

Tema 5. Plataforma Java EE

SCS – Sistemas Cliente/Servidor
4º informática

<http://ccia.ei.uvigo.es/docencia/SCS>

septiembre 2011

5.1 Introducción a Java EE

Java EE (*Java Enterprise Edition*): Plataforma Java para el desarrollo de aplicaciones empresariales

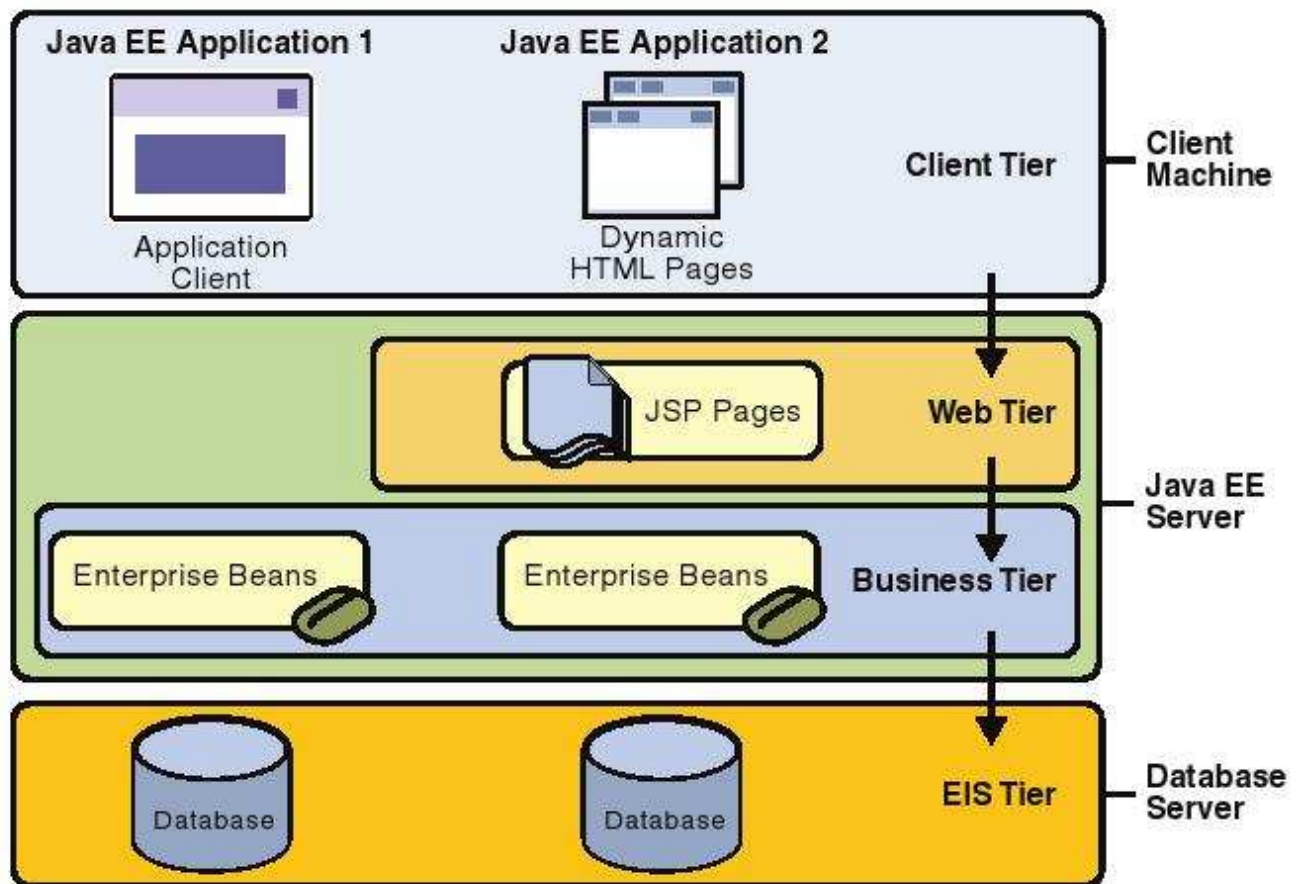
- Ofrece un *framework* para el desarrollo de aplicaciones distribuidas multicapa (*3-tier*, *n-tier*) basadas en Web
 - Define una infraestructura común básica para el acceso a bases de datos, gestión de la persistencia, control de seguridad, gestión de transacciones, ...
 - Separación clara entre presentación (interfaz), modelo (lógica) de negocio y datos
 - Plataforma basada en componentes
 - Entes (objetos) intercambiables que residen en un servidor de aplicaciones y son gestionados por él
 - <http://www.oracle.com/technetwork/java/javaee/>
- Java EE es un conjunto de especificaciones de APIs Java (no es un producto/aplicación en si mismo)
 - Describe el conjunto de paquetes, interfaces y clases Java que debe de ofrecer un *framework* Java EE
 - Define también una *test suite* (ejemplos de referencia) que todo servidor de aplicaciones Java EE debe soportar y una colección de documentos (guías de diseño) explicativos
 - La implementación de esas API es responsabilidad del fabricante
 - Implementaciones propietarias
 - ◇ IBM: WebSphere AS 8.0
 - ◇ Oracle: Oracle Application Server
 - Implementaciones libres:
 - ◇ Red Hat: JBoss AS 7.0 (<http://www.jboss.org>)
 - ◇ Oracle: GlashFish v3.x (<http://glassfish.java.net>)
 - ◇ Apache: Gerónimo [Tomcat+OpenEJB] (<http://geronimo.apache.org>)
 - Servidores certificados:

<http://www.oracle.com/technetwork/java/javaee/overview/compatibility-jsp-136984.html>

- Punto clave: independencia del fabricante
 - Una aplicación distribuida Java EE que use las APIs estándar se podrá desplegar en distintos servidores de aplicaciones sin necesidad de modificación

- Java EE se asienta sobre Java SE (*Java Standard Edition*)
 - Java SE provee la infraestructura de ejecución (*Java Virtual Machine* + APIs básicas) y de compilación (`javac`, `rmic`, ...)
 - Variantes de la plataforma Java
 - *Java EE*: aplicaciones distribuidas multicapa sobre Web
 - *Java SE*: aplicaciones de escritorio y applets
 - *Java ME*: (*Java MicroEdition*) aplicaciones para dispositivos móviles
 - ◇ Versión simplificada de Java SE + APIs específicas (localización, etc...)
- Versión actual: **Java EE 6.0**
 - A partir de Java EE 5.0 cambios importantes respecto a versiones previas: J2EE 1.4 y anteriores
 - **Objetivo**: simplificación del modelo de desarrollo
 - Uso extensivo de nuevas funcionalidades introducidas en la versión 5 de Java SE (jdk 1.5)
 - Uso de **anotaciones** Java para añadir metainformación al código fuente que será explotada por el entorno de ejecución Java EE (contenedores de aplicaciones)
 - Reemplazan en parte a los *descriptores de despliegue* (ficheros XML de configuración)
 - Uso de **inyección de dependencias** (*Dependency Injection*) para simplificar el desarrollo de las aplicaciones Java EE, facilitar su instalación (despliegue) y reducir el acoplamiento
 - Ciclo vida de los objetos gestionado por el servidor de aplicaciones
 - A partir de la metainformación (anotaciones) el contenedor/servidor Java EE es capaz de "*inyectar*" referencias a otros objetos en determinados atributos de los componentes JEE sin necesidad de que el propio componente lo tenga que hacer por sí mismo
 - En el código de los componentes JEE se usan anotaciones especiales para marcar atributos cuyos valores serán "*rellenados*" por el contenedor JEE en el momento en que sean desplegados (tiempo de ejecución).
 - **Idea**: el componente no tiene que preocuparse de hacer `new()` o consultar un servidor de nombres (JNDI), ni de configurar el objeto referenciado

