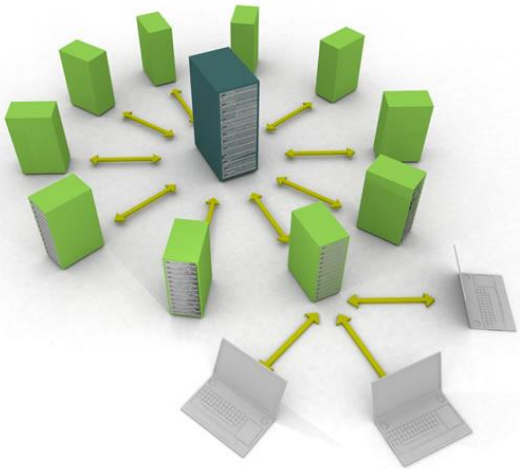
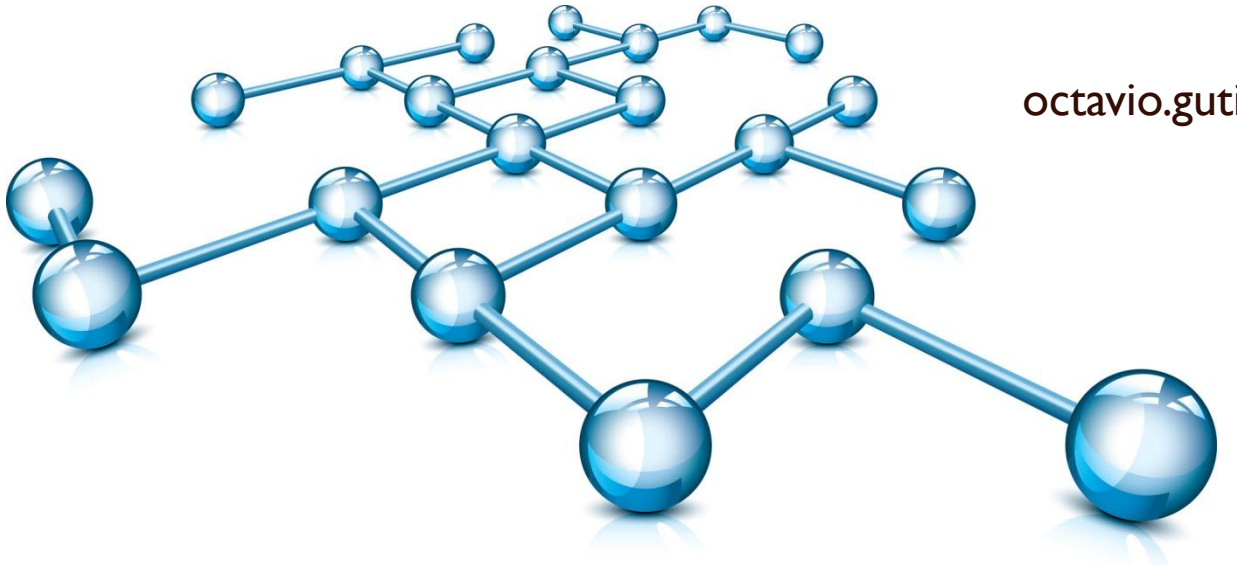


Sistemas de objetos y componentes distribuidos

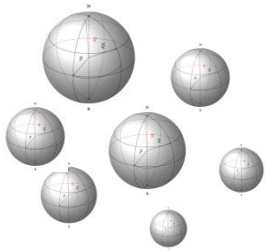
Profesor:
Dr. J. Octavio Gutiérrez García

octavio.gutierrez@itam.mx



Sistemas de objetos distribuidos y componentes

- La tarea de un middleware es proveer un alto nivel de abstracción en la programación de sistemas distribuidos para soportar heterogeneidad, interoperabilidad y portabilidad.



- Middleware de objetos distribuidos

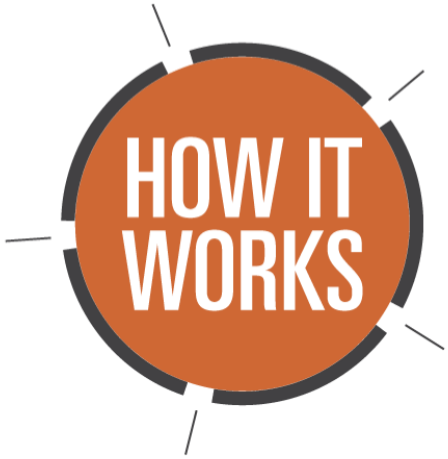
- Middleware basado en componentes

Middleware de objetos distribuidos

- Adopción de paradigma **orientado a objetos**.
- Objetos se comunican vía
  - RMI
  - Eventos
- Ejemplos: Java RMI y CORBA

Middleware de objetos distribuidos

- Referencias a objetos remotos
- Interfaces remotas
- Acciones distribuidas (RMI)
- Excepciones distribuidas, ej. pérdida de mensajes.
- Recolección distribuida de basura



Middleware de objetos distribuidos

Complejidad

- Comunicación inter-objetos.
- Gestión distribuida del ciclo de vida de los objetos.
- Activación y desactivación de objetos.
  - Persistencia del estado de los objetos
- Más servicios:
  - Servicios de nombres, seguridad, etc.

Middlewares basados en componentes

Desarrollados para superar limitantes de los middlewares de objetos distribuidos:

- Dependencias implícitas

¿Con qué objetos se comunican?

- Complejidad de programación

Los programadores son expuestos a cuestiones del middleware

- Falta de separación de aspectos de distribución: seguridad, manejo de fallas, concurrencia, etc.
- Sin soporte para el despliegue.

Middlewares basados en componentes

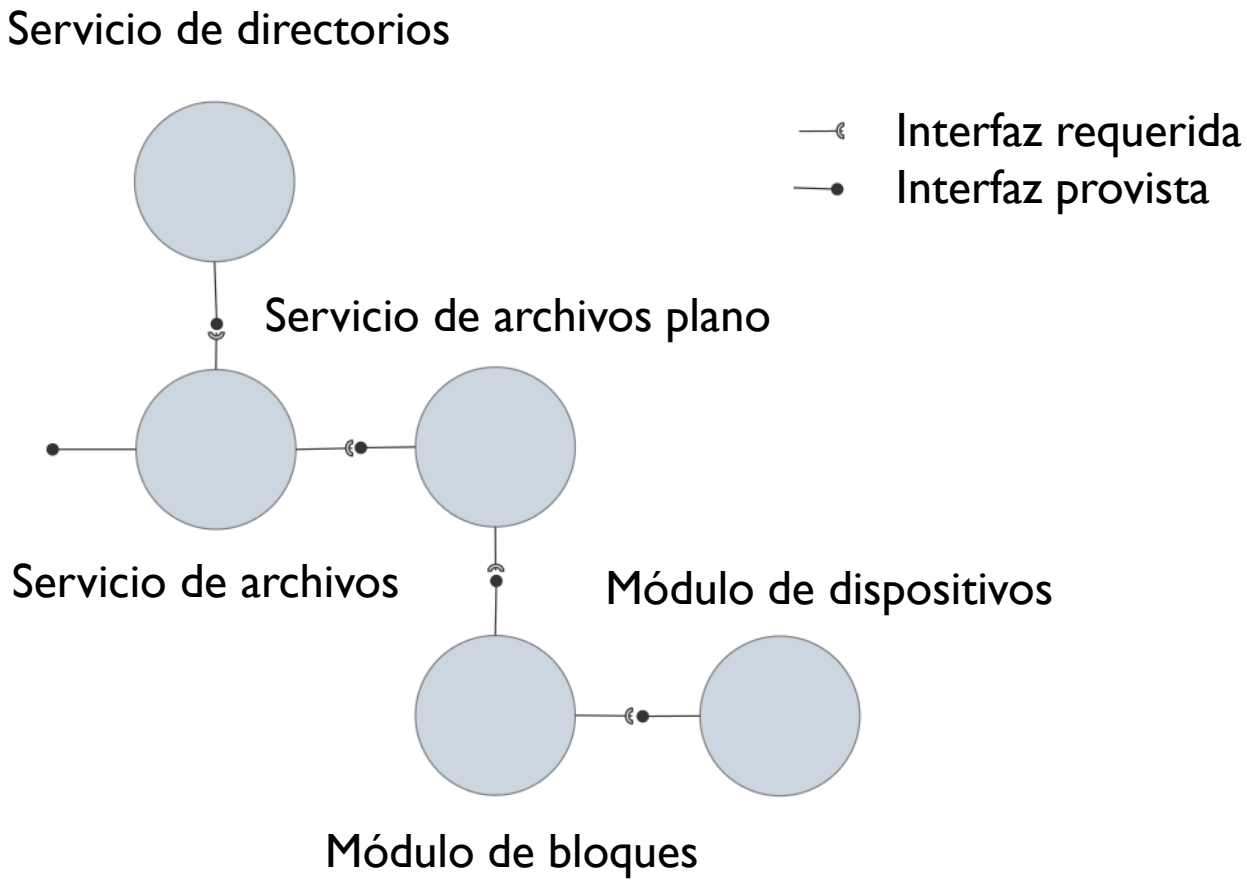
Un componente de software es una unidad de composición con interfaces especificadas contractualmente y dependencias explícitas.

