aula MENTOR

GOBIERNO DE ESPAÑA

MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

Programación web en Java

AULA MENTOR

educacion.es

ISBN: 978-84-369-5430-2
Nipo: 030-12-335-9

Autores:
José Miguel Ordax Cassá
Pilar Aranzazu Ocaña Díaz-Ufano

Ilustración de portada:
María Guija Medina

Unidad de Aprendizaje 1

INTRODUCCIÓN A JAVA EE

ÍNDICE

1.1 Introducción

Debido a la naturaleza del lenguaje Java: portable, seguro, multithread, etc…, está siendo utilizado en multitud de ámbitos y tecnologías, desde el chip de una tarjeta de crédito hasta un servidor de la más alta gama.

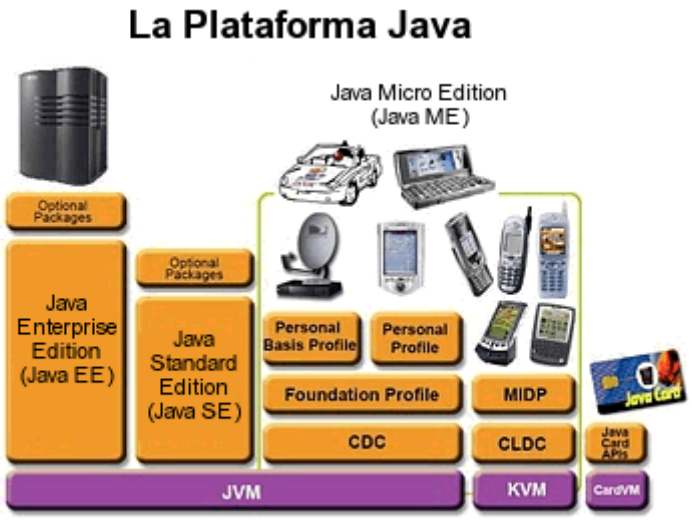
Evidentemente, estos distintos ámbitos o entornos, tienen unas características y peculiaridades muy distintas entre sí. Por ejemplo, la cantidad de memoria disponible en el chip de una tarjeta de crédito y la de un servidor es muy distinta, por lo que habrá que tenerlo en cuenta a la hora de desarrollar las aplicaciones.

Es por ello que existen distintas plataformas Java, dependiendo del ámbito en el que se vaya a trabajar.

Son las siguientes:

- Java Card.
- Java Micro Edition (Java ME).
- Java Standard Edition (Java SE).
- Java Enterprise Edition (Java EE).

Podemos verlas resumidas en el siguiente gráfico:



1.1.1 Java Card

La plataforma Java Card define las APIs y requerimientos necesarios para poder ejecutar aplicaciones Java en los chips de las tarjetas. Debido a las mínimas prestaciones del entorno de ejecución contiene el API más escueto.

El estudio de esta plataforma no es el objetivo de este curso, pero si el alumno quiere profundizar en este tema podrá encontrar más información en la siguiente URL:

http://www.oracle.com/technetwork/java/javame/javacard/overview/getstarted/index.html

1.1.2 Java Micro Edition (Java ME)

La plataforma Java Micro Edition define las APIs y requerimientos necesarios para poder ejecutar aplicaciones Java en dispositivos embebidos. Debido a la gran diversidad de estos dispositivos, desde teléfonos móviles o buscas con pocas prestaciones hasta televisores o automóviles mucho más potentes, se definieron distintas configuraciones con más o menos APIs y por tanto, posibilidades.

Por defecto existen estos dos:

- CLDC (Connected Limited Device Configuration): Define las APIs y la JVM (denominada KVM) para dispositivos con muy pocas prestaciones.
- CDC (Connected Device Configuration): Define las APIs para dispositivos con pocas prestaciones pero conectados a la red. No requiere una JVM especial.

El estudio de esta plataforma no es el objetivo de este curso, pero si el alumno quiere profundizar en este tema podrá encontrar más información en la siguiente URL:

http://www.oracle.com/technetwork/java/javame/index.html

1.1.3 Java Standard Edition (Java SE)

La plataforma Java Standard Edition (Java SE) define las APIs y requerimientos necesarios para poder ejecutar aplicaciones Java de escritorio en ordenadores personales o portátiles.

El estudio de esta plataforma no es el objetivo de este curso, pero si el alumno quiere profundizar en este tema, Aula Mentor cuenta con otros dos cursos que la cubren:

- Programación en Java – Inicial
- Programación en Java – Avanzado

Puedes consultar la información de estos cursos en www.aulamentor.es, en el apartado “Cursos” – “Programación”.

1.1.4 Java Enterprise Edition (Java EE)

La plataforma Java Enterprise Edition (Java EE) define las APIs y requerimientos necesarios para poder ejecutar aplicaciones Java servidoras, con todo lo que ello supone: cliente-servidor, multiusuario, transaccionalidad, escalabilidad, etc…en definitiva, características que no eran importantes o imprescindibles en aplicaciones de escritorio.

Se apoya en la plataforma Java SE, por lo que es imprescindible conocer y dominar dicha plataforma antes de aventurarse en esta otra. Utiliza la misma Máquina Virtual Java (JVM).