(a) Esquema general Java EE

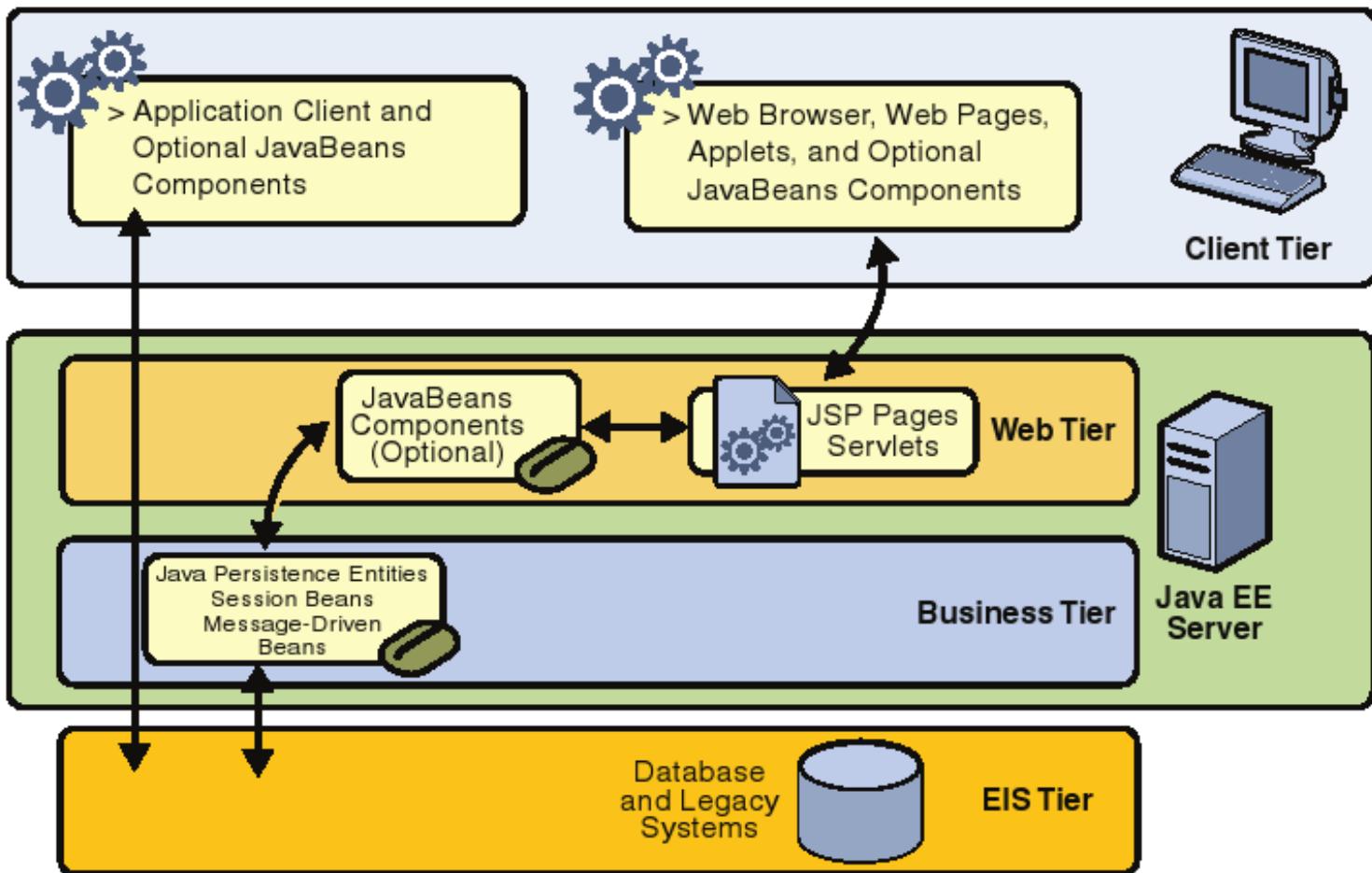


(b) Componentes de la arquitectura Java EE

Tipos de componentes:

- Clientes de la aplicación {
 - clientes web [navegador]
 - clientes de escritorio Java [swing, awt]
 - clientes de escritorio no Java
- Componentes Web: *servelts*, páginas JSP (*Java Server Pages*), JSF (*Java Server Faces*), ...
 - Se ejecutan en el contenedor Web
 - Responsables de componer la presentación de datos en formato HTML
 - Suelen apoyarse en el uso de componentes Java Beans
 - **Java Beans:** objetos Java que verifican 2 requisitos
 1. tienen un constructor sin argumentos
 2. todos sus atributos son accesibles mediante pares de métodos *get()* y *set()*

- Componentes de negocio: EBJ (*Enterprise Java Beans*) y entidades JPA (*Java Persistence API*)
 - Se ejecutan en el contenedor de EBJs
 - Componentes (objetos Java) responsables de implementar la lógica de la aplicación
 - EBJ gestionan interacciones con los clientes e implementan reglas de negocio
 - Entidades Java: objetos persistentes que representan los datos de la capa EIS
- Capa EIS (*Enterprise Information Systems*)
 - Capa de datos ⇒ gestiona la información permanente del sistema
 - Bases de datos o aplicaciones empresariales "heredadas" (*legacy systems*) que actúan como almacenes de datos



Nota: En general, se entiende por *componente* un objeto Java con restricciones especiales que se ensambla y ejecuta dentro de un servidor de aplicaciones Java EE (contenedor)