Un componente se específica en términos de un contrato que incluye:

- Interfaces provistas

- Interfaces requeridas

Middlewares basados en componentes



Middlewares basados en componentes

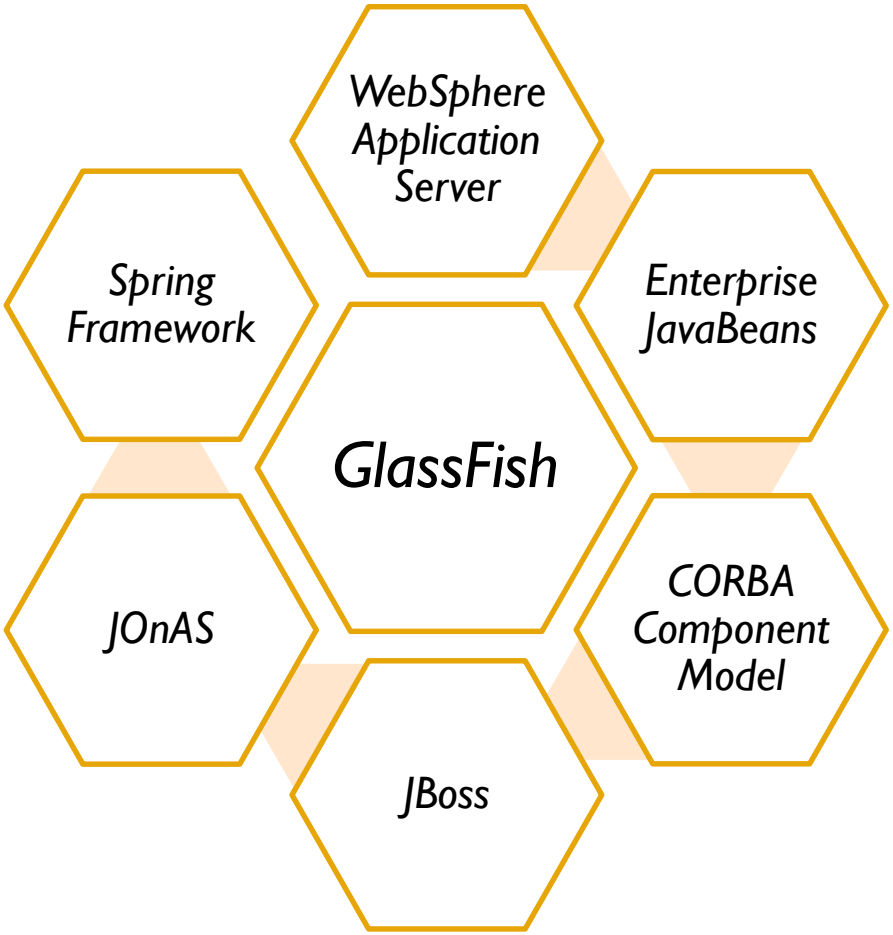
- Los contenedores proveen un patrón común, que consiste en:
  - Un front-end.
  - Un recipiente que contiene uno o más componentes que implementan la aplicación o la lógica de negocios.
  - Servicios del sistema que administran los datos asociados en almacenamiento persistente.

Middlewares basados en componentes

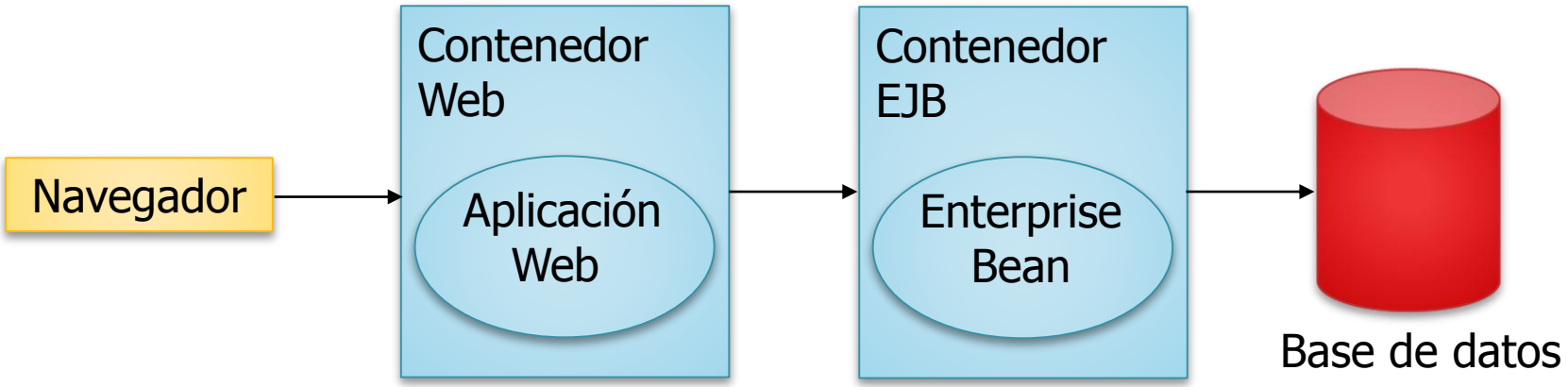
Los contenedores separan la funcionalidad de la aplicación (que implementan los componentes) de las cuestiones del middleware distribuido.

Middlewares basado en componentes

- Los middlewares que soportan el uso de componentes se conocen como servidores de aplicaciones



Enterprise Java Beans (EJB)



Arquitectura de 4 niveles

Enterprise Java Beans

Modelo de programación, constituido por convenciones/ protocolos y un conjunto de clases e interfaces sobre las cuales se apoya el EJB API.

Portabilidad

Convenciones
Protocolos
Interfaces

Especificaciones

Objetos distribuidos y componentes del lado del servidor

EJB Container

EJB

Enterprise Java Beans

- Tecnología para el desarrollo de componentes en la parte del servidor con la plataforma J2EE (Java EE)

- Los contenedores de EJBs soportan:
  - Transacciones
  - Seguridad
  - Persistencia

El objetivo de EJBs es mantener una fuerte separación de las preocupaciones entre los diversos roles implicados en el desarrollo de aplicaciones distribuidas.

¿Cuándo usar EJBs?

- No siempre, para simples aplicaciones web con servlets y JSPs es más que suficiente.

- Solución para alta disponibilidad y escalabilidad.
- Cuando el sistema debe manejar transacciones distribuidas
- Cuando hay múltiples clientes heterogéneos

Categorías de EJBs

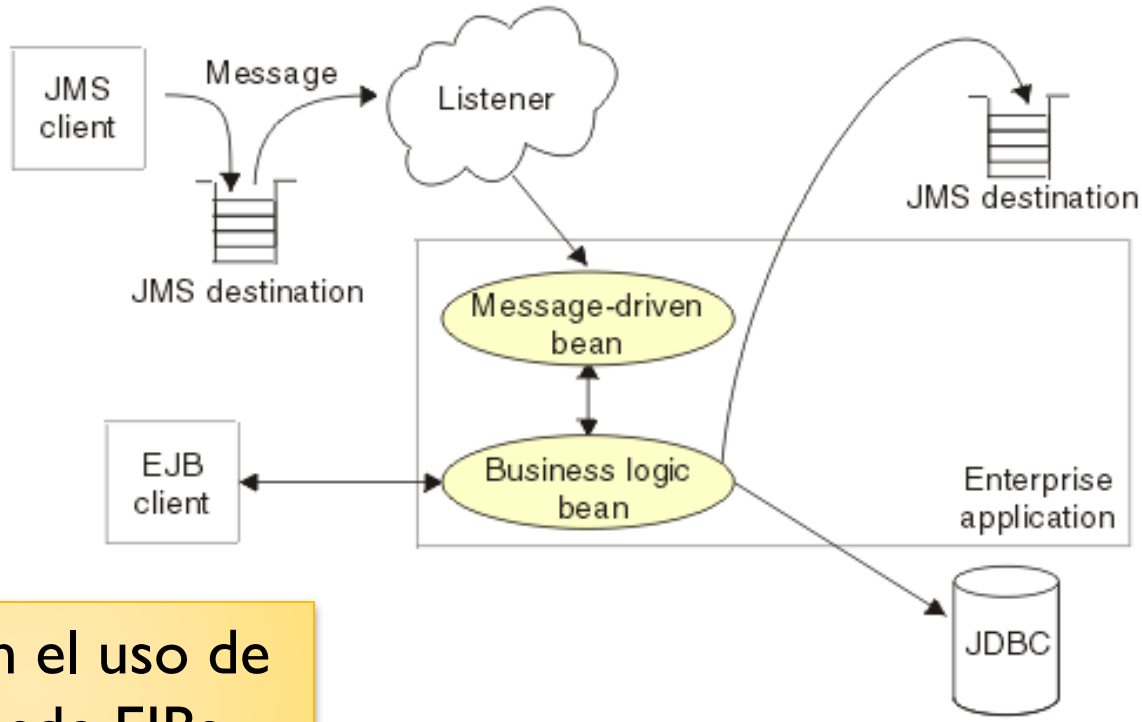
- Session Beans: Un bean de sesión es un componente que implementa una tarea específica dentro de la lógica de la aplicación de un servicio.
- Son usados para modelar procesos de negocio que son accedidos de manera síncrona.
- No son persistentes

Categorías de EJBs

- Session Beans
  - Stateful: Mantener el estado de una conversación con un solo cliente. El bean puede manejar el flujo de trabajo de varios beans.
  - Stateless: Accedidos concurrentemente por muchos clientes pero sin estado

Categorías de EJBs

- Message-driven Beans: Diseñados para modelar procesos de negocio que son accedidos de manera asíncrona a través de mensajes.



Permiten el uso de JMS desde EJBs

Categorías de EJBs

- Singleton Session Beans: Un bean de sesión singleton se instancia una vez por cada aplicación y existe para el ciclo de vida de la aplicación.

