Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces

Fernando Pech-May[1], Mario A. Gomez-Rodriguez[1], Luis A. de la Cruz-Diaz[1], Salvador U. Lara-Jeronimo[1]

[1]Instituto Tecnológico Superior de los Ríos.
86930 Km. 3 Carretera Balancán – Villahermosa, Balancán,
Tabasco, México
{fpech, mgomez}@tamps.cinvestav.mx, {aceletes, uciellara}@gmail.com

Resumen. En este artículo se presentan diversas tecnologías de la plataforma Java EE para el desarrollo de aplicaciones web robusta, potente, de alta disponibilidad y que simplifica enormemente su desarrollo. Además se analizan diversas APIs para el desarrollo de aplicaciones empresariales tales como JPA, EJB, JSF y JNDI que cumplan con las especificaciones de los estándares de la tecnología de la plataforma Java.

Keywords: JPA, EJB, JSF, Primefaces

1 Introducción

Las empresas de hoy en día viven en un mundo global competitivo que necesitan aplicaciones para satisfacer las necesidades de negocio, que son cada vez más complejas. Con el avance de las tecnologías web y la Internet, se han abierto nuevas oportunidades para los desarrolladores de aplicaciones empresariales; permitiéndoles el uso de las nuevas tecnologías web en el desarrollo de aplicaciones mucho más robustas, escalables y con un mayor rendimiento. Algunas de las nuevas tecnologías que han surgido son: *JavaServer Faces* (JSF) que es la tecnología estándar de la edición empresarial de Java (*Java Enterprise Edition*, Java EE) para la creación de interfaces de usuario en la web y que permite integrar otras tecnologías como las hojas de estilo en cascada (*Cascade Style Sheet, CSS*) que describen como se va a mostrar un documento, Ajax (*Asynchronous JavaScript And XML*); un modelo de desarrollo web para crear aplicaciones interactivas, JavaBeans empresariales (*Enterprise JavaBeans*, EJB) y el API (*Application Programming Interface*) de Java para el manejo de entidades persistentes (*Java Persistence API*, JPA) sobre bases de datos relacionales.

La Figura 1.1 muestra el conjunto de tecnologías de Java EE que pueden utilizarse para el desarrollo de aplicaciones web. Todas estas tecnologías serán descritas en este documento.

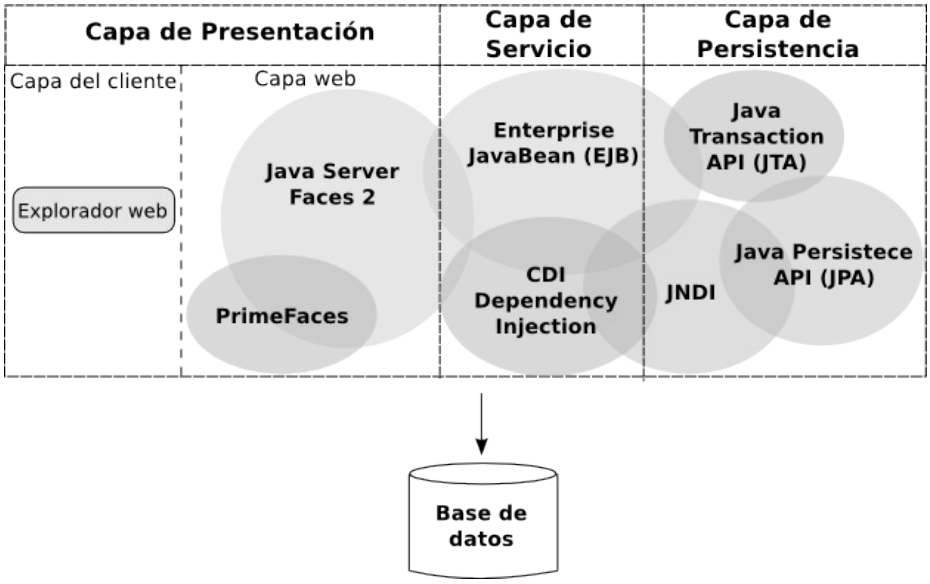

Fig. 1.1. Tecnologías Java EE para el desarrollo de aplicaciones web.

