ESCUELA DE SISTEMAS Y TECNOLOGÍAS

Transparencias de **ANALISTA DE SISTEMAS**

Edición 2020 Materia: **Java Web**

TEMA: JSP y MVC

Agenda

- ¿Qué es una Página JSP?
- Elementos JSP
- Expresiones EL
- Ciclo de Vida de una Página JSP
- Manejo de Errores
- JavaBeans
- JSP y JavaBeans
- Servlets y JavaBeans
- Arquitectura MVC

¿Qué es una Pagina JSP? (1)

- Una página JSP es un documento de texto que contiene dos tipos de texto:
  - **Información estática**: que puede ser expresada en cualquier formato basado en texto (ej: HTML, WML, XML). También se lo conoce como template text.
  - **Elementos JSP**: que construyen contenido dinámico. Estos elementos JSP pueden ser expresados de dos formas (sintaxis): estándar o XML. Se abordará aquí la forma estándar. Una página JSP con sintaxis XML es un documento XML que puede ser manipulado por APIs y herramientas para manejo de XML (ver Referencias).

Elementos JSP (1)

- Existen 3 tipos de elementos JSP:
  - **Directivas** (***directives***): especifican información acerca de la página que se mantiene igual a lo largo de todos los pedidos.
  - **Acciones** (***actions***): permite realizar alguna acción basada en información al momento de hacer el pedido.
  - **Scriplets** (***scripting***): permiten incorporar código Java dentro de una página JSP.
- A continuación se presenta una breve descripción de cada tipo de elemento JSP. Varios de ellos serán posteriormente estudiados en mayor detalle

Elementos JSP (2)

- **Directivas:**

Directiva	Descripción
`<%@page ... %>`	Define atributos de la página que controlan cómo el contenedor web traduce y ejecuta la página JSP.
`<%@include ... %>`	Incluye otra página JSP durante la fase de traducción.
`<%@taglib ... %>`	Declara una librería de tags, conteniendo acciones, que es utilizada en la página.

Elementos JSP (3)

- **Acciones estándar**:

Acción	Descripción
`<jsp:useBean>`	Permite utilizar un JavaBean desde la página JSP.
`<jsp:getProperty>`	Permite obtener el valor de una propiedad del JavaBean.
`<jsp:setProperty>`	Permite establecer el valor de una propiedad del JavaBean.
`<jsp:include>`	Incluye la respuesta de un servlet o página JSP durante la fase de pedido de la página.
`<jsp:forward>`	Redirige el pedido a un servlet o página JSP.
`<jsp:param>`	Agrega un parámetro a un pedido redirigido mediante `<jsp:include>` o `<jsp:forward>`
`<jsp:plugin>`	Permite incluir un applet.

Elementos JSP (4)

- **Scriplets**:

Elemento	Descripción
<% ... %>	Se denomina *scriplet* (surge de *scripting element*) y permite colocar código Java dentro.
<%= ... %>	Expresión: permite incorporar una expresión Java que es evaluada y su resultado (en *String*) es incorporado a la salida HTML.
<%! ... %>	Declaración: permite incorporar una declaración a ser utilizada por los *scriplets*.

Expresiones EL (1)

- Desde la versión JSP 2.0 existe un lenguaje de expresiones denominado **EL** (***Expression Language***).
- Dicho lenguaje permite a los autores de páginas JSP (particularmente a quienes no son programadores) escribir expresiones relativamente sencillas para leer valores de un JavaBean dinámicamente y/o establecer valores de atributos de acciones basados en datos recolectados en tiempo de ejecución.
- Estas expresiones permiten potenciar el poder de las acciones JSP, ya sean las acciones estándar o las incluidas en **JSTL** (Java Standard Tag Library).

Expresiones EL (2)

- Dicho lenguaje (EL) contiene características similares a JavaScript.
- Una expresión EL comienza con los caracteres "**${**" y termina con el carácter "**}**"
- Ejemplo de uso de una expresión EL para el seteo de un atributo de un JavaBean (*el uso de JavaBeans será abordado más adelante*):

```
<jsp:useBean name="miBean" class="mipaquete.MiBean"
             scope="session">
    <jsp:setProperty name="miBean" property="prop"
                     value="${expresionEL}" />
</jsp:useBean>
```

Expresiones EL (3)

- Desde la versión JSP 2.1 existe el **Unified Expression Language** (conocido como ***Unified*** EL) el cual representa una unión del lenguaje EL utilizado por **JSP** 2.0 más el lenguaje de expresiones utilizado por **JSF** (*JavaServer Faces*).
- Los agregados son:
  - Evaluación diferida de expresiones (pues el ciclo de vida de una JSF es más complejo que el de una JSP).
  - Setear datos además de obtenerlos (permitir que dentro de una expresión EL se establezcan nuevos valores).
  - Poder invocar métodos (permitir que dentro de una expresión EL se invoquen métodos).