Este curso se centra precisamente en esta plataforma, aunque no en su totalidad. Como veremos un poco más adelante, las aplicaciones Java EE pueden dividirse en varias capas y este curso se centra la denominada capa web o de presentación.

La plataforma Java Enterprise Edition ha pasado por distintas nomenclaturas y versiones a lo largo de su vida, de manera que hubo momentos en el tiempo en el que se denominó Java 2 y luego simplemente Java. Las versiones también han sufrido modificaciones, comenzando a numerarse como 1.2, 1.3… para a posteriori pasar a numerarse como 5.0, 6.0…

Actualmente estamos en la versión 6.0, pero en su totalidad han existido:

- Java 2 EE 1.2

- Java 2 EE 1.3
- Java 2 EE 1.4
- Java EE 5.0
- Java EE 6.0

La definición de esta y las otras plataformas se realiza mediante un proceso colaborativo entre distintas empresas denominado Java Community Process (JCP). Para cada plataforma, API, funcionalidad… se crea lo que se denomina como Java Specification Request (JSR) donde se sientan las bases y especificaciones de dicha plataforma, API o funcionalidad.

En concreto la discusión y definición de la futura plataforma Java EE 7.0 se puede seguir en la siguiente URL: http://www.jcp.org/en/jsr/detail?id=342

1.2 El modelo de aplicación Java EE

El modelo de aplicaciones Java EE define una arquitectura para implementar servicios como aplicaciones multicapa que aseguren la escalabilidad, accesibilidad y facilidad de gestión necesarias en un ámbito empresarial.

El modelo divide el trabajo a realizar en la implementación en dos partes:

- La lógica de presentación y de negocio a implementar por el desarrollador.
- Los servicios estándar que ofrece la plataforma Java EE.

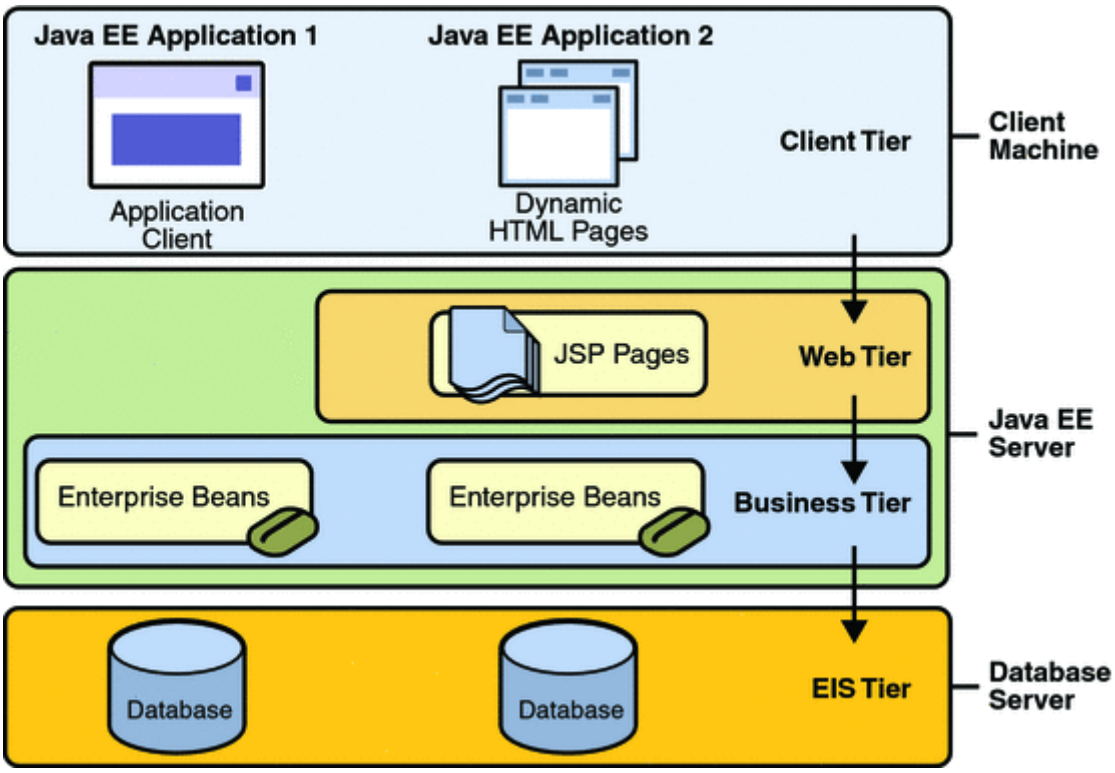
El desarrollador puede apoyarse en los servicios ofrecidos por la plataforma Java EE en vez de reinventar la rueda una y otra vez, facilitándole así el concentrarse únicamente en la lógica específica de su aplicación.

La plataforma Java EE utiliza un modelo de programación distribuido en distintas capas. La lógica de la aplicación se divide en distintos componentes dependiendo de su funcionalidad, y estos son desplegados en las distintas capas dependiendo de a cuál pertenecen.