2 Java Enterprise Edition

Java Enterprise Edition [5, 8] (Java EE), fue desarrollado por Sun Microsystems y lanzado en 1999 con el nombre de J2EE. Proporciona un conjunto de especificaciones técnicas para el desarrollo de aplicaciones empresariales. Puede ser visto como una extensión de Java SE para facilitar el desarrollo de aplicaciones distribuidas, robustas, potentes y de alta disponibilidad.

Java EE define cuatro tipos de componentes:

1. *Applets:* Aplicaciones GUI que se ejecutan en un navegador.
2. *Aplicaciones:* Son programas que se ejecutan en un cliente
3. *Aplicaciones web:* (servlets, páginas JSP y JSF) Se ejecutan en un contenedor web y responden a las peticiones HTTP del cliente.
4. *Aplicaciones Empresariales*: (EJB, JMS, JTA, etc.) Son ejecutadas en un contenedor EJB.

Java EE se divide en dominios lógicos llamados contenedores (ver Figura 2.1). Cada contenedor tiene una función específica, soporta un conjunto de APIs y ofrece servicios a los componentes tales como *seguridad*, *acceso a base de datos*, *gestión de transacciones*, *nombres de directorios*, e *inyección de recursos*. Los contenedores

ocultan la complejidad técnica y mejoran la portabilidad. El contenedor EJB es responsable de administrar la ejecución de los *beans*[1] que contiene la lógica de negocio.

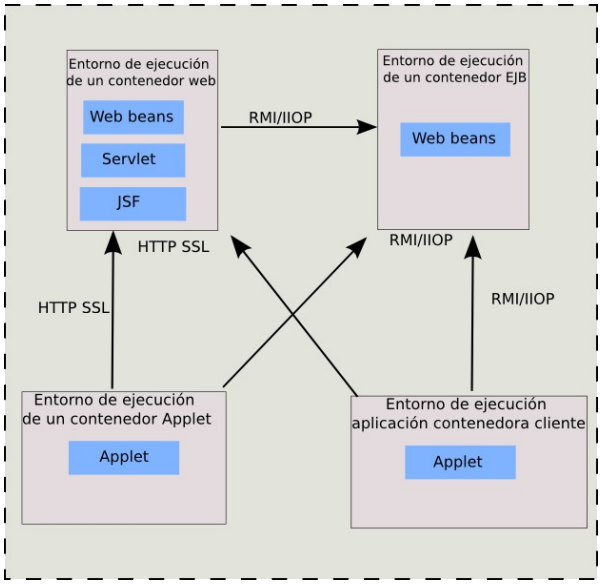

Fig. 2.1. Estándar de los contenedores Java EE.

Los contenedores proporcionan servicios subyacentes a sus componentes desplegados (ver Figura 2.2), esto permite al desarrollador centrarse en la lógica de aplicación en lugar de resolver problemas técnicos. Algunos de los servicios que proporciona Java EE se describen a continuación:

1. *Java Transaction API (JTA)*: Este servicio ofrece una demarcación de transacciones API utilizada por el contenedor y la aplicación. También proporciona una interfaz entre el administrador de transacciones y el administrador de recursos en el nivel *Service Provider Interface* (SPI).
2. *Java Persistence API (JPA):* API estándar para el mapeo de objeto-relacional (ORM). Con Java Persistence Query Language (JPQL), se puede consultar objetos almacenados en la base de datos subyacente.
3. *Validación*: El Bean de validación proporciona un nivel de declaración de restricción de la clase y la facilidad de validación.
4. *Java Message Service (JMS)*: Permite que los componentes se comuniquen de forma asincrónica a través de mensajes.
5. *Java Naming and Directory Interface (JNDI):* Esta API, incluida en Java SE, se utiliza para acceder a los sistemas de nombres y directorios. La aplicación se utiliza para asociar (enlazar) los nombres de los objetos y luego encontrar

[1] Componente de software reutilizable que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.

estos objetos (búsqueda) en un directorio. Puede buscar fuentes de datos, JMS, EJB y otros recursos.