Expresiones EL (4)

- Para que las expresiones EL puedan ser interpretadas y su resultado pueda ser devuelto al browser en forma de HTML, el atributo **isELIgnored** de la directiva **@page** debe estar habilitado.
- De no estarlo, las expresiones EL se mostrarán como texto plano en la página JSP.
- Uso:

```
<%@page isELIgnored="[true|false]" %>
```

- Por defecto, las expresiones EL están habilitadas (es decir con el valor **false**).

Expresiones EL (5)

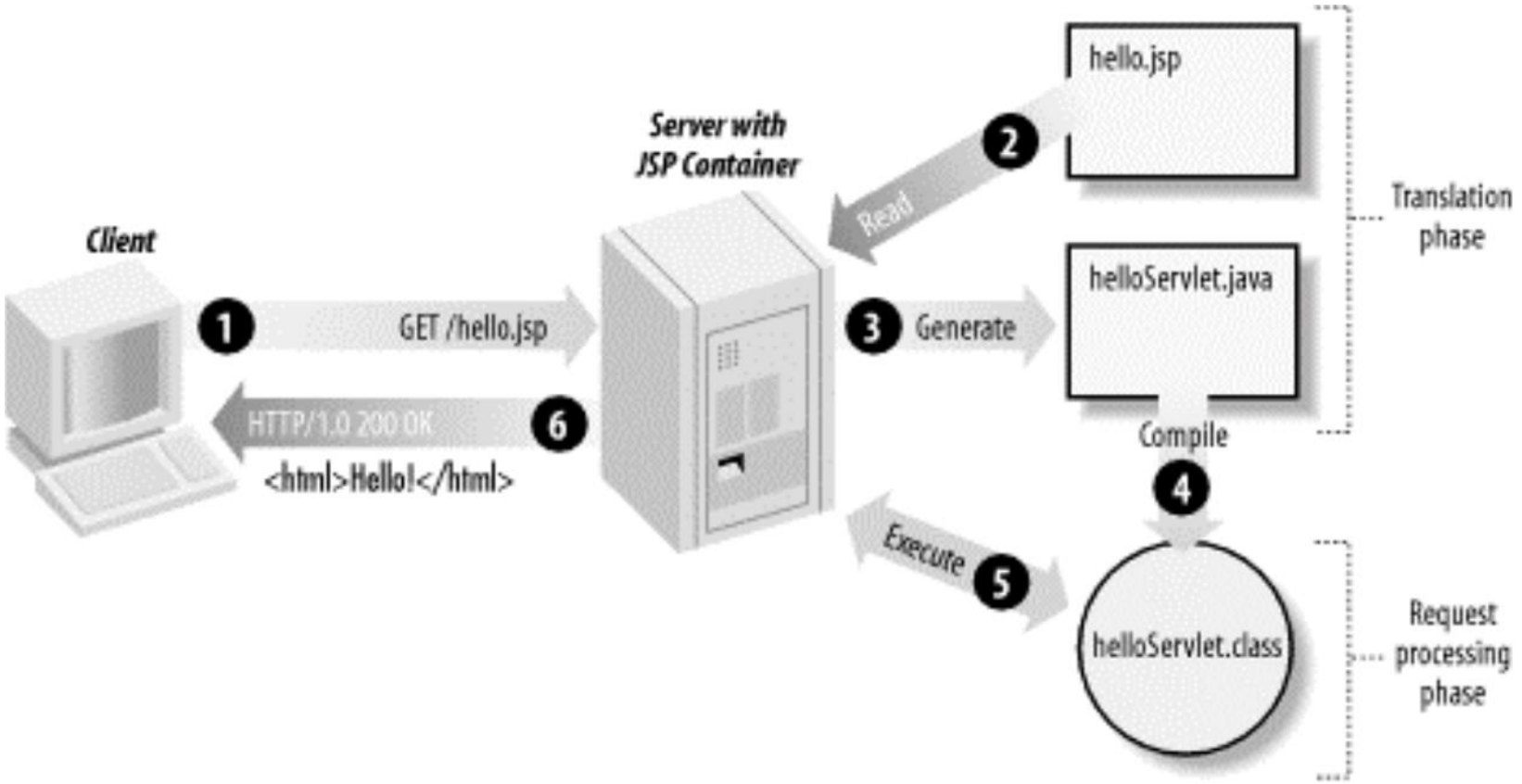
- El **Unified EL** permite:
  - Que los autores de páginas web utilicen conjuntamente contenido JSP y tags JSF.
  - Que tecnologías relacionadas con JSP hagan uso de las nuevas características del Unified EL. Si bien JSTL continúa utilizando únicamente las características del lenguaje EL de JSP 2.0, nuevos tags personalizados (custom tags) pueden tomar ventaja de las nuevas características.
- Por más detalles sobre expresiones EL ver Referencias, particularmente el JavaEE Tutorial

