ESCUELA DE SISTEMAS Y TECNOLOGÍAS

Transparencias de **ANALISTA DE SISTEMAS**

Edición 2020 Materia: **Java Web**

TEMA: Introducción Java Web

Agenda

- Introducción
- Servlet vs. JSP
- JavaBeans
- Relación entre Tecnologías Web Java
- JavaEE Containers
- Ciclo de Vida de una Aplicación Web
- Deployment Descriptors
- Instalación de un Módulo Web

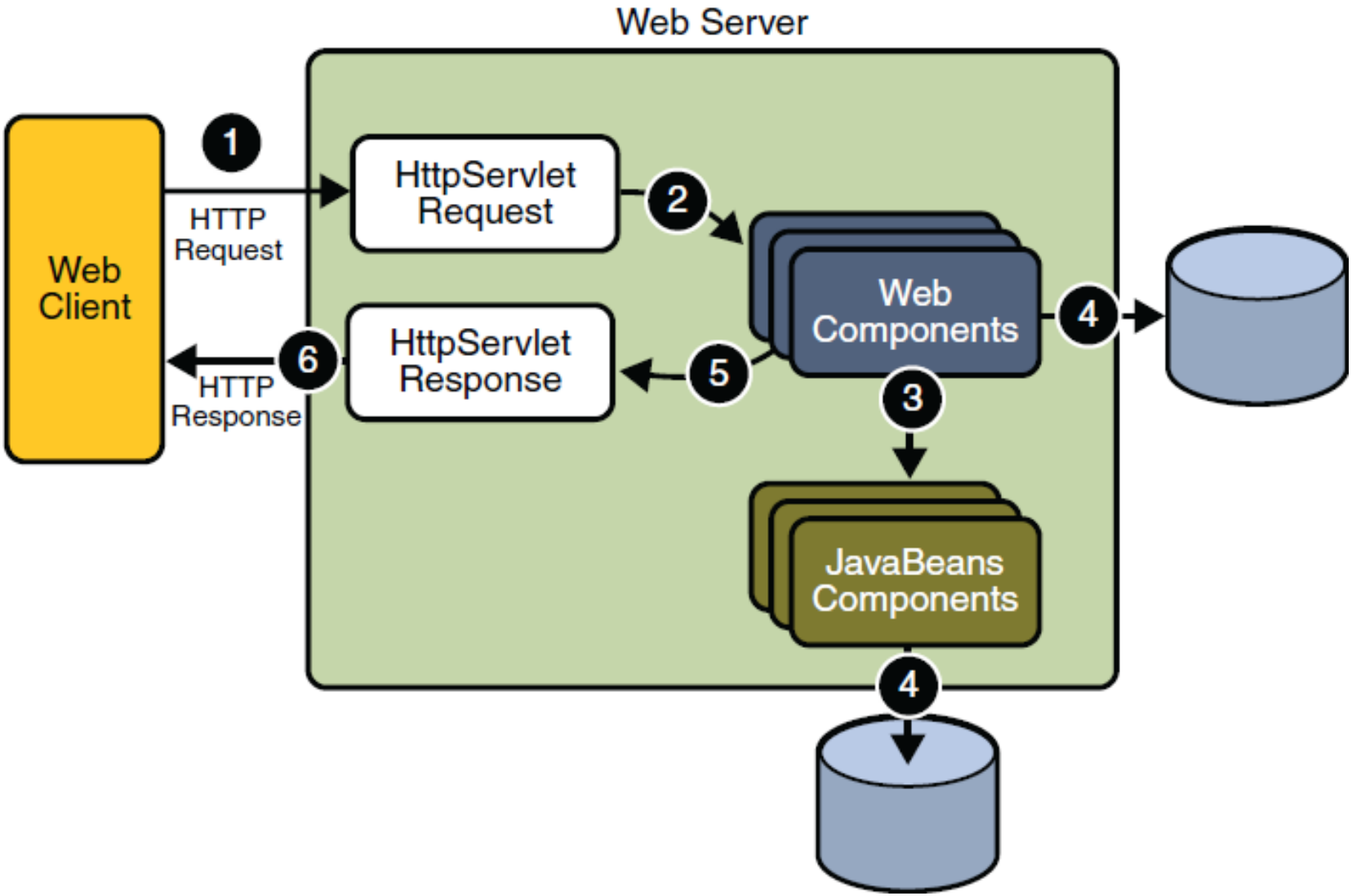
Introducción (1)

- Java Web se enmarca dentro de la plataforma **JavaEE**.
- JavaEE (anteriormente conocido como J2EE) es una especificación de una plataforma para el desarrollo de aplicaciones empresariales de gran porte (no sólo web).
- Provee APIs para el desarrollo de componentes, instalados y ejecutados sobre un servidor de aplicaciones.
- Existen múltiples empresas (vendors) que han desarrollado implementaciones particulares de JavaEE (Oracle GlassFish, RedHat JBoss, IBM WebSphere, Apache Geronimo, etc.)

