Desarrollo de Software para Sistemas Distribuidos

Curso 2015

Desarrollando páginas JSP

- JSP (JavaServer Page) permiten escribir paginas HTML estándar que contienen tags que ejecutar programas Java. El objetivo de las paginas JSP es separar la logica de presentación de la lógica de negocios.
- Los diseñadores Web pueden diseñar y actualizar paginas sin saber programar en Java y los programadores pueden escribir código Java sin necesidad de diseñar paginas Web.

Servlet "Hola Mundo"

```
public void generarRespuesta    (HttpServletRequest
   request, HttpServletResponse response)
            Throws IOException {
// Determina el nombre especificado
String name = request.getParameter ("name");
If ( ( name == null ) ||  (name.length() == 0 ) ) {
   name = DEFAULT_NAME;
}
// Especifica que el tipo de contenido es HTML
response.setContentType ("text/html");
PrinterWriter out = response.getWriter () ;
```

Servlet "Hola Mundo" (cont.)

```
// Generar respuesta HTML
out.println ("<HTML>");
out.println ("<HEAD>");
out.println ("<TITLE>Hola Mundo</TITLE>");
out.println ("</HEAD>");
out.println ("<BODY BGCOLOR = 'white'>");
out.println ("<B>Hello, " + name + "</B>");
out.println ("</BODY>");
out.println ("</HTML>");

out.close();
}
```

"Hola Mundo" JSP

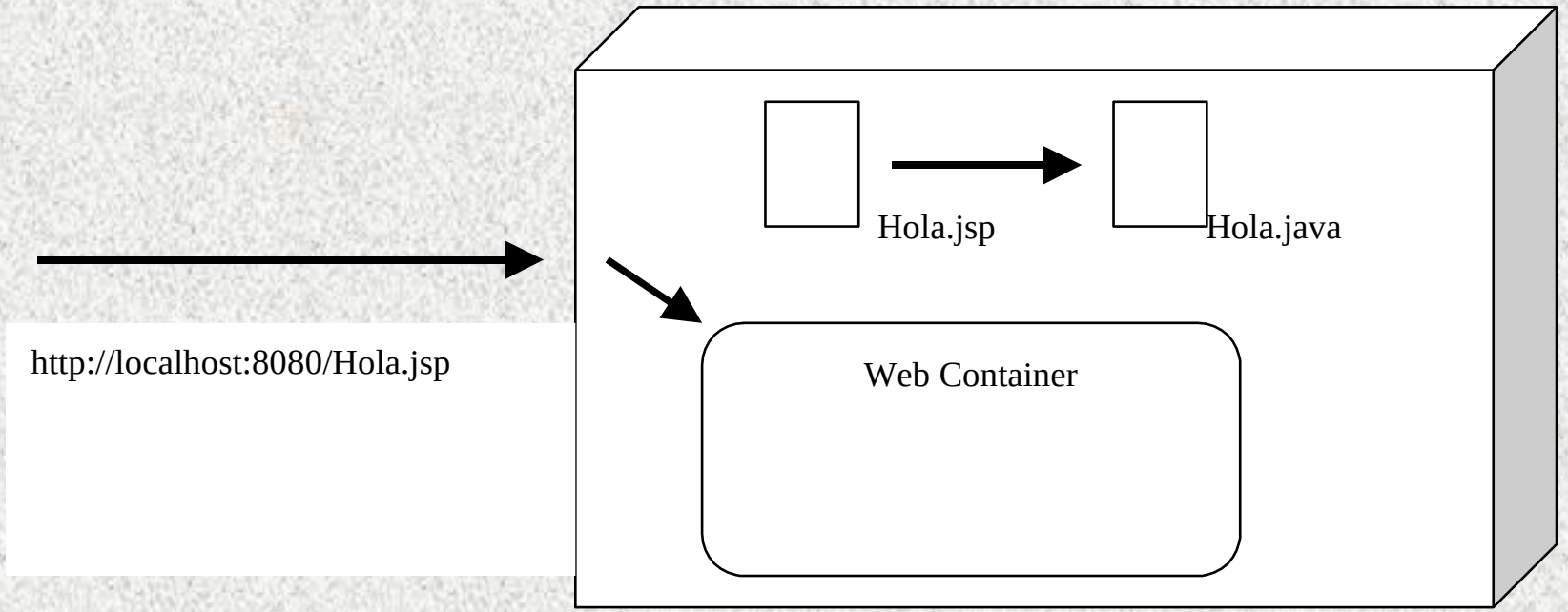
```
<%! private static final String DEFAULT_NAME = "Mundo" ; %>
<HTML>
<HEAD>
<TITLE> Hola Mundo en JSP </TITLE>
</HEAD>
<% -- Determinar el nombre especificado -- %>
<%
    String name = request.getParameter ("name");
    if ( ( name == null ) || (name.length () == 0 ) ) {
        name = DEFAULT_NAME;
    } %>
<BODY BGCOLOR = "white">
<B> Hello, <% =name %> </B>
</BODY>
</HTML>
```

