

Integración de Tecnologías

Curso 2017-2018

¿JSP? ¡¡¡¡¡Vamos a ello!!!!



TEMA 1

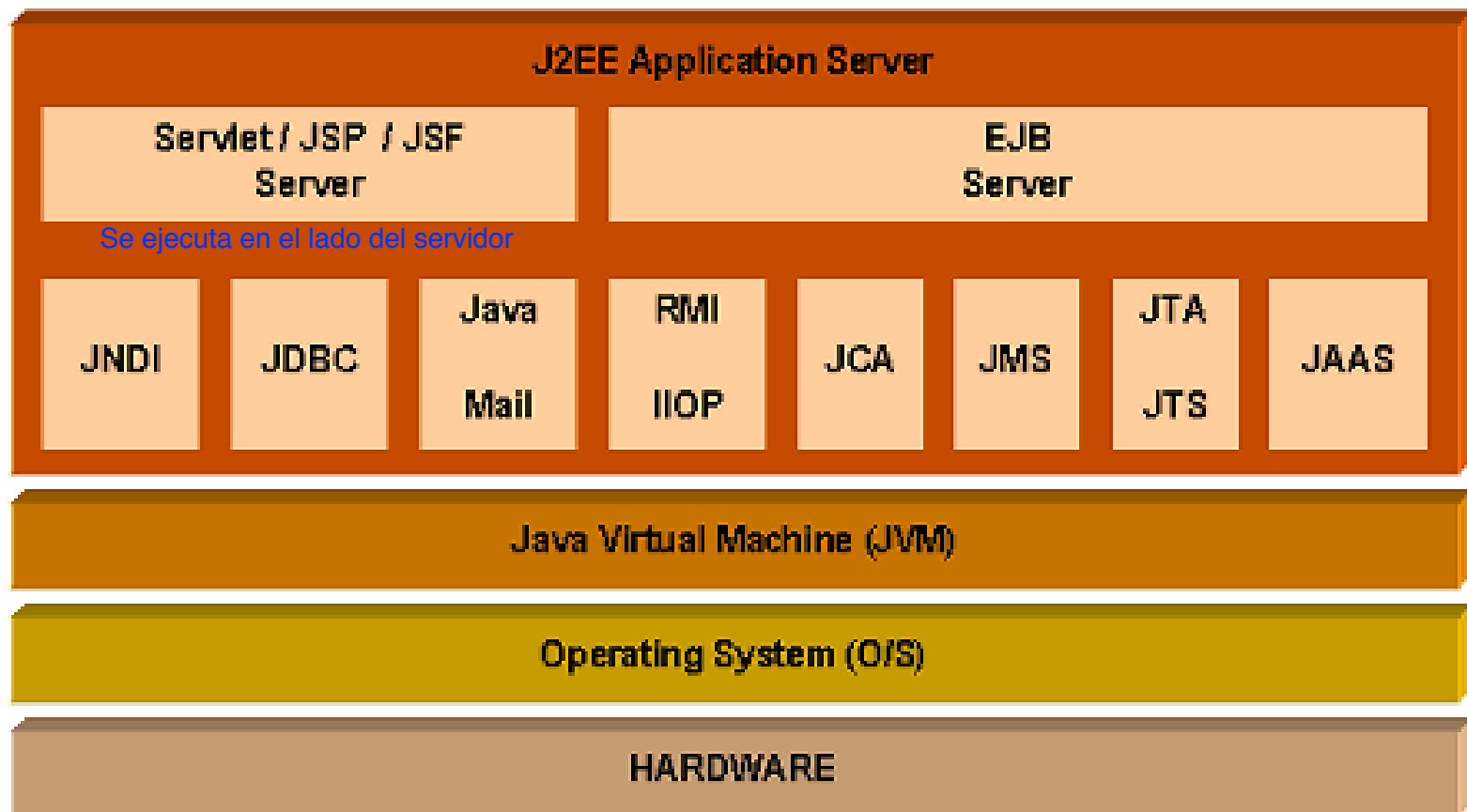
Desarrollo en arquitecturas
homogéneas integradas

J2EE

- *“Plataforma para el desarrollo en Java de aplicaciones divididas en múltiples capas que pueden estar distribuidas”.*
- Ello permite al desarrollador crear una aplicación de empresa portable entre plataformas y escalable, a la vez que integrable con otras tecnologías.
- Tecnologías:

J2EE

- Tecnologías:



Integración de tecnologías

J2EE

- Tecnologías:

//EJB example

@Stateless

public class CustomerService {

@PersistenceContext

private EntityManager entityManager;

public void addCustomer(Customer customer) {

entityManager.persist(customer);

