JSTL

CONCEPTOS DEL PATRÓN DE DISEÑO MVC

- Los Servlets están enfocados en controlar el flujo de la petición HTTP.
- Los JSP's están enfocados en desplegar la información de la aplicación Web.
- La información que se comparte entre los componentes (Servlets y JSP's) suele manejarse con JavaBeans.
- El Patrón de Diseño MVC (Modelo Vista Controlador) nos permite integrar a los JSP's (Vista), a los Servlets (Controlador) y a los JavaBeans (Modelo)

En esta lección vamos a revisar los conceptos del patrón de diseño en Modelo-Vista-Controlador, también conocido como MVC.

Un patrón de diseño permiten solucionar problemas comunes que se presentan al momento de crear aplicaciones, y en particular en aplicaciones Web nos interesa separar la vista de los datos (modelo) y unirlos por medio de un componente que hace la vez de controlador.

Según hemos estudiado hasta el momento los Servlets están enfocados en controlar el flujo de la aplicación y en este caso procesan las peticiones HTTP, así como utilizar los JavaBean para almacenar información y finalmente redireccionar al JSP respectivo.

Los JSP’s están enfocados en desplegar la información de la aplicación Web, en este caso la información es proveída por medio de los Servlets y la información que se comparte entre esos componentes, es decir, entre los Servlets y JSPs suele manejarse con JavaBeans.

En resumen el patrón de diseño MVC modelo-vista-controlador nos permite integrar a los JSPs, Servlets y a los JavaBean en un solo modelo para poder interactuar y así crear aplicaciones web cada vez más robustas y más fáciles de mantener.

FRAMEWORKS QUE UTILIZAN EL PATRÓN MVC

- **JSP/Servlets**: Se implementa manualmente con ayuda del objeto RequestDispatcher para controlar el flujo de la aplicación

- **Struts**: Es un framework de Apache, el cual utiliza JSPs(Vista) con tags de Struts, ActionForm (Modelo), Action (Controlador), entre otros componentes.

- **JavaServer Faces (JSF):** Es una tecnología de Sun Microsystems, que utiliza conceptos como JSPs (Vista) con tags de JSF, ManagedBean (Controlador) y JavaBeans (Modelo)

- **SpringMVC**: Es una extensión del framework de Spring, que utiliza JSP (Vista) con tags de Spring, Clases Java (Controladores) y JavaBeans (Modelo)

- Otros…

El patrón de diseño MVC lo podemos implementar de manera manual utilizando JSPs y Servlets y con la ayuda del objeto RequestDispatcher vamos a poder controlar el flujo de nuestra aplicación.

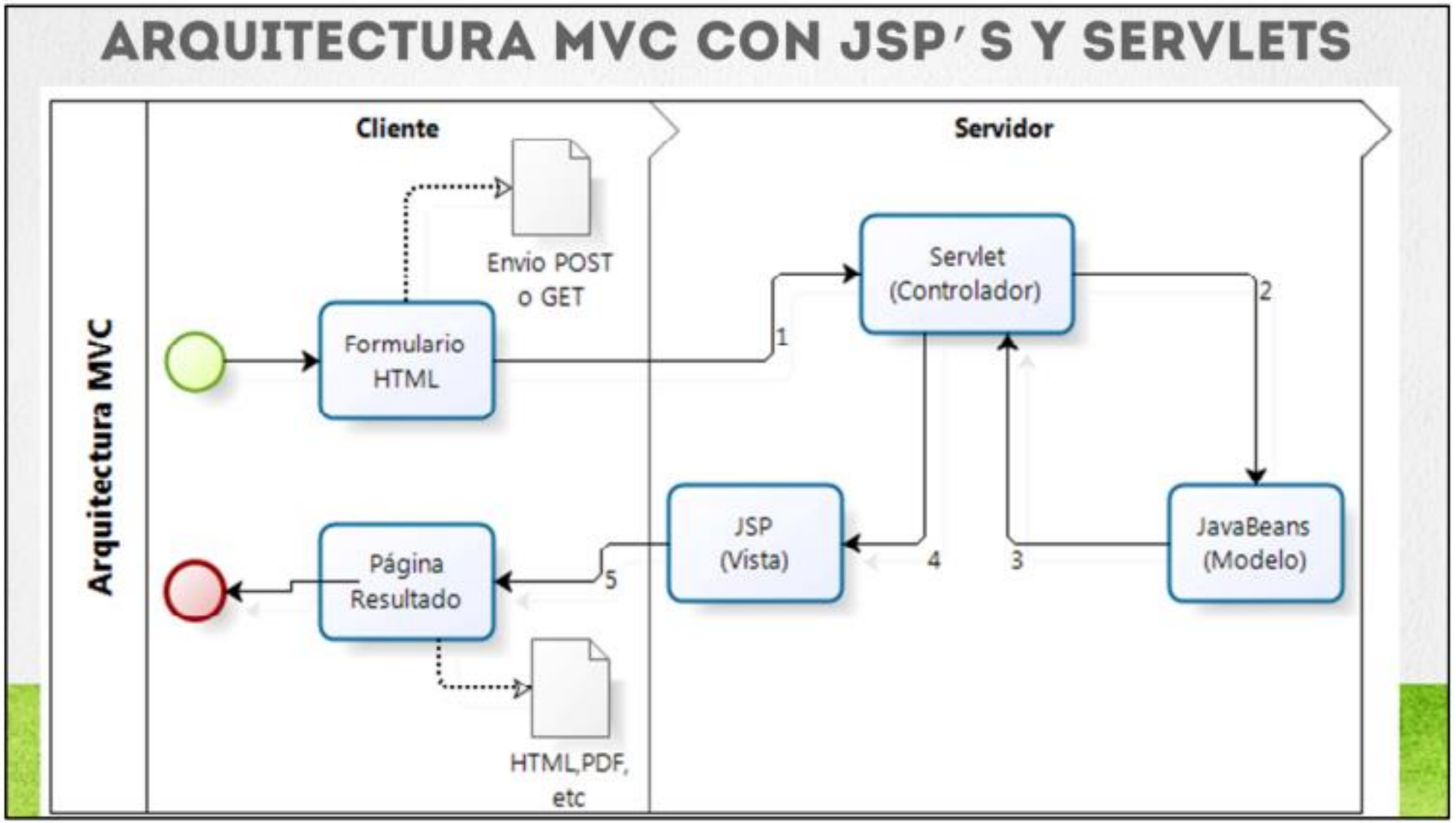
Existe varios Frameworks que implementan ya este patrón, un patrón de diseño es simplemente una guía, por lo tanto cada uno de estos Frameworks tanto Struts, JavaServer Faces, Spring MVC, entre otros, únicamente siguen esta guía o recomendaciones pero realmente no es una especificación que nos indique paso a paso como implementar este patrón, sino simplemente es una serie de pasos genéricos con los cuales cada uno de estos Frameworks implementa según las mejores prácticas desde el punto de vista de cada uno de estos Frameworks.