Introducción (2)

- Una aplicación web es una extensión dinámica de un servidor web o de un servidor de aplicaciones.
- Existen 2 tipos de aplicaciones web:
  - **Orientada a presentación** (presentation - oriented): genera páginas web interactivas conteniendo varios tipos de lenguajes de marcas así como contenido dinámico.
  - **Orientadas a servicios** (service - oriented): exporta e implementa servicios web.
- A continuación se presenta un esquema de la interacción entre un cliente web y una aplicación web...

Introducción (3)



Introducción (4)

1. El cliente web (ej: browser) envía un pedido HTTP al servidor web.
2. El servidor web (si implementa Servlets y JSPs) convierte el pedido en un objeto **HTTPServletRequest** el cual es enviado al componente web solicitado (ej: Servlet o JSP).
3. El componente web puede interactuar con otros componentes web así como con JavaBeans.
4. También puede conectarse a una BD, ya sea directamente o mediante JavaBeans.
5. Finalmente, algún componente web (el originalmente solicitado u otro invocado por éste) generará un objeto **HTTPServletResponse**.
6. Esa respuesta será convertida por el servidor web en una respuesta HTTP hacia el cliente web (ej: HTML).

Servlet Vs. JSP (1)

- Un **Servlet** es una clase Java que procesa pedidos generando respuestas dinámicas.
- Una **página JSP** es un documento basado en texto que se traduce (internamente) a un Servlet (por lo que es ejecutado como Servlet).
- Una página JSP permite, a diferencia de un Servlet, generar el contenido estático en forma natural y sencilla, pues por ser un documento basado en texto puede ser utilizado por personas que no programen (ej: diseñadores).
- Dado que una página JSP es finalmente traducida a un Servlet, ambas tecnologías poseen las mismas potencialidades, y lo que se logre con un Servlet también puede lograrse con una página JSP y viceversa.

Servlet Vs. JSP (2)