6. *JavaMail*: Muchas aplicaciones requieren la capacidad de enviar correos electrónicos que pueden ser implementadas a través del uso de la API JavaMail.

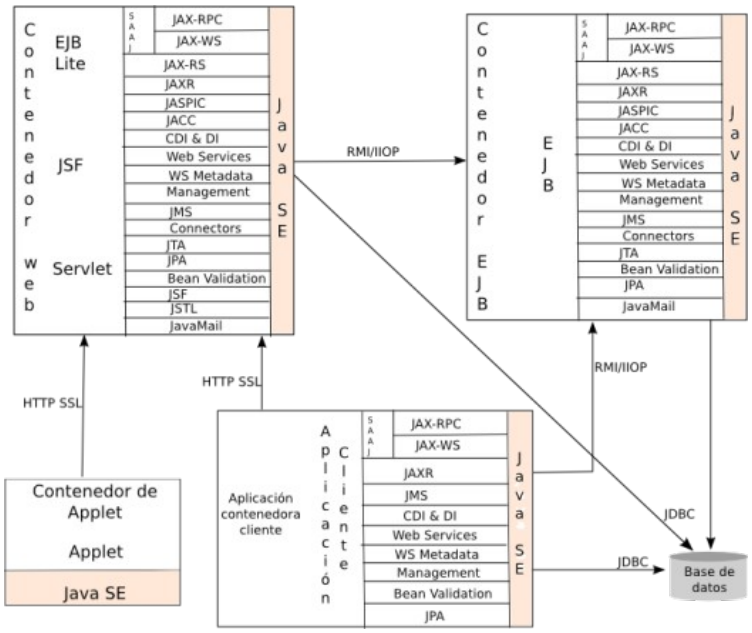

Fig. 2.2. Servicios proporcionados por los contenedores.

3 Java Server Faces (JSF)

Java Server Faces [3, 4] (JSF) es un estándar de Java hacia la construcción de interfaces de usuario para aplicaciones web que simplifican el desarrollo de aplicaciones web del lado del cliente, JSF está basado en la tecnología Java EE. En el 2009 se dio a conocer la nueva versión JSF 2.0, que contiene algunas características y/o mejoras con respecto a las versiones anteriores (JSF 1.0, JSF 1.1 y JSF 1.2) como son: *Mejoras en la navegación:* navegación condicional, inspección en tiempo de ejecución en las reglas de navegación. *Control de excepciones:* permite fácilmente la creación de una página de error que utiliza componentes JSF. *Mejoras en la expresión del lenguaje:* compatibilidad con métodos arbitrarios incluyendo el paso de parámetros. *Validación:* es una nueva especificación java desarrollada para la validación de beans.

Una página JSF utiliza la extensión *.xhtml, es decir, una combinación de XML con HTML y puede incluir componentes como CSS, JavaScript, entre otros.

La especificación de JSF define seis fases distintas en su ciclo de vida:

1. *Restauración de la vista:* Crea un árbol de componentes en el servidor para representar la información de un cliente.
2. *Aplicar valores de la petición:* Actualiza los valores del servidor con datos del cliente.
3. *Proceso de validación:* Valida los datos del usuario y hace la conversión.
4. *Actualización de valores del modelo:* Actualiza el modelo del servidor con nuevos datos.
5. *Invocar la aplicación:* Ejecutar cualquier lógica de aplicación para cumplir con la solicitud.
6. *Procesar la respuesta:* Guarda un estado y da una respuesta al cliente.