(c) Contenedores Java EE

- Ofrecen servicios a los componentes, actuando como interfaz entre un componente y los aspectos de bajo nivel de cada plataforma
 - Simplifican el desarrollo de los componentes ocultando los detalles complejos
 - Sus tareas concretas depende de la capa a la que pertenezcan
- Servicios ofrecidos: seguridad, gestión de transacciones, consulta de directorios de nombres (JNDI), etc
- Tipos de contenedores en la plataforma Java EE
 - **Servidor de Aplicaciones Java EE:** entorno de ejecución de aplicaciones JEE
 - Proporciona el contenedor Web y/o el contenedor de EJBs
 - **Contenedor EJB** (*Enterprise Java Beans*): gestiona la ejecución (ciclo de vida) de los EJBs
 - Da soporte a los componentes que implementan la capa de lógica de negocio
 - Ofrece servicios de control de acceso y seguridad, control de transacciones y ejecución concurrente, etc
 - **Nota:** Hay otras alternativas que ofrecen funcionalidades similares para implementar la lógica de negocio sin tener que usar EJBs (ej.: framework *Spring*)
 - Ejemplos libres: JBoss, OpenEJB, ...
 - **Contenedor Web:** gestiona la ejecución de los *servlets* y paginas JSP
 - Da soporte a los componentes que implementan la capa Web
 - API de *Servlets*, APIs JSP, JSTL, Framework JSF, JavaBeans
 - Ejemplos libres: Tomcat, Jetty, ...
 - **Contenedor aplicación cliente:** ofrece la infraestructura necesaria para la ejecución del cliente

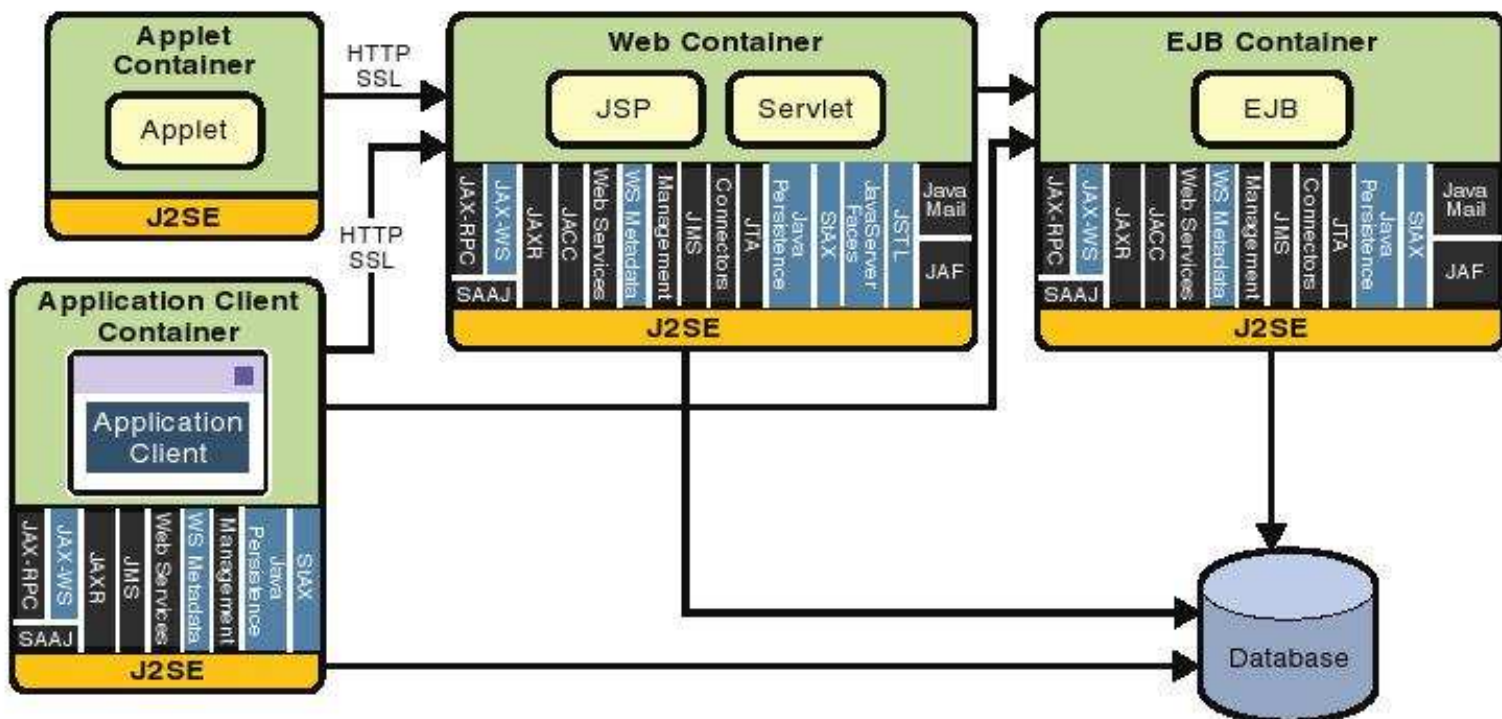
(d) APIs de Java EE 6.0

Java EE se basa en Java SE

- Todas las APIs de Java SE están disponibles
- En Java EE se definen nuevas API específicas para el desarrollo de sistemas distribuidos multicapa
 - Algunas han acabado pasando a Java SE (ej. JDBC, JPA)