Introduce en la BD

}

}

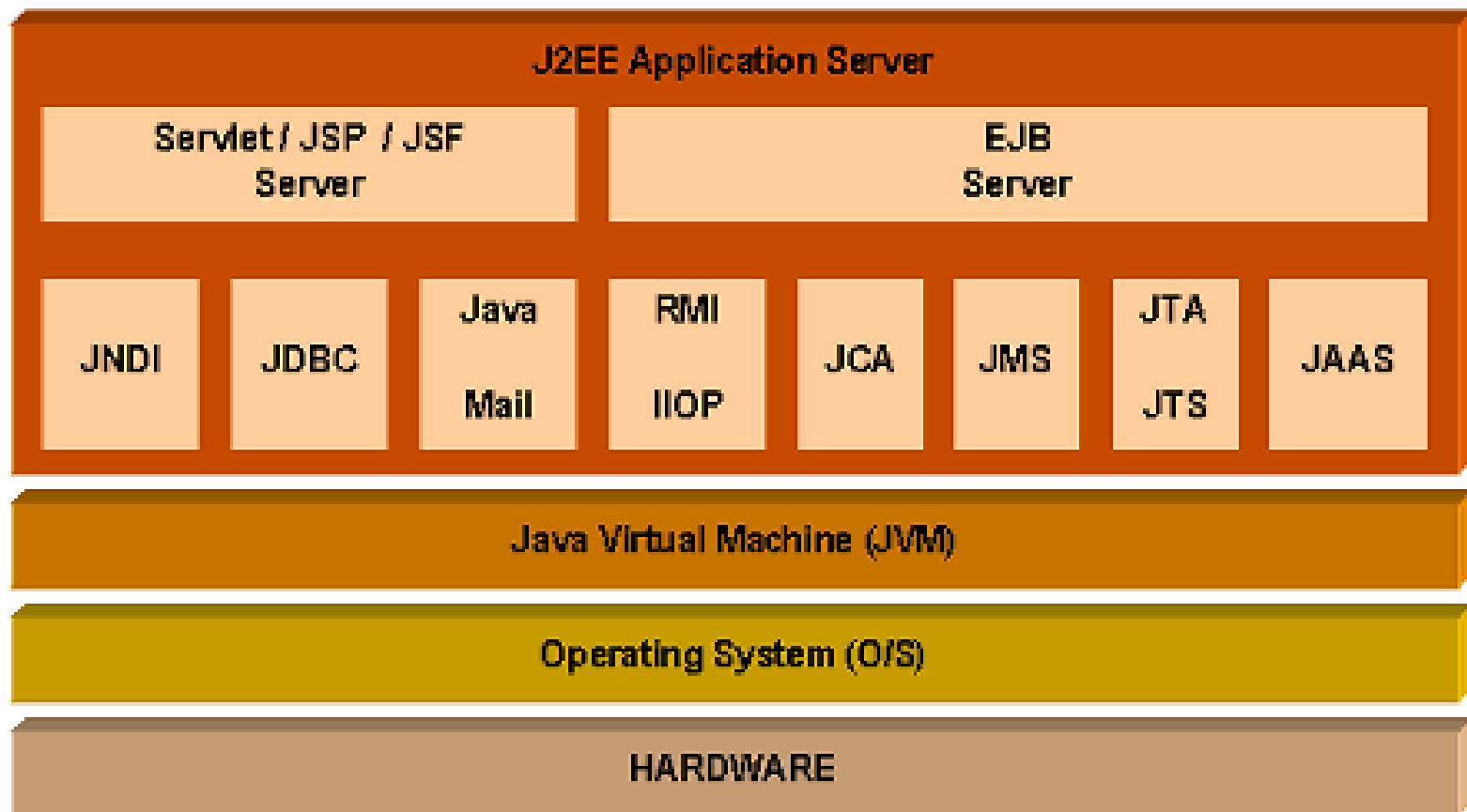
Operating System (O/S)