- Los Beans de sesión Singleton se diseñaron para circunstancias en las que una instancia de bean enterprise es compartida entre y accedida concurrentemente por múltiples clientes

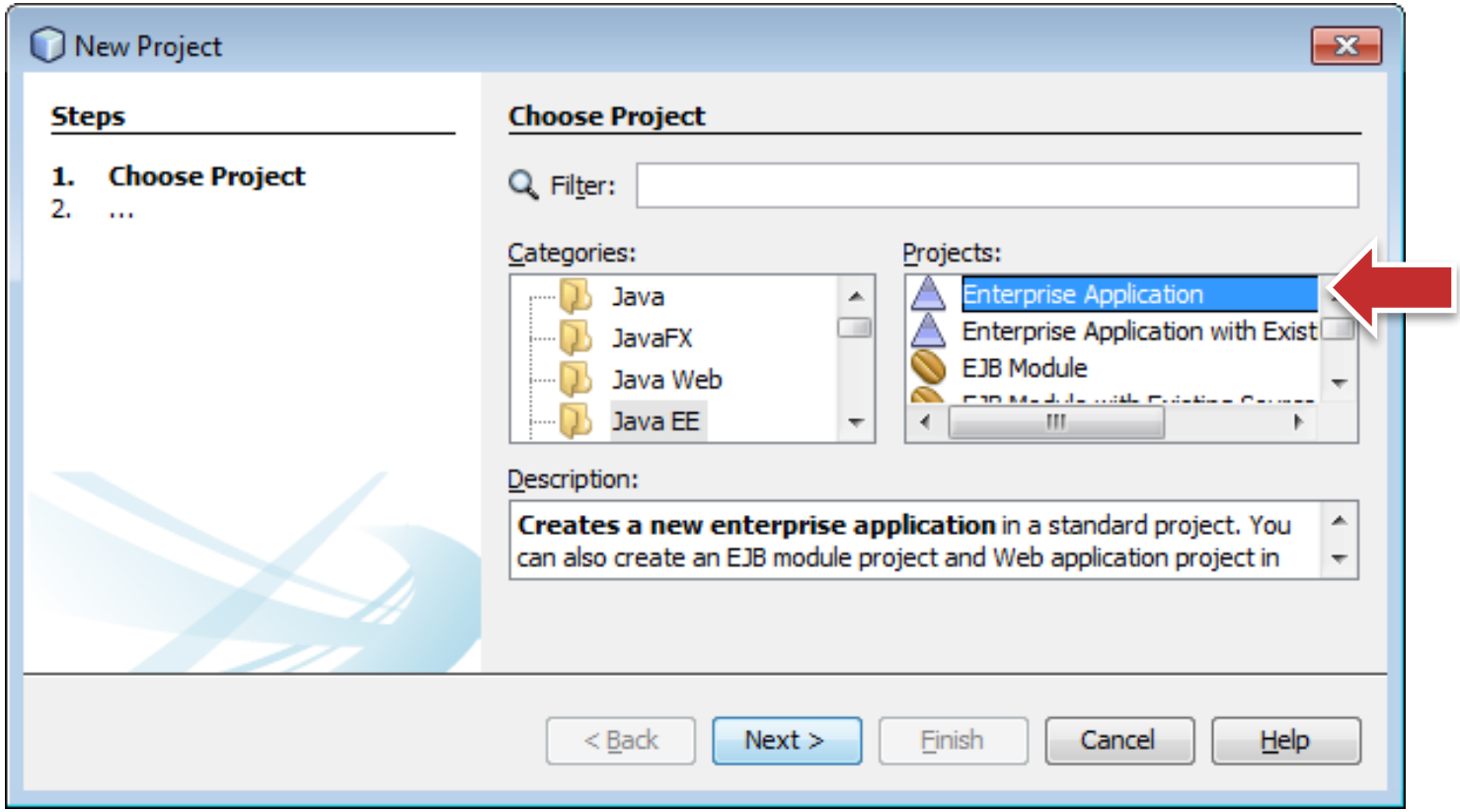
Desventajas de EJBs

- Tiempo de desarrollo

- Conocimiento avanzado de JAVA
  - EJBs
  - RMI
  - JNDI
  - JDBC

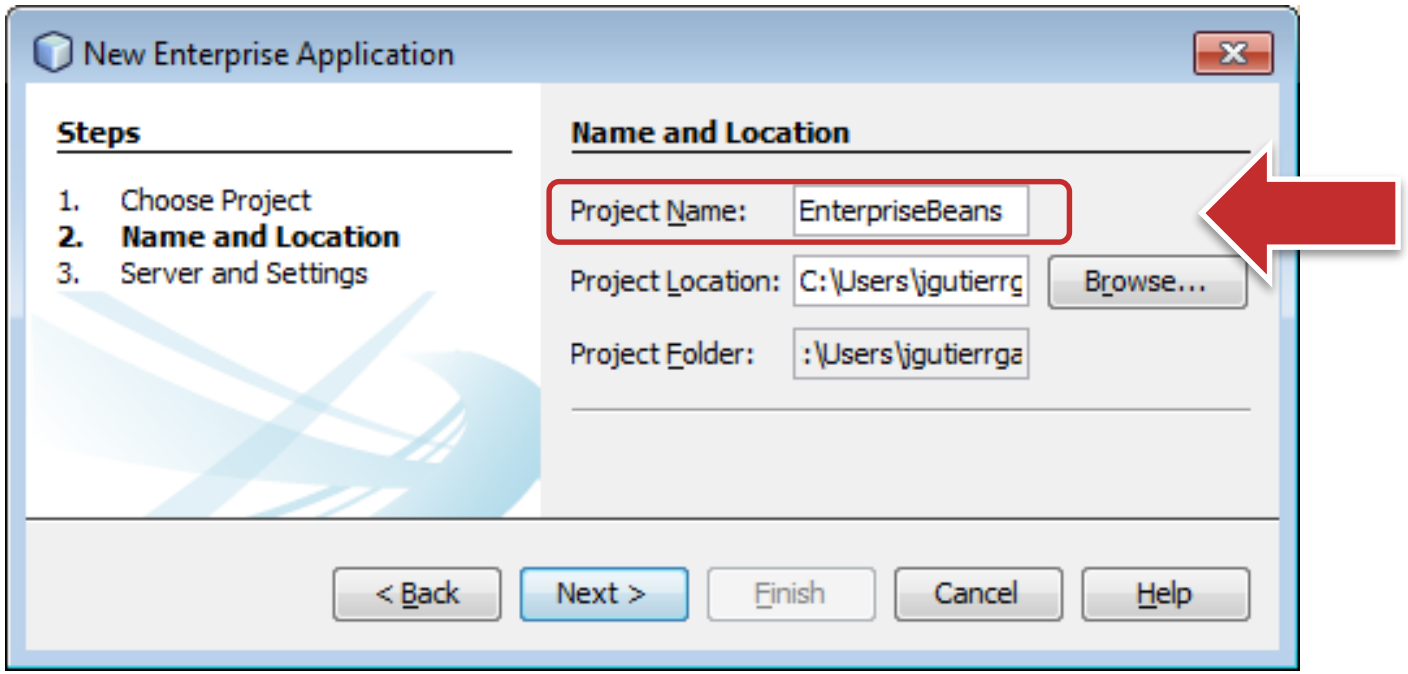
Session Bean

Paso 1. Crear una ”Enterprise Application”



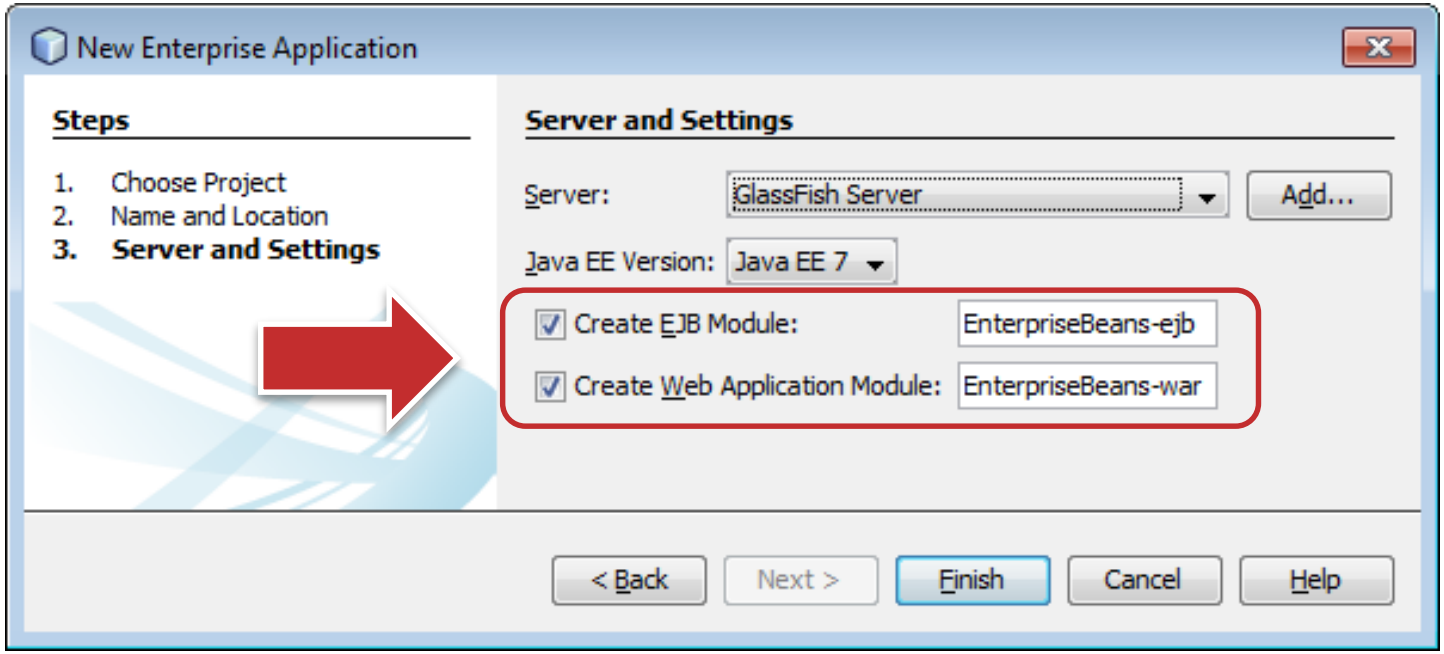
Session Bean

Paso 1. Crear una "Enterprise Application"