Las distintas capas son:

- Capa cliente (Client Tier): responsable de la interacción con el usuario.
- Capa web (Web Tier): responsable del control de la aplicación y en ocasiones también de la interacción con el usuario.
- Capa de negocio (Business Tier): responsable de la lógica de la aplicación propiamente dicha.
- Capa de datos (EIS Tier): responsable de la persistencia de datos y/o lógica especializada (conocida con el nombre de EIS: Enterprise Information System, o Sistema de Información Empresarial). Por ejemplo ERPs, BBDD, Motores Transaccionales (CICS, IMS, Tuxedo…).

A continuación mostramos este concepto de multicapa en un diagrama:



Es muy importante tener en cuenta, que esta división es puramente lógica y no física. Es decir, físicamente cada capa no tendrá por qué estar en máquinas independientes, sino que podrán compartir hardware. Por ejemplo, veremos que lo normal será que el entorno de desarrollo que montaremos para resolver las distintas actividades de este curso tendrá todas las capas físicamente en la misma máquina.

Adicionalmente al tema multicapa, el modelo de aplicación Java EE define otros tres conceptos claves para entender la plataforma:

- Componentes: Unidades de software que forman o componen la aplicación.
- Contenedores: Entorno de ejecución donde se ejecutan los componentes.
- Servicios: Funcionalidades estándar que todo contenedor debe proveer a los componentes.

Veamos qué son y en qué consisten cada uno de estos conceptos.

1.2.1 Componentes Java EE

Una aplicación Java EE está compuesta de componentes. Un componente Java EE es una unidad de software funcional auto contenida que se ensambla como parte de una aplicación Java EE y que puede interactuar con otros componentes.

Las especificaciones Java EE definen lo siguientes tipos de componentes:

- Componentes cliente: son aplicaciones Java SE (AWT/Swing, Applets) o un navegador web (Firefox, Chrome, IExplorer…). Se despliegan en la capa cliente.

- Componentes web: son Java Servlets, JavaServer Pages (JSP) o JavaServer Faces (JSF). Se despliegan en la capa web.
- Componentes de negocio: Enterprise JavaBeans (EJB). Se despliegan en la capa de negocio.

1.2.2 Contenedores Java EE

Normalmente el desarrollo de una aplicación empresarial es muy complicado dado que el desarrollador tiene que tener en cuenta temas muy importantes como la gestión multiusuario, la gestión de la transaccionalidad, la gestión de la seguridad, la compartición de recursos, etc…

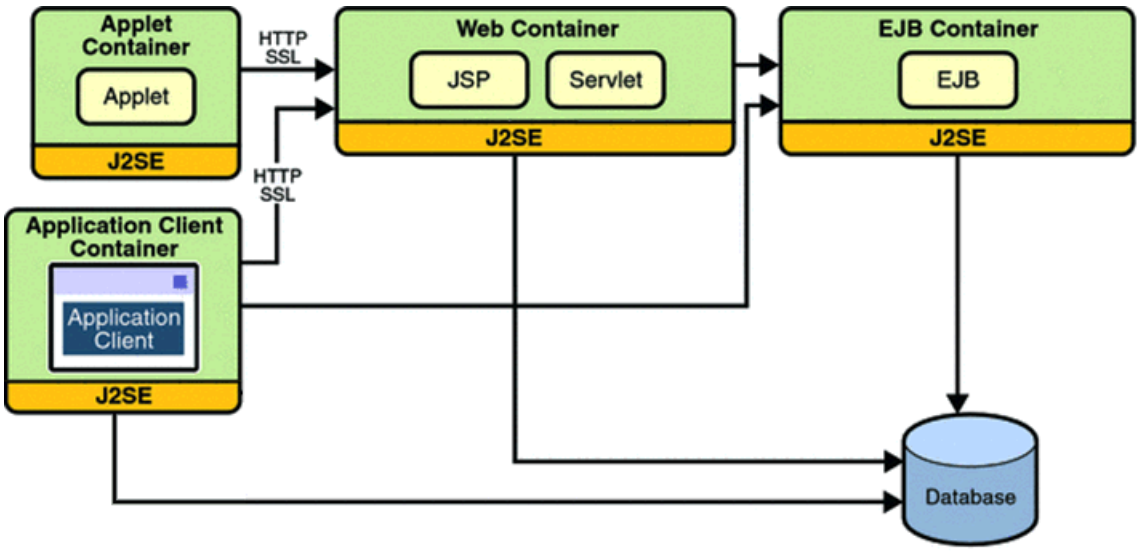
El modelo de programación de la plataforma Java EE facilita enormemente esta tarea con la definición de los contenedores Java EE. Estos contenedores ofrecen al desarrollador una serie de servicios sobre los que se puede apoyar permitiéndole centrarse en el desarrollo de la lógica de negocio de la aplicación propiamente dicha.

Dependiendo del tipo de contenedor, ofrecerá unos servicios u otros, y permitirá desplegar en él un tipo de componente u otro. Los tipos de contenedores Java EE son:

- Contenedor cliente (Application Client Container o Applet Container).
- Contenedor web (Web Container).
- Contenedor de negocio o de EJBs (EJB Container).

Como podemos ver, cada tipo de contenedor corresponde con una de las capas definidas, a excepción de la capa de datos que está implementada por otro tipo de productos (ya mencionados anteriormente) ajenos a la plataforma Java EE.

