

Java EE – Enterprise Beans (EJB)

Aplicaciones Web/Sistemas Web



Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

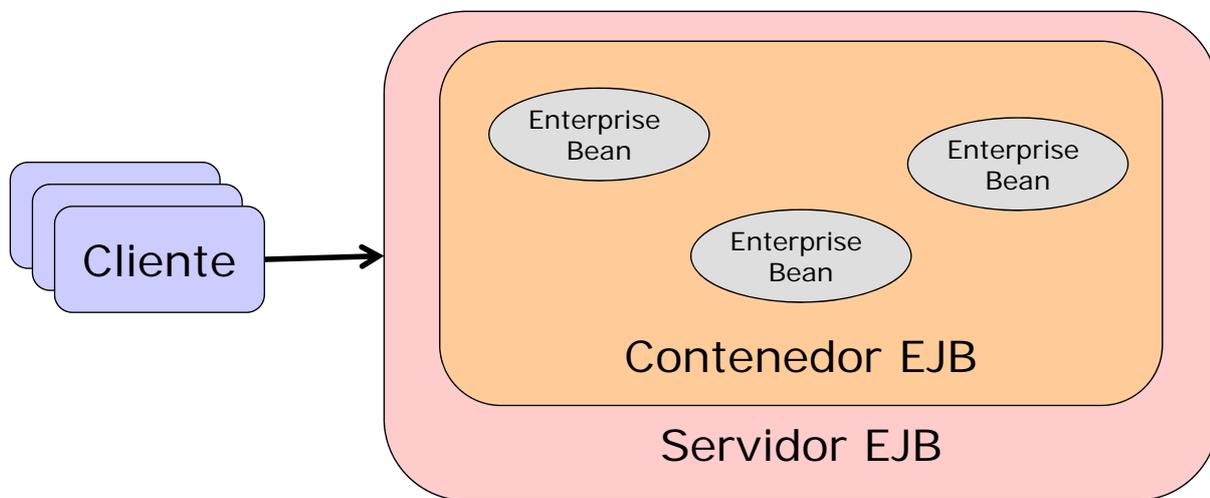
Material bajo licencia Creative Commons, Juan Pavón 2013



Enterprise Beans

- Componentes Java que implementan la tecnología Enterprise Java Beans (EJB)
 - Se ejecutan en un contenedor EJB (en un servidor de aplicaciones)
 - Implementan la lógica de negocio de la aplicación
 - Son reusables
 - Se pueden adaptar y configurar en el despliegue
- El contenedor de EJB proporciona varios servicios
 - Ciclo de vida de los EJB
 - Gestión de la persistencia
 - Servicio de nombres
 - Mensajería asíncrona
 - Seguridad de acceso
 - Gestión de transacciones
 - Balanceo de carga

