaultDataSource	✓	DerbyPool	
<input type="checkbox"/>	jdbc/sample		✓	SamplePool	

PASO 18. CONFIGURACION CONEXIÓN JTA

Creamos ahora el recurso de JDBC:



PASO 18. CONFIGURACION CONEXIÓN JTA

Con esto ya tenemos la conexión de JDBC y el pool de conexiones de MySQL y podemos utilizarlo para conectarnos desde nuestra aplicación de Java vía JTA (Java Transaction API).

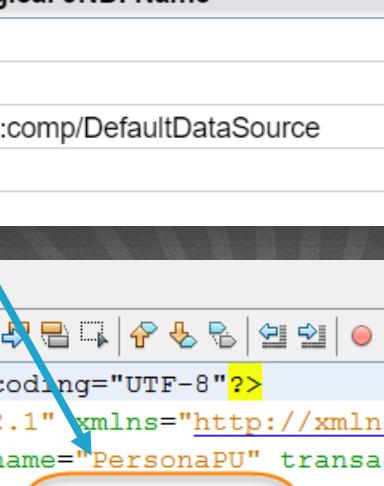
The screenshot shows the GlassFish Admin Console interface. The title bar says "JDBC Resources". The URL in the browser is "localhost:4848/common/index.jsf". The page header includes "User: admin | Domain: domain1 | Server: localhost" and "GlassFish™ Server Open Source Edition". The left sidebar has a tree view with nodes like Server (Admin Server), Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources, Concurrent Resources, Connectors, and JDBC. Under JDBC, "JDBC Resources" is selected. The main content area is titled "JDBC Resources" with the sub-instruction "JDBC resources provide applications with a means to connect to a database." Below this is a table titled "Resources (4)". The table columns are Select, JNDI Name, Logical JNDI Name, Enabled, Connection Pool, and Description. The rows show:

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool	Description
<input checked="" type="checkbox"/>	jdbc/PersonaDb		<input checked="" type="checkbox"/>	PersonaPool	
<input checked="" type="checkbox"/>	jdbc/__TimerPool		<input checked="" type="checkbox"/>	__TimerPool	
<input checked="" type="checkbox"/>	jdbc/__default	java:comp/DefaultDataSource	<input checked="" type="checkbox"/>	DerbyPool	
<input checked="" type="checkbox"/>	jdbc/sample		<input checked="" type="checkbox"/>	SamplePool	

PASO 18. CONFIGURACION CONEXIÓN JTA

Podemos observar que el mismo nombre configurado en Glassfish, es el nombre usado en el archivo persistence.xml:

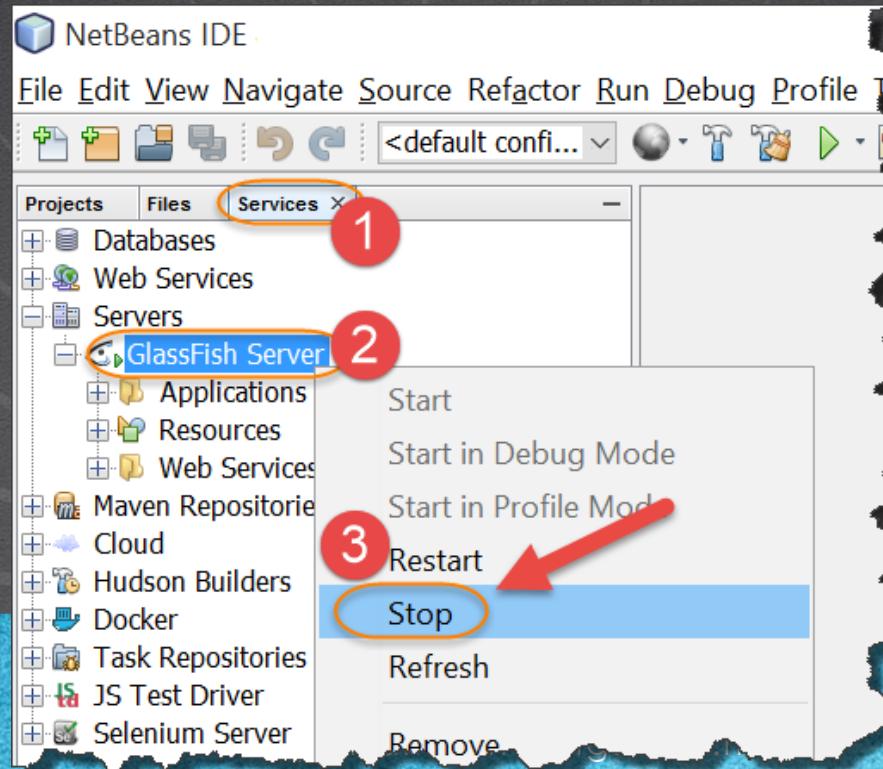
Resources (4)					
Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool	
<input type="checkbox"/>	jdbc/PersonaDb		✓	PersonaPool	
<input type="checkbox"/>	jdbc/_TimerPool		✓	_TimerPool	
<input type="checkbox"/>	jdbc/_default	java:comp/DefaultDataSource	✓	DerbyPool	
<input type="checkbox"/>	jdbc/sample		✓	SamplePool	



```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <persistence-unit name="personaPU" transaction-type="JTA">
        <jta-data-source>jdbc/PersonaDb</jta-data-source>
    </persistence-unit>
</persistence>
```