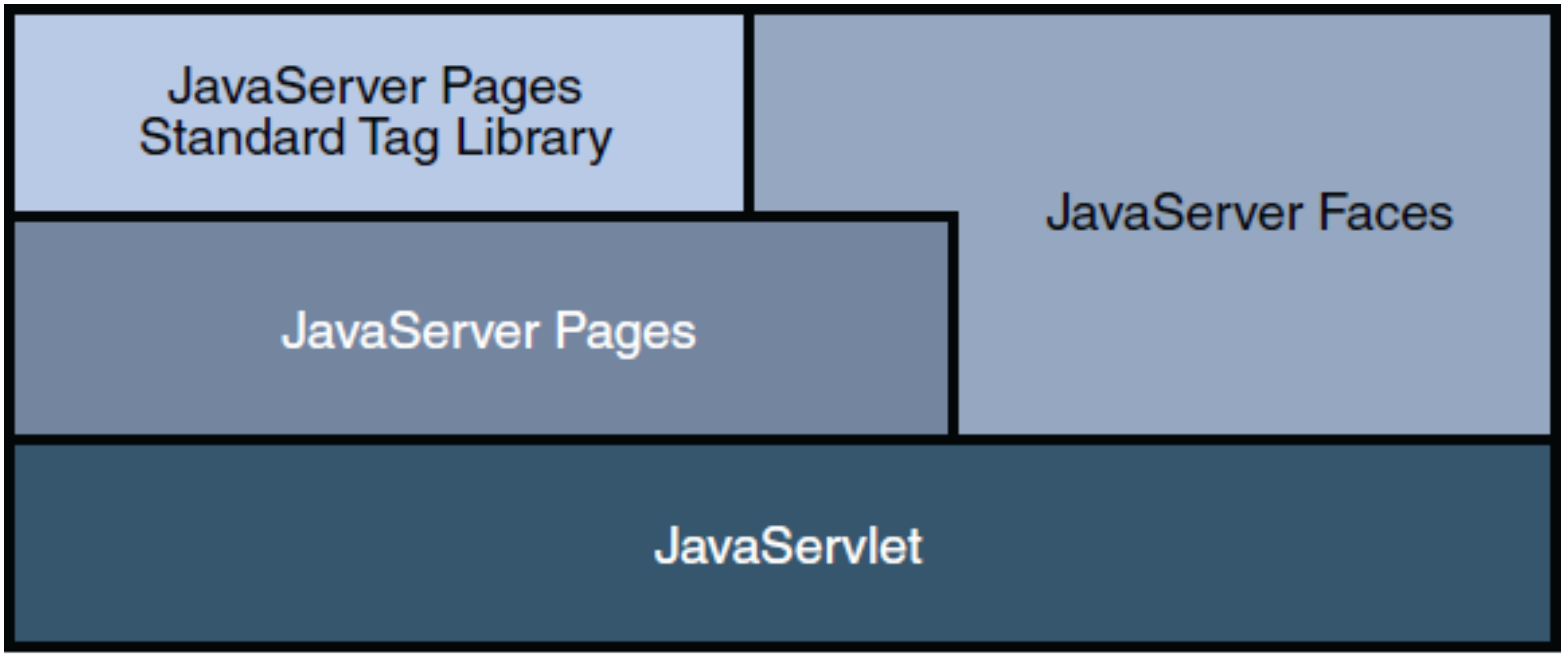
- Un **Servlet** es una clase Java que procesa pedidos generando respuestas dinámicas.
- Por lo tanto: ¿**Porqué entonces JavaEE provee dos tecnologías diferentes con el mismo poder**?
  - Para que sean utilizadas con distinto enfoque y posiblemente por distintas personas (con distintos perfiles dentro del equipo de desarrollo).
- Una ***página JSP*** permite desarrollar de forma sencilla contenido web (ej: HTML) para una aplicación web orientada a presentación.
- Un ***Servlet*** es más adecuado para una aplicación web orientada a servicios, o bien para controlar una aplicación web orientada a presentación (en particular todo aquello que no implique generar contenido web como HTML).

JavaBeans

- Si bien los **JavaBeans** no son parte de la especificación JavaEE, su uso dentro de la arquitectura JavaEE amerita su introducción.
- Un ***JavaBean*** es una clase Java que sigue cierto estándar (ej: contener un constructor por defecto, atributos privados y getters/setters públicos).
- Los JavaBeans típicamente se utilizan para el flujo de datos entre componentes de una aplicación JavaEE.
- Incluso pueden ser utilizados declarativamente (ej: desde una página JSP).
- No confundir con los Enterprise Java Beans (EJB)