Ciclo de Vida de una pagina JSP (1)

- Dado que una página JSP se convierte en un servlet, éstos comparten muchos de los pasos en el ciclo de vida (incialización, servicio, destrucción).
- No obstante, el paso de traducción de JSP a servlet y su posterior **compilación** merecen especial atención (*translation phase*).
- Cuando un pedido se mapea con una página JSP, el contenedor web verifica si el servlet disponible del JSP se encuentra actualizado.
- De no estarlo (o de no existir) el contenedor web traduce la página JSP en una clase servlet y la compila

Ciclo de Vida de una pagina JSP (2)

- Procesamiento de un pedido a una página JSP, incluyendo su traducción y compilación:



Ciclo de Vida de una pagina JSP (3)

- Durante la traducción, cada uno de los 2 tipos de texto que contiene la página JSP son traducidos de forma diferente:
  - La información estática se transforma en código que escribirá los datos estáticos en el flujo de salida (ej: out.println("..."))
  - Los elementos JSP: son traducidos de diferente forma; la siguiente tabla muestra varios casos…

Ciclo de Vida de una pagina JSP (4)

Elemento JSP	Traducción
Directivas	Son utilizadas para controlar cómo el contenedor web traduce y ejecuta la página JSP.
Scriplets	Son insertados dentro del servlet generado.
Expresiones EL	Son pasadas por parámetro a llamadas al *JSP Expression Evaluator*.
`jsp:[get|set]Property`	Son convertidos en llamadas a métodos de JavaBeans (a los *getters* y *setters* respectivamente)
`jsp:[include|forward]`	Son convertidos en invocaciones a la API de servlets (*include* y *forward* respectivamente)
`jsp:plugin`	Es convertido en código de marcas (*markup language*) para activar el applet.
Custom Tags	Son convertidos en llamadas al *tag handler* que implementa el *custom tag*.

Ciclo de Vida de una pagina JSP (5)

- **NetBeans** permite ver el servlet a generar haciendo clic derecho sobre la página JSP, opción View Servlet.
- El archivo .**java** con el código resultado de la traducción lleva el nombre de la página JSP original agregándose *_jsp* al final (ej: *index.jsp* → *index_jsp.java*) y se aloja dentro del servidor web.
- Tanto la traducción como la compilación pueden producir errores que son observados cuando la página sea solicitada por primera vez.
- Frente a estos errores, el servidor lanza una **JasperException**

Ciclo de Vida de una pagina JSP (6)

- La nueva clase Java resultante de la traducción contiene los métodos:
  - public void _jspInit()
  - public void _jspService(HttpServletRequest, HttpServletResponse)
  - public void _jspDestroy()

- Estos métodos son los análogos a los vistos para servlets (pues de hecho se ejecutará como servlet).

Manejo de Errores (1)

- JSP permite redirigir automáticamente al usuario a una página determinada cuando un error ocurra durante la ejecución de una página.
- Esto se determina en la directiva **@page** dentro de la página que lanza el error:

```
<%@page errorPage="paginaError.jsp"%>
```

- Así como también en la página de error:

```
<%@page isErrorPage="true"%>
```

- Esta última directiva le permite a la página de error tener acceso al objeto de error, el cual es de tipo **javax.servlet.jsp.ErrorData**

Manejo de Errores (2)

- Este objeto de error permite, por ejemplo, poder interpretar el error ocurrido y brindar al usuario un mensaje detallado.
- Importante: si utiliza Internet Explorer, deshabilite la opción avanzada “Mostrar mensajes de error HTTP descriptivos”; de lo contrario el IE mostrará su propia página de error y no la que Ud. programó.