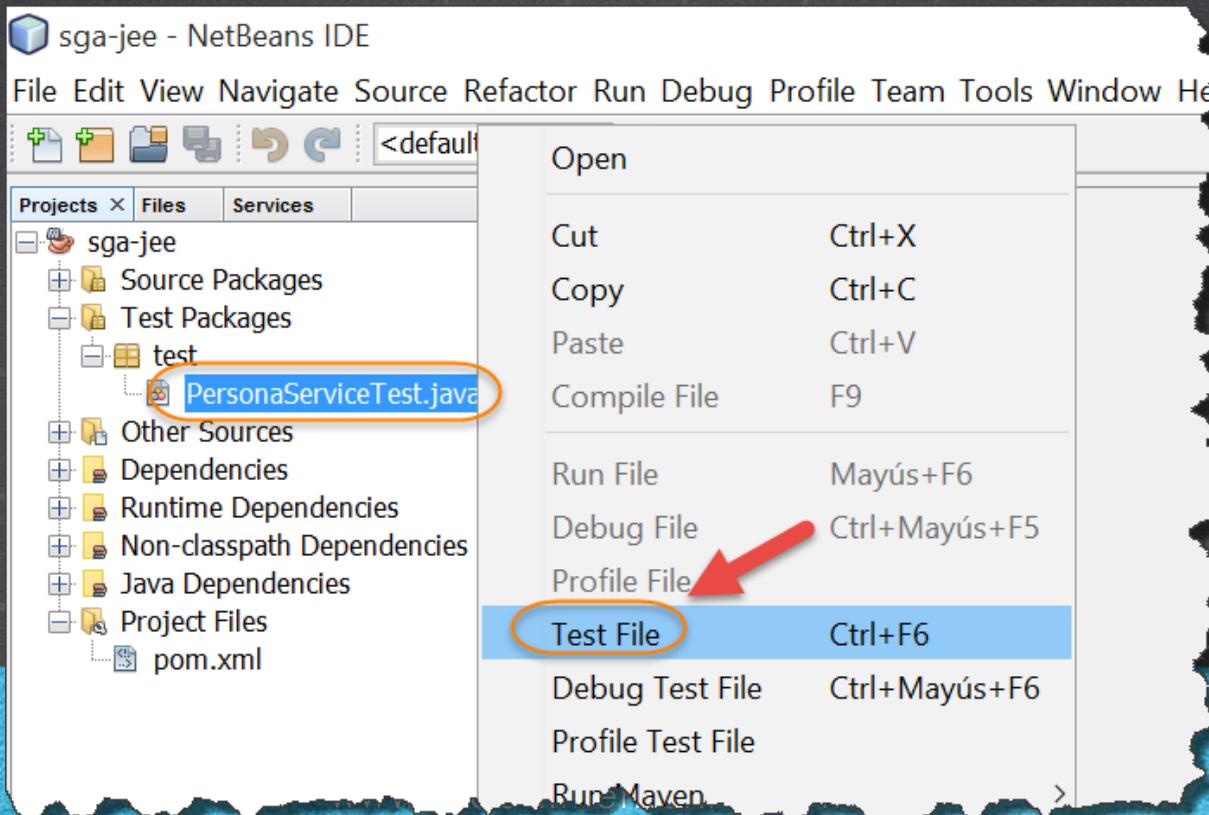
PASO 19. CONFIGURACION CONEXIÓN JTA

Antes de ejecutar la prueba, debemos detener el servidor GlassFish si es que estuviera en modo Start, ya que el contenedor embebido utiliza la misma JVM.



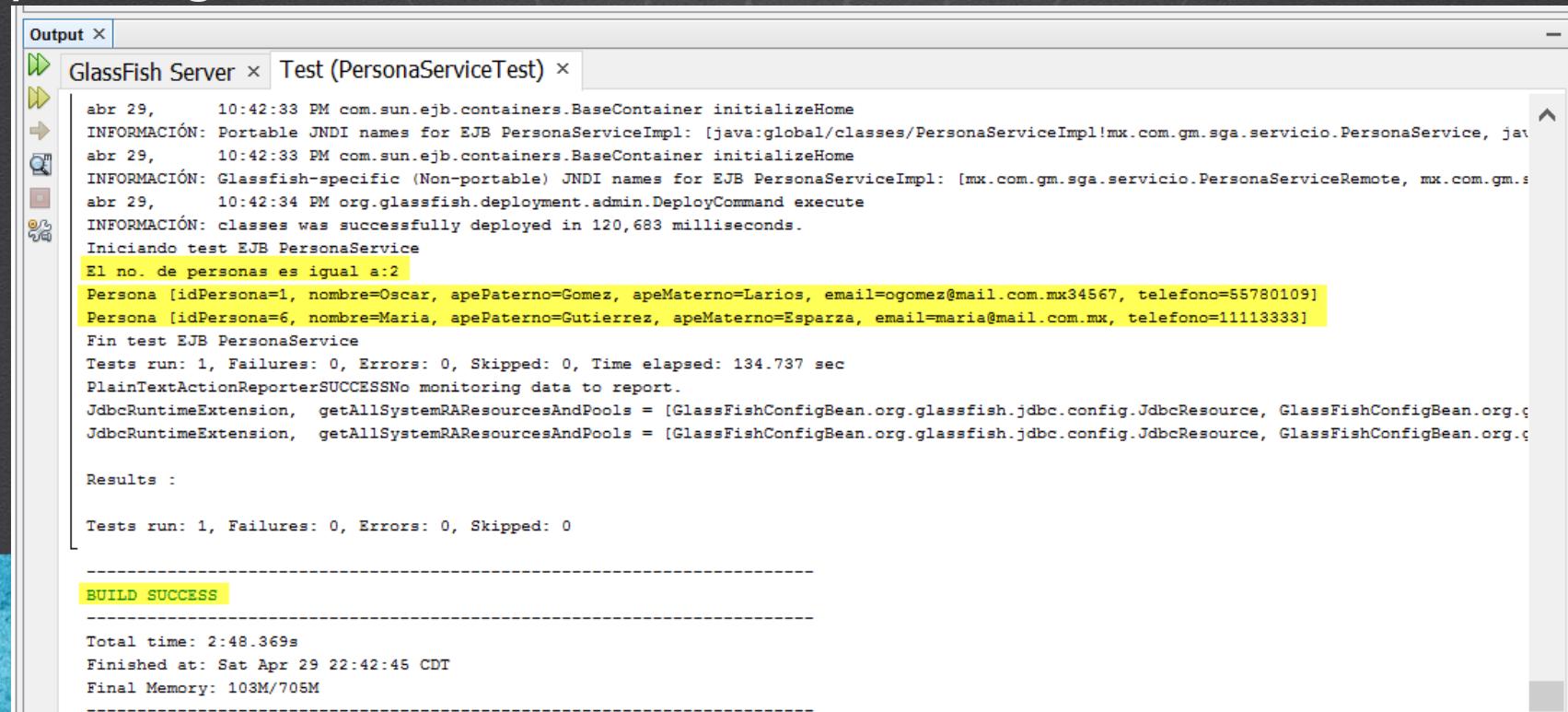
PASO 20. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:



PASO 20. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto. El test depende del no. de personas que tengamos en la base de datos:



The screenshot shows the 'Output' window of an IDE, specifically for a GlassFish Server deployment and a PersonaServiceTest.