En el siguiente diagrama, podemos observar la relación entre los distintos tipos de contenedores Java EE:



1.2.3 Servicios Java EE

Las especificaciones Java EE, definen una serie de funcionalidades que los distintos tipos de contenedores deberán implementar y ofrecer a los desarrolladores de aplicaciones Java EE.

Existen multitud de servicios, pero simplemente destacaremos algunos:

- De directorio: para la indexación y búsqueda de componentes y recursos.
- De despliegue: para facilitar la descripción y personalización de componentes a la hora de su instalación.
- De transaccionalidad: para poder ejecutar distintas acciones en una misma unidad transaccional.
- De seguridad: para poder autenticar y autorizar a los usuarios de una aplicación.
- De acceso a datos: para facilitar el acceso a las Bases de Datos.
- De conectividad: para facilitar el acceso a los distintos Sistemas de Información Empresarial (EIS).
- De mensajería: para poder comunicarse con otros componentes, aplicaciones o EISs.

1.3 El diseño de aplicaciones Java EE

Todos los informáticos saben, que antes de ponerse a programar hay una fase muy importante de análisis y de diseño donde se estudia y define la solución. Para ello, contamos con el lenguaje UML (Unified Modeling Language) que define una serie de diagramas y notaciones para poder plasmar estos análisis y diseños.

Nota: El lenguaje UML no es objeto de este curso pero se cubre con cierta profundidad en el curso de Aula Mentor: Programación en Java – Inicial.

Adicionalmente, existe el concepto de Patrones de Diseño, que son un conjunto de soluciones (o diseños) a los problemas más comunes en la programación de aplicaciones, que han demostrado ser útiles y eficientes en la resolución de dichos problemas. Su existencia, permite poder reutilizar y aplicar soluciones existentes, y no estar reinventando la rueda constantemente.

Pues bien, existe un conjunto importante de estos patrones de diseño muy relacionados con el desarrollo de aplicaciones Java EE. A lo largo del curso iremos viendo algunos, pero si el alumno quiere profundizar en este tema tiene el catálogo completo en las Blue Prints de Java EE:

http://www.oracle.com/technetwork/java/catalog-137601.html

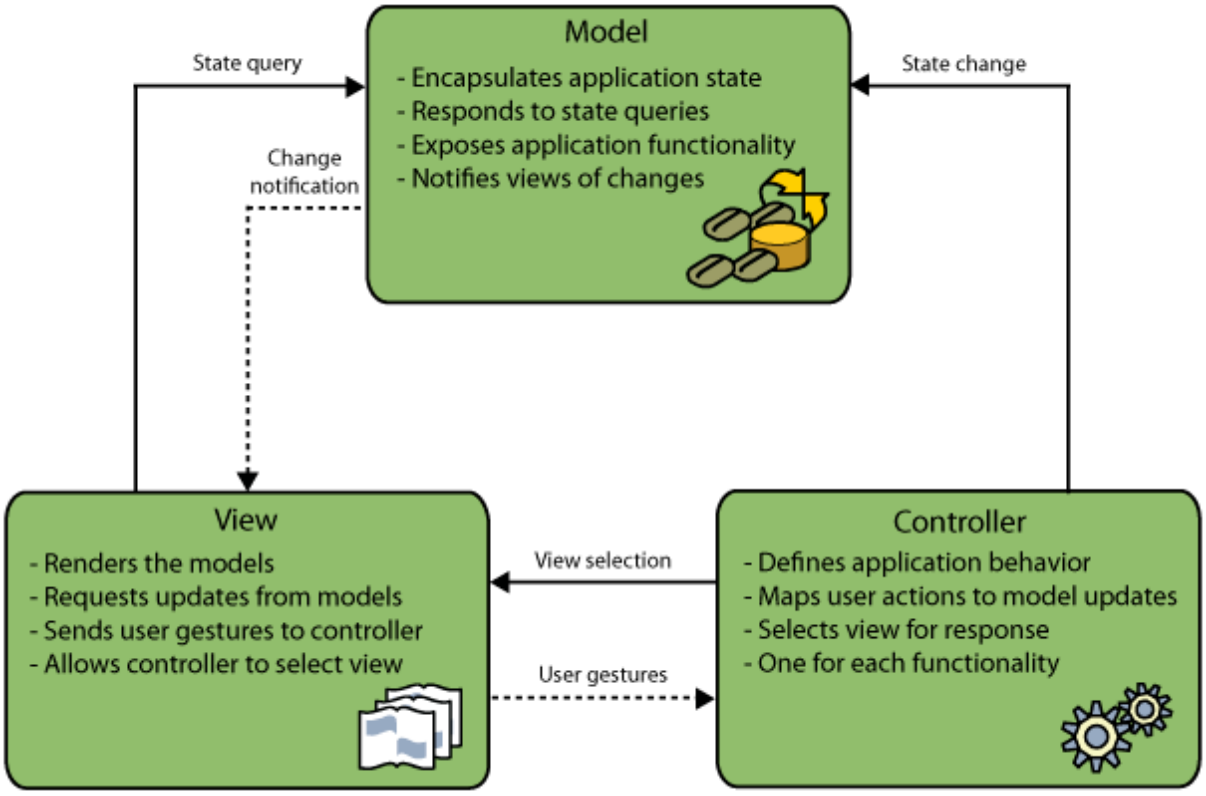
En este capítulo de introducción a Java EE, vamos a hablar de uno que ataca directamente las problemáticas derivadas del modelo de una aplicación Java EE multicapa. Es conocido con el nombre de: Modelo-Vista-Controlador (MVC).