En Java EE 6.0 se introducen el concepto de *profiles* → los servidores de aplicaciones no están obligados a implementar todas las APIs de la especificación

1. *Web profile*: incluye sólo parte de las APIs, fundamentalmente las orientadas a construcción de aplicaciones Web en Java
 - a) APIs incluidas: Servlet 3.0, JSF 2.0, JSP 2.2, JSTL 1.2, CDI, EJB 3.1 lite, JPA, JTA, BeanValidation
 - b) Omite: EJB 3.1 full, APIs de servicios web (JAX-WS, JAX-RS), JCA, JMS, APIs de seguridad, etc
2. *Full profile*: incluye todas las APIs de la especificación



Tecnologías y APIs más relevantes

- **Enterprise Java Beans (ver. 3.1):** Definición de *componentes* EJB que ejecutan los procesos de negocio
 - *session beans*: componentes de negocio que gestionan la "conversación" con el cliente
 - Ofrecen un punto de entrada (interfaz) mediante el cual los clientes (Web, escritorio) invocan los procesos de la lógica de negocio
 - 3 tipos: $\left\{ \begin{array}{l} \text{con estado (stateful session bean)} \\ \text{sin estado (stateless session bean)} \\ \text{singleton (singleton session bean)} \end{array} \right.$
 - Permiten la invocación local (dentro del contenedor JEE) y/o remota (mediante RMI/IIOP)
 - *message-driven beans*: componentes de negocio que reciben y gestionan invocaciones asíncronas
 - **Nota:** hasta J2EE 1.4 existían los *entity beans*, desde JEE 5 han sido reemplazados por las *entidades* JPA (objetos persistentes)
- **Java Persistence API (JPA) (ver. 2.0):** soporte estándar para persistencia de objetos
 - Permite definir un mapeo objeto-relacional para tipos de entidades (mediante anotaciones o ficheros XML)
 - *Entidad*: objeto Java que representa los datos almacenados en una tupla de una BD
 - JPA gestiona la consulta, carga, modificación y escritura de entidades
 - Salva las diferencias entre modelo relacional y modelo Orientado a Objetos
 - Se asienta sobre el API JDBC (acceso a BD SQL)
- **Contexts and Dependency Injection for the Java EE Platform (CDI):** amplía y generaliza el soporte para la inyección de dependencias y la gestión del ciclo de vida de los objetos basada en contextos.
 - Simplifica la integración de las capas Web (JSF) y de lógica de aplicación (EJB)
- **Java Database Connectivity (JDBC):** soporte para acceso a sistemas gestores de base de datos e invocación de sentencias SQL
- **API de Servlets (ver. 3.0):** *servlets* son objetos Java que manejan peticiones HTTP

- **Java Server Pages (JSP)** (ver. 2.2): lenguaje de marcado que combina Java y HTML
 - Internamente se compilan en *servlets*
- **JSP Standard Tag Library (JSTL)** (ver. 1.2): juego de etiquetas estándar para definir páginas JSP a alto nivel
- **Java Server Faces (JSF)** (ver 2.0): Framework MVC (modelo-vista-controlador) basado en componentes del lado servidor.
- **Java API for XML Processing (JAXP)** y **Java Architecture for XML Binding (JAXB)** (ver. 2.2): soporte para el procesamiento de documentos XML
- **Java API for XML Web Services (JAX-WS)** (ver. 2.2): soporte para invocación y publicación de Servicios Web (protocolos SOAP, WSDL, UDDI)
- **Java API for RESTful Web Services (JAX-RS)** (ver. 1.1): soporte para invocación y publicación de Servicios Web basados en la arquitectura REST
- **Java Naming and Directory Interface (JNDI)**: soporte para el acceso (inserción y consulta) a servicios de directorio
 - Permite almacenar (asociar nombre) y recuperar información, recursos, referencias a objetos (EJB), etc
- **Otras:** *Java Message Service(JMS)*, *Java Transaction API(JTA)*, *Java Mail*, *Java Authentication and Authorization Service(JAAS)*, *Java EE Connector Architecture (JCA)* ...