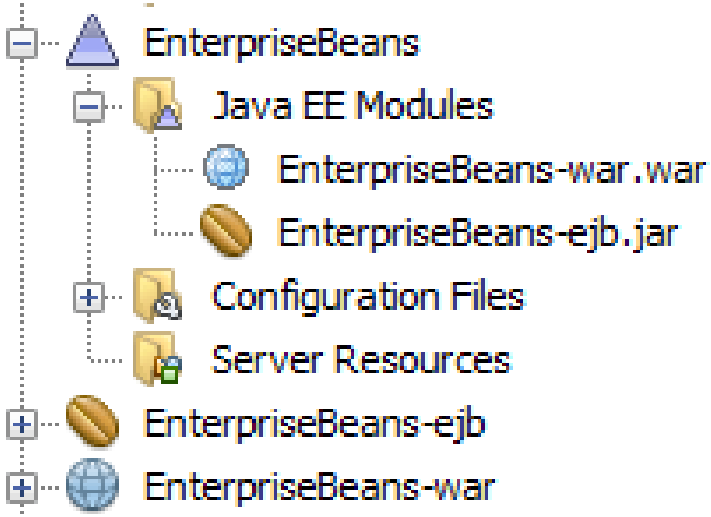
Session Bean

Paso 1. Crear una "Enterprise Application"



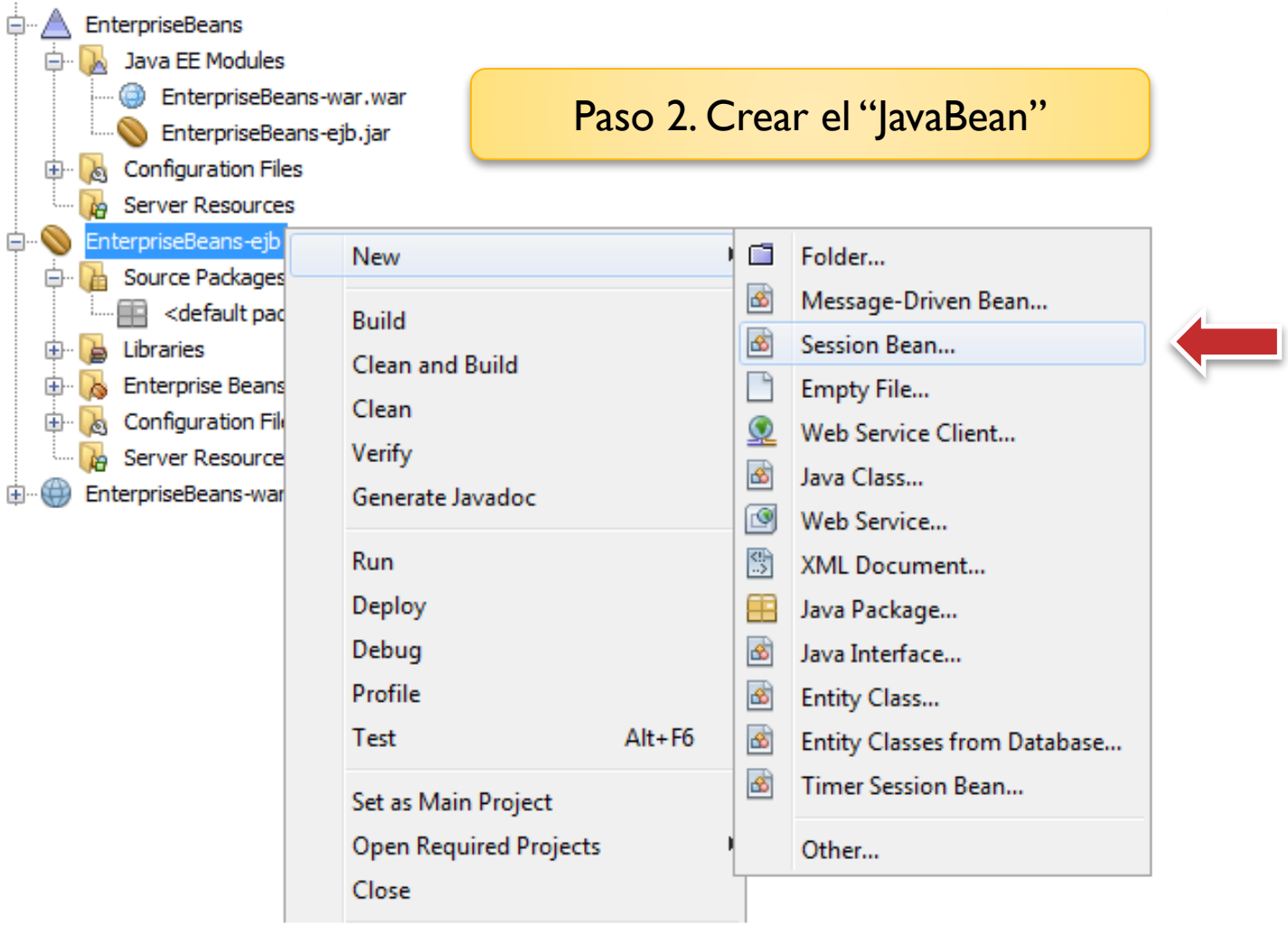
Session Bean

3 proyectos



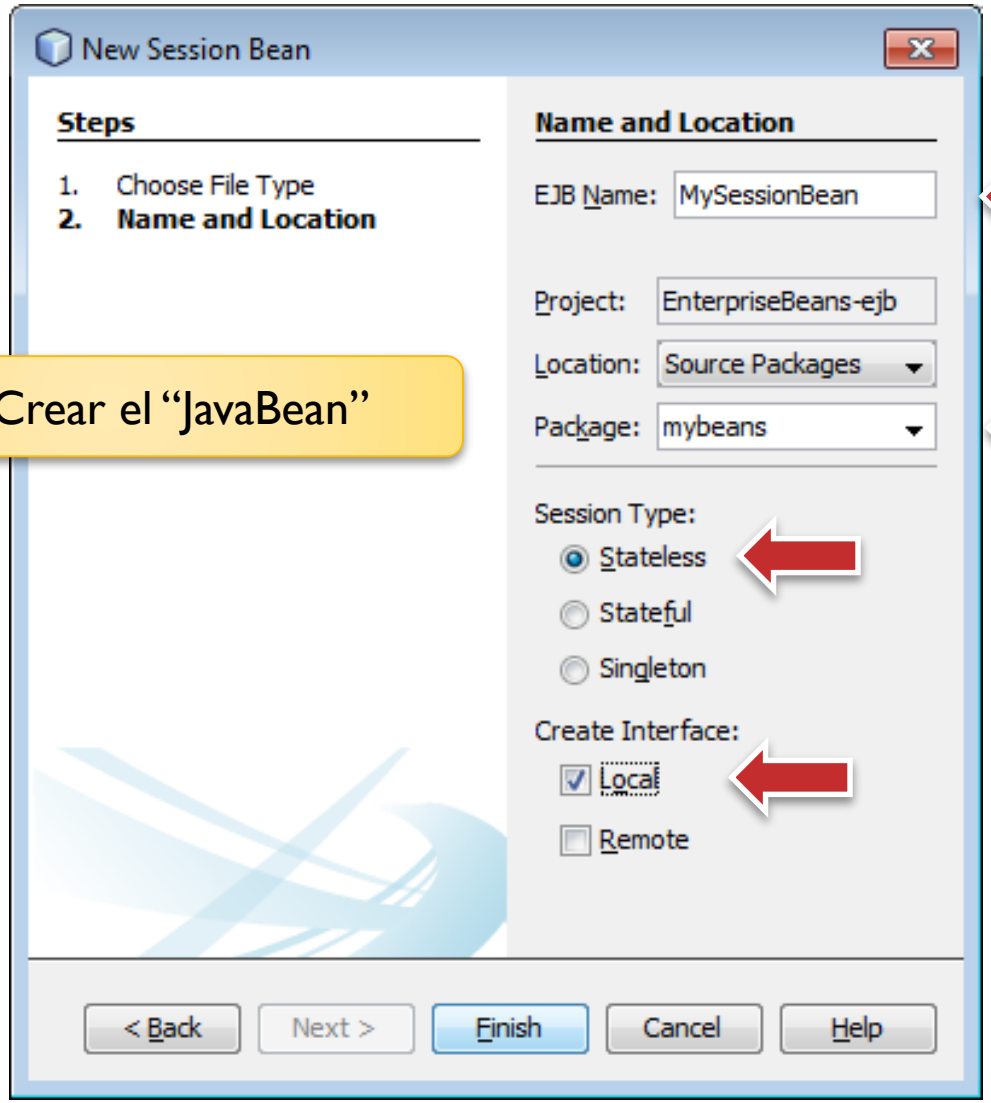
Session Bean

Paso 2. Crear el "JavaBean"



Session Bean

Paso 2. Crear el "JavaBean"



Session Bean

Paso 3. Especificar la "Lógica de Negocios"