Procesamiento de JSP

- La primera vez que una página JSP es requerida, el Web Container convierte el archivo JSP en un servlet que puede responder a los requerimientos HTTP.

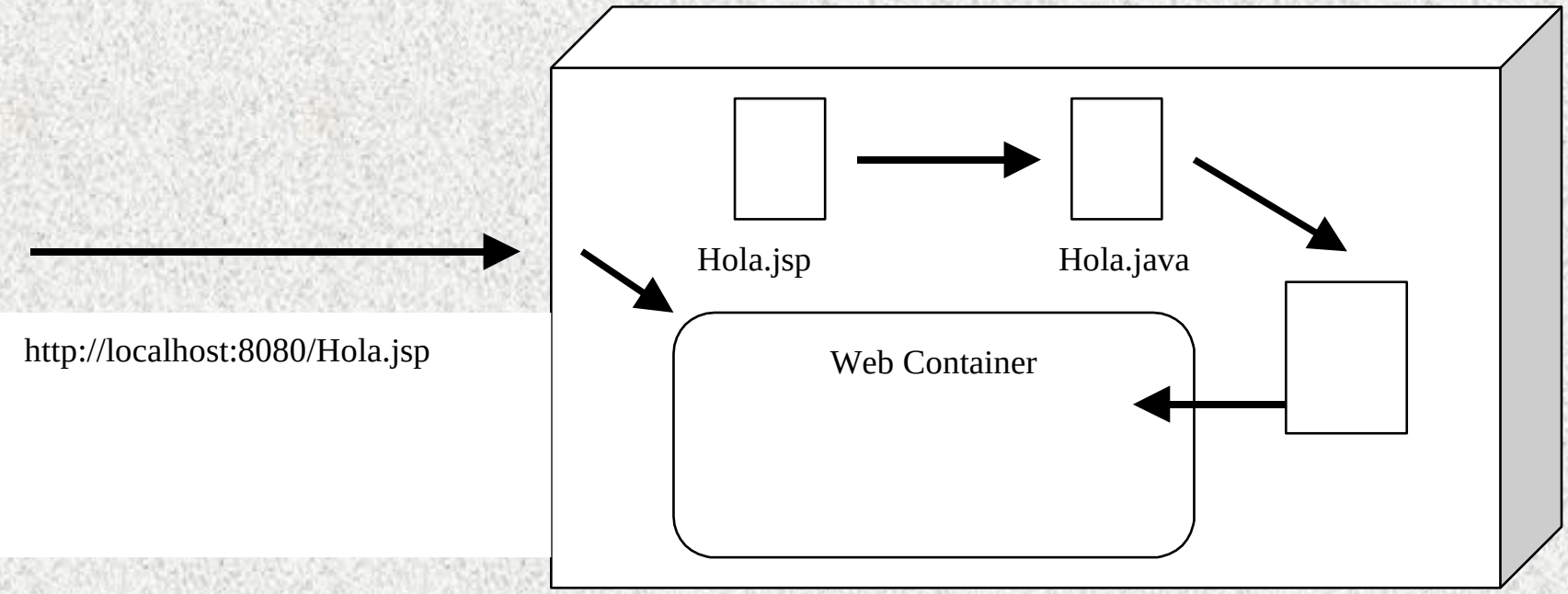
Procesamiento JSP - Paso 1

- En el primer paso el Container traduce el archivo JSP en código fuente Java que contiene la definición de una clase Servlet.