Arquitectura EJB

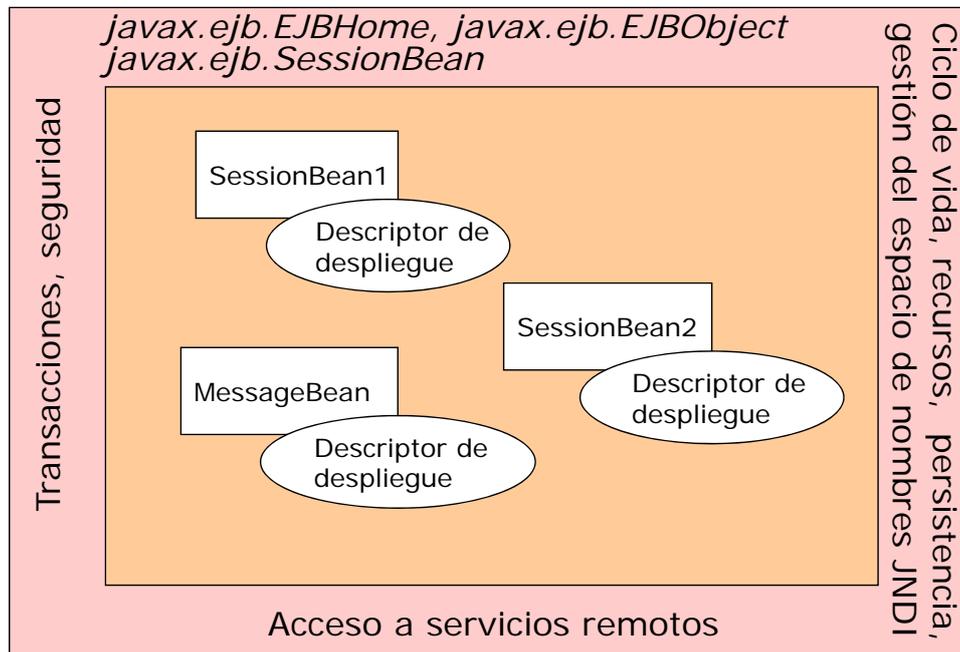


Servidor EJB

- Entorno de ejecución
 - Gestión de procesos y threads
 - Gestión de recursos del sistema
 - Conexión a bases de datos
 - API de gestión

Contenedor de EJB

- Entorno de ejecución de los EJBs



Cientes EJB

- Pueden ser servlets o aplicaciones
- El contenedor de EJBs puede controlar el acceso de los clientes
- Para localizar los EJBs los clientes usan el Java Naming and Directory Interface (JNDI)
- El acceso a los EJBs se puede hacer remotamente con Java RMI (Remote Method Invocation)

Enterprise Beans

- Implementan la lógica de negocio
- Tipos (EJB 3.1)
 - Enterprise Bean de Sesión
 - Gestionan la lógica de negocio para un cliente
 - Tipos
 - Stateless Session Beans: no mantienen estado entre invocaciones
 - Stateful Session Beans: guardan estado entre invocaciones de un mismo cliente
 - Singletons: una sola instancia que es compartida por muchos clientes
 - Enterprise Bean dirigido por Mensajes
 - Oyente (listener) para un mensaje particular (generalmente a través del API Java Message Service)
 - No mantiene estado entre invocaciones
 - Entity Bean
 - Representaciones de datos almacenados en una BD
 - Recientemente se recomienda usar Java Persistence API (JPA)

Enterprise Beans

- Gran parte de la funcionalidad la gestiona el contenedor
- En ejecución, el contenedor intercepta todas las llamadas al EJB
 - Ejecuta código relacionado con la política de multi-threading, transacciones, y seguridad
 - Llamará al método correspondiente
 - Realiza las tareas de limpieza correspondientes tras la llamada
- Se define mediante declaraciones
 - En el desarrollo: anotaciones en el código
 - En descriptores de despliegue

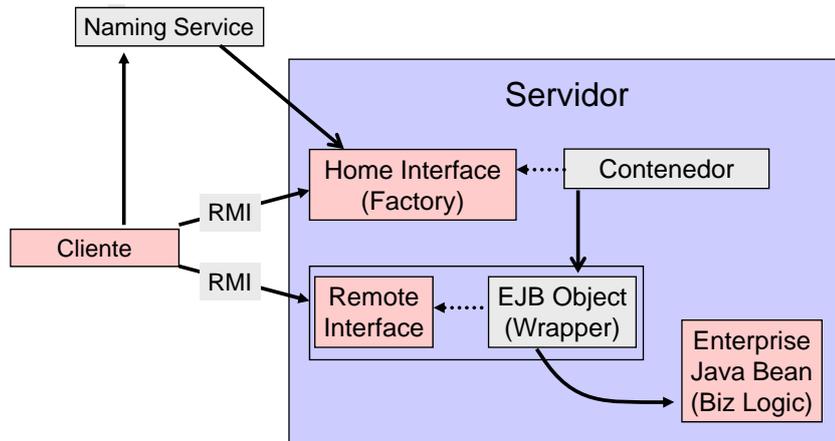
Enterprise Beans

- **Persistencia**
 - El almacenamiento de los datos de los beans
 - Los puede gestionar el servidor: Container-Managed Persistence
 - O el propio bean: Bean-Managed Persistence
- **Transacciones distribuidas**
 - El servidor gestionará todas las transacciones
 - Se puede gestionar también directamente (hay que tener las ideas claras)
- **Seguridad**
 - Transmisión de datos por SSL/RMI
 - Autenticación de cliente y servidor con SSL
 - Control de acceso a objetos, métodos y servicios con ACLs (listas de control de acceso)
- **Multithreading**
 - Las políticas de multithreading a aplicar se declaran y el servidor se encarga de ejecutarlas
 - El programador no tiene que ocuparse de este aspecto