HARDWARE

Integración de tecnologías

J2EE

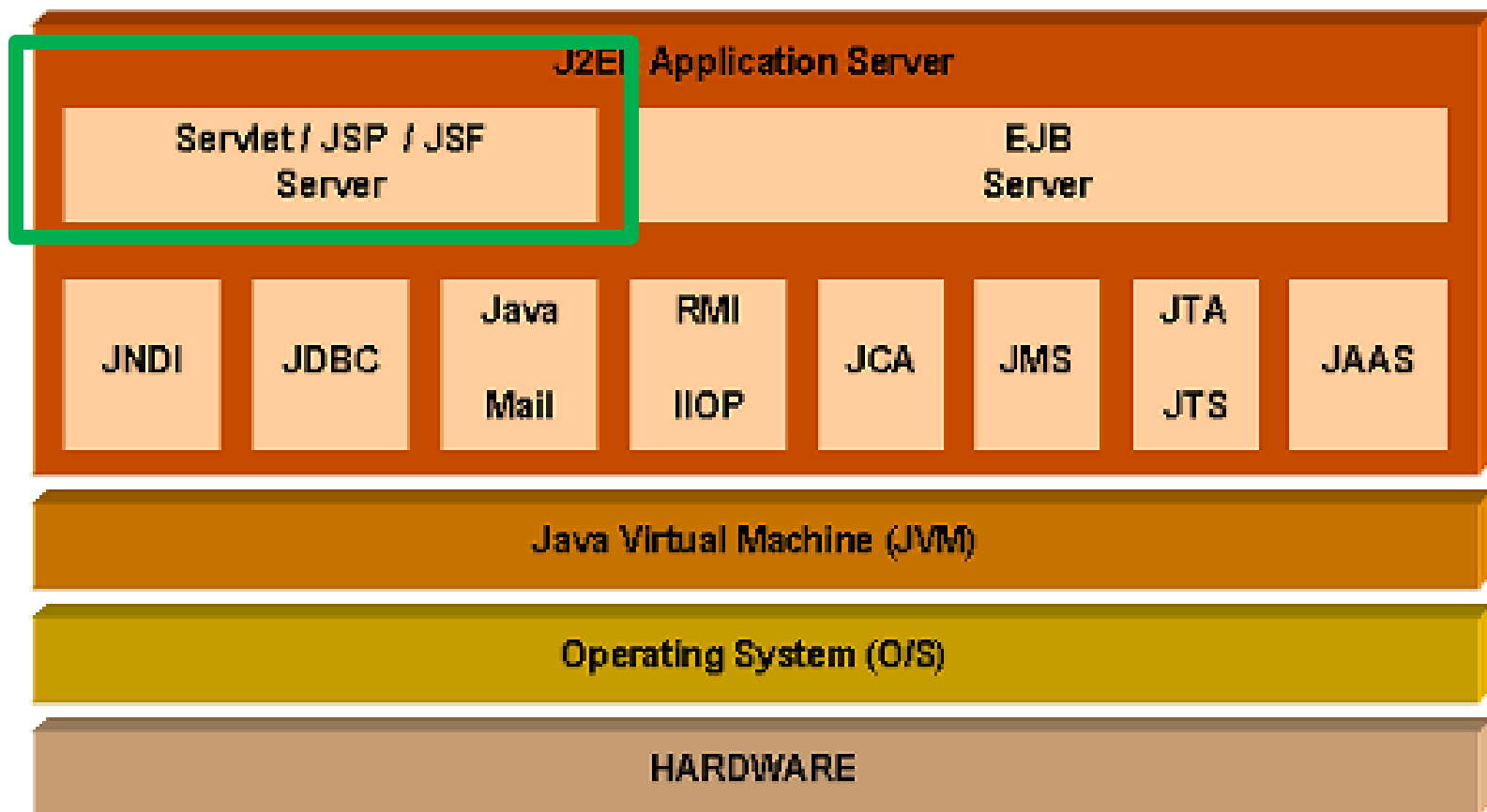
- Tecnologías:



Integración de tecnologías

J2EE

- Tecnologías:



Integración de tecnologías

1.1 JSP Básico

1.1.1 Inmersión en JSP

- ¿Qué es JSP?
- Funcionamiento de JSP
- Ventajas/Desventajas de JSP
- Primera página JSP

¿Qué es JSP?

- **JSP: JavaServer Pages.**
 - Primeros trabajos en 1994/1995. Servidor web escrito en Java.
 - 1999 Los *servlets* (spec. 2.1) habían madurado.
 - Un servlet es un programa Java para gestionar peticiones web.
 - 2001 Se publica la primera especificación de JSP (1.2).
 - http://raibledesigns.com/rd/entry/the_history_of_jsp
- Diseñado para la generación dinámica de páginas web.
 - De forma más sencilla a como se podría hacer con servlets.
- Se ejecuta en el lado del servidor.
- El código va "incrustado" en las páginas web. Como PHP
 - Básicamente incrusta Java dentro de páginas HTML. Cuando el servidor lee los fragmentos de Java los ejecuta.
 - Los códigos al ejecutarse pueden escribir contenido en la página web de una forma muy similar a como se imprime por pantalla.

¿Cómo funciona JSP?

- Las páginas JSP se compilan:
 - Al compilarse genera un servlet equivalente. Éste, a su vez, se compila en bytecode.
 - JSP no es interpretado.
 - Al basarse en Java se considera de propósito general.
 - De mayor rendimiento que otros lenguajes interpretados:
 - No necesita ser interpretado al vuelo.
 - Las páginas sólo se cargan la primera vez que se las llama. El resto del tiempo permanecen a la espera de nuevas peticiones:
 - Podríamos mantener disponibles recursos. P. ej. Conexiones a las bases de datos, ficheros, etc ...
- Para compilar las páginas y gestionar la ejecución de los servlets que de ellas derivan se necesita un **“contenedor”** (servidor):
 - Apache Tomcat (Open source y no comercial).
 - WebSphere (IBM)
 - Oracle Application Server
- **Se integra como parte de una arquitectura que permite dividir la aplicación en diferentes capas**
 - Qué capa sería JSP en el MVC?



Palabro!!!