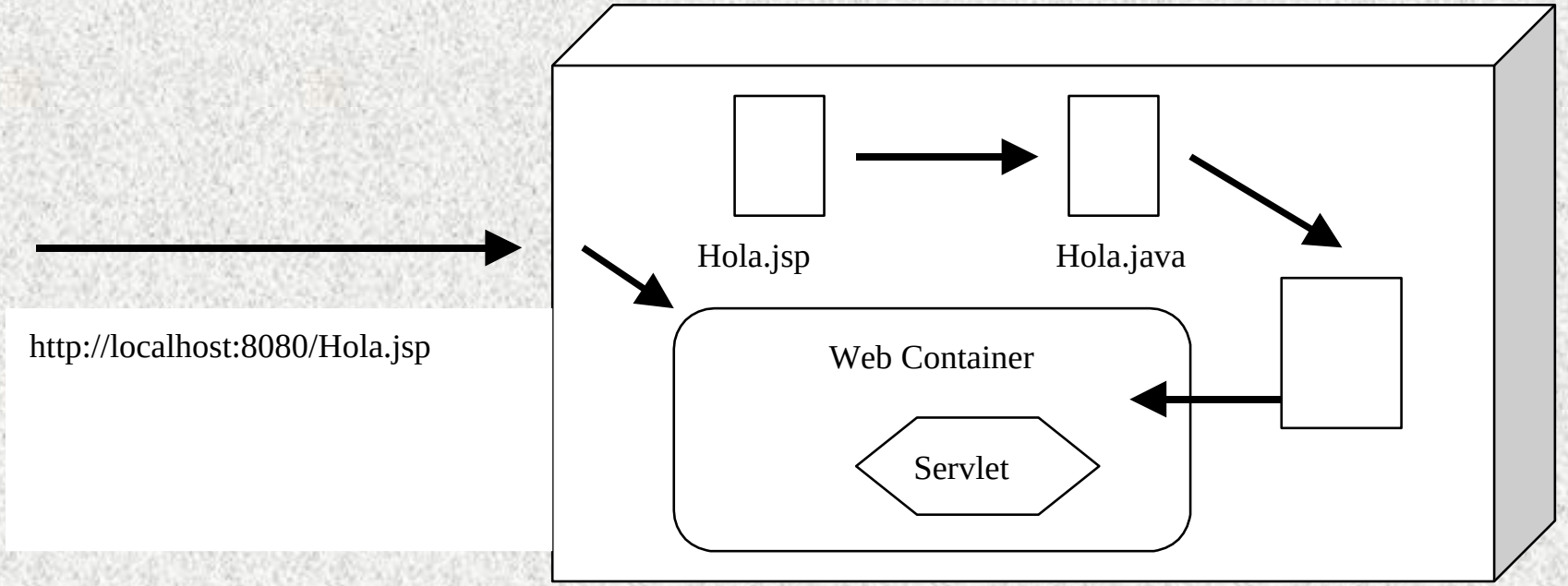
Procesamiento JSP - Paso 2

- En el segundo paso, el Web Container compila el código fuente Java en una archivo de clase Java.



Procesamiento JSP - Paso 3

- En el tercer paso, el Container crea una instancia de la clase servlet y realiza los pasos del ciclo de vida del mismo invocando un método especial que es *jspInit*.



Elementos de scripting

- Los elementos de scripting de JSP se escriben encerrados entre <% y %> y son procesados durante el tiempo de traducción de la página JSP a código fuente Java.
- Cualquier otro texto dentro la página JSP es considerado parte de la respuesta y es copiado tal cual.

Elementos de scripting

- Comentarios <% -- texto del comentario --%>
- Directiva <%@ directiva %>
- Declaración <%! Declaracion %>. Permite incluir atributos o metodos de otra clase Java.
- Scriptlet <% codigo fuente Java %>
- Expresión <%= expresión %>. Es una expresión Java que es evaluada al durante un requerimiento HTTP.

Variables implícitas

- **Request**: El objeto *HTTPServletRequest* asociado con el requerimiento
- **Response**: El objeto *HTTPServletResponse* asociado con la respuesta que envía el Browser
- **Out**: El objeto *JspWriter* asociado con el stream de salida de la respuesta
- **Session**: El objeto *HttpSession* asociado con la sesión para el requerimiento de un usuario dado.