4 Java Persistence API (JPA)

JPA [2, 7] 1.0 fue creado con Java EE 5 para resolver problemas de persistencia de datos. Proporciona un modelo de persistencia para mapear bases de datos relacionales (ver Figura 4.1). En java EE 6, JPA 2.0 sigue el mismo camino de simplicidad y robustez y agrega nuevas funcionalidades. Se puede utilizar para acceder y manipular datos relacionales de Enterprise Java Beans (EJBs), componentes web y aplicaciones Java SE.

JPA es una abstracción que está por encima de JDBC lo que permite ser independiente de SQL. Todas las clases y anotaciones de esta API se encuentran en el paquete *javax.persistence*. Los principales componentes de JPA son:

- Mapeo de base de datos relacionales (ORM). Es el mecanismo para mapear objetos a los datos almacenados en una base de datos relacional.
- Un API administrador de entidad para realizar operaciones en la base de datos tales como crear, leer, actualizar, eliminar (CRUD).
- El Java Persistence Query Language (JPQL) que permite recuperar datos con un lenguaje de consultas orientado a objetos.
- Las transacciones y mecanismos de bloqueo cuando se accedan a los datos concurrentemente, la API Java Transaction (JTA).

Una entidad es objeto de dominio de persistencia. Por lo general, una tabla en el modelo de datos relacional es representada por una entidad y sus instancias corresponden a los registros de dicha tabla. El estado de persistencia de una entidad es representado por propiedades persistentes, estas propiedades a su vez usan anotaciones para el mapeo de las entidades y relaciones entre entidades. Las

relaciones entre entidades persistentes deben mapearse explícitamente como llaves foráneas o uniones de tablas, la manera en que se estructura una entidad, sus atributos y sus relaciones (ver Figura 4.2).

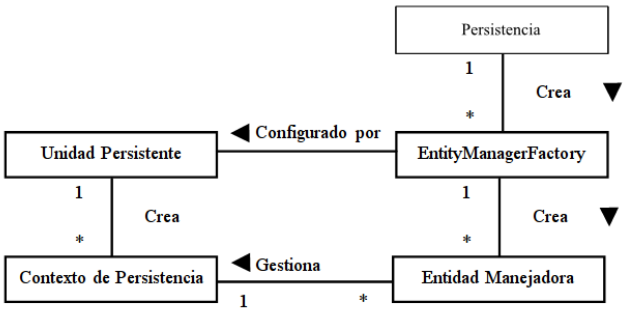

Fig. 4.1. Relación entre los conceptos de JPA.

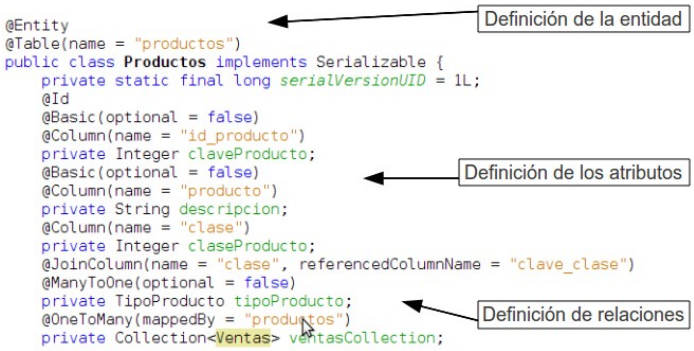

Fig. 4.2. Definición de atributos y relaciones de una Entidad.

5 EJB

Los JavaBeans empresariales (*Enterprise JavaBeans* [1, 6], EJB) son una tecnología (API) que forma parte del estándar de Java EE. Están diseñados para desarrollo y despliegue de aplicaciones (distribuidas) de negocio basadas en componentes del lado del servidor. Una vez que se desarrolla una aplicación, ésta puede ser desplegada en cualquier servidor que soporte la especificación de EJB. Con esta tecnología es posible desarrollar aplicaciones empresariales sin tener que crear de nuevo los servicios de transacción, seguridad, persistencia, concurrencia y lo que se pueda necesitar en el proceso de creación de una aplicación; permitiendo a los desarrolladores enfocarse en la implementación de la lógica de negocio.