Relación entre Tecnologías

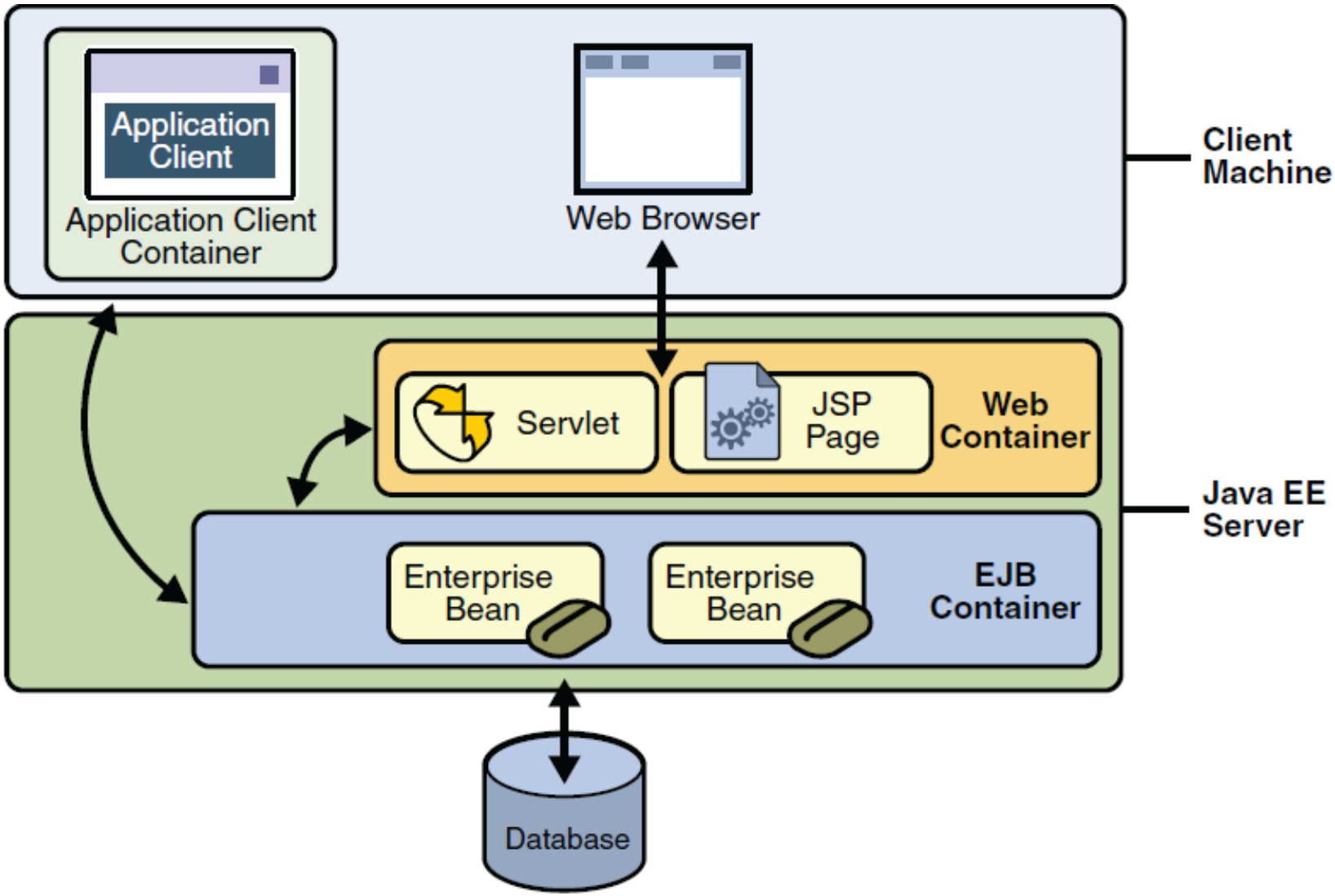
- La tecnología web de base en JavaEE son los Servlets.
- Sobre ella, las demás tecnologías (JSP, JSTL y JSF) agregan niveles de abstracción que facilitan el desarrollo de diferentes aspectos de una aplicación web:



JavaEE Containers (1)

- Un Servidor **JavaEE** provee servicios a los diferentes componentes (servlets, JSPs, etc.) a través de los denominados contenedores (*containers*).
- Un contenedor es la interfaz entre un componente y la funcionalidad (de bajo nivel y específica de la plataforma) que da soporte a ese componente.
- Los componentes deben ser instalados (*deployed*) en sus respectivos contenedores, los cuales les proveerán de los servicios necesarios para su correcta ejecución.
- Cada componente o conjunto de componentes puede especificar su configuración con respecto a su contenedor (*container settings*).

JavaEE Containers (2)



Ciclo de Vida de una App WEB (1)

- Una aplicación web se constituye de:
  - **Componentes web** (ej: Servlets y JSPs)
  - **Archivos** (ej: imágenes)
  - **Clases y librerías de soporte** (ej: clases y JARs)
  - **Deployment descriptor** (web.xml)
- Debido a que los componentes web se ejecutan sobre el contenedor web (web container) el proceso de desarrollo e instalación varía con respecto a las clases Java estándar:

Ciclo de Vida de una App WEB (2)

1. Codificar los componentes web.
2. Crear el **deployment descriptor**.
3. Compilar los componentes web así como las clases de soporte referenciadas por éstos.
4. Empaquetar la aplicación web en una unidad de ***deployment*** (llamada *módulo*).
5. Hacer el deployment de la aplicación web en el ***container*** del servidor a utilizar.
6. Acceder a la URL correspondiente

Deployment Descriptors (1)

- Un ***deployment descriptor*** es un documento XML con extensión *.xml* que describe la configuración de deployment de una aplicación, módulo o componente.
- En el caso de una aplicación Java Web, el archivo deployment descriptor se llama *web.xml*.
- Dicho deployment descriptor forma parte de la especificación JavaEE, y por lo tanto es reconocido por cualquier implementación de servidor JavaEE.
- También pueden existir un deployments descriptors específicos de la implementación de servidor JavaEE donde se esté desplegando la aplicación.