Las aplicaciones que manejan acceso a datos, gestionan distintas presentaciones y tienen lógica de negocio compleja, suelen sufrir un problema serio a la hora de mantenerlas debido a interdependencias entre todos los componentes. Dichas interdependencias también dificultan la reutilización de código, obligando a rescribir más veces de las deseadas una lógica muy parecida.

El patrón de diseño MVC resuelve estos problemas desacoplando el acceso a datos de la lógica de negocio y esta de la presentación. De esta forma, se podrá reutilizar un acceso desde distintas funcionalidades, o reutilizar la misma funcionalidad desde distintos tipos de presentación, etc… facilitando también el mantenimiento posterior:

- El Modelo (Model): Representa los datos y cualquier lógica de negocio relacionada con ellos.
- La Vista (View): renderiza el contenido de los modelos dependiendo de la tipología de cliente (navegador web, teléfono móvil, etc…), permitiendo su visualización.
- El Controlador (Controller): define el comportamiento general de la aplicación coordinando a las otras dos partes (Modelo y Vista).

Veamos el patrón de diseño en un diagrama:



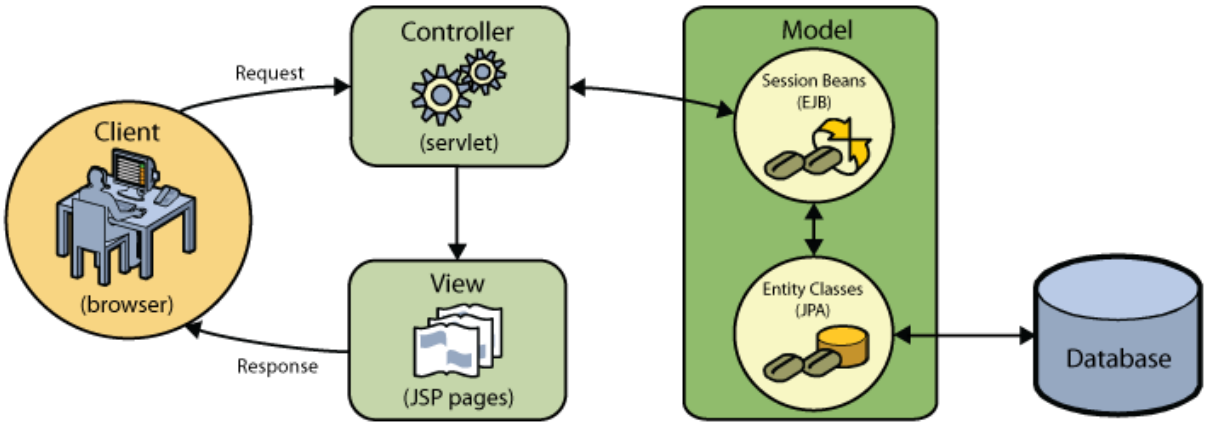
Pues bien, los distintos tipos de componentes que hemos introducido en el apartado del modelo de aplicación Java EE, encajan perfectamente en este diseño:

- Modelo: Enterprise JavaBeans, POJOs (Plain Old Java Objects).
- Vista: JavaServer Pages (JSP), JavaServer Faces (JSF).
- Controlador: Java Servlets.

Un ejemplo del flujo de una aplicación Java EE sería algo así:

1. El cliente, por ejemplo un navegador, solicitará una funcionalidad desde el interface visual (Vista).
2. Dicha petición entrará a través de un Java Servlet (Controlador).
3. Dicho Java Servlet, analizará qué se está pidiendo, qué información adicional aporta y decidirá que Enterprise JavaBean o POJO (Modelo) cubre dicha petición.
4. Lo invocará, y tras recibir un resultado, decidirá qué JavaServer Page (JSP) muestra dicho resultado al cliente (Vista).
5. El resultado será devuelto y mostrado.

Y visualmente:



Evidentemente, las aplicaciones se pueden desarrollar sin tener en cuenta estos patrones de diseño, pero está demostrado sobradamente en el mercado, el aumento de productividad y mejora del mantenimiento con su uso.

A lo largo del curso iremos insistiendo en este punto y detallando y practicando tanto este patrón como otros.

1.4 Las especificaciones Java EE

Las especificaciones Java EE son el conjunto de las definiciones detalladas de los conceptos que forman parte de la plataforma Java EE: componentes, contenedores y servicios.

No entraremos en el detalle de cada una. A lo largo del curso, iremos desgranando aquellas especificaciones relacionadas con la programación web.