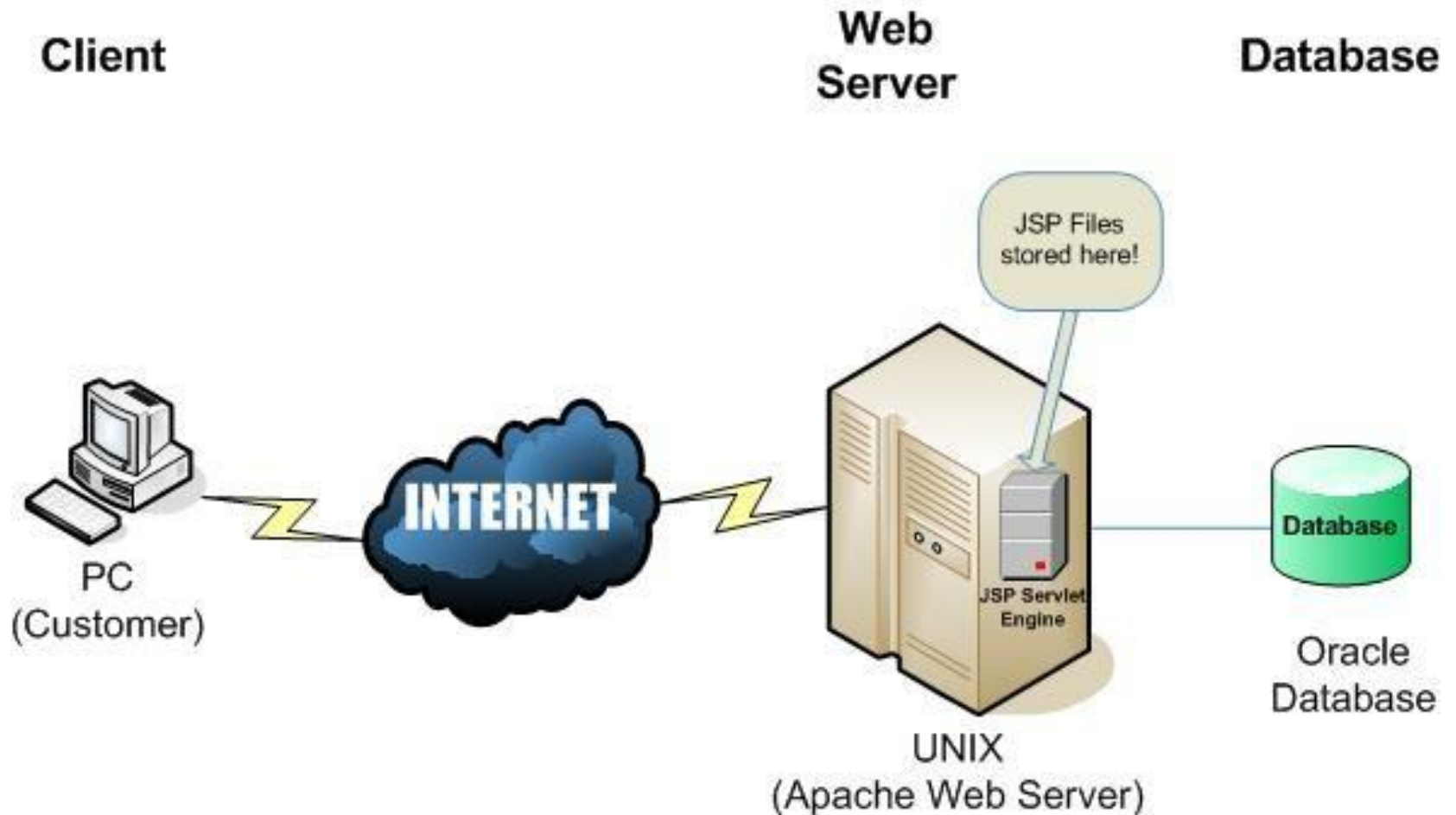
¿Cómo funciona JSP?

- Las páginas JSP se compilan:
 - Al compilarse genera un servlet equivalente. Éste, a su vez, se compila en bytecode.
 - JSP no es interpretado.
 - Al basarse en Java se considera de propósito general.
 - De mayor rendimiento que otros lenguajes interpretados:
 - No necesita ser interpretado al vuelo.
 - Las páginas sólo se cargan la primera vez que se las llama. El resto del tiempo permanecen a la espera de nuevas peticiones:
 - Podríamos mantener disponibles recursos. P. ej. Conexiones a las bases de datos, ficheros, etc ...
- Para compilar las páginas y gestionar la ejecución de los servlets que de ellas derivan se necesita un **contenedor** (servidor):
 - Apache Tomcat (Open source y no comercial).
 - WebSphere (IBM)
 - Oracle Application Server
- **Se integra como parte de una arquitectura que permite dividir la aplicación en diferentes capas**
 - JSP es la capa de vista de la aplicación.