GlassFish Server x Test (PersonaServiceTest) x

```
abr 29, 10:42:33 PM com.sun.ejb.containers.BaseContainer initializeHome
INFORMACIÓN: Portable JNDI names for EJB PersonaServiceImpl: [java:global/classes/PersonaServiceImpl!mx.com.gm.sga.servicio.PersonaService, java:global/classes/PersonaServiceImpl!mx.com.gm.sga.servicio.PersonaServiceRemote]
abr 29, 10:42:33 PM com.sun.ejb.containers.BaseContainer initializeHome
INFORMACIÓN: Glassfish-specific (Non-portable) JNDI names for EJB PersonaServiceImpl: [mx.com.gm.sga.servicio.PersonaServiceRemote, mx.com.gm.sga.servicio.PersonaService]
abr 29, 10:42:34 PM org.glassfish.deployment.admin.DeployCommand execute
INFORMACIÓN: classes was successfully deployed in 120,683 milliseconds.

Iniciando test EJB PersonaService
El no. de personas es igual a:2
Persona [idPersona=1, nombre=Oscar, apePaterno=Gomez, apeMaterno=Larios, email=o Gomez@mail.com.mx34567, telefono=55780109]
Persona [idPersona=6, nombre=Maria, apePaterno=Gutierrez, apeMaterno=Esparza, email=maria@mail.com.mx, telefono=11113333]
Fin test EJB PersonaService
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 134.737 sec
PlainTextActionReporterSUCCESSNo monitoring data to report.
JdbcRuntimeExtension, getAllSystemRAResourcesAndPools = [GlassFishConfigBean.org.glassfish.jdbc.config.JdbcResource, GlassFishConfigBean.org.glassfish.jdbc.config.JdbcConnectionPool]
JdbcRuntimeExtension, getAllSystemRAResourcesAndPools = [GlassFishConfigBean.org.glassfish.jdbc.config.JdbcResource, GlassFishConfigBean.org.glassfish.jdbc.config.JdbcConnectionPool]

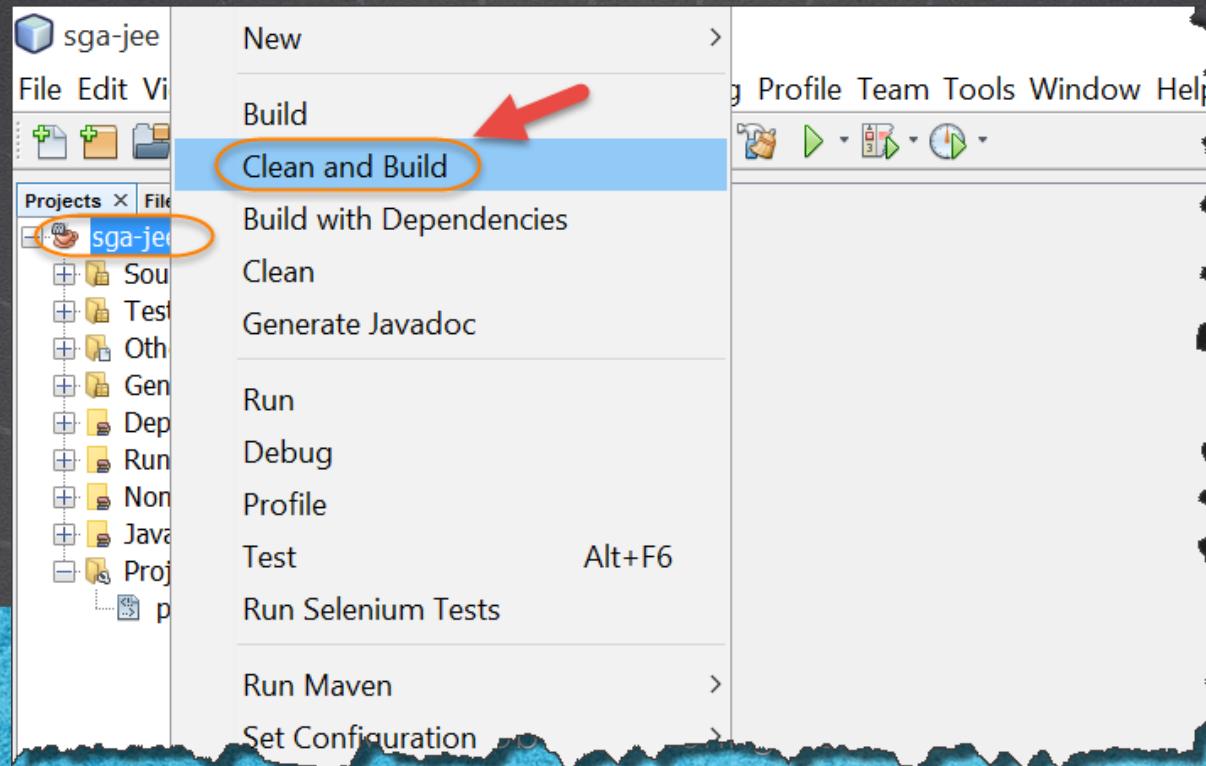
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 2:48.369s
Finished at: Sat Apr 29 22:42:45 CDT
Final Memory: 103M/705M
-----
```