En concreto, las especificaciones Java EE 6.0 (última versión en el momento de la redacción de este manual) están compuestas por las siguientes definiciones (en negrita las que cubriremos en el curso):

- Java API for RESTful Web Services (JAX-RS) 1.1
- Implementing Enterprise Web Services 1.3
- Java API for XML-Based Web Services (JAX-WS) 2.2
- Java Architecture for XML Binding (JAXB) 2.2
- Web Services Metadata for the Java Platform
- Java API for XML-Based RPC (JAX-RPC) 1.1
- Java APIs for XML Messaging 1.3
- Java API for XML Registries (JAXR) 1.0
- **Java Servlet 3.0**
- JavaServer Faces 2.0
- **JavaServer Pages 2.2/Expression Language 2.2**
- **Standard Tag Library for JavaServer Pages (JSTL) 1.2**
- Debugging Support for Other Languages 1.0
- Contexts and Dependency Injection for Java (Web Beans 1.0)
- Dependency Injection for Java 1.0
- Bean Validation 1.0
- Enterprise JavaBeans 3.1 (incluye Interceptors 1.1)
- Java EE Connector Architecture 1.6
- Java Persistence 2.0
- Common Annotations for the Java Platform 1.1
- Java Message Service API 1.1
- Java Transaction API (JTA) 1.1
- JavaMail 1.4
- Java Authentication Service Provider Interface for Java Authorization Contract for Containers 1.3
- Java EE Application Deployment 1.2

- J2EE Management 1.1

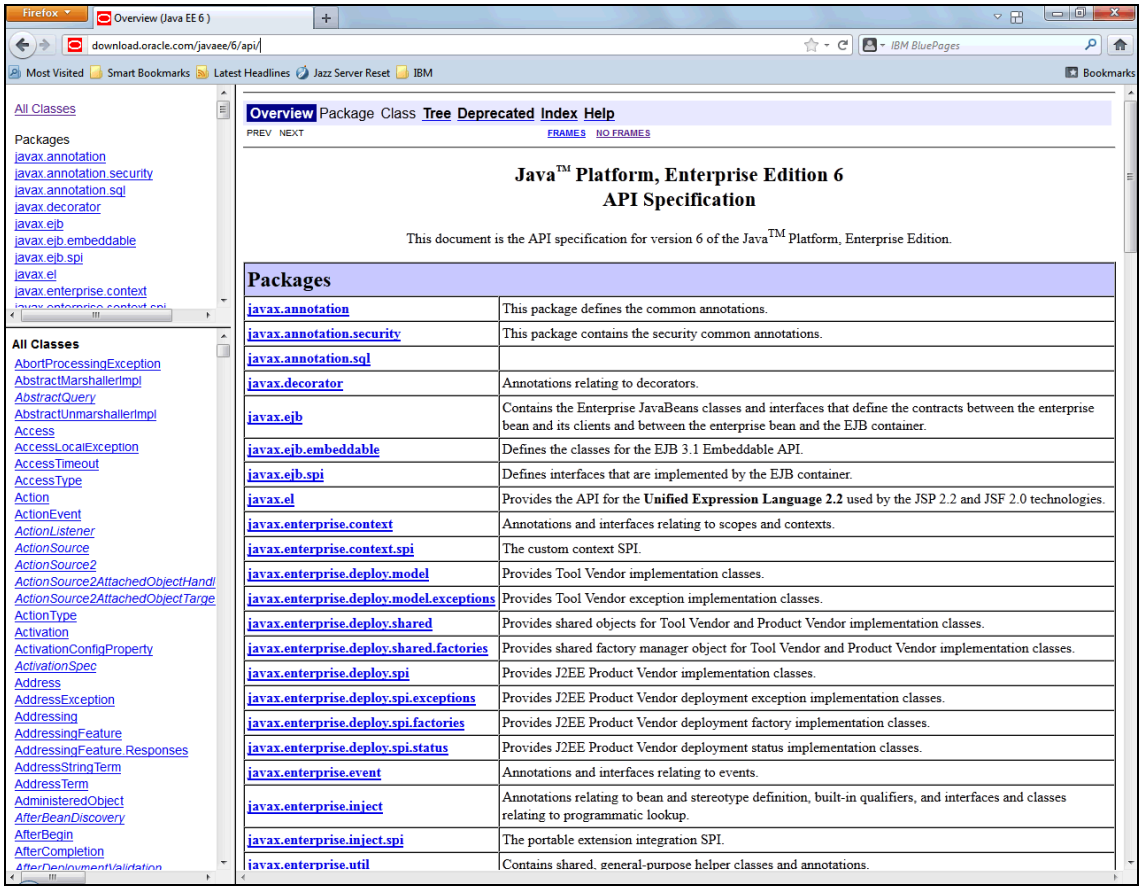
Dado que la plataforma Java EE extiende la Java SE, existen especificaciones Java SE incluidas en Java EE:

- Java API for XML Processing (JAXP) 1.3
- **Java Database Connectivity 4.0**
- Java Management Extensions (JMX) 2.0
- JavaBeans Activation Framework (JAF) 1.1
- Streaming API for XML (StAX) 1.0
- **Java Naming and Directory Interface (JNDI) 1.2**

Como ya hemos comentado, la plataforma Java EE es un tema muy extenso que toca gran cantidad de ámbitos y tecnologías. Nosotros en este curso titulado "Programación Web" nos centraremos exclusivamente en aquellas especificaciones relacionadas con la presentación y el acceso a Bases de Datos.