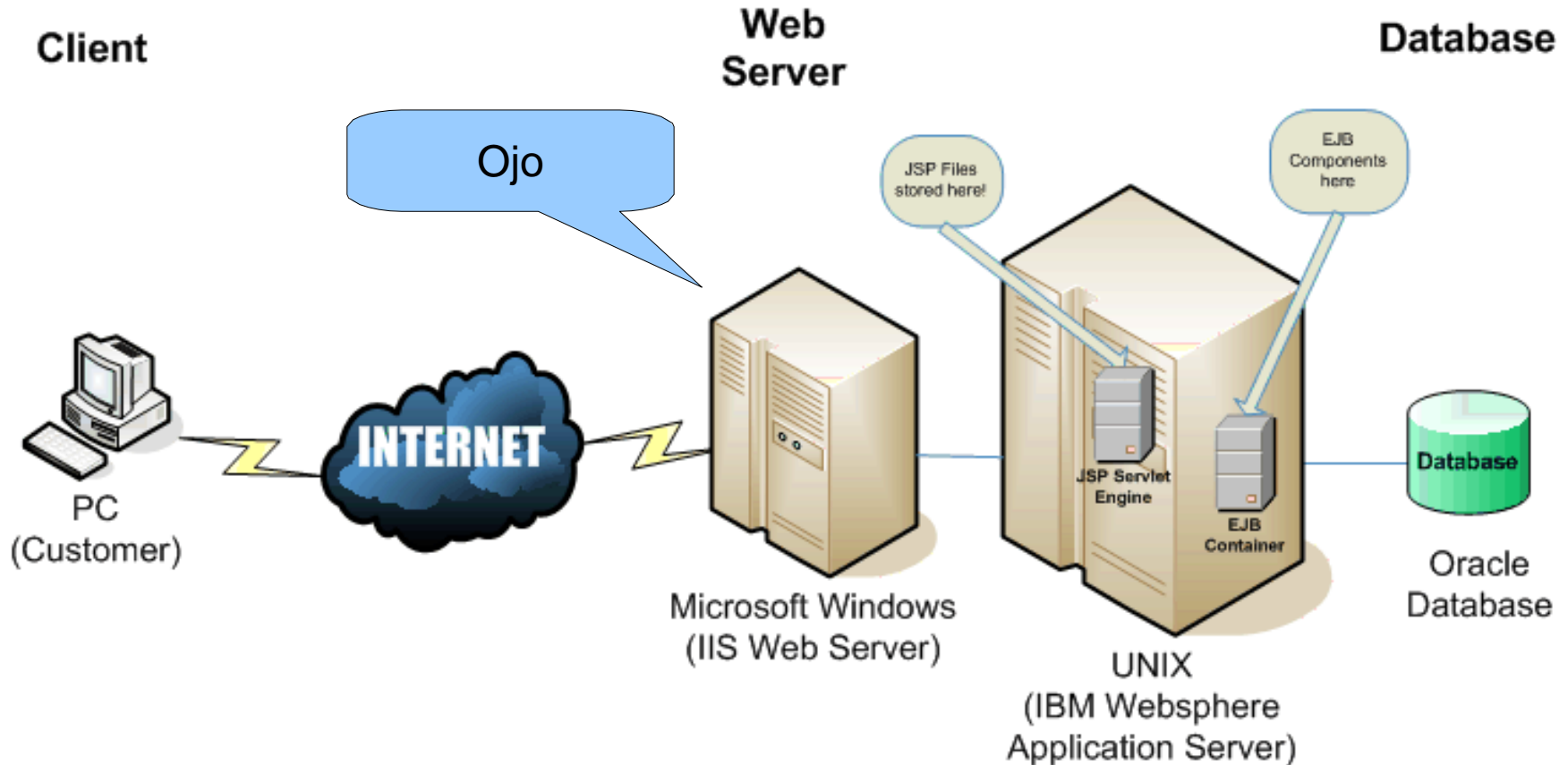


Palabro!!!

Funcionamiento de JSP (básico)