Deployment Descriptors (2)

- Dado que la información del deployment descriptor es declarativa, puede ser cambiada sin la necesidad de modificar el código fuente ni volver a compilar.
- En tiempo de ejecución, el Servidor JavaEE lee el deployment descriptor y aplica su configuración a la aplicación, módulo o componente que corresponda.
- Esta configuración puede hacerse:
  - Directamente modificando el archivo **web.xml**
  - Mediante el NetBeans IDE (que provee una interfaz gráfica que permite al programador establecer las configuraciones al tiempo que éstas son automáticamente traducidas al archivo **web.xml**).

Deployment Descriptors (3)

- **Expiración de Sesión**: Determina los minutos dentro de los cuales permanecerá activa la sesión del usuario mientras éste tenga actividad en la aplicación web. Luego de ese tiempo de inactividad, la sesión será automáticamente finalizada.

```
<session-config>
     <session-timeout>30</session-timeout>
</session-config>
```

Deployment Descriptors (4)

- **Servlets**: Determina el nombre interno del servlet, el nombre de la clase Java que lo implementa y parámetros a ser utilizados por el mismo (desde su código Java):

```
<servlet>
    <servlet-name>ServletProductos</servlet-name>
    <servlet-class>paq.SerProd</servlet-class>
    <init-param>
        <param-name>codigo_inicial</param-name>
        <param-value>10000</param-value>
    </init-param>
</servlet>
```

Deployment Descriptors (5)

- **Mapeo de Servlets**: Todos los pedidos que terminen con *.jsp son automáticamente redirigidos a páginas JSP. Sin embargo con los Servlets esto no ocurre, por lo que deben mapearse en el web.xml:

```
<servlet-mapping>
    <servlet-name>ServletProductos</servlet-name>
    <url-pattern>/Productos</url-pattern>
</servlet-mapping>
```

Deployment Descriptors (6)

- **Página de Bienvenida**: Permite especificar un conjunto de nombres de páginas para intentar cargar (en orden, hasta encontrar un archivo que coincida con ese nombre de página) en caso que no se especifique ningún recurso en la URL solicitada

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>
        bienvenidaAlternativa.jsp
    </welcome-file>
</welcome-file-list>
```

Deployment Descriptors (7)

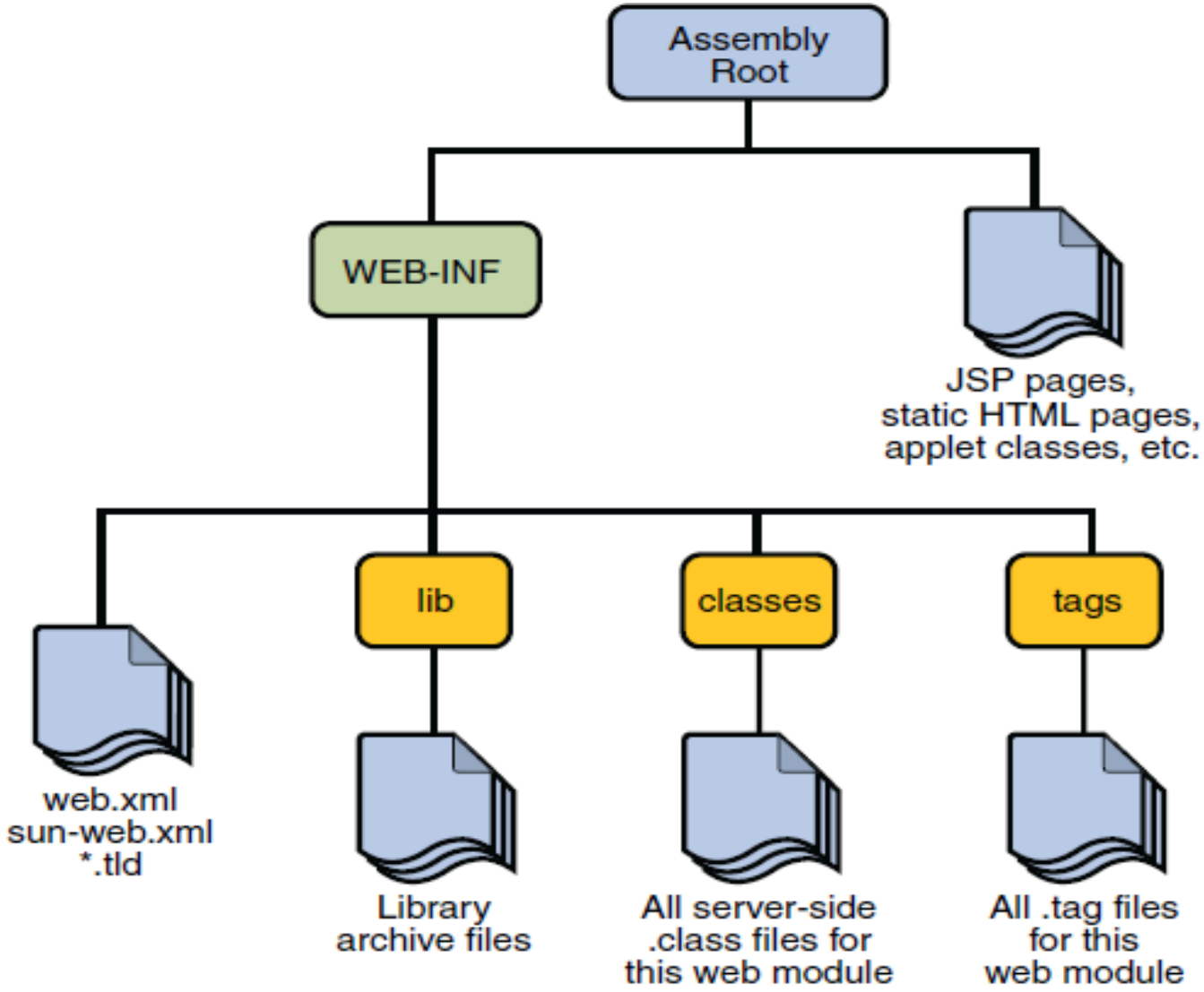
- **Página de Error**: Permite redirigir al usuario a una página de error dependiendo del tipo de error ocurrido (ya sea el código HTTP o la excepción de Java)