Por ejemplo en el caso del Framework de Struts es un framework de Apache, el cual utiliza JSPs como la Vista utilizando también a su vez tags de Struts, posteriormente utiliza el concepto llamado ActionForm que de alguna manera sustituye a los JavaBeans, siendo el modelo de nuestra aplicación y finalmente tenemos el concepto de Action, el cual cubre el rol del controlador. Estos son simplemente algunos componentes de los que se manejan dentro del Frameworks de Struts.

JavaServer Faces es una tecnología definida por Sun Microsystems, en el cual se utilizan conceptos como son los mismos JSPs pero utilizando tags de JSF, también vamos a utilizar el concepto de ManagedBean los cuales cubren el rol del controlador y finalmente tenemos el concepto de JavaBeans para manejar el concepto de modelo aunque también cabe resaltar que los ManagedBean pueden jugar el rol tanto de controlador como de modelo, entonces todo podría mezclarse en un solo bean y se podría omitir el uso de los JavaBeans.

Por último mencionaremos el framework de Spring MVC. Este framework es una extensión de Spring, en el cual se utilizan JSPs como parte la vista y se pueden utilizar los tags de Spring para robustecer estos JSPs. Posteriormente se utilizan clases de Java, también conocidas como POJOs (Plain Old Java Objects) o clases puras de Java y se utilizan como controladores y también se utilizan JavaBeans como parte del modelo.

Existen muchos frameworks más que implementan el patrón MVC utilizando el lenguaje Java, esto son tan solo algunos de los más representativos y como hemos comentado únicamente el patrón de diseño es una guía general y cada framework define la especificación según las mejores prácticas desde el punto de vista de cada framework.


ARQUITECTURA MVC CON JSP´S Y SERVLETS
Arquitectura MVC
Cliente
Servidor
Envio POST
o GET
Formulario
HTML
Servlet
(Controlador)
1
2
JSP
(Vista)
JavaBeans
(Modelo)
4
3
Página
Resultado
5
HTML,PDF,
etc

Como podemos observar, el flujo inicia con un formulario HTML, esta información está almacenada en nuestro cliente y del lado derecho vamos a tener el servidor. Una vez que nuestro cliente envía la petición del formulario hacia el servidor web, quien va a procesar esta petición es un Servlet (controlador) y a pesar que en ejercicios anteriores hemos colocado a un JSP para manejar directamente formularios, según hemos comentado, esta no es buena práctica. Por lo tanto, las peticiones las debe de procesar un Servlet controlador, incluso podemos observar los números con los cuales vamos a ir siguiendo nuestro flujo (1 al 5).

Una vez que el Servlet controlador ha recibido la petición, una de sus tareas puede ser procesar los parámetros del formulario HTML si es que aplica y una vez que ya tenemos procesado los parámetros del formulario, normalmente lo que hacemos es apoyarnos de JavaBeans para almacenar o procesar la información de lógica de negocio o lógica de presentación de nuestra aplicación Web.

Ya que hemos creado y almacenado la información en nuestros JavaBeans, regresamos el control al Servlet y este Servlet controlador puede colocar estos JavaBean en algún alcance para compartir información hacia un JSP. Los alcances pueden ser request, session o application. Los Servlets no conocen el alcance de page, ya que eso pertenece sólo a los JSPs, por ello únicamente podemos manejar los 3 alcances mencionados.