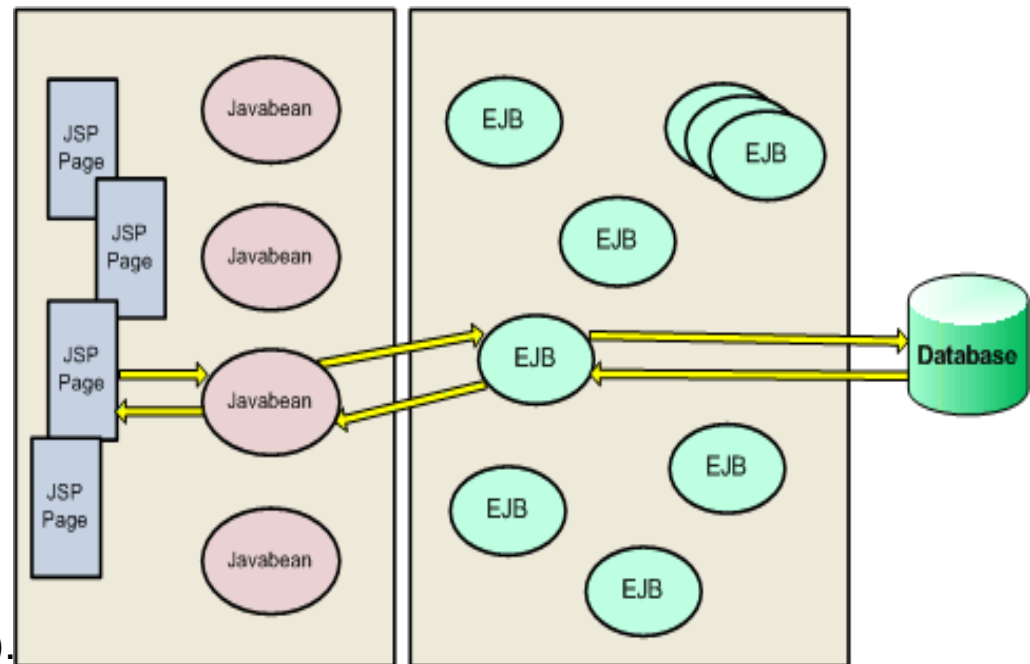


Funcionamiento de JSP (detallado)

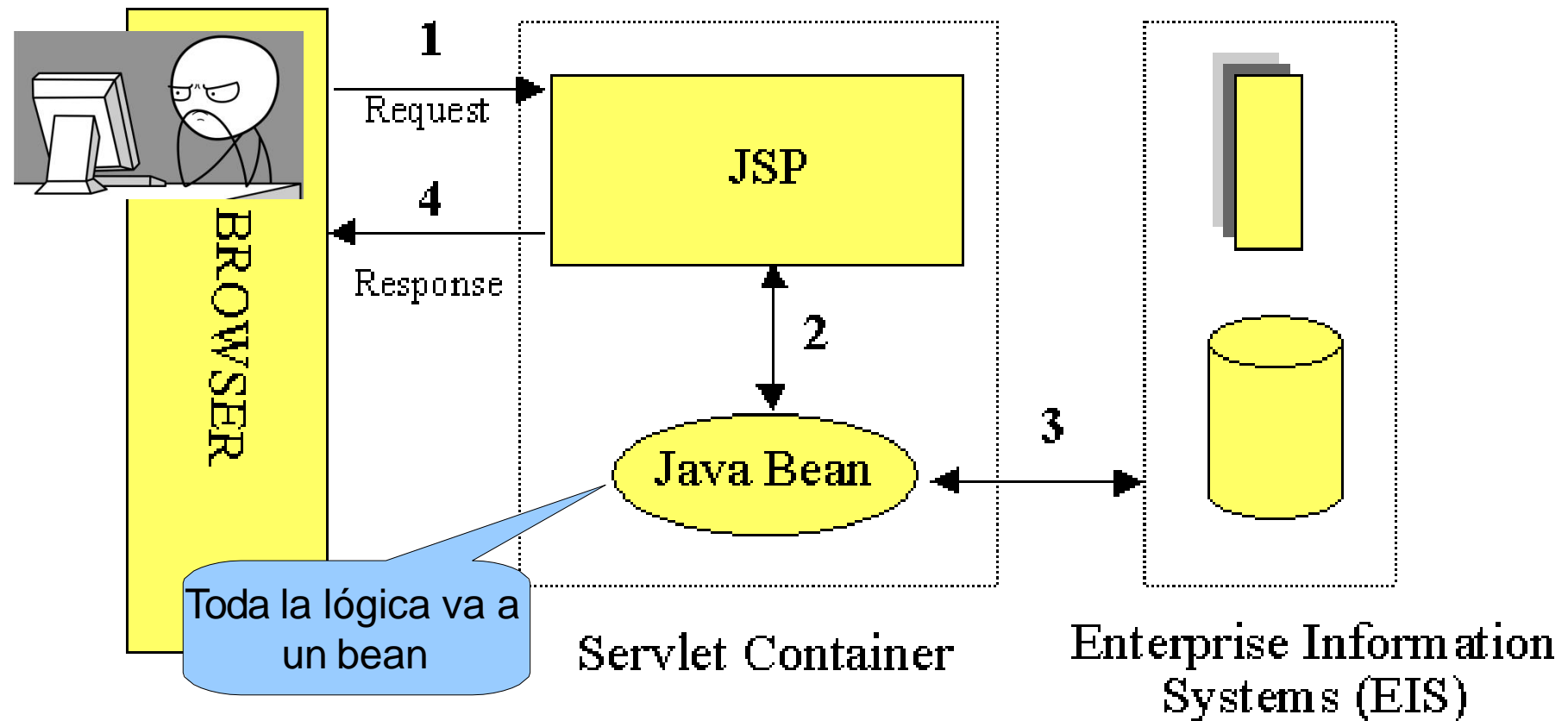


Arquitectura de una aplicación JSP

- Cuatro capas:
 - Vista de la presentación (JSP).
 - Lógica de la presentación (JavaBean)
 - Se encarga de preparar los datos eliminando cualquier código del JSP. P. ej. cálculo de un total.
 - Lógica de negocio (Enterprise JavaBeans).
 - Modelo de datos (Base de datos).