Variables implícitas

- **Application**: El objeto *ServletContext* asociado con el servlet para esta página JSP
- **Config**: El objeto *ServletConfig* asociado con el servlet para esta página JSP
- **pageContext**: Este objeto encapsula el ambiente de un requerimiento único para la pagina JSP

Detrás de escena

Lo que sucede al momento de "ejecutar" una página JSP está representado por tres etapas o tiempos:

- **Tiempo de traducción**: el trabajo que hace el Container para transformar la pagina JSP en código fuente Java.
- **Tiempo de compilación**: el trabajo que hace el Container para transformar el código fuente Java en una clase en código objeto (bytecode).
- **Tiempo de ejecución**: el trabajo que hace el Container para administrar el ciclo de vida del servlet. Esto es: instanciarlo si es la primera vez, o bien usarlo si la instancia ya existe.

JavaBean

- Una componente JavaBean es una clase Java que posee al menos las siguientes características:
  - Las propiedades se definen con métodos para accederlas y modificarlas (llamadas comunmente “seters” y “geters”
  - Posee un método constructor sin argumentos que permite instanciar objetos de esa clase.
  - Las variables de instancia no son públicas, es decir, sólo se acceden y modifican por los “seters” y “geters”

JavaBean

- Un JavaBean es más que nada una técnica de diseño y puede utilizarse dentro de una página JSP bien escribiendo código Java o bien utilizando tags XML.
- En este último caso, el tag estándar es:

  <jsp: *acción* [*atributo* = *"valor"*]/>

JavaBean: tag XML Vs scriplet

Se utiliza el tag jsp:useBean. Por ejemplo:

```
<jsp:useBean id=”libro” class = “Libro” />
```

que es equivalente al siguiente scriplet dentro de la pagina JSP

```
<%
  Libro libro = new Libro();
%>
```

Alcances del JavaBean

- **Aplicación**: los objetos son accesibles desde paginas que pertenecen a la misma aplicación Web donde fueron creadas y son almacenadas en el objeto ServletContext.
- **Sesión:** los objetos son accesibles desde páginas que pertenecen a la misma sesión donde fueron creadas y son almacenadas en el objeto HttpSession.

Alcances del JavaBean (cont.)

- **Request:** los objetos son accesibles desde páginas que procesan un requerimiento donde ellas fueron creadas y son almacenadas en el objeto SevletRequest.
- **Page**: los objetos son accesibles solamente dentro de las paginas que ella misma ha creado y se almacenan en el objeto PageContext.

JavaBean en JSP reduciendo scripting

- Crear una componente JavaBean usando el tag *jsp:useBean*
- Establecer propiedades en el Bean, usando el tag *jsp:setProperty*.
- Redirigir desde una pagina JSP a otra usando el tag *jsp:forward*
- Acceder a las propiedades del Bean usando el tag *jsp:getProperty*

Model View Controler

- Motivación:
  - Una aplicación Web interactúa con un usuario que accede con un Web Browser y envía requerimientos HTTP
  - El diseño de una aplicación Web tiene ciertas limitaciones. La más importante es que el método doPost hace todo el trabajo. Esto atenta contra el principio de modularización y además colapsa cuando crece la complejidad de la aplicación