Una vez que el Servlet Controlador ya ha colocado los JavaBeans en algún alcance, hace un redireccionamiento por medio del objeto RequestDispatcher que vamos a ver más adelante. En este punto también cabe mencionar que el Servlet Controlador toma la decisión de cual JSP se va a utilizar, por ejemplo podríamos tener un JSP1 un JSP2 un JSP3 etc. todo va a depender del flujo de la aplicación Web, pero en este caso el Servlet Controlador es quien va a decidir cuál es Vista o JSP es el que se va a utilizar.

Finalmente, una vez que ya estamos dentro del JSP seleccionado, lo que va a hacer el JSP es jugar el rol de la Vista, esto implica que únicamente va a mostrar la información que le compartió el Servlet. Los JSP's en teoría no deberían de crear nuevos objetos Java, para ello toda la información que va a utilizar el JSP ya debió de haber sido proporcionada por el Servlet controlador.

Una vez que el JSP genera el HTML utilizando la información de los JavaBeans que el Servlets le proporcionó, lo que hace es regresar el contenido al cliente y en este momento es cuando se genera el Render de nuestra aplicación según el Content Type que hayamos utilizado. Por ejemplo, puede ser una salida en HTML, PDF, Video, un archivo de Excel, etc. según hemos visto anteriormente.

El punto es que el JSP únicamente va a desplegar la información que recibió del Servlet y enviará esta información al cliente. Con esto termina el flujo y si el cliente necesitara de realizar una nueva petición el proceso se repite nuevamente.

PASOS GENERALES DE UN SERVLET CONTROLADOR

a) Procesamos y validamos los parámetros (si aplica)

```
request.getParameter("nombreParametro");
```

b) Realizamos la lógica de presentación almacenando el resultado en JavaBeans

```
Rectangulo rec = new Rectangulo();
```

c) Compartimos el objeto bean a utilizar en algún alcance (scope)

```
request.setAttribute("rectanguloBean", rec );
```

d) Redireccionamos al JSP seleccionado

```
RequestDispatcher dispatcher = request.getRequestDispatcher("resultado.jsp");
dispatcher.forward( request, response );
```

Según revisamos en la teoría de los Servlets, para procesar un parámetro podemos utilizar la siguiente notación: **request.getParameter("nombreParametro")**; podemos hacer uso del objeto request y del método getParameter indicando el nombre del parámetro que queremos procesar.

Posteriormente, podemos validar los parámetros para saber si la información que estamos recibiendo es correcta. Una vez que ya hemos procesado los parámetros podemos realizar la lógica de presentación respectiva o la lógica de negocio utilizando JavaBeans. El resultado de procesar esta información la vamos a almacenar en objetos de tipo JavaBean, en este caso simplemente estamos creando un nuevo objeto de tipo Rectangulo Rectangulo rec = new Rectangulo(). Únicamente estamos creando un nuevo JavaBean para tratar de ejemplificar que estos JavaBeans van a tener la información de nuestra aplicación Web.

Posteriormente, antes de redireccionar y seleccionar el JSP que vamos a utilizar, debemos de compartir el objeto que estamos creando en algún alcance, en este caso estamos seleccionado el alcance de request, pero podemos usar los alcances de session o el de servlet context. En este caso usaremos el método setAttribute del objeto request e indicamos el nombre del bean que se va a recuperar por parte del JSP.

Finalmente seleccionaremos el JSP que desplegará la información al cliente, por medio del código:

RequestDispatcher dispatcher = request.getRequestDispatcher("resultado.jsp"); lo que hacemos es seleccionar el JSP apoyándonos del objeto request y del método getRequestDispatcher, aquí lo que vamos hacer es proporcionar el JSP que estamos seleccionando como la Vista que va a desplegar la información que estamos compartiendo por medio del objeto request y una vez que hemos seleccionado el JSP que vamos a utilizar, lo asignamos a un objeto llamado RequestDispatcher. Este es el objeto que hemos venido comentando y que nos va a permitir redireccionar hacia el JSP respectivo.

Finalmente usamos el método **dispatcher.forward(request, response);** lo que hacemos es utilizar el objeto dispatcher para ejecutar el método forward. Aquí lo importante es que podemos observar que tenemos 2 argumentos, el argumento de request y el argumento de response. Si recordamos, una vez que estamos procesando un método de un Servlet Controlador, por ejemplo si tenemos el método doGet o el método doPost, estamos recibiendo como argumentos el objeto request y el objeto response, en pocas palabras lo que estamos haciendo aquí es únicamente proporcionar los mismos objetos que ya hemos recibido en este Servlet, con la intención de que la información que hemos compartido en nuestros objetos de request, session o application se sigan compartiendo con los demás elementos que vamos a seguir utilizando. En este caso, por medio del método forward estamos proporcionando y enviando toda la información necesaria al JSP para que no tenga ningún problema y pueda acceder a la información que hemos compartido previamente por medio del Servlet. A continuación vamos hacer un ejercicio para poner en práctica el concepto del patrón de diseño MVC.