Arquitectura simplificada



Ventajas de JSP

- Arquitectura multicapa:
 - **Encapsulación:** ahorramos detalles a los usuarios. Los cambios a un componente no afectan a los demás.
 - **Separación:** Permite desarrollar sistemas complejos, definiendo claramente las responsabilidades de cada componente.
 - **Reusabilidad:** De los componentes para otras aplicaciones
- **Alto rendimiento con respecto a otras plataformas.**
 - **Poco uso de memoria. Cada petición es una hebra en lugar de un proceso.**
- Software libre: (También hay plataformas privativas)
 - Java.
 - Servidor Tomcat.
- Portable: Cualquier arquitectura para que la que esté disponible la máquina virtual Java
- Dispone de todas las funcionalidades y librerías disponibles en Java.

Desventajas de JSP

- La arquitectura multicapa puede ser muy compleja para proyectos pequeños/medianos.
 - Podemos usar las capas que consideremos necesarias.
- La curva de aprendizaje depende totalmente de la de Java (Alta?).
- Realizar ciertas tareas comunes (p.ej. Acceso a BD, envío de email, etc...) suele ser más complejo que en lenguajes de propósito específico.
 - Es el precio que se ha de pagar por tener un lenguaje de propósito general.

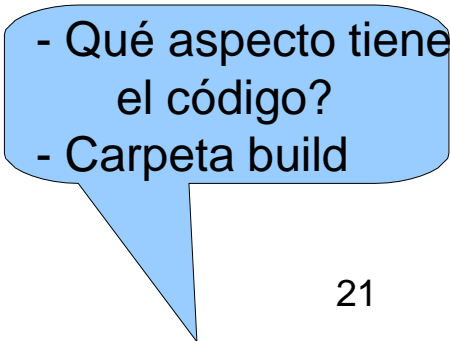
Primera página JSP

EJ1

```
<html>
  <body>
    <p> Hola mundo </p>
    <p>
      <%= new java.util.Date() %>
    </p>
  </body>
</html>
```

Recordatorio

- Las páginas JSP se convierten en código Java.
 - En programas que atienden las peticiones web.
 - Usar JSP nos evita el trabajo de tener que programarlos.
- Este posteriormente se compila y se ejecuta.
 - Todo ello se hace en el servidor.



- Qué aspecto tiene el código?
- Carpeta build

1.1.2 Introducción al lenguaje JSP

- Etiquetas básicas
- Errores
- Procesamiento de formularios

Etiquetas básicas de JSP

- `<% ... %>`: Incrustación de código (Scriptlet) Trozo de código java
- `<%= ... %>`: Expresión.
- `<%@ ... %>`: Directiva.
- `<%-- ... --%>`: Comentario.
- `<%! ... %>`: Declaración de atributos y métodos propios.

Incrustación de código Java

EJ3

```
<html>
  <body>
    <p>
```

Imprime numeros del 0 al 10 y la fecha abajo

Vamos a contar:

```
<% for(int i=1;i<=10;i++) {
    out.write(Integer.toString(i));
    if(i<10)
        out.write(", ");
    }
%>
```

```
</p>
<p>
```

Generada:

```
<% out.write( new java.util.Date().toString() );%>
```

```
</p>
</body>
</html>
```


Expresiones

EJ4

```
<html>
  <body>
    <p>
      Vamos a contar:
      <% for(int i=1;i<=10;i++) {
        %><%=i %><% Solamente podemos poner valor
        if(i<10) {
          %>, <%
        }
      }
      %>
    </p>
    <p>Generada: <%=new java.util.Date() %></p>
  </body>
</html>
```

Directivas

- Dan indicaciones **globales** sobre la página JSP:
- Sintaxis:
 - `<%@ directiva atributo="valor"%>`
 - `<%@ directiva atrib1="valor1" atrib2="valor2" ...%>`
- Directivas comunes:
 - page: Modifica algún parámetro de la página:
 - Inclusión de librerías.
 - Configuración del buffer.
 - ...
 - include: permite incluir algún otro archivo.
 - taglib: Permite cargar librerías de etiquetas.