EJB divide la capa de negocio en dos partes: *Capa de lógica de negocio* donde se encuentra EJB y *capa de persistencia*. EJB cuenta con dos componentes de proceso de negocio, los beans de sesión (*Session Beans*) y los beans dirigidos por mensajes (*Message-Driven Beans*, MDBs), ambos son desarrollados por una aplicación de

negocio e implementados y ejecutados por el Contenedor de EJB. A continuación se explican los dos componentes:

5.1 Session Beans

Los beans de sesión (Session Beans) son componentes Java que se pueden ejecutar tanto en contenedores EJB independientes como en contenedores EJB que son parte del estándar Java EE. Dichos componentes son típicamente utilizados para modelar una tarea particular del usuario o un caso de uso, tal como introducir la información del cliente o implementar un proceso que mantiene un estado de la conversación con una aplicación cliente.

Existen varios tipos de beans de sesión, como los beans de sesión sin estado (Stateless Session Beans, SLSBs), beans de sesión con estado (Stateful Session Beans , SFSBs) y un tipo particular de beans; los beans de sesión singleton (Singleton Beans) [1].

- SLSBs: Son útiles para los casos en que el estado no necesita ser mantenido de invocación a invocación. El cliente no puede asumir que las solicitudes posteriores utilizarán una instancia particular del bean. El contenedor puede destruir o crear nuevas instancias según determine que es más eficiente (Figura 5.1 a).
- SFSBs: Difieren de los SLSBs en que se garantiza que todas las peticiones invoquen la misma instancia del bean, tal como se muestra en la Figura 5.1 b. Cada SFSB contiene un contexto de sesión aislado por lo que las llamadas de una sesión no afectan a las demás. Las sesiones de estado y sus correspondientes instancias de bean son creadas en algún momento antes de la primera invocación a su instancia objetivo. Viven hasta que el cliente invoca un método que el proveedor del bean ha marcado como un evento para remover el bean, o hasta que el contenedor decide desalojar la sesión (usualmente debido a un tiempo de espera).
- Singleton beans: Se utilizan cuando solamente se necesita mantener una sola instancia de los objetos de negocio. Debido a que todas las peticiones hacia un singleton son dirigidas hacia la misma instancia, el contenedor no realiza mucho trabajo en seleccionar la instancia objetivo, como se ve en la Figura 5.1 c.

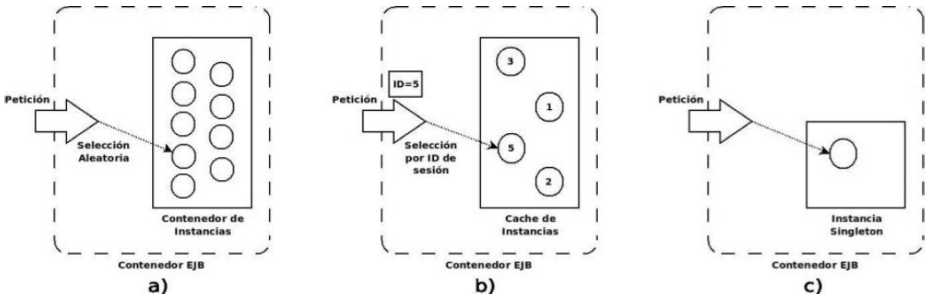

Fig. 5.1. Tipos de beans de sesión. a) Selección aleatoria de una instancia SLSB. b) Selección de una instancia SFSB de acuerdo a su ID de sesión. c) Singleton Bean que cuenta con una sola instancia de soporte.