Interfaces

- **Inicialmente los EJBs tenían que implementar varias interfaces**
 - Desde la especificación 3.0 pueden ser objetos POJO pero con anotaciones
- **Interfaz Home**
 - Extiende la interfaz javax.ejb.EJBHome
 - Interfaz remota para crear, encontrar y eliminar beans
- **Interfaz Remote**
 - Extiende la interfaz javax.ejb.EJBObject
 - Interfaz remota que define los métodos concretos que implementará el bean
- **Clase Bean**
 - Extiende la clase javax.ejb.EnterpriseBean
 - Implementa la lógica de negocio

Funcionamiento



- Escrito por el programador
- ←..... Implements
- Invoca
- Crea/Usa

Adaptado de jGuru (2000): *Enterprise JavaBeans Fundamentals*

Programación de EJB

- **Interfaz Remote**
 - Extiende la interfaz `javax.ejb.EJBObject`
 - Interfaz remota que define los métodos concretos que implementará el bean

```
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Cliente extends EJBObject {

    public Name getNombre() throws RemoteException;
    public void setNombre(String nombre) throws RemoteException;
    public Address getEmail() throws RemoteException;
    public void setEmail(String email) throws RemoteException;

}
```

Programación de EJB

■ Interfaz Home

- Extiende la interfaz javax.ejb.EJBHome
- Interfaz remota para crear, encontrar y eliminar beans
- Se genera automáticamente la clase que la implementa

```
import javax.ejb.EJBHome;
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import java.rmi.RemoteException;

public interface ClienteHome extends EJBHome {
    public Cliente create(Integer idCliente)
        throws RemoteException, CreateException;

    public Cliente findByPrimaryKey(Integer idCliente)
        throws RemoteException, FinderException;

    public Enumeration findByNombre(String nombre)
        throws RemoteException, FinderException;
}
```

Programación de EJB

■ Clase Bean

- Extiende la interfaz javax.ejb.EJBHome
- Interfaz remota para crear, encontrar y eliminar beans

```
import javax.ejb.EntityBean;

public class ClienteBean implements EntityBean {
    String nombre;
    String email;
    public String getNombre() {
        return nombre;
    }
    public void setNombre (String nombre) {
        this.nombre = nombre;
    }
    public Address getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

Entorno de trabajo

- Para desarrollar, Eclipse o Netbeans

- Como servidor de aplicaciones que tenga contenedor de EJBs
 - JBoss: <http://www.jboss.org>
 - Hay una versión para desarrolladores: JBoss Developer Studio (Eclipse + JBoss Tools)
 - Glassfish: <https://glassfish.java.net/es/>

- Una comparativa interesante:
<http://antoniogoncalves.org/2011/10/20/o-java-ee-6-application-servers-where-art-thou/>

- Bases de datos
 - Apache Derby: <http://db.apache.org/derby/>
 - H2: <http://www.h2database.com>

Primera aplicación EJB con Eclipse y JBoss

- Una posibilidad es instalar JBoss Developer Studio

- Otra es configurar Eclipse para trabajar con JBoss Tools
 - Help → Eclipse Marketplace
 - Buscar: JBoss
 - Seleccionar JBoss Tools para la versión de eclipse actual (Juno)
 - Se pueden seleccionar únicamente Java Development Tools y JBoss AS Tools
 - Otros plugins se podrán instalar posteriormente si fueran necesarios
 - Al acabar, reinicializar eclipse

- Como BD se puede instalar Derby, muy eficiente y escrita en Java
 - Instalar los plugins Derby 10 Core Plug-in y Derby 1.0 UI
 - Descargarlos de http://db.apache.org/derby/derby_downloads.html
 - Y descomprimirlos en la carpeta de instalación de eclipse
 - Alternativa: Eclipse Data Tools Platform
<http://www.vogella.com/articles/EclipseDataToolsPlatform/article.html>

Primera aplicación EJB con Eclipse y JBoss

- Instalar JBoss
 - <http://sourceforge.net/projects/jboss/files/JBoss/JBoss-6.0.0.Final/jboss-as-distribution-6.0.0.Final.zip/download>
- Añadir el servidor JBoss en eclipse
 - En Servers o File → New → Other → Server → Server
 - Añadir el JBoss AS 6.x
 - Indicar el directorio donde se ha descomprimido el JBoss
 - Para arrancar el servidor, seleccionarlo y con el menú contextual elegir Start
- Crear un nuevo proyecto EJB
 - New → EJB Project
 - Indicar que se usará el servidor JBoss en Target Runtime
- Crear una clase Java que será el primer EJB
 - Crear un paquete (es.ucm.jpavon.ejb)
 - En ejbModule crear el primer EJB con New → Session Bean (EJB 3.x)

Primera aplicación EJB con Eclipse y JBoss