JavaBeans (2)

- En lo que respecta a las propiedades de un JavaBean, éstas pueden ser:
  - De sólo lectura, de sólo escritura o de lectura / escritura.
  - Simple (contiene un solo valor) o indexada (representa una colección de valores).
- Las propiedades no tienen por qué ser implementadas dentro del JavaBean (ej: mediante atributos de instancia) pero sí deben ser accesibles mediante *getters* y *setters* públicos de la forma habitual:
  - Tipo getPropiedad()
  - void setPropiedad(Tipo)

JSP y JavaBeans (1)

- Para que una página JSP utilice un JavaBean se utiliza la acción estándar **<jsp:useBean>**
- Sintaxis:

```
<jsp:useBean id="nombre" class="clase" scope="ambito"/>
```

1. **id**: determina el nombre del bean en el ámbito especificado (también para las expresiones EL y *scriplets*).
2. **class**: *fully qualified classname* (es decir el nombre de la clase que implementa el bean incluyendo su *package*).
3. **scope**: el ámbito del bean (si el bean no existe previamente en ese ámbito, se lo crea).

JSP y JavaBeans (2)

- Debido a que el atributo **class** requiere el *fully qualified classname*, las clases que implementan los JavaBeans deben estar dentro de un *package* para poder ser utilizadas desde una página JSP.
- También se pueden especificar **parámetros de inicialización** para el bean, los cuales serán seteados sólo si el bean no existía previamente en el ámbito especificado.
- Sintaxis:

```
<jsp:useBean id="nombre" class="clase" scope="ambito">
    <jsp:setProperty ... >
</jsp:useBean>
```

JSP y JavaBeans (3)

- Notar que el **<jsp:setProperty>** fue colocado dentro del **<jsp:useBean>** lo cual tendrá el efecto de setear valores únicamente al momento de instanciar el bean.
- Esto en el caso de un **scope="session"** significa una sola vez (mientras la *Session* esté activa) y en el caso de un **scope="request"** significa cada vez que se solicita el recurso (pues es un nuevo *request* cada vez).
- No obstante, el **<jsp:setProperty>** también puede ser utilizado fuera del **<jsp:useBean>** lo cual tendrá el efecto de setear valores cada vez que se solicita la página.

JSP y JavaBeans (4)

- La sintaxis del **<jsp:setProperty>** depende del origen del valor a setear , es decir de dónde se obtiene el valor que se seteará en el bean.
- En la tabla siguiente el atributo **name** debe contener el **id** del bean a setear:

Origen	Sintaxis
Un texto	`<jsp:setProperty name="id_del_bean" property="nombre_prop" value="un_string"/>`
Un parámetro del *request*	`<jsp:setProperty name="id_del_bean" property="nombre_prop" param="un_request_param"/>`

JSP y JavaBeans (5)

- (cont.)

Origen	Sintaxis
Un parámetro del *request* que coincide con el nombre de la propiedad	`<jsp:setProperty name="id_del_bean" property="nombre_prop"/>`
Todos los parámetros del *request*	`<jsp:setProperty name="id_del_bean" property="*"/>`
Expresión EL	`<jsp:setProperty name="id_del_bean" property="nombre_prop" value="expresion_EL"/>`

JSP y JavaBeans (6)

- Para obtener el valor de una propiedad de un bean se utiliza la acción **<jsp:getProperty>** luego de haber declarado el bean mediante **<jsp:useBean>**
- Sintaxis:

```
<jsp:getProperty name="id_del_bean"
                 property="nombre_prop"/>
```

- También se puede utilizar una expresión EL:

```
${id_del_bean.nombre_prop}
```

- O incluso un *scriplet*:

```
<%= id_del_bean.getNombre_prop() %>
```

JSP y JavaBeans (7)

- **Observación**:
  - Dado que la página JSP se ejecuta en el servidor, cuando la tabla anterior habla de request se refiere al pedido por la propia página JSP.
  - Por lo tanto cuando se habla de setear una propiedad del bean con un parámetro del request, son los parámetros del request por el cual se llegó a la página, y no (por ejemplo) los parámetros del request generado por un formulario dentro de la página.