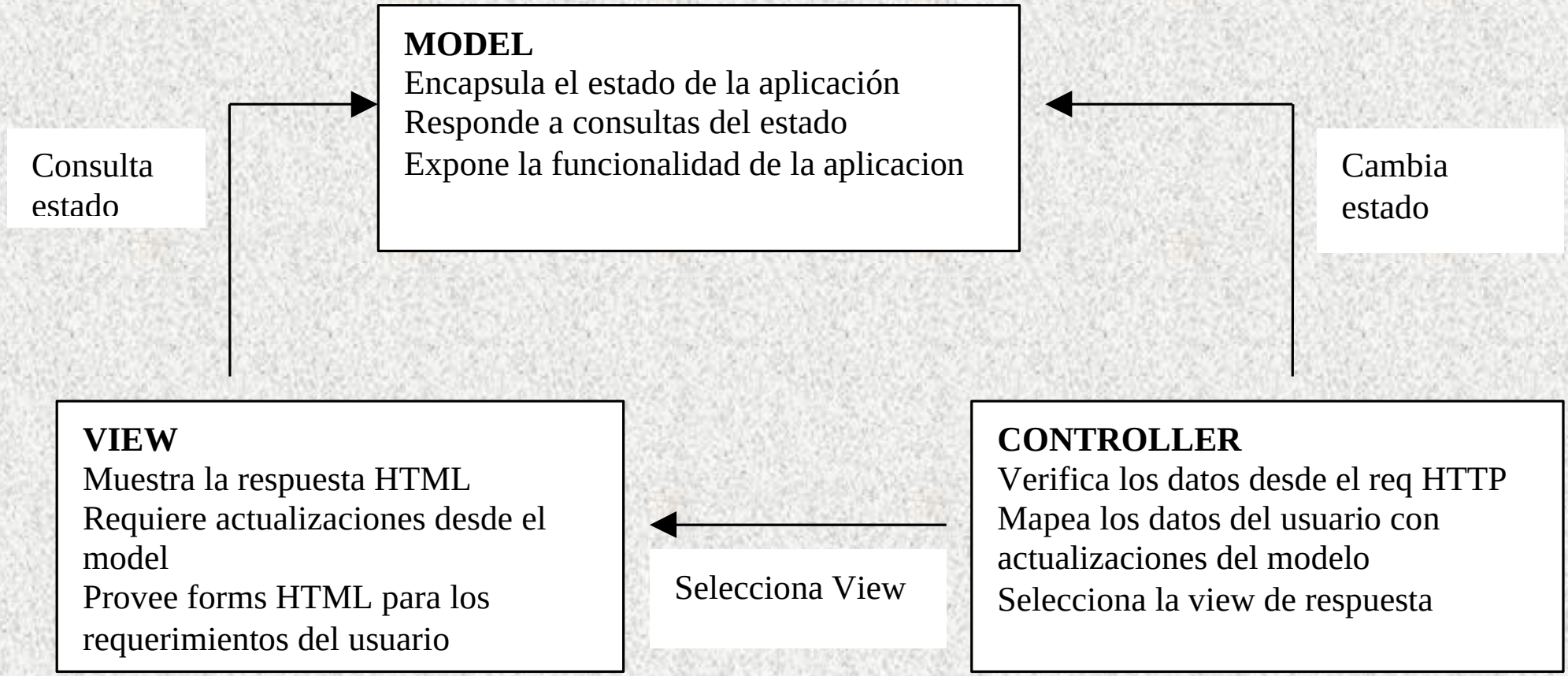
Model View Controler

- Una aplicación Web procesa los requerimientos utilizando un servlet que realiza los siguientes pasos:
  - Verificar los datos de un formulario HTML
  - Enviar una página de error si los datos fallan en la verificación
  - Procesar los datos, que pueden almacenar información persistente
  - Enviar una página de error si el procesamiento falla.
  - Enviar una página de respuesta si el procesamiento tuvo éxito.

Patrón de Diseño MVC

- Tiene sus raíces en el desarrollo de GUI en Smalltalk.
- El propósito es separar tres clases de tareas de una aplicación:
  - Model: los servicios de negocios y objetos del dominio de aplicación
  - View: la “ventana” dentro de la aplicación que se presenta al usuario
  - Controler: la lógica que acepta las acciones del usuario, realiza la operación y selecciona la próxima View para el usuario.