Existe documentación on-line de todas estas especificaciones o APIs que serán de enorme utilidad a la hora de desarrollar aplicaciones Java EE:

http://download.oracle.com/javaee/6/api/



1.5 El ensamblado y despliegue de aplicaciones Java EE

Una aplicación Java EE está formada por un empaquetamiento de una o varias unidades conocidas con el nombre de módulos. Estos módulos contienen a su vez:

- Uno o varios componentes (Java Servlets, Enterprise JavaBeans (EJB)…).
- Un descriptor de despliegue que describe el contenido y características del módulo (desde la versión 5.0 estos descriptores son opcionales, ya que el propio código puede ser auto descriptivo mediante el uso de anotaciones).

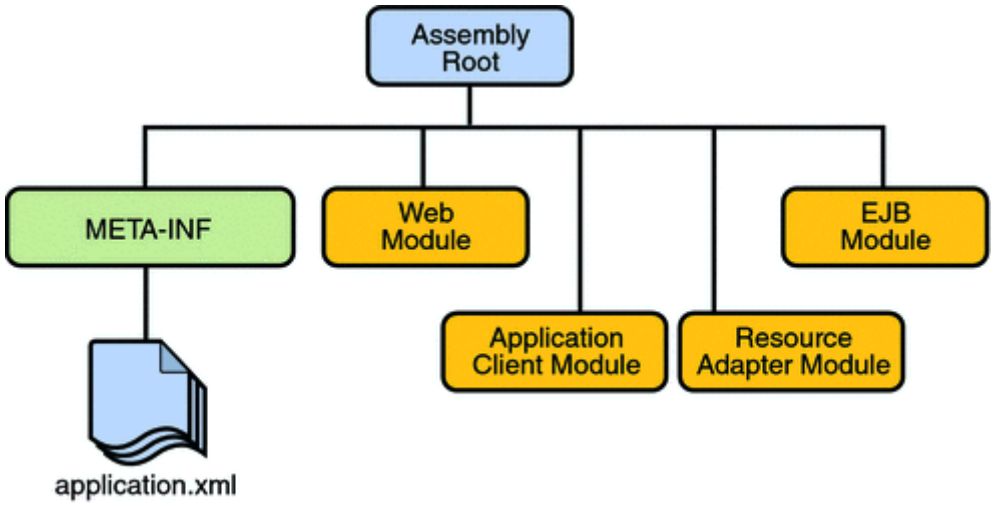
Existen distintos tipos de módulos dependiendo de su contenido y el contenedor donde se vaya a ejecutar. Los distintos módulos van empaquetados en un fichero JAR (Java ARchive). No obstante, la extensión de dicho fichero dependerá del tipo de módulo:

- Módulo Web (Web Module): contiene normalmente Java Servlets, JavaServer Pages (JSP), JavaServer Faces (JSF), contenidos estáticos como imágenes, HTMLs, CSSs… La extensión del fichero empaquetado será WAR (Web ARchive).
- Modulo de EJBs (EJB Module): como su nombre indica, contiene Enterprise JavaBeans (EJB). La extensión del fichero empaquetado es la de por defecto, JAR (Java ARchive).
- Modulo cliente (Application Client Module): contiene el código de la aplicación cliente. La extensión del fichero empaquetado es la de por defecto, JAR (Java ARchive).
- Modulo Adaptador (Resource Adapter Module): contiene un conector para comunicarse con un Sistema de Información Empresarial (EIS). La extensión del fichero empaquetado será RAR (Resource ARchive).

La aplicación Java EE a su vez, también es un empaquetado de los distintos módulos que la forman. La extensión del fichero empaquetado será EAR (Enterprise ARchive).

Una aplicación Java EE no tiene por qué contener módulos de todos los tipos, lo que si es necesario, es que tenga como mínimo uno independientemente del tipo.

Gráficamente, la estructura de una aplicación Java EE empaquetada sería la siguiente, partiendo desde la raíz del EAR (Assembly Root):



El descriptor de despliegue de una aplicación Java EE es un fichero XML llamado application.xml que reside en un directorio denominado META-INF.

Como ya hemos mencionado, desde la versión 5.0 de las especificaciones Java EE, este descriptor de despliegue es opcional si se han usado anotaciones en el código. No obstante, no está mal seguir utilizándolos para que los administradores de los sistemas de una manera legible y sencilla puedan entender el contenido y las características de la aplicación.

Un ejemplo de descriptor de despliegue sería:

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:application="http://java.sun.com/xml/ns/javaee/application_5.xsd
" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_6.xsd" id="PruebaEAR"
version="6">
  <module>
    <connector>PruebaConnector.rar</connector>
  </module>
  <module>
    <java>PruebaClient.jar</java>
  </module>
  <module>
    <ejb>PruebaEJB.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>PruebaWeb.war</web-uri>
      <context-root>PruebaWeb</context-root>
    </web>
  </module>
</application>
```

Se trata de una aplicación Java EE que contiene un módulo de cada tipo. No hay que preocuparse si no se entiende el contenido del fichero como tal, ya que de momento, simplemente se trata de ver un ejemplo del concepto de descriptor de despliegue.

Debido a que este curso es sobre programación web, nos centraremos únicamente en los módulos web, aunque no está de más conocer la existencia de otro tipo de módulos.

1.6 El Servidor de Aplicaciones Java EE

Hasta aquí, todo han sido especificaciones, definiciones… pero para poder ejecutar una aplicación Java EE necesitamos un entorno de ejecución. Dicho entorno de ejecución se conoce con el nombre de Servidor de Aplicaciones.

Un Servidor de Aplicaciones por tanto, es un producto de software que implementa todas las especificaciones Java EE. De manera que al desplegar o instalar una aplicación Java EE en el servidor, sabemos seguro que va a encontrarse con todos los contenedores y servicios definidos por la especificación y que seguramente utiliza y necesita la aplicación.

Existe una batería de pruebas estándar que todo proveedor de Servidores de Aplicaciones debe pasar satisfactoriamente para poder decir que es Java EE. Es lo que se conoce con el nombre de JCK (java Compatibility Kit o Kit de Compatibilidad Java).

Gracias a la existencia del estándar Java EE, podemos tener la tranquilidad de que nuestra aplicación debe funcionar perfectamente en el Servidor de Aplicaciones de cualquier proveedor, asegurándonos así que no debemos de trabajar con ninguno en concreto.

Existen multitud de Servidores de Aplicaciones en el mercado. Podríamos categorizarlos de la siguiente forma:

- Gratuitos o de pago.
- Certificados Java EE o no.

A continuación comentamos algunos. No están todos los que son, ni son todos los que están, pero si los más usados y conocidos:

- Apache Tomcat: Es gratuito de código abierto. No cumple con el 100% de las especificaciones Java EE por lo que no se puede decir que sea un Servidor de Aplicaciones Java EE. Pero si que cubre algunas de las especificaciones relacionadas con la programación web que estudiamos en este curso: Java Servlet 3.0, JavaServer Pages 2.2, Java Database Connectivity 4.0 y Java Naming and Directory Interface 1.2

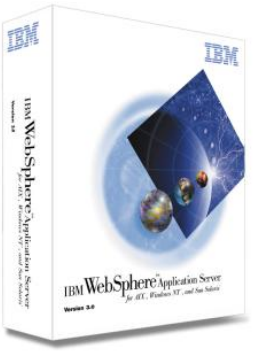
  Este es el Servidor de Aplicaciones que usaremos en este curso. En los Materiales de Apoyo de la Mesa del Alumno, encontraréis las instrucciones de descarga, instalación y configuración del servidor.

URL: http://tomcat.apache.org

- Jetty: Es gratuito y de código abierto. Al igual que pasara con Apache Tomcat, tampoco implementa el 100% de las especificaciones Java EE. Una vez más, se centra exclusivamente en algunas especificaciones web.

URL: http://jetty.codehaus.org/jetty/

- WebSphere Application Server: Es de pago de la empresa IBM. Se trata de un Servidor de Aplicaciones Java EE completo.



URL: http://www-01.ibm.com/software/webservers/appserv/was/

- Apache Gerónimo: Es gratuito y de código abierto. Se trata de un Servidor de Aplicaciones Java EE completo.

URL: http://geronimo.apache.org

- Oracle Weblogic Server: Es de pago de la empresa Oracle. Se trata de un Servidor de Aplicaciones Java EE completo.

URL: http://www.oracle.com/us/products/middleware/application-server/

- GlassFish: Es gratuito y de código abierto. Se trata de un Servidor de Aplicaciones Java EE completo.

URL: http://glassfish.java.net/

- JBoss: Es gratuito y de código abierto. Se trata de un Servidor de Aplicaciones Java EE completo.

URL: http://www.jboss.org/jbossas

para recordar

- ✓ En esta unidad hemos visto la diferencia entre las distintas versiones de Java: **Java Card, Java Micro Edition, Java Standar Edition** y **Java Enterprise Edition (Java EE).**

- ✓ Las aplicaciones Java EE, están distribuidas en distintas capas para una mejor organización funcional: **capa cliente** (para interactuar con el usuario), **capa web** (para llevar el control de la aplicación y a veces interactuar con el usuario), **capa de negocio** (contiene la lógica del negocio como tal) y la **capa de datos** (contiene la información de negocio).

- ✓ A su vez la plataforma Java EE, para poder ejecutar las aplicaciones Java EE multicapa, está formada por:
  - **Componentes**: Unidades de software que forman o componen la aplicación
  - **Contenedores**: Entorno de ejecución donde se ejecutan los componentes.
  - **Servicios**: Funcionalidades estándar que todo contenedor debe proveer a los componentes.

- ✓ Para poder diseñar aplicaciones Java EE, ya existen distintos patrones de diseño muy extendidos que posibilitan la reutilización de los componentes y un mejor mantenimiento de los mismos, como es el patrón **Modelo-Vista-Controlador** (MVC).

- ✓ Para que una aplicación Java EE, pueda ejecutarse en un servidor Java EE, estos deben de cumplir las especificaciones determinadas en una versión dada o al menos aquellas necesarias para el tipo de aplicación que se desea implementar.

- ✓ Las aplicaciones Java EE (fichero EAR) están formadas por al menos un módulo de los siguientes tipos**: Módulo Web** (fichero WAR)**, Modulo de EJBs** (fichero JAR), **Modulo cliente** (fichero JAR), **Modulo Adaptador** (fichero RAR).