Servlets y JavaBeans

- Debido a que un servlet es una clase Java y un JavaBean también, éstos se utilizan como clases normales dentro del código del servlet.
- Por tanto, para obtener un bean que una página colocó, basta con obtener una referencia al ámbito (ej: Session) y solicitar el bean según el **id** con el cual éste fue guardado.
- Para setear un bean desde el servlet, basta con instanciarlo, cargarlo y dejarlo disponible en el ámbito deseado.

Arquitectura MVC (1)

- Si bien se puede desarrollar un sitio web completamente funcional únicamente utilizando páginas JSP o únicamente utilizando clases servlet, existe la posibilidad de lograr una arquitectura que aproveche las ventajas de cada tecnología, combinándolas en una misma solución, junto con los JavaBeans Components.
- Dicha arquitectura (diseño de alto nivel) se conoce como **MVC** (*Model - View - Controller*) o también como ***Model-2 Architecture***.

Arquitectura MVC (2)

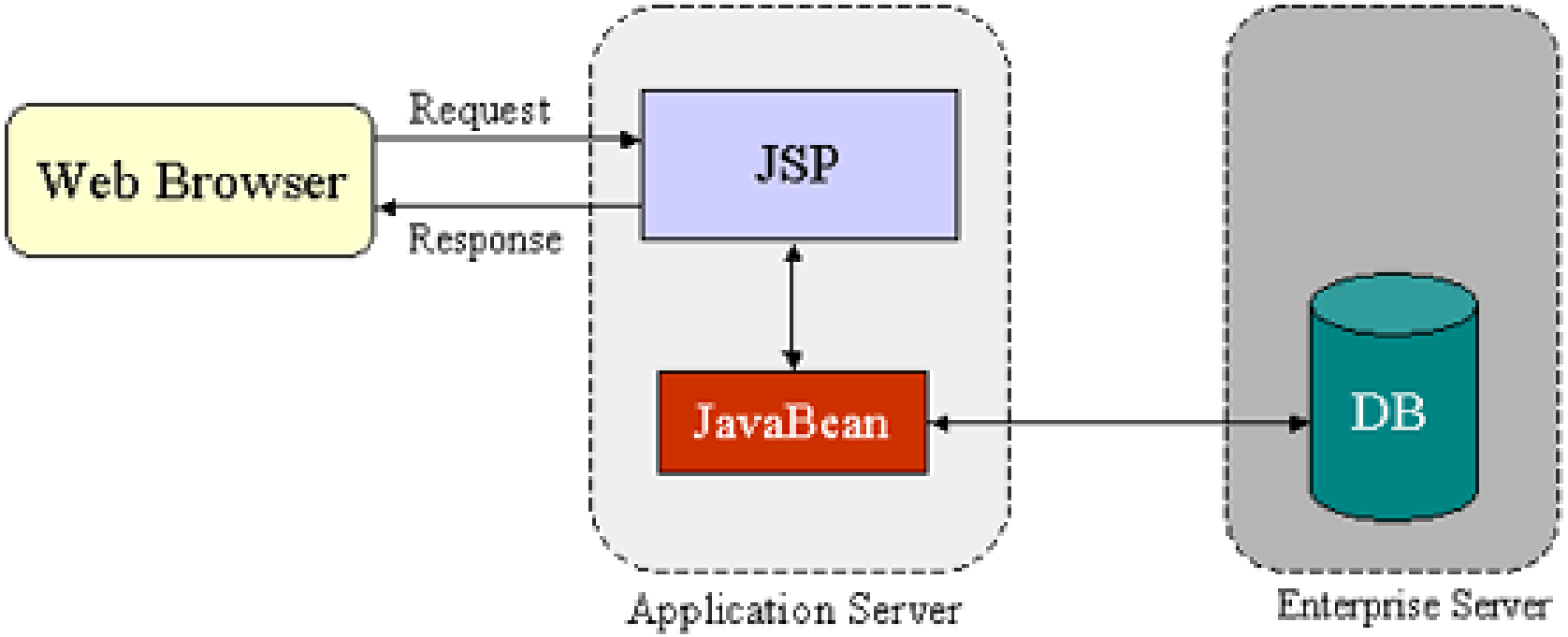
- En una arquitectura MVC las responsabilidades se separan de la siguiente forma:
  - **Model**: representa datos del negocio (opcionalmente con cierta lógica y con operaciones que regulan el acceso y modificación de dichos datos).
  - **View**: obtiene los datos de un Model y determina cómo esos datos deben ser presentados. También redirige las entradas del usuario hacia el Controller.
  - **Controller**: define el comportamiento de la aplicación. Responde a los pedidos del usuario y selecciona las vistas (View) a retornar utilizando beans para el envío de información

Arquitectura MVC (3)

- En términos muy generales, la relación entre MVC y las tecnologías Web Java es:
  - **Model**: JavaBeans Components
  - **View**: Java Server Pages
  - **Controller**: Servlets
- La arquitectura MVC permite:
  - Que cada componente evolucione de forma independiente.
  - Mantener separada la presentación de la lógica.