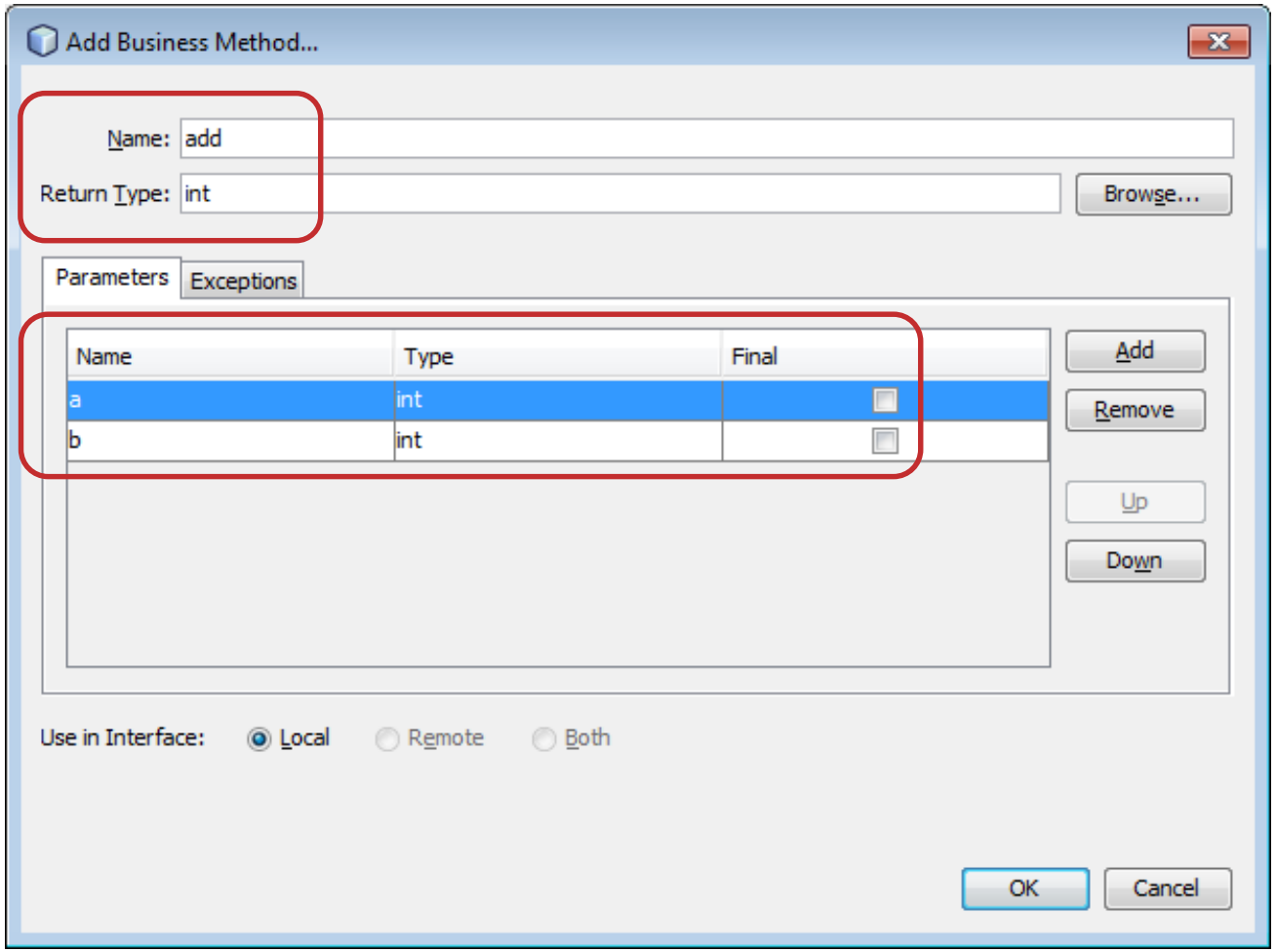
```
@Stateless
public class MySessionBean implements MySessionBeanLocal {

    ic below. (Right-click in editor and choose
    ness Method")
}
```

Generate

- Add Business Method...
- Constructor...
- Logger...
- toString()...
- Override Method...
- Add Property...
- Call Enterprise Bean...
- Use Database...
- Send JMS Message...
- Send E-mail...
- Call Web Service Operation...
- Generate REST Client...

Session Bean

Paso 3. Especificar la "Lógica de Negocios"



Session Bean

Paso 3. Especificar la "Lógica de Negocios"

```
@Stateless
public class MySessionBean implements MySessionBeanLocal {

    @Override
    public int add(int a, int b) {
        return a+b;
    }
```

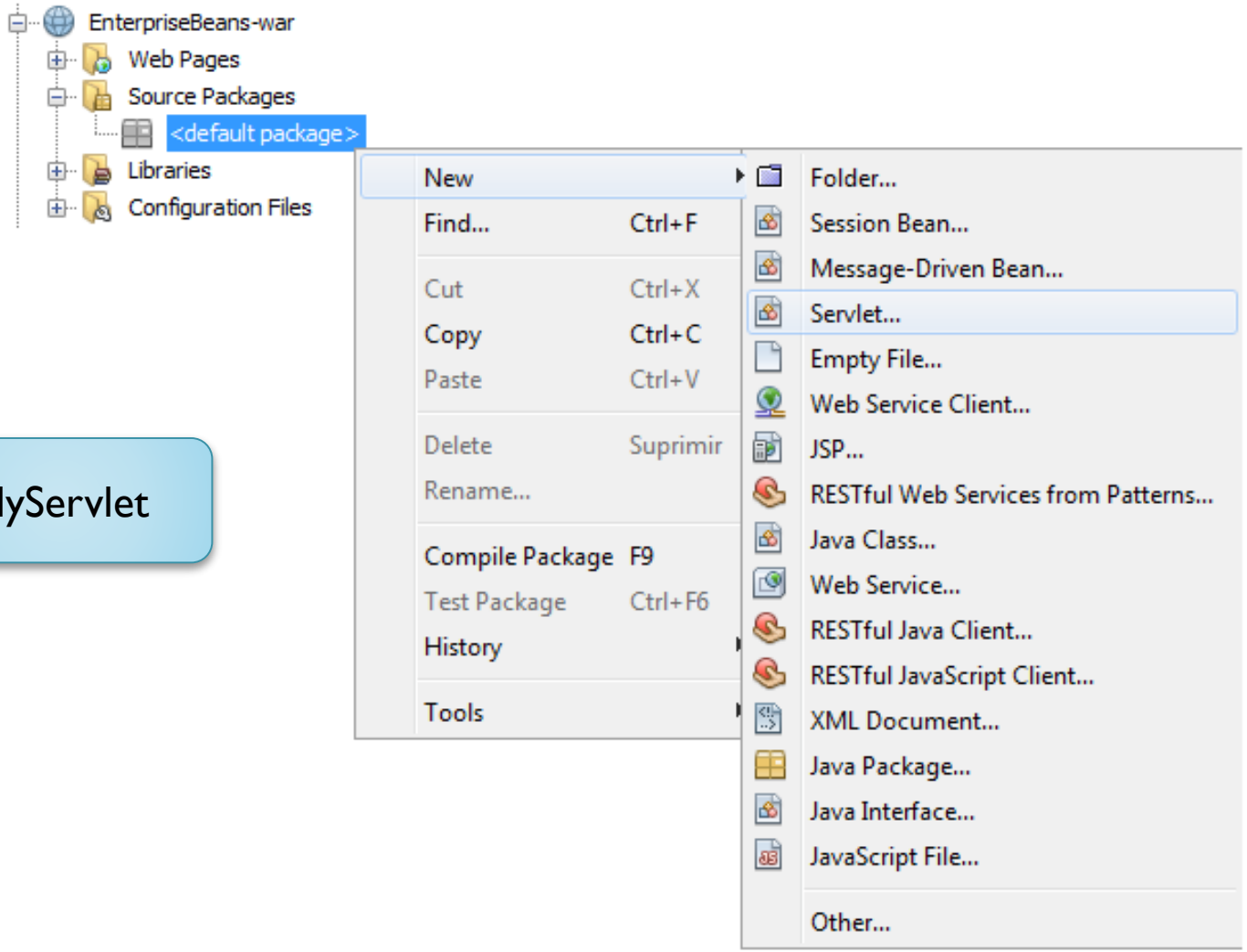
Session Bean

Paso 4. 1. Clean & Build
Paso 4. 2. Deploy

```
@Stateless
public class MySessionBean implements MySessionBeanLocal {

    @Override
    public int add(int a, int b) {
        return a+b;
    }
```