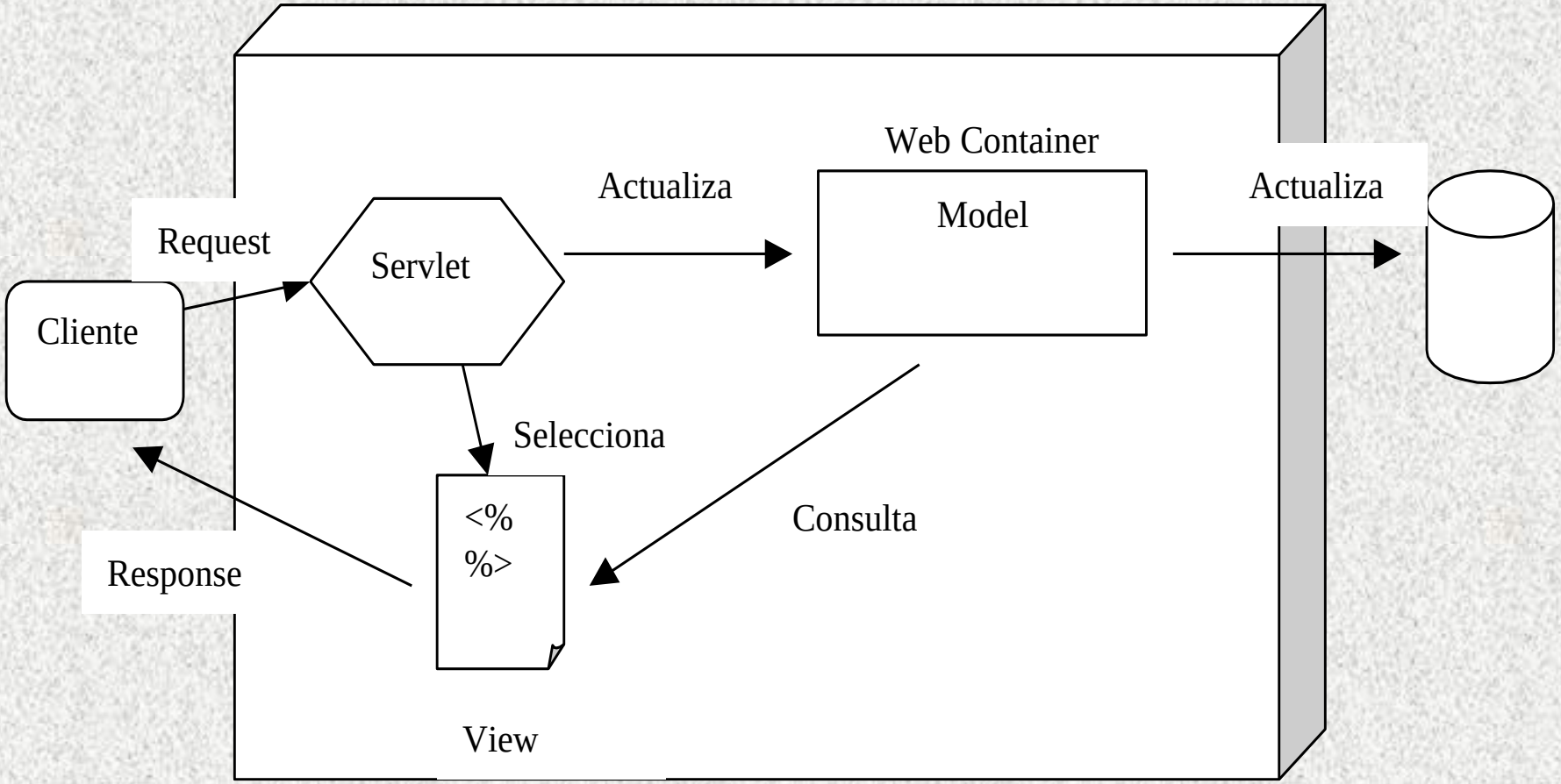
MVC


MODEL
Encapsula el estado de la aplicación
Responde a consultas del estado
Expone la funcionalidad de la aplicacion
Consulta estado
Cambia estado
VIEW
Muestra la respuesta HTML
Requiere actualizaciones desde el model
Provee forms HTML para los requerimientos del usuario
Selecciona View
CONTROLLER
Verifica los datos desde el req HTTP
Mapea los datos del usuario con actualizaciones del modelo
Selecciona la view de respuesta

Diseño utilizando MVC

- El Model es construído con clases Java o con componentes JavaBean.
- El View es construido con páginas JSP
- El Controller se implementa con servlets.