```
package es.ucm.jpavon.ejb;
import javax.ejb.Remote;
import javax.ejb.Stateless;
```

```
@Remote
@Stateless
public class PrimerBean implements PrimerBeanRemote {
    public String saluda(String nombre) {
        return "Hola " + nombre;
    }
}
```

Indica que el POJO es un Bean stateless para que el contenedor lo trate como un EJB

Al haber declarado @Remote eclipse añade la interfaz

Se genera al haber declarado la clase como Remote

```
package es.ucm.jpavon.ejb;

import javax.ejb.Remote;

@Remote
public interface PrimerBeanRemote {
    public String saluda(String nombre);
}
```

Hay que declarar los métodos que vaya a ofrecer el Bean

Primera aplicación EJB – Cliente

- El cliente puede ser otra aplicación
- Crear en eclipse un nuevo proyecto Java
- Para que pueda usar la configuración del proyecto anterior (y acceder a las librerías correspondientes de EJB)
 - En el proyecto, seleccionar Build Path → Configure Build Path
 - Pestaña Projects, botón Add, seleccionar el anterior
- Crear la clase del programa cliente
- Ejecutar como una aplicación normal

Primera aplicación EJB – Cliente

```
package es.ucm.jpavon.cliente;
import java.util.Properties;
import javax.naming.*;
import es.ucm.jpavon.ejb.PrimerBeanRemote;

public class ClienteEJB {
    private static final String JNDI_PRIMER_BEAN = "PrimerBean/remote";

    public static void main(String[] args) throws NamingException {
        Properties p = new Properties();
        p.put("java.naming.factory.initial", org.jnp.interfaces.NamingContextFactory);
        p.put("java.naming.factory.url.pkgs", "org.jboss.naming:org.jnp.interfaces");
        p.put("java.naming.provider.url", "jnp://localhost:1099");
        Context contexto = new InitialContext(p);