Session Bean

Paso 5. Crear Servlet para invocar al JavaBean



MyServlet

Session Bean

Paso 5. Invocar al JavaBean

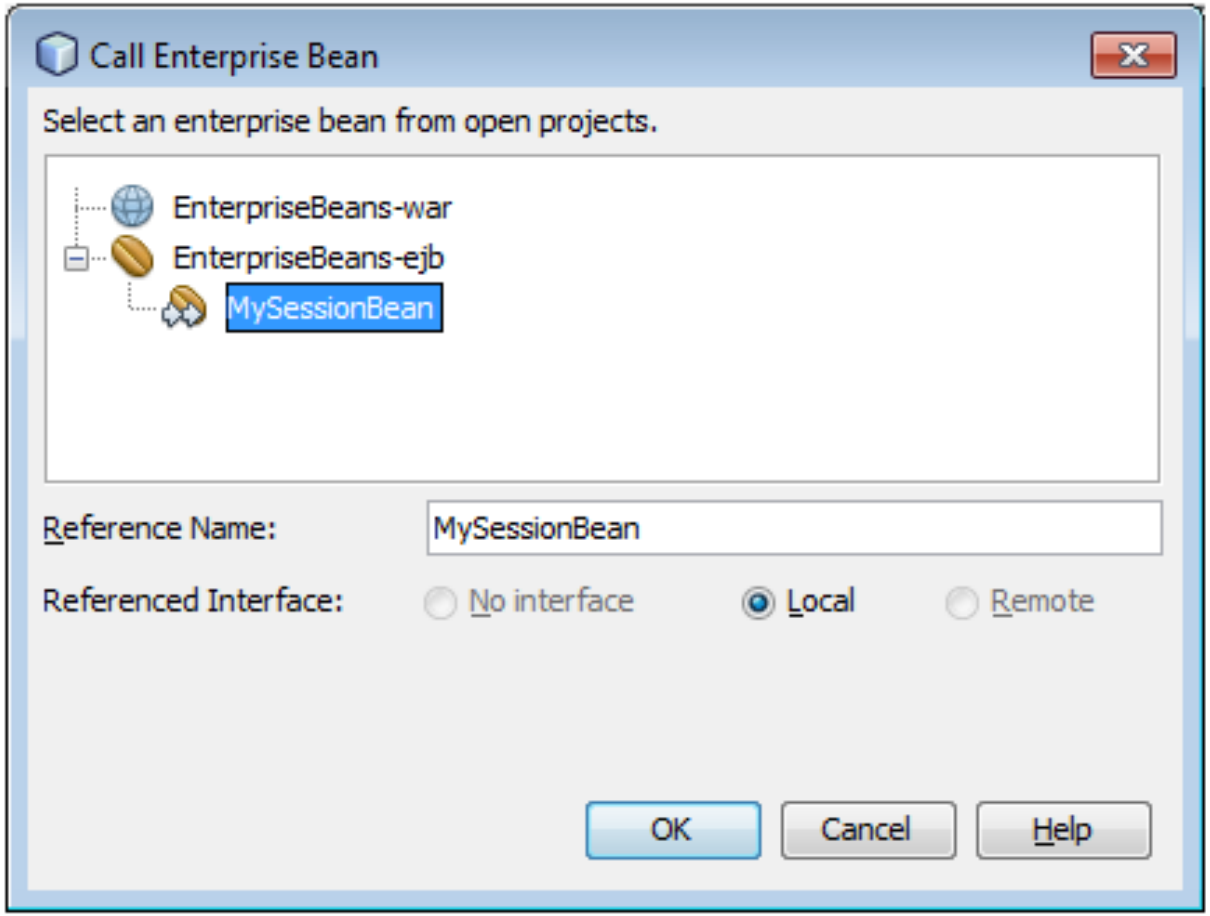
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        YPE html>");
        ");
        ");
        >Servlet MyServlet</title>");
        >");
        ");
        rvlet MyServlet at " + request.getContextPath() + "</h1>");
        >");
        >");
    }
}
```

Generate

- Constructor...
- Logger...
- toString()...
- Override Method...
- Add Property...
- Call Enterprise Bean...
- Use Database...
- Send JMS Message...
- Send E-mail...
- Call Web Service Operation...
- Generate REST Client...

Session Bean

Paso 5. Invocar al JavaBean



Session Bean

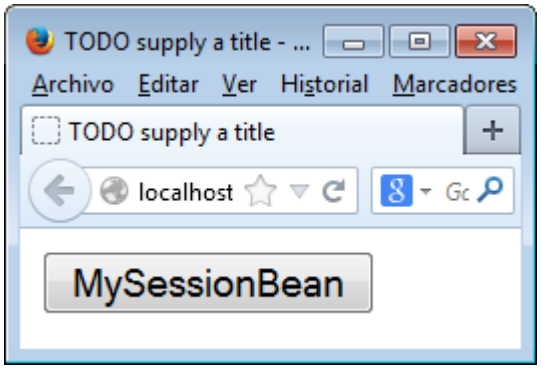
Paso 5. Invocar al JavaBean

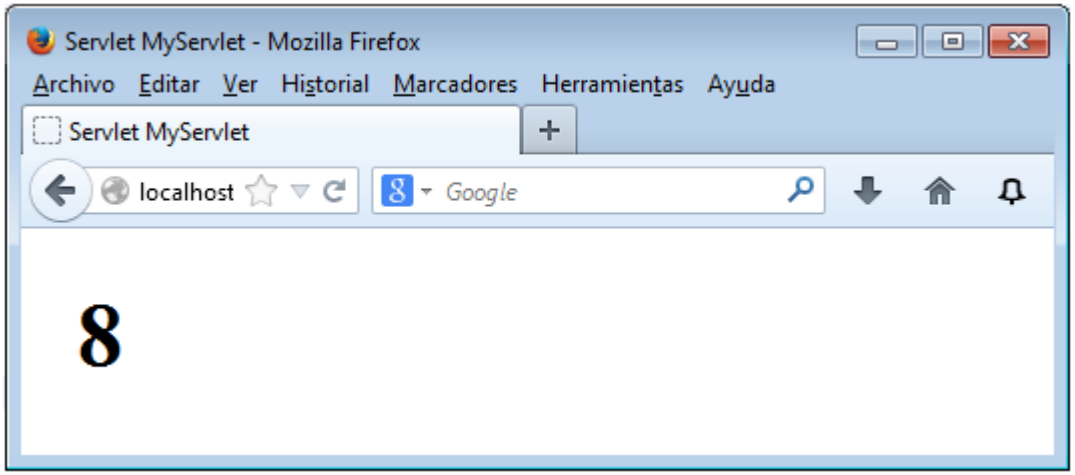
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {


        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet MyServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet MyServlet at " + mySessionBean.add(3, 5) + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Session Bean

Paso 6. Incluir forma en index.html para llamar al Servlet

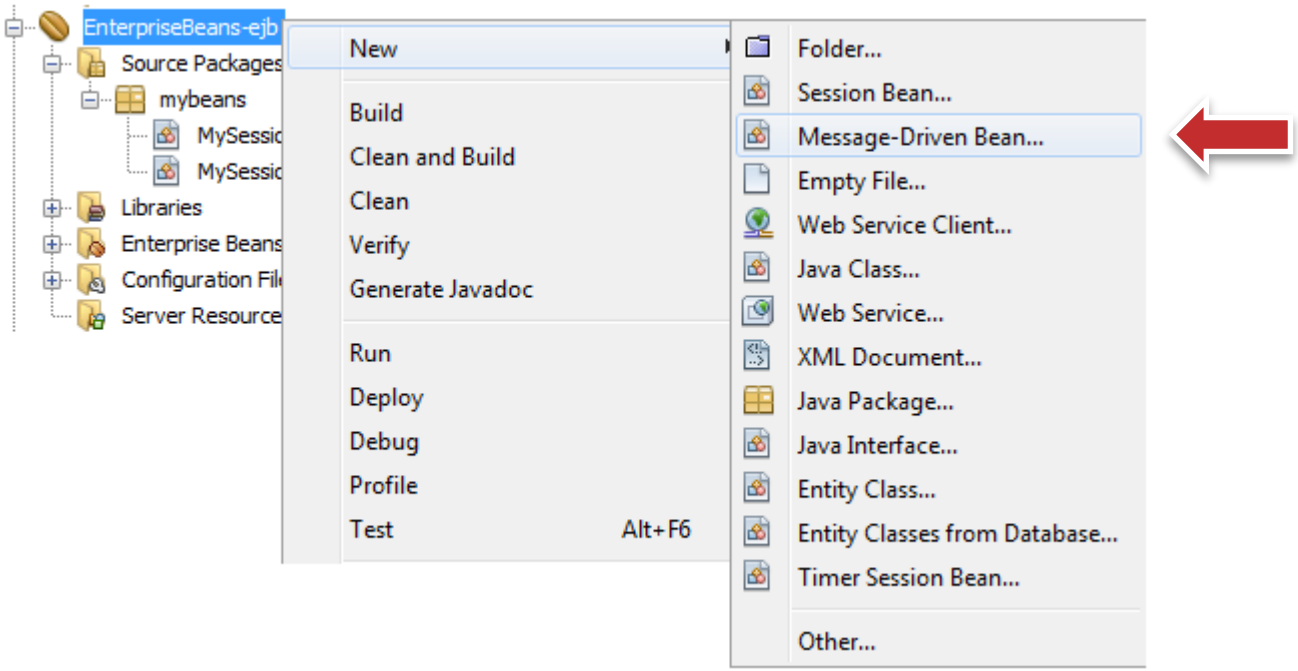
```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="MyServlet" method="POST">