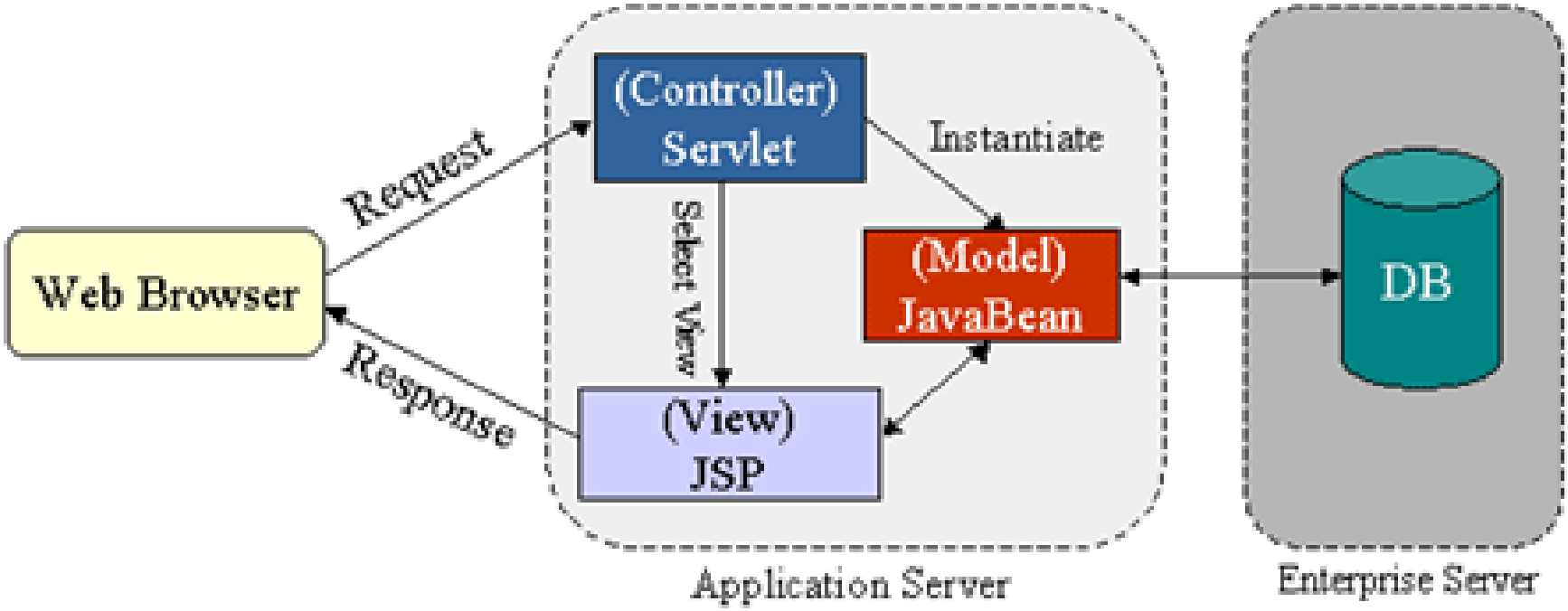
Arquitectura MVC (4)

- La arquitectura **Model-1** se puede representar de la siguiente manera:



Arquitectura MVC (5)

- La arquitectura **Model-2 (MVC)** se puede representar de la siguiente manera:



Arquitectura MVC (6)

- Como puede apreciarse del diagrama, en la arquitectura **Model-1** las páginas JSP son responsables por el procesamiento de los pedidos y la generación de las respuestas, utilizando JavaBeans para colocar cierta lógica así como conexiones a BD.
- Por otro lado, la arquitectura **Model-2** se encuentra en línea con la idea propuesta por **MVC**, donde las páginas JSP son utilizadas únicamente para aspectos de presentación, los servlets para el procesamiento de pedidos, instanciando JavaBeans que son enviados hacia las páginas JSP