Directiva “page”

- `<%@ page import="java.libreria.algo" %>`:
Importa la librería especificada.
- `<%@ page language="java" %>`: Indica el lenguaje de los scriptlets.
- `<%@ page contentType="text/html; charset=ISO-8859-1" %>`: Indica el tipo de contenido que generará la página.
- `<%@ page pageEncoding="ISO-8859-1" %>`: Indica la codificación de la página. (p.ej. Cod. latina).
- Se pueden combinar los elementos anteriores en una sola directiva:
 - Ej: `<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>`

Ejemplo Directiva “page”

EJ5

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <body>
    <p>
      Vamos a contar:
      <% for(int i=1;i<=10;i++) {
          %><%=i %><%
          if(i<10) {
              %>, <%
          }
      }
      %>
    </p>
    <p>Generada: <%=new Date() %></p>
  </body>
</html>
```

Ejemplo directiva “include”

EJ6

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <body>
    <p>
      Vamos a contar:
      <% for(int i=1;i<=10;i++){
        %><%=i %><% Se puede usar .out o imprimir la variable
        if(i<10){
          %>, <%
        }
      }
      %>
    </p>
    <%@ include file="pie.jsp" %>
  </body>
</html>
```

EJ6_Pie.jsp:

```
<%@ page import="java.util.Date"%>
<p>Generada en pie: <%=new Date() %></p>
```

Comentarios

EJ7

```
<html>
  <body>
    <!-- Este comentario se vera en el
           navegador -->
    Mira el codigo fuente,
    <%--  Pero este otro comentario no se
           vera --%
```

Declaraciones.

EJ8

```
<%! variable1; [variable2;] ... %>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%! int inicio = 1; %> Todas las variables se declaran arriba antes de usarlas
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
    <body>
        <p>
            Vamos a contar:
            <% for(int i=inicio;i<(inicio+10);i++){
                %><%=i %><%
                if(i<(inicio+9)){
                    %>, <%
                }
            }
            inicio+=10;
            %>
        </p>
        <%@ include file="pie2.jsp" %>
    </body>
</html>
```

Declaraciones: Pié de página

pie2.jsp

```
<%@ page import="java.util.Date"%>
<%! Date horaInicio = new Date(); %>
<p>Generada: <%=new Date() %> -
Cargada: <%=horaInicio%> </p>
```


¿Qué pasa con los errores?

- NetBeans (o Eclipse) **NO** compila (de verdad) las páginas JSP.
 - Lo intenta con buena intención.
- La compilación la hace el mismo servidor.
- Si hay algún error, éste se muestra al visualizar la página con el navegador.

- New
- version
- – L
- La
- Si
- vis

avaJava.com Web Tutorials - JSPs

Apache Tomcat/5.5.20 - Error report - Windows Internet Explorer

http://localhost:8080/web-archive-test/test

File Edit View Favorites Tools Help

Apache Tomcat/5.5.20 - Error report

HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

```

javax.servlet.ServletException: This is a ServletException
    com.cakes.TestServlet.performTask(TestServlet.java:30)
    com.cakes.TestServlet.doGet(TestServlet.java:17)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:689)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:802)
  
```

note The full stack trace of the root cause is available in the Apache Tomcat/5.5.20 logs.

Apache Tomcat/5.5.20

How do I create a JSP error page to handle exceptions?

Páginas JSP Interactivas

- Las páginas JSP pueden leer y tratar los valores de los campos de un formulario para interactuar con el usuario.
- Se puede acceder a dichos valores usando la referencia *request* que está definida para cualquier página JSP.
- *request* referencia a un objeto de la clase *HttpServletRequest*.
- Se emplea el método `getParameter("identificador")`
- Lectura recomendada: Documentación de la clase
 - <http://java.sun.com/javaee/5/docs/api/javax/servlet/http/HttpServletRequest.html>

Ejemplo: Formulario

EJ10

```
<html>
  <body>
    <form action="EJ11.jsp" method="get">
      Dime tu nombre:
      <input name="nombre" type="text"/>
      <input name="enviar" type="submit"
        value="enviar" />
    </form>
  </body>
</html>
```

Ejemplo: Procesamiento del formulario

EJ11

```
<%@ page language="java" contentType="text/html;  
    charset=ISO-8859-1"  
    pageEncoding="ISO-8859-1"%>  
<html>  
    <body>  
        Hola <%= request.getParameter("nombre") %>  
    </body>  
</html>
```

Ejercicio 1

- Cree una página JSP que muestre una tabla de números de 1 al 99 y su correspondencia en español.
 - Claves:
 - Considerar sólo los casos especiales estrictamente necesarios.
 - El resto de cifras se crean escribiendo un prefijo para las decenas y otro para las unidades.
 - Usar la estructura de programación más adecuada (¿Switch?).
- ¿Qué editor uso para el código JSP? ¿Cómo ejecuto el código?
 - NetBeans. Necesitáis una distribución que incluya “Java web and EE” y es recomendable que uséis Tomcat como servidor en lugar de Glass Fish **(Para ello es necesario hacer una instalación personalizada y seleccionarlo)**.
 - Cread un proyecto de tipo “Java Web” de la categoría “Web Application”. Podemos crear un archivo JSP nuevo pulsando el botón derecho sobre la carpeta “Web Pages”.
 - Podemos ejecutar dándole a la opción “Run File” al pulsar el botón derecho sobre un archivo JSP. O darle al “play” si queremos ejecutar la index.jsp.