        PrimerBeanRemote bean = (PrimerBeanRemote)contexto.lookup(JNDI_PRIMER_BEAN);
        String respuesta = bean.saluda("Cliente Java");
        System.out.println(respuesta);
    }
}
```

Referencia JNDI del componente

Contexto del contenedor

Anotaciones EJB

- El uso de anotaciones de metadatos (desde EJB 3.0) simplifica la codificación de EJBs
 - Se genera código automáticamente
 - Cambios de las anotaciones requieren recompilar
- Definidas como clases del paquete `javax.ejb`
 - Por ejemplo, la anotación `@Remote` corresponde a `javax.ejb.Remote`
- Se pueden definir para afectar a clases o métodos
- Descripción completa:
http://docs.oracle.com/cd/E11035_01/wls100/ejb30/annotations.html

Anotaciones EJB

- Anotaciones de Beans de Sesión
 - `@Stateful`: Indica que el Bean de Sesión es con estado
 - `@Stateless`: Indica que el Bean de Sesión es sin estado
 - `@Init`: Especifica que el método se corresponde con un método `create` de un `EJBHome` o `EJBLocalHome` de EJB 2.1
 - `@Remove`: Indica que el contenedor debe llamar al método cuando quiera destruir la instancia del Bean
 - `retainIfException` - indica si el Bean debe mantenerse activo si se produce una excepción. Por defecto a `false`
 - `@Local`: Indica que la interfaz es local
 - `@Remote`: Indica que la interfaz es remota
 - `@PostActivate`: Invocado después de que el Bean sea activado por el contenedor
 - `@PrePassivate`: Invocado antes de que el Bean esté en estado `passivate`

Adaptado de: <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEJB3>

Anotaciones EJB

- Anotaciones de un Bean de Entidad
 - @Entity: Indica que es un Bean de Entidad
 - Anotaciones del ciclo de vida: @EntityListeners, @PrePersist, @PreRemove, @PostUpdate, @PersistenceContext, etc.
- Anotaciones de un Bean Manejador de Mensajes
 - @MessageDriven: Indica que la clase es un Bean Manejador de Mensajes
 - Atributos
 - name – por defecto el nombre de la clase pero se puede especificar otra
 - messageListenerInterface –interfaz del Bean message listener
 - activationConfig – otras opciones de configuración para JMS
 - mappedName – si se quiere que el contenedor maneje el objeto indicado de manera específica
 - description - descripción del Bean

Adaptado de: <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEJB3>

Anotaciones EJB

- Anotaciones comunes a los Beans de Sesión y Beans Manejadores de Mensajes
 - @Timeout: Asigna un tiempo de ejecución a un método.
 - @ApplicationException: Excepción a enviar al cliente cuando se produzca.
- Transacciones
 - @TransactionManagement: Especifica quién va a manejar la transacción: el contenedor (TransactionManagementType.CONTAINER) o el Bean (TransactionManagementType.BEAN).
 - @TransactionAttribute: Especifica el método de negocio invocado por el contexto de la transacción únicamente si es manejada por el contenedor. (TransactionAttributeType.REQUIRED (por defecto) | MANDATORY | REQUIRES_NEW | SUPPORTS | NOT_SUPPORTED | NEVER).

Adaptado de: <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEJB3>

Anotaciones EJB

- Interceptores (métodos invocados automáticamente al invocar a los métodos de negocio de un Bean)
 - `@Interceptors`: Interceptores asociados con el Bean
 - `@AroundInvoke`: Designado a un método de un Interceptor
 - `@ExcludeDefaultInterceptors`: Se utiliza para excluir la invocación de un Interceptor en caso de que para un Bean de negocio en concreto no queramos que se llame a ningún método de los Interceptores.
 - `@ExcludeClassInterceptors`: Excluye la invocación de los Interceptores a nivel de clase.
- Métodos del ciclo de vida
 - `@PostConstruct`: Invocado después de que el Bean sea creado (después de hacerse las inyecciones de dependencias).
 - `@PreDestroy`: Invocado después de que el Bean sea destruido del pool del contenedor.

Adaptado de: <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEJB3>

Anotaciones EJB

- Seguridad
 - `@DeclareRoles`: Especifica los roles de seguridad asociados con el Bean.
 - `@RolesAllowed`: Sólo los roles declarados en esta anotación podrá;n invocar el método. Se pueden especificar los roles permitidos a nivel de clase o a nivel de método.
 - `@PermitAll`: Permite la invocación por todos los roles definidos. Se puede especificar a nivel de clase o a nivel de método.
 - `@DenyAll`: Bloquea a todos los roles de manera que ninguno podrá; llamarlo. Únicamente a nivel de método.
 - `@RunAs`: Se aplica a nivel de clase e indica el rol con el que se ejecuta el Bean: `@RunAs("admins")`

Adaptado de: <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEJB3>

Conclusiones

- El modelo EJB permite programar POJOs y simplemente añadiendo anotaciones convertirlos en componentes que pueda manejar un contenedor EJB
 - El contenedor proveerá las propiedades y comportamientos especificados sin necesidad de programarlos explícitamente
 - Ciclo de vida
 - Distribución
 - Transacciones
 - Seguridad
 - Persistencia

Bibliografía

- Eric Jendrock et al. The Java EE 6 Tutorial (2013)
<http://docs.oracle.com/javaee/6/tutorial/doc/>
- Introducción a EJB 3.1 en davidmarco.es
<http://www.davidmarco.es/blog/entrada.php?id=239>
- Otras introducciones sencillas:
<http://www.slideshare.net/PacoGarat/ejb-con-eclipse-y-jboss>
<http://theopentutorials.com/examples/java-ee/ejb3/how-to-create-a-simple-ejb3-project-in-eclipse-jboss-7-1/>
- Documentación oficial
 - Java EE Specifications
<http://www.oracle.com/technetwork/java/javaee/tech/index.html>
 - Enterprise JavaBeans technology:
<http://www.oracle.com/technetwork/java/javaee/ejb-141389.html>
 - API specification for version 6 of Java EE
<http://docs.oracle.com/javaee/6/api/>