MODELO VISTA CONTROLADOR

La arquitectura MVC (Modelo Vista Controlador) filosofía para el