5.2 Message-Driven Beans

La mensajería asíncrona es un paradigma en el cual dos o más aplicaciones se comunican a través de mensajes que describen un evento de negocio. Un proveedor común de mensajería asíncrona es el servicio de mensajería de Java (Java Message Service, JMS), y la especificación de EJB2 dicta que JMS es soportado de manera implícita. Si un mensaje es enviado a una fila JMS, un MDB puede ser creado para manejar el evento.

Al igual que los SLSBs, en los MDBs cualquier instancia puede ser utilizada para atender un mensaje, el cliente no tiene conocimiento del MDB.

En términos generales, EJB se puede ver como una plataforma para la creación de aplicaciones empresariales portables, reutilizables y escalables utilizando el lenguaje de programación Java y el estándar Java EE.

6 PrimeFaces

PrimeFaces [11] es una librería de componentes visuales de código abierto para el conjunto Java Server Faces 2.0 desarrollada y mantenida por Prime Technology. Su objetivo principal es ofrecer un conjunto de componentes para facilitar la creación y diseño de aplicaciones web.
Los componentes de PrimeFaces cuentan con soporte nativo de Ajax, pero no se encuentra implícito, de tal manera que se tiene que especificar que componentes se deben actualizar al realizar una petición proporcionando así mayor control sobre los eventos. Cuenta también con un modulo adicional TouchFaces para el desarrollo de aplicaciones web para dispositivos móviles con navegadores basados en WebKit.

Las principales características de PrimeFaces son:

- Soporte nativo de Ajax, incluyendo Push/Coment.
- Kit para crear aplicaciones web móviles.
- Es compatible con otras librerías de componentes como Jboss RichFaces.
- Uso de JavaScript no intrusivo.
- Es un proyecto open source, activo y estable.

7 Conclusiones

Java EE, es sin duda la plataforma de desarrollo de aplicaciones más popular en el ámbito empresarial. Ofrece una serie de ventajas que hacen posible el desarrollo de aplicaciones potentes, flexibles, extensibles, ligeras, etc. Elimina la complejidad en cuanto a código y cuenta con la posibilidad de reutilizar código en los archivos de aplicaciones web, además de incorporar anotaciones mediante la plataforma para facilitar el desarrollo de aplicaciones. Actualmente, hemos utilizado estas tecnologías para el desarrollo de un sistema de control de actividades científicas y tecnológicas de los investigadores del estado de Tabasco.

Agradecimientos. Esta investigación fue parcialmente financiada mediante el proyecto No. TAB-2010-C19-144199 del Fondo Mixto CONACYT-Gobierno del Estado de Tabasco.

Referencias

1. A. Lee Rubinger E Bill Burke. Enterprise JavaBeans 3.1. Safari, 6 Ed.(2010)
2. D. Yang. Java Persistence with JPA. Outskirts Press. (2010)
3. C. Schalk and E. Burns. JavaServer Faces 2.0: The Complete Reference. MC Graw Hill (2010)
4. D. Geary and C. Horstmann. Core JavaServer Faces. Prentice hall, 3 ed. (2010)
5. A. Goncalves. Beginning Java EE 6 Platform with GlassFish 3, Apress, 2 ed. (2010)
6. M. Keith and M. Schincariol. Pro EJB 3 Java Persistence API, Apress, (2006)
7. M. Keith and M. Schincariol. Pro JPA 2 Mastering the Java Persistence API. Apress, (2009)
8. N., C. Zakas, J. Mcpeak and J. Fawcett. Professional Ajax. Wiley, 2 ed. (2007)
9. Y. Vasiliev. Beginning Database-Driven Application Development in Java EE Using GlassFish. Apress, (2008)
10. J. J. Garrett. Ajax: A New Approach to Web Applications. url http://www.adaptivepath.com/ideas/essays/archives/000385.php, 2005.
11 Tutorial PrimeFaces, Url: http://primefaces.org/documentation.html