            <input type="submit" value="MySessionBean" />
        </form>
    </body>
</html>
```





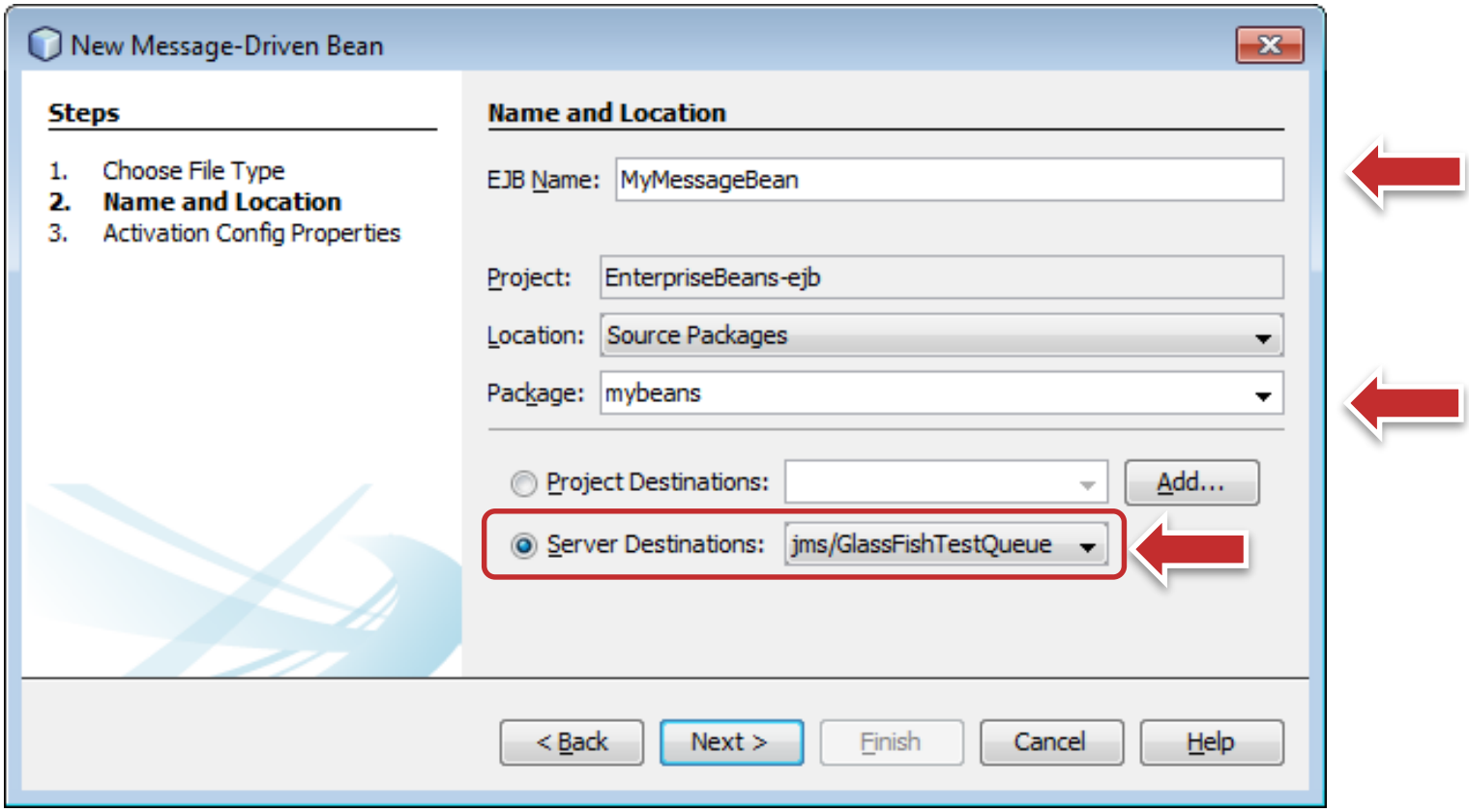
Message-Driven Bean

Paso 1. Crear el "JavaBean"



Message-Driven Bean

Paso 1. Crear el "JavaBean"



Message-Driven Bean

Paso 2. Definir el manejo de los mensajes entrantes

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "jms/GlassFishTestQueue")
})
public class MyMessageBean implements MessageListener {

    public MyMessageBean() {
    }

    @Override
    public void onMessage(Message message) {
        try {
            System.out.println("A message: "+((TextMessage)message).getText());
        } catch (JMSException ex) {
            Logger.getLogger(MyMessageBean.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

Message-Driven Bean

Paso 3. Desplegar el JavaBean

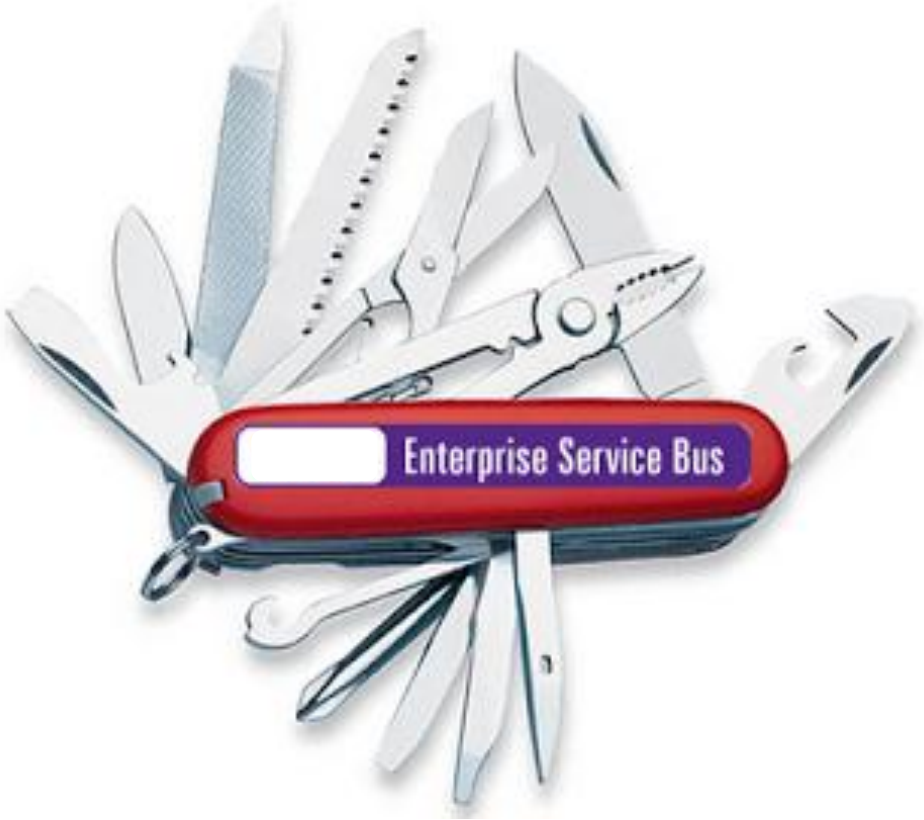
Paso 4. Probar el JavaBean ejecutando un MessageProducer(Queue)

NetBeans

GlassFishServer 4.0
EnterpriseBeans
EnterpriseBeans-ejb
EnterpriseBeans-war
JSMQueue

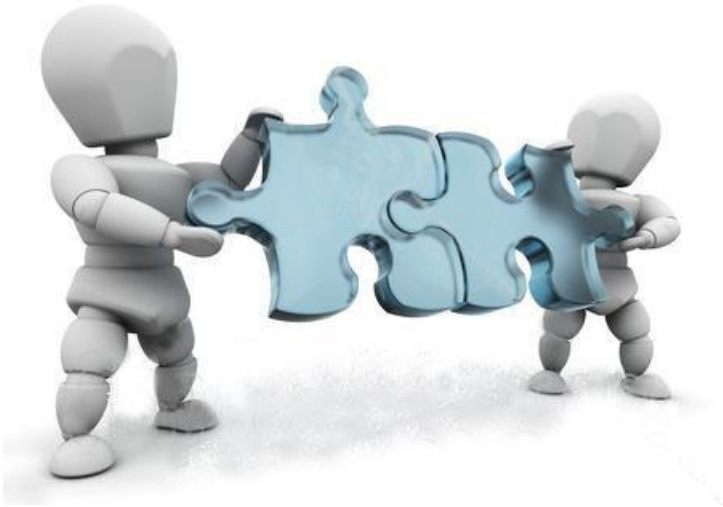
ESB

Enterprise Service Bus



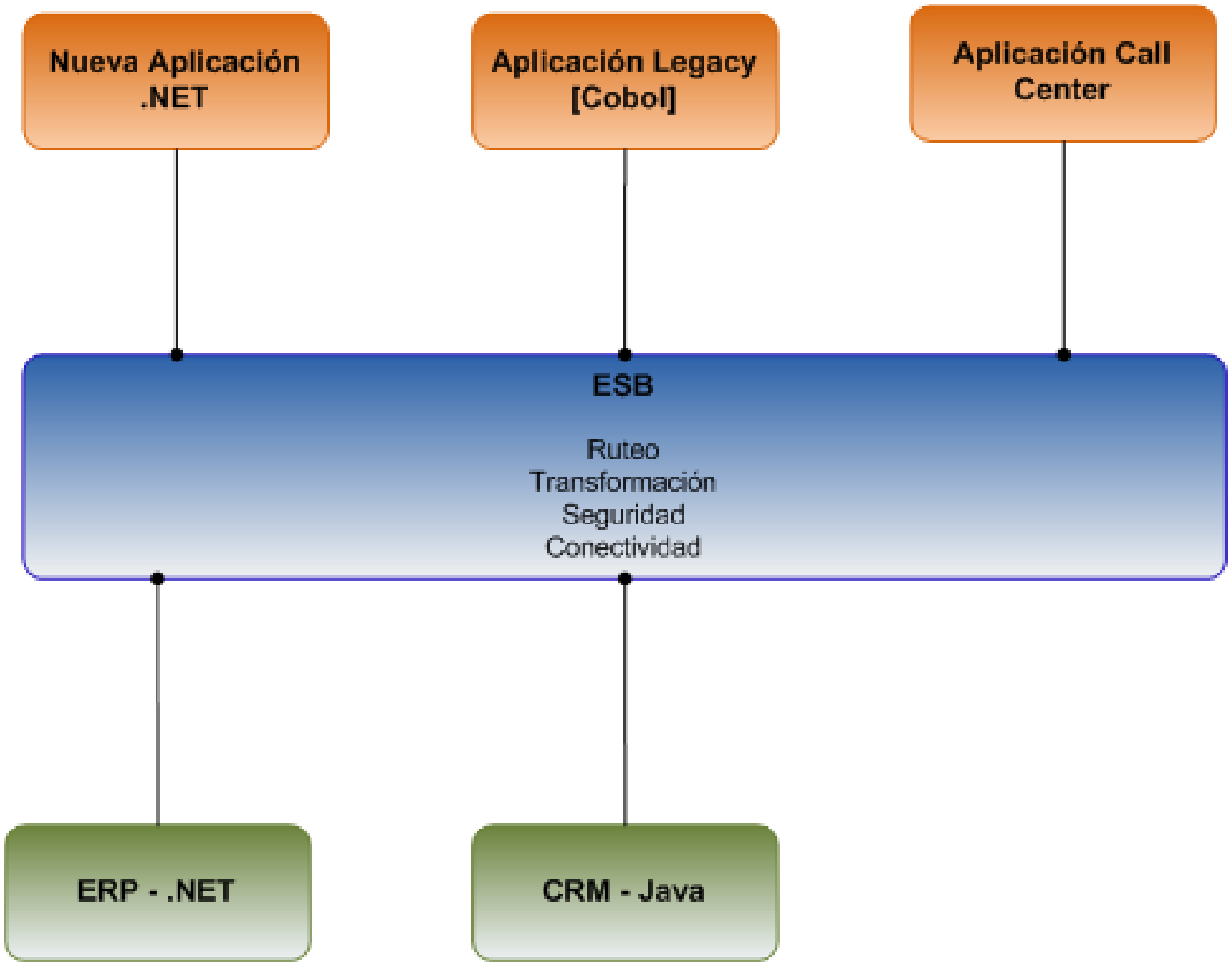
Enterprise Service Bus

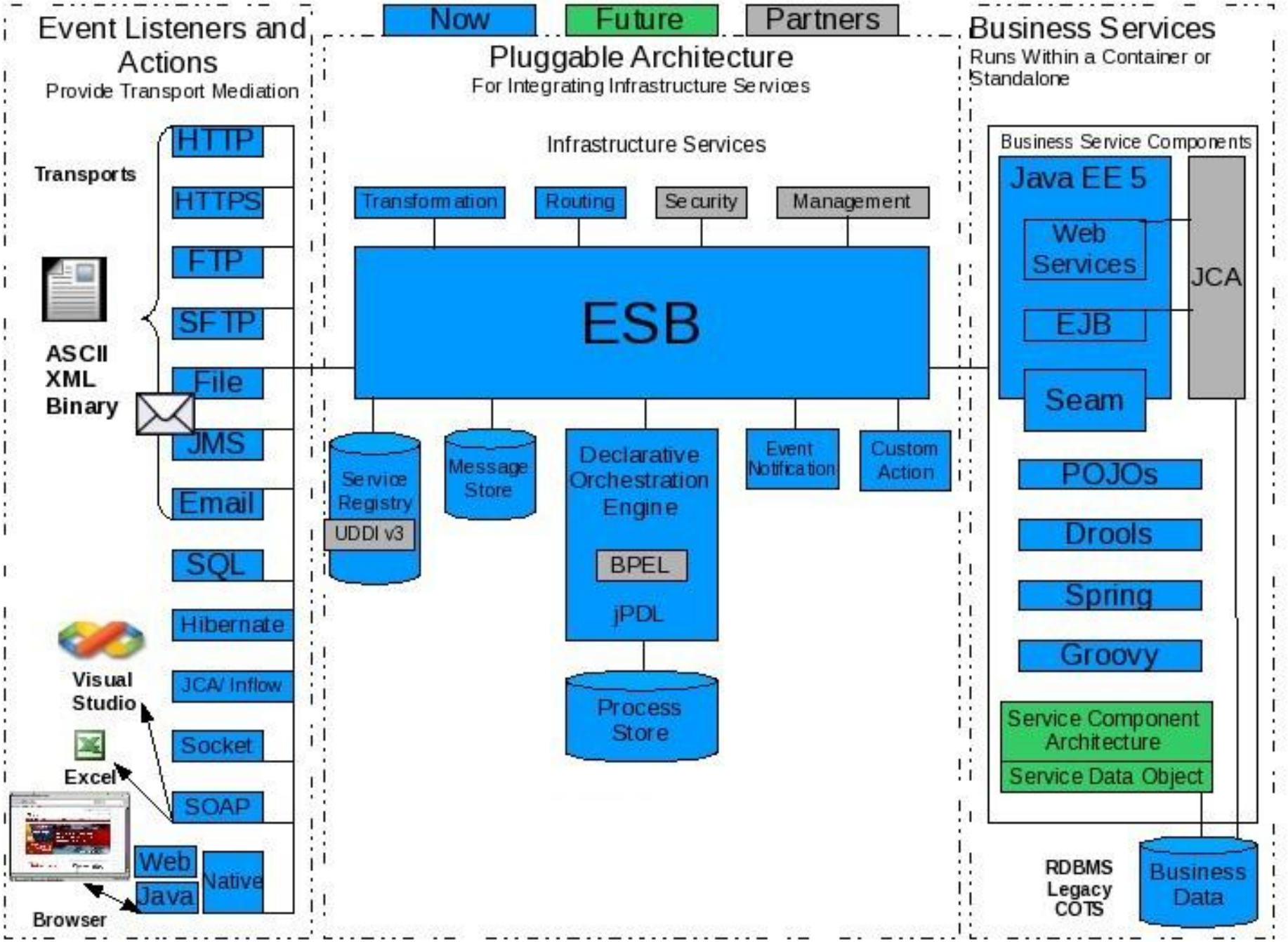