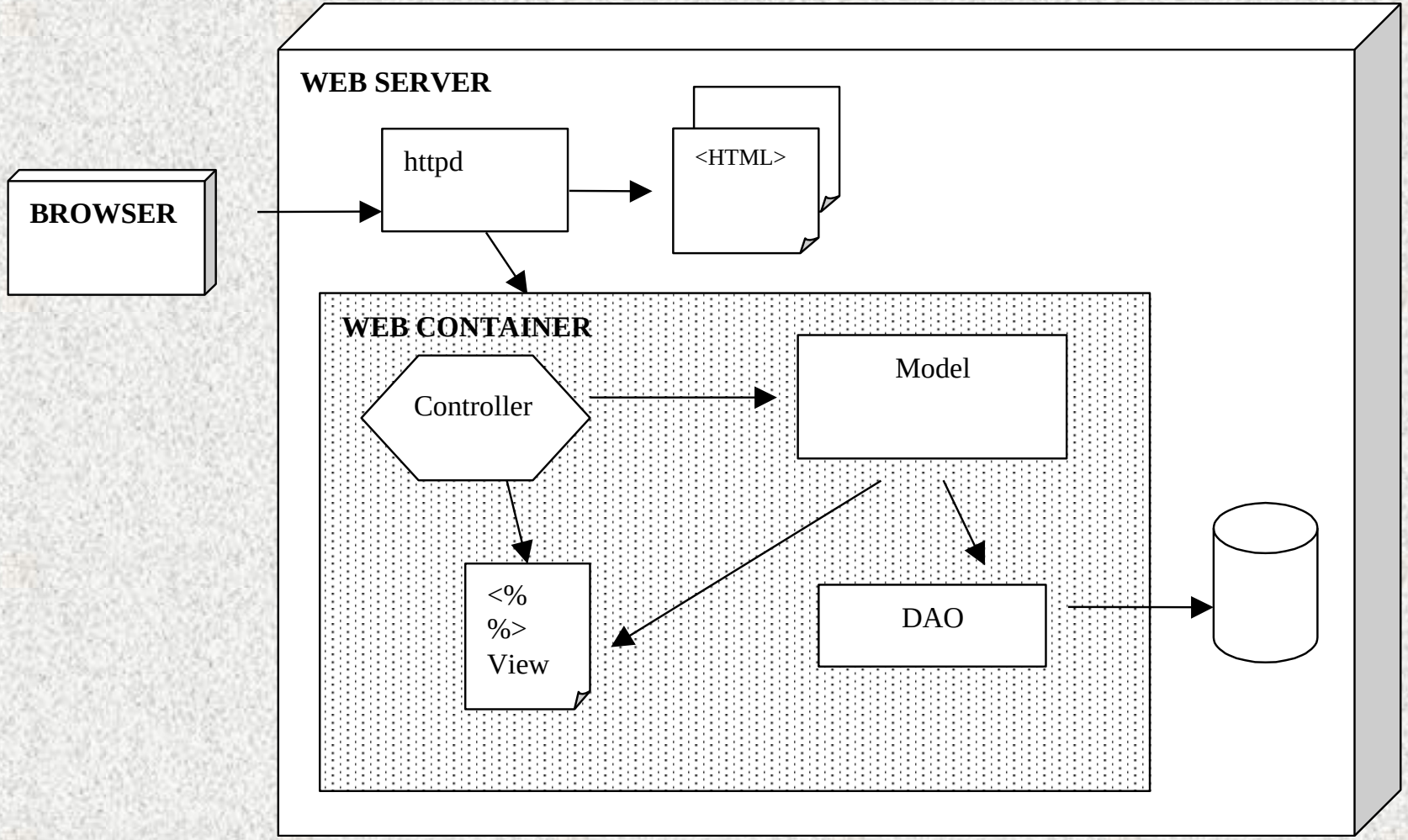
Ciclo de vida



Servlet Controller y JSP View

- El servlet Controller debe:
  - Verificar el formulario de datos HTML
  - Invocar los servicios de negocios en el Model
  - Almacenar los objetos del dominio en el alcance del request (o sesión)
  - Seleccionar la próxima View para el usuario
- La JSP View debe:
  - Presentar la interfase de usuario (en HTML)
  - Acceder a los objetos del dominio para generar el contenido dinámico