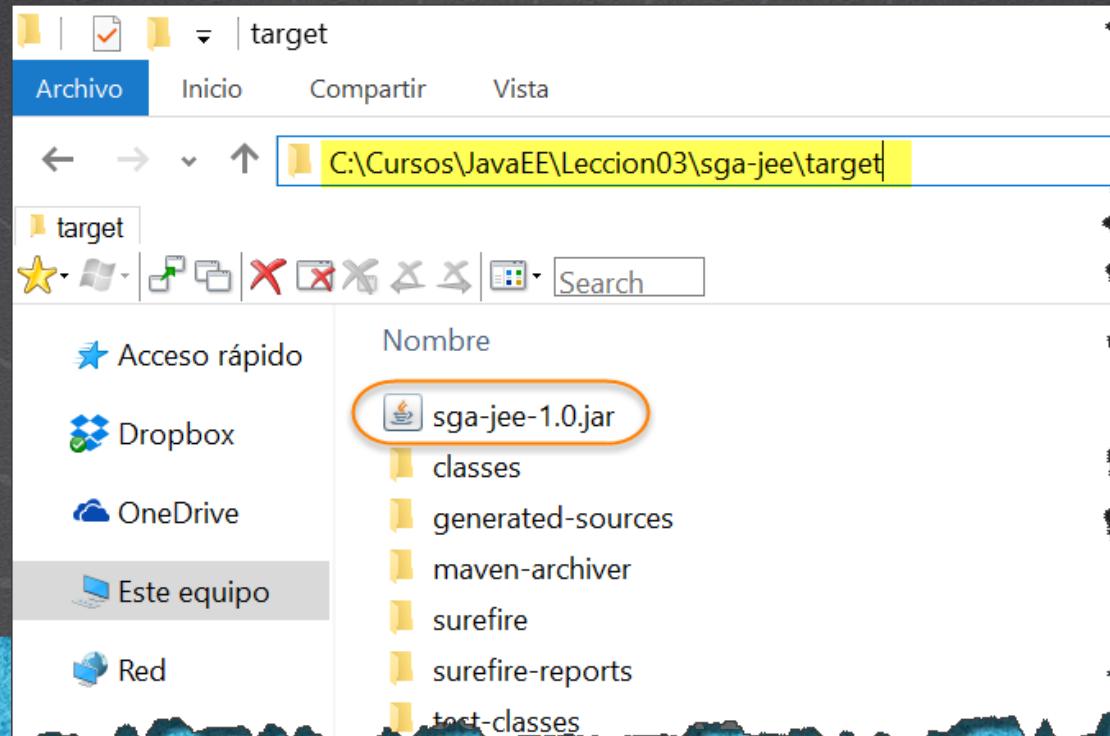
PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Empaquetamos el EJB en un archivo .jar, hacemos un clean & build para generar el archivo .jar. **Nota: Debe estar detenido el servidor GlassFish:**



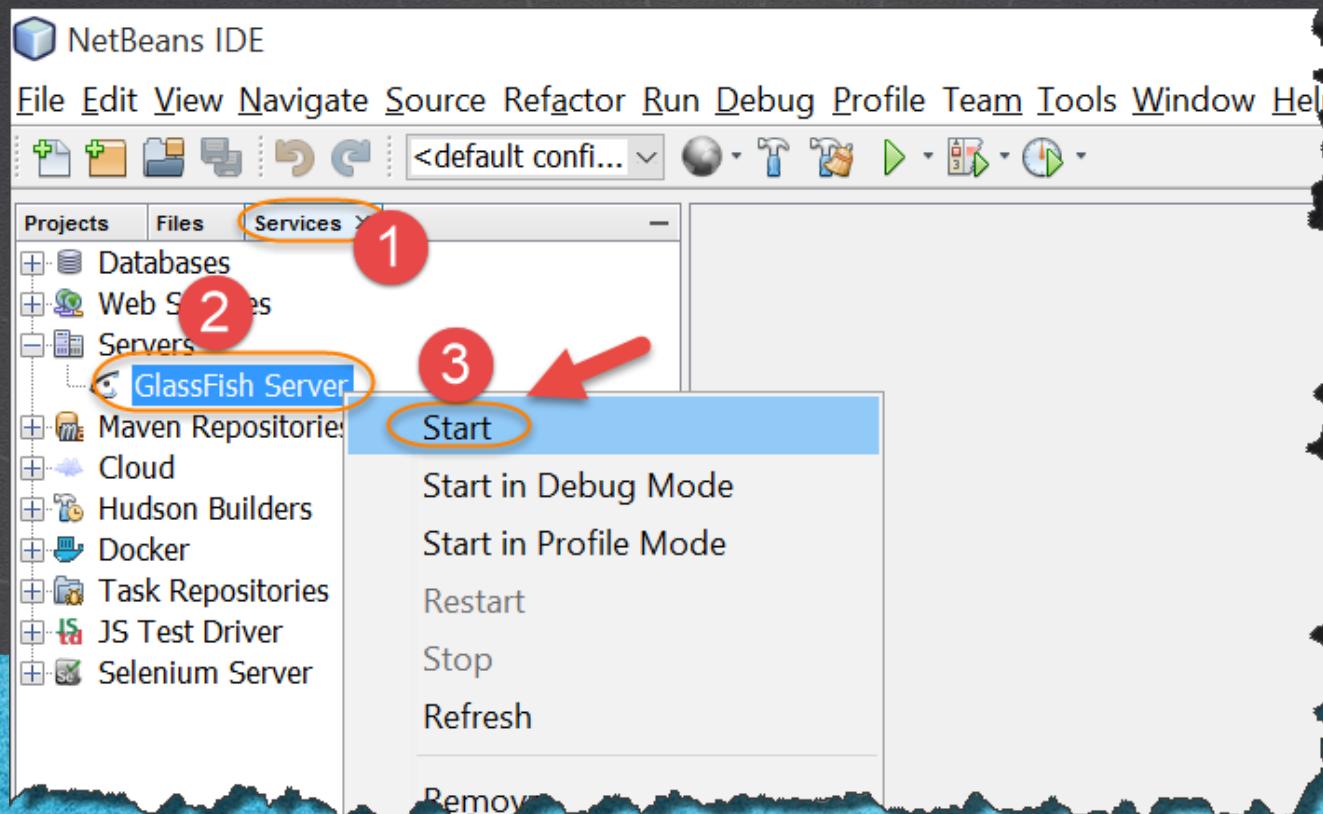
PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Se genera el archivo sga-jee-1.0.jar:



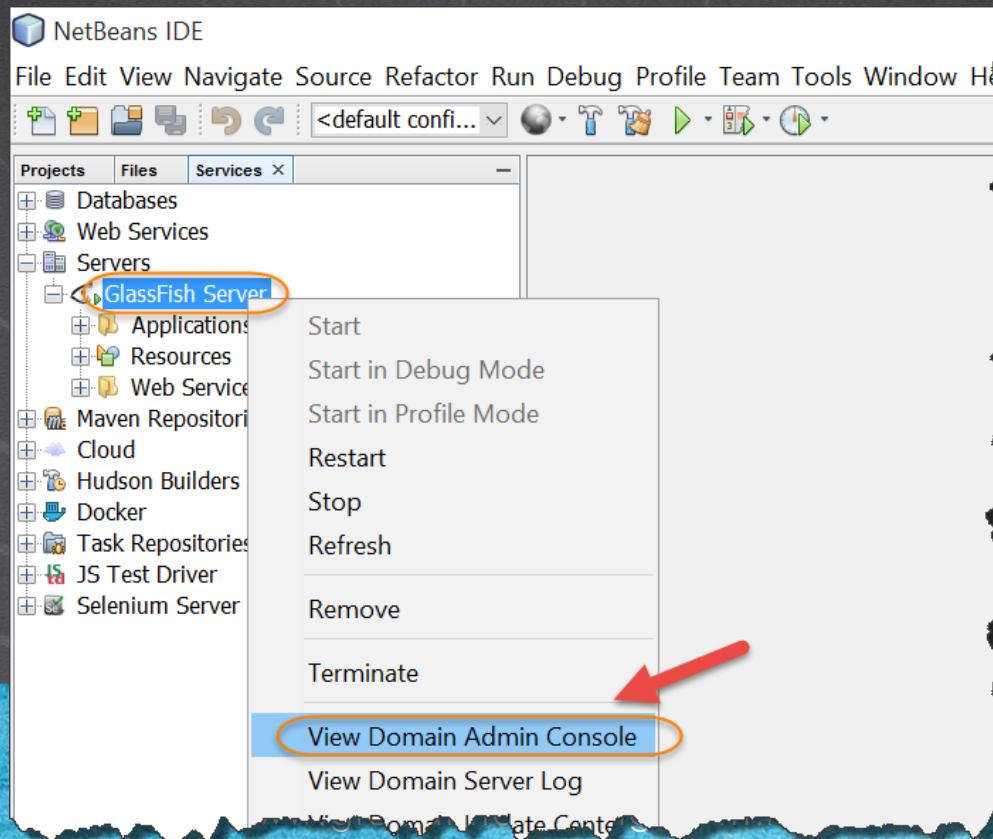
PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Levantamos el servidor de Glassfish:



PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Entramos a la consola de Glassfish:



PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Entramos a la aplicación y hacemos Redeploy en la aplicación de sga-jee:

The screenshot shows the GlassFish Admin Console interface. On the left, there's a sidebar with 'Common Tasks' and several navigation items: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes (with a red circle labeled 1), Applications (selected and highlighted with a blue bar, with a red circle labeled 1), Lifecycle Modules, Monitoring Data, and Resources.

The main content area is titled 'Applications'. It contains a brief description: 'Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.' Below this is a table titled 'Deployed Applications (2)'. The table has columns: Select, Name, Deployment Order, Enabled, Engines, and Action. There are two rows:

Select	Name	Deployment Order	Enabled	Engines	Action
<input type="checkbox"/>	HolaMundoJavaEE	100	<input checked="" type="checkbox"/>	web	Launch Redeploy Reload
<input checked="" type="checkbox"/>	sga-jee	100	<input checked="" type="checkbox"/>	ejb	Redeploy Reload

A red arrow points from the bottom right towards the 'Redeploy' link for the 'sga-jee' application row. A red circle labeled 2 is placed over the 'Redeploy' link for the 'sga-jee' row.

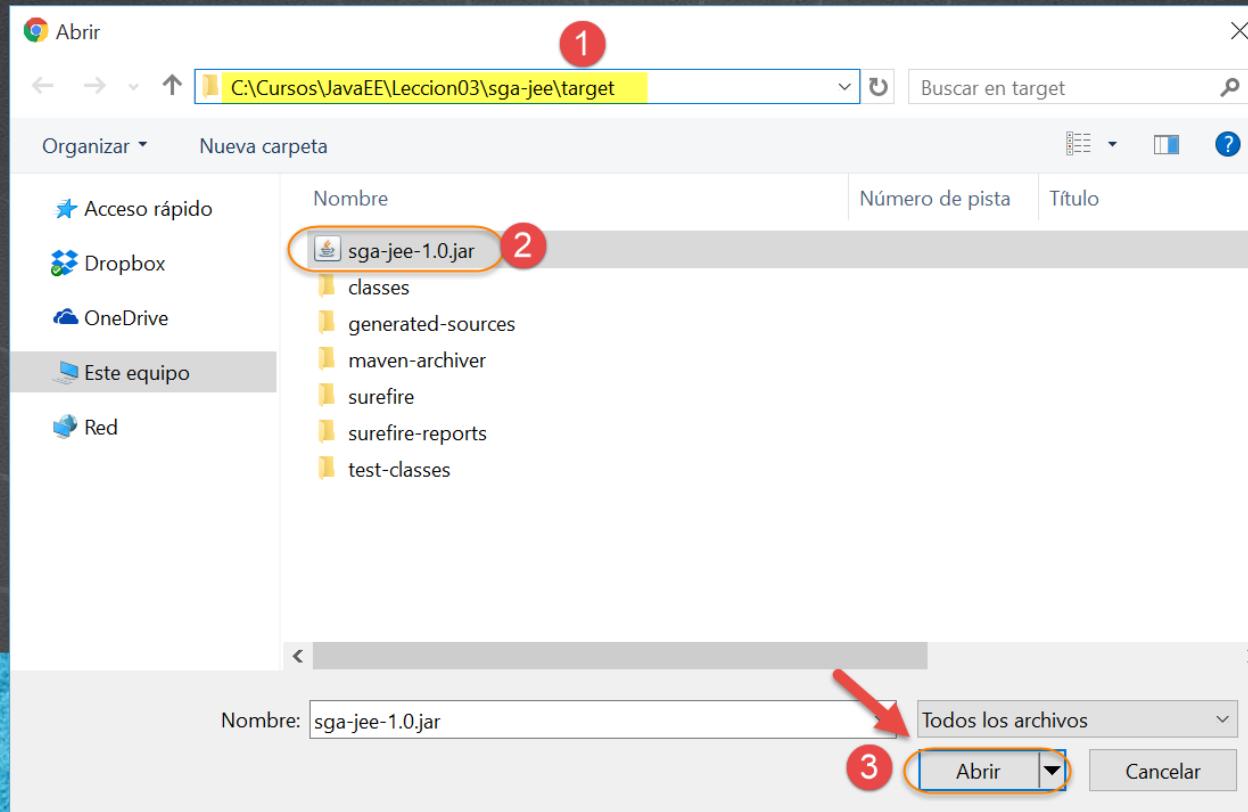
PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Seleccionamos el nuevo archivo para hacer redeploy:

The screenshot shows the GlassFish Server Open Source Edition interface. The left sidebar lists common tasks such as Domain, Clusters, Standalone Instances, Nodes, Applications (which is selected), Lifecycle Modules, Monitoring Data, Resources, Concurrent Resources, and Connectors. The main content area displays the 'Redeploy Applications or Modules' dialog. The dialog has two tabs: 'Packaged File to Be Uploaded to the Server' (selected) and 'Local Packaged File or Directory That Is Accessible from GlassFish Server'. A red arrow points to the 'Seleccionar archivo' button under the first tab, which is highlighted with an orange circle. The application name is set to 'sga-jee' and precompile JSPs is set to 'Enabled'. The top navigation bar shows the URL 'localhost:4848/common/index.jsf' and the user 'Ubaldo'.

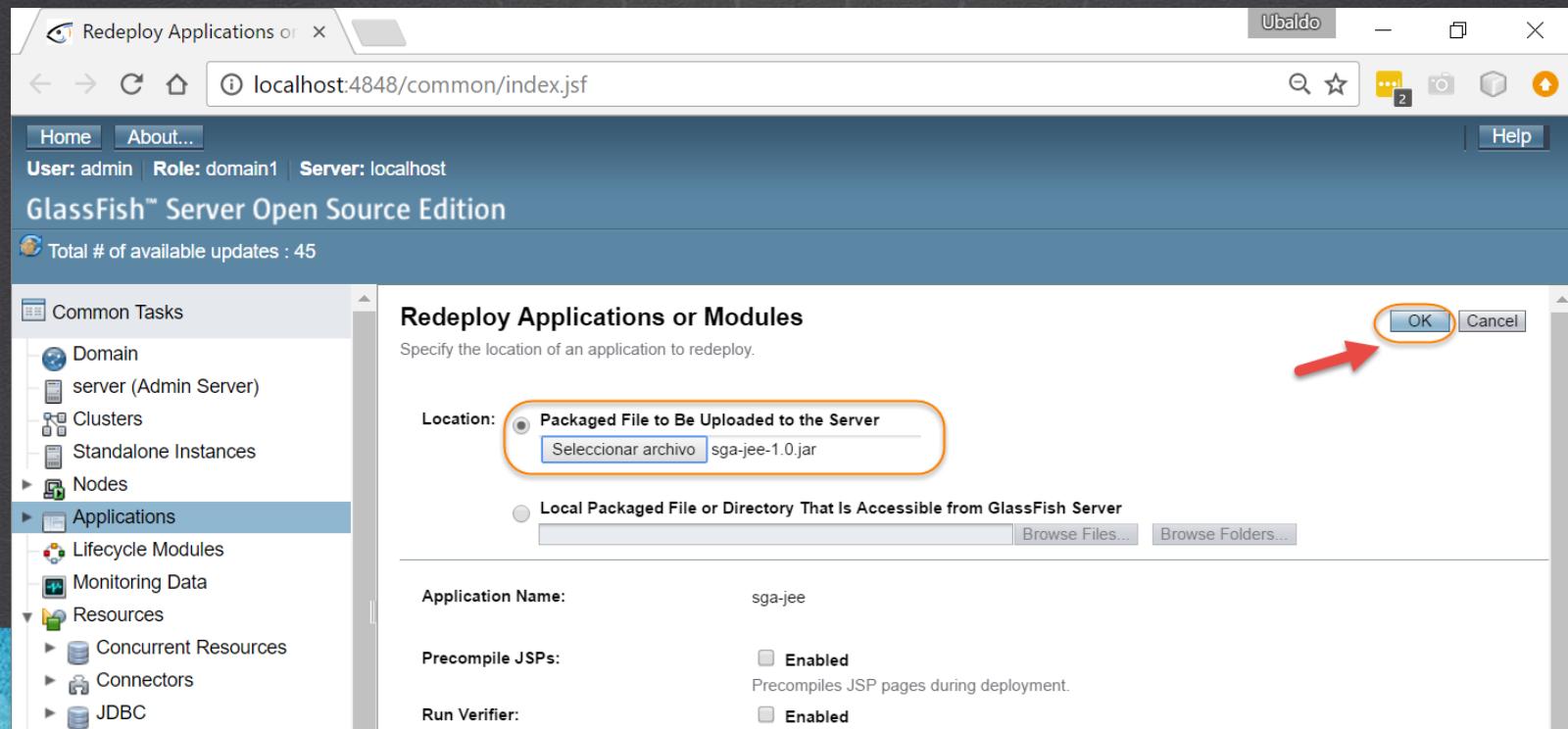
PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Seleccionamos el archivo sga-jee-1.0.jar:



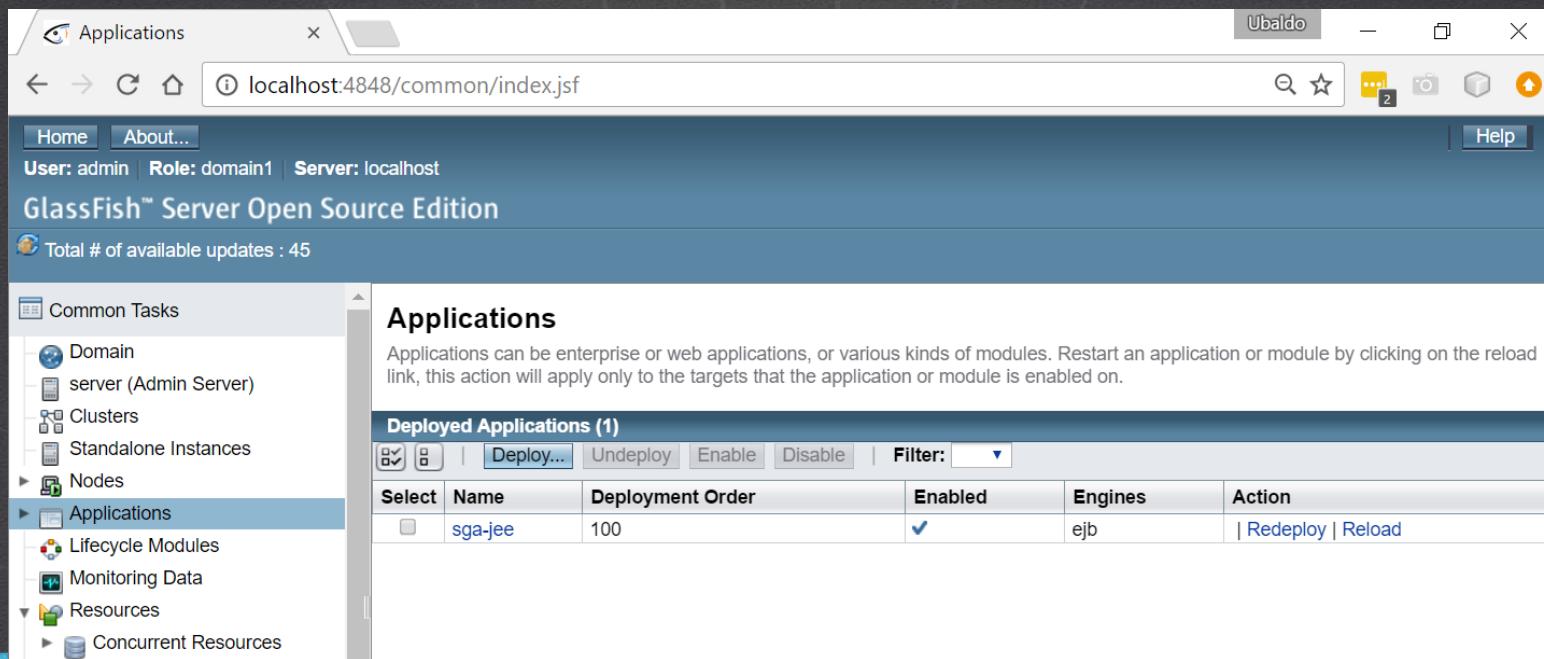
PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Seleccionamos el nuevo archivo para hacer redeploy:



PASO 21. EMPAQUETAMIENTO Y DESPLIEGUE EJB

Ya quedó nuevamente desplegada nuestra aplicación:

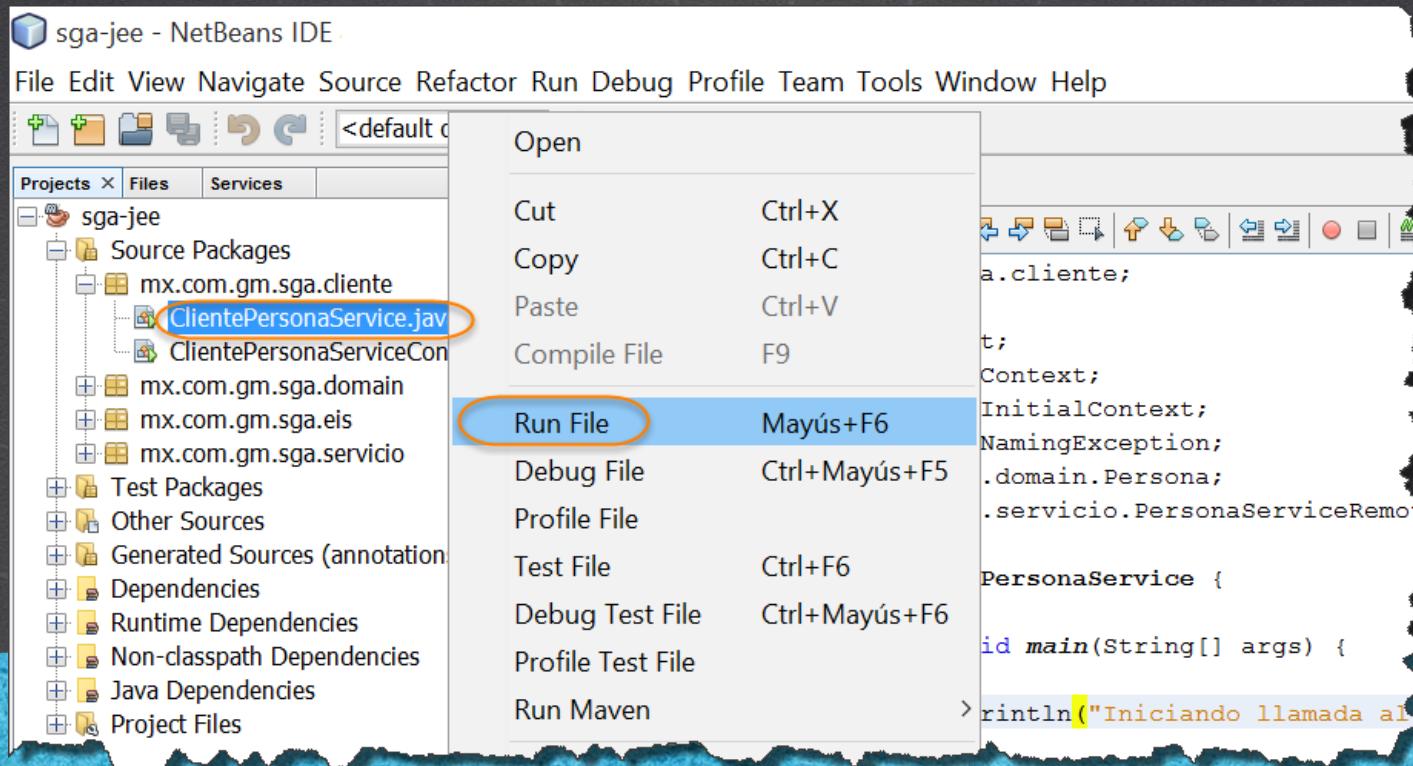


CURSO JAVA EE

www.globalmentoring.com.mx

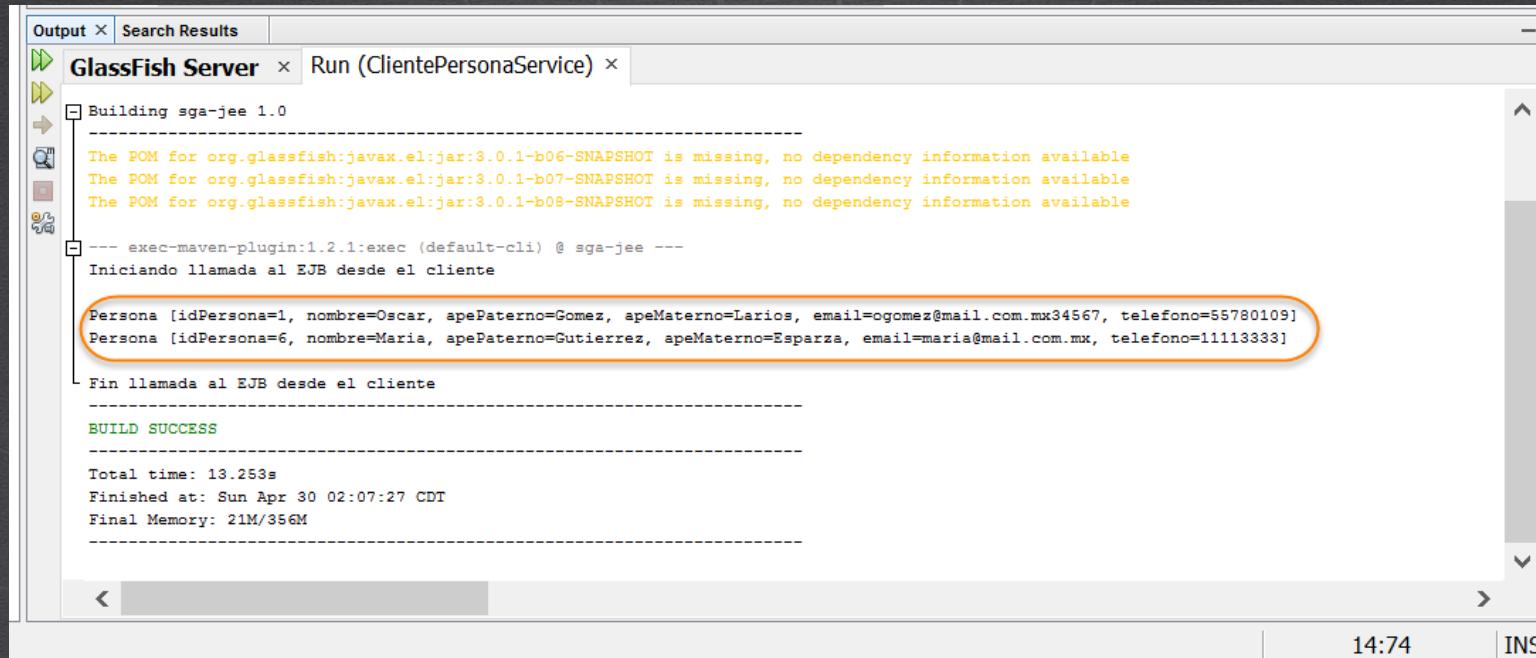
PASO 22. EJECUTAMOS EL CLIENTE EJB

Ejecutamos nuestro cliente EJB:



PASO 22. EJECUTAMOS EL CLIENTE EJB

Ejecutamos nuestro cliente EJB:



The screenshot shows the GlassFish Server Output window in a Java IDE. The title bar says "Output X Search Results" and "GlassFish Server x Run (ClientePersonaService) x". The main pane displays the build logs for "Building sga-jee 1.0". It shows Maven dependency errors for javax.el, followed by the command "exec-maven-plugin:1.2.1:exec (default-cli) @ sga-jee". The log then indicates "Iniciando llamada al EJB desde el cliente". Two Person objects are printed to the console, highlighted with an orange oval: "Persona [idPersona=1, nombre=Oscar, apePaterno=Gomez, apeMaterno=Larios, email=ogomez@mail.com.mx, telefono=55780109]" and "Persona [idPersona=6, nombre=Maria, apePaterno=Gutierrez, apeMaterno=Esparza, email=maria@mail.com.mx, telefono=11113333]". The log concludes with "Fin llamada al EJB desde el cliente", "BUILD SUCCESS", and build statistics: "Total time: 13.253s", "Finished at: Sun Apr 30 02:07:27 CDT", and "Final Memory: 21M/356M". The bottom status bar shows the time "14:74" and an "INS" indicator.

```
The POM for org.glassfish:javax.el:jar:3.0.1-b06-SNAPSHOT is missing, no dependency information available
The POM for org.glassfish:javax.el:jar:3.0.1-b07-SNAPSHOT is missing, no dependency information available
The POM for org.glassfish:javax.el:jar:3.0.1-b08-SNAPSHOT is missing, no dependency information available
--- exec-maven-plugin:1.2.1:exec (default-cli) @ sga-jee ---
Iniciando llamada al EJB desde el cliente

Persona [idPersona=1, nombre=Oscar, apePaterno=Gomez, apeMaterno=Larios, email=ogomez@mail.com.mx, telefono=55780109]
Persona [idPersona=6, nombre=Maria, apePaterno=Gutierrez, apeMaterno=Esparza, email=maria@mail.com.mx, telefono=11113333]

Fin llamada al EJB desde el cliente
BUILD SUCCESS
Total time: 13.253s
Finished at: Sun Apr 30 02:07:27 CDT
Final Memory: 21M/356M
```

CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos agregado JPA a nuestro ejercicio de sga-jee.
- De esta manera podemos comunicarnos con la base de datos de mysql, además de configurar nuestra conexión a base de datos utilizando JTA (Java Transaction API), el cual nos permite delegar los datos de conexión a Glassfish y así evitar configurar la conexión a base de datos desde nuestra aplicación.
- Posteriormente realizamos una prueba unitaria para verificar el funcionamiento de la integración de JPA y nuestro EJB de Sesión.
- Finalmente hicimos el deploy de nuestra aplicación a Glassfish usando el archivo .jar y ejecutamos la clase cliente para verificar el funcionamiento de la aplicación.

CURSO ONLINE

JAVA EMPRESARIAL

JAVA EE

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO JAVA EE

www.globalmentoring.com.mx