```
<error-page>
    <error-code>404</error-code>
    <location>
        /paginaDeError404.jsp
    </location>
</error-page>
```

Instalación Modulo Web (1)

- La mínima unidad de deployment en un servidor JavaEE es un **módulo** (*module*).
- Por tanto, para hacer el deployment de una aplicación web (incluyendo sus componentes web, archivos, clases y librerías) se debe construir un módulo web.
- Éste contendrá todos los recursos utilizados por la aplicación web más el deployment descriptor (archivo **web.xml**).
- Únicamente en el caso en que la aplicación web sólo contenga páginas JSP y archivos estáticos (ej: imágenes) entonces el deployment descriptor no sería necesario.
- A continuación se presenta la **estructura** de un módulo

Instalación Modulo Web (2)



Instalación Modulo Web (3)

- Un módulo web puede ser instalado (*deployed*) simplemente copiando una estructura de directorios y archivos como la indicada, en el directorio de trabajo del servidor web o servidor de aplicaciones.
- También el módulo web puede ser empaquetado en un archivo **WAR** (*Web Archive*) que es un archivo **JAR** pero con extensión **.war**
- El archivo WAR facilita el transporte e instalación del módulo web en cualquier servidor compatible (ej: en Tomcat o GlassFish basta con elegir el archivo WAR para deployment).

Instalación Modulo Web (4)

- El archivo WAR puede ser creado mediante:
  - El comando jar (http://docs.oracle.com/javase/tutorial/deployment/jar/)
  - El utilitario Ant (http://ant.apache.org)
  - El NetBeans IDE (http://www.netbeans.org)
- Para instalar (deploy) el archivo WAR:
  - Desde **Tomcat**: abrir el Tomcat Manager (http://localhost:8084/manager/html), ir a Archivo WAR a desplegar y elegir el archivo .war
  - Desde **GlassFish**: abrir la Administration Console (http://localhost:4848), ir a Applications / Deploy… y elegir el archivo .war
  - Desde **NetBeans**: eligiendo la opción Deploy sobre el proyecto web o simplemente corriéndolo con Run.

Instalación Modulo Web (5)

- Para desinstalar (*undeploy*) el archivo WAR:
  - Abrir el Manager o Administration Console.
  - Seleccionar la aplicación que se desea desinstalar y pulsar el botón Undeploy.
- También puede hacerse con NetBeans desde Prestaciones / Servidores.