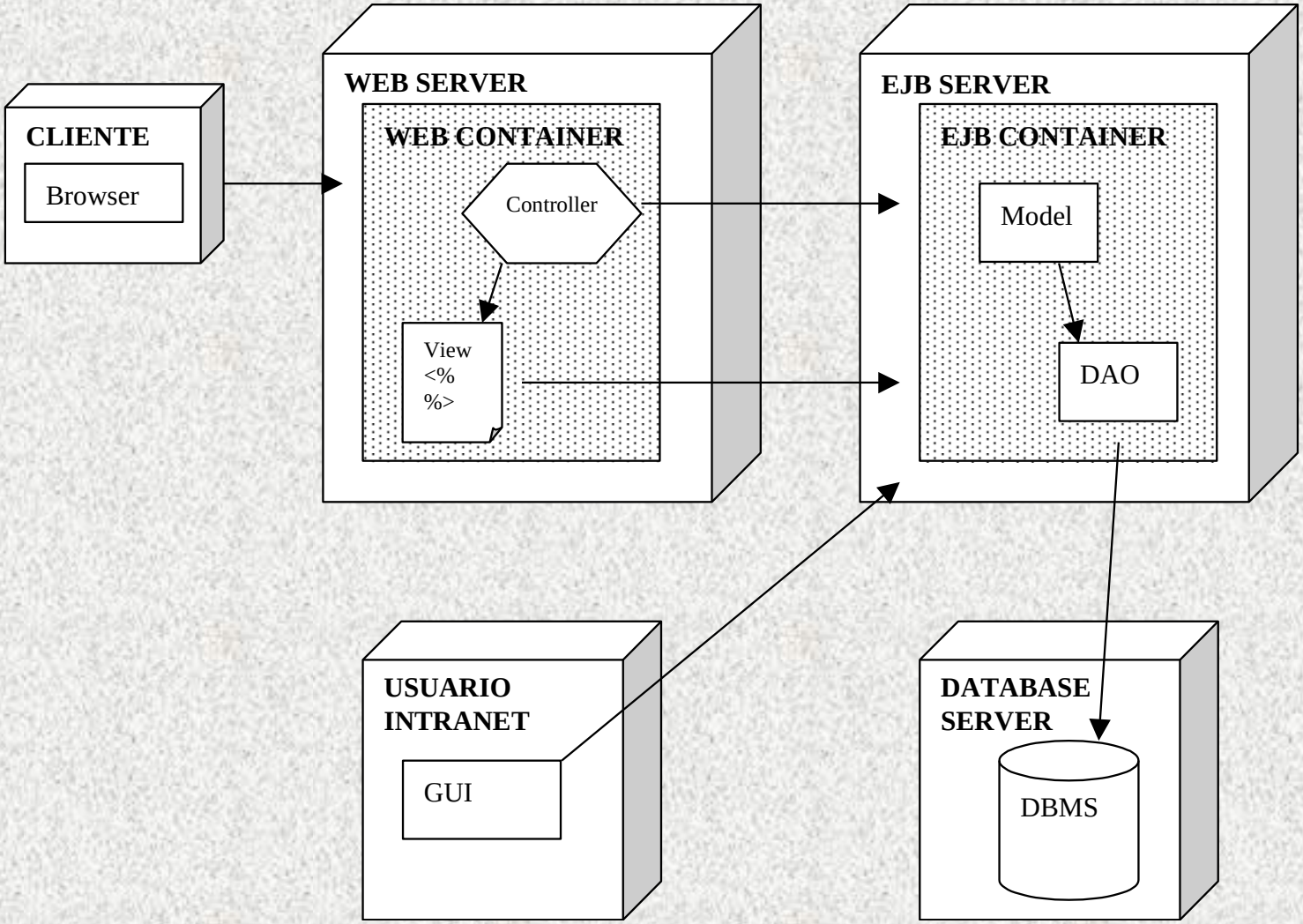
Aplicación Web con Servlet y JSP



Aplicación Web con Servlet y JSP

- La lógica de presentación, la lógica de negocios y la de acceso a los datos son resueltas por el Web Server.
- Inconvenientes:
  - Incrementa la carga de trabajo en un único host
  - El Web Server es eficiente en la atención de requerimientos concurrentes, mientras que lógica de negocios se caracteriza por consumir mucha CPU.
  - Si el proyecto requiere de clientes GUI “standalone” (que se ejecuten fuera del Web Browser), deberán duplicarse la lógica de negocios y acceso a datos, con el correspondiente manejo de transacciones.

Plataforma J2EE



Tecnología EJB

- La plataforma J2EE incluye tecnología EJB que son JavaBeans que incluyen control de transacciones, seguridad y concurrencia.
- Los EJB pueden ser:
  - de entidad: encapsulan objetos del dominio como las entidades del modelo ER.
  - de sesión: se utilizan para encapsular servicios de negocios