Enterprise Service Bus (ESB) es una infraestructura de conectividad flexible para integrar aplicaciones y servicios.



Un ESB proporciona la conectividad para implementar una arquitectura orientada a servicios (SOA), lo que reduce la complejidad de la integración de aplicaciones y servicios.

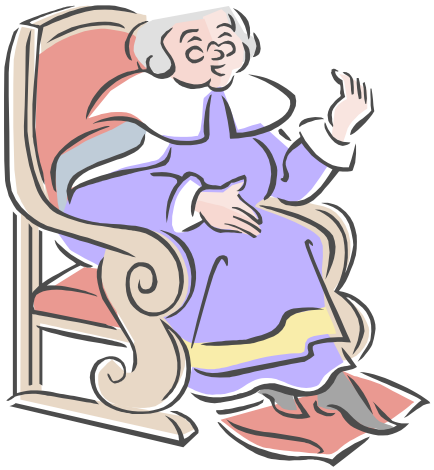
Enterprise Service Bus




Now
Future
Partners
Event Listeners and Actions
Provide Transport Mediation
Pluggable Architecture
For Integrating Infrastructure Services
Business Services
Runs Within a Container or Standalone
Transports
HTTP
HTTPS
FTP
SFTP
File
JMS
Email
SQL
Hibernate
JCA/ Inflow
Socket
SOAP
Web
Java
Native
ASCII
XML
Binary
Visual Studio
Excel
Browser
Infrastructure Services
Transformation
Routing
Security
Management
ESB
Service Registry
UDDI v3
Message Store
Declarative Orchestration Engine
BPEL
jPDL
Process Store
Event Notification
Custom Action
Business Service Components
Java EE 5
Web Services
EJB
Seam
JCA
POJOs
Drools
Spring
Groovy
Service Component Architecture
Service Data Object
RDBMS
Legacy
COTS
Business Data

¿Cuándo usar JBOSS – ESB?

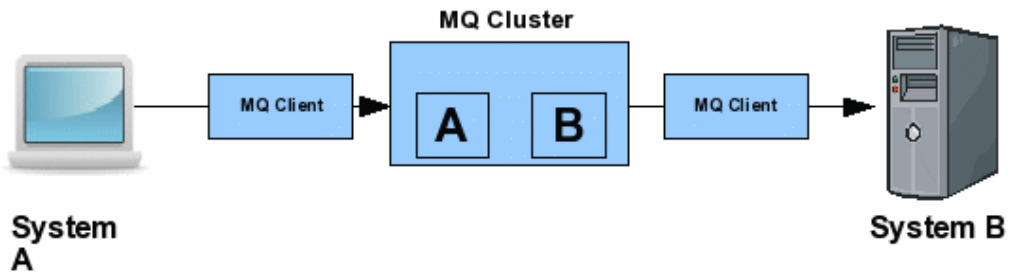
Legacy Software

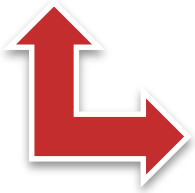


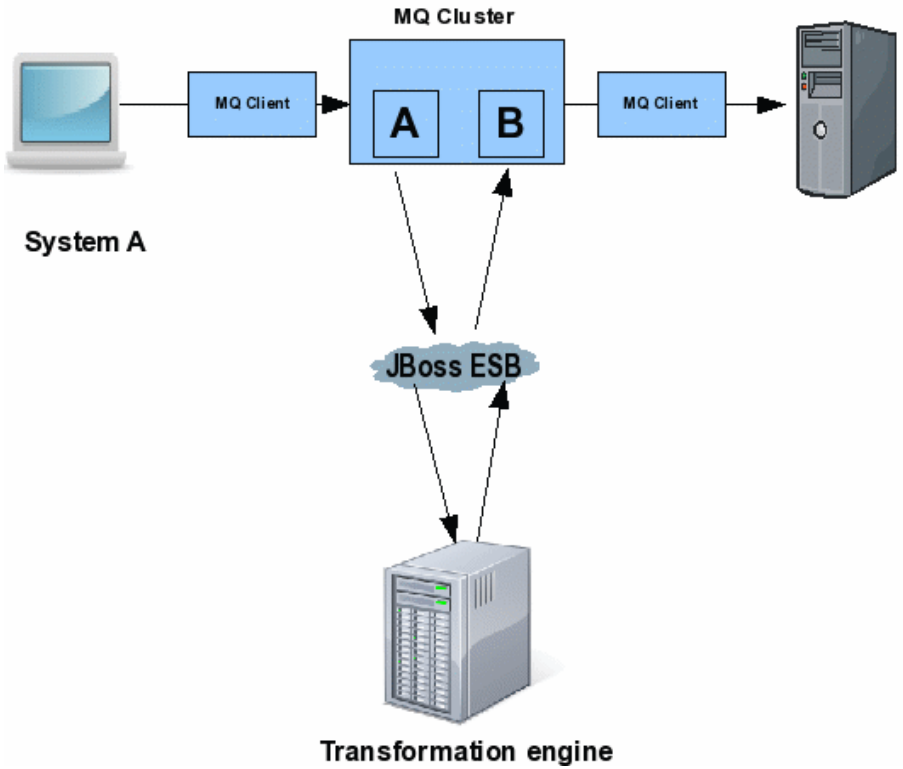
JBoss ESB

Cutting edge Software

¿Cuándo usar JBOSS – ESB?







¿Cuándo usar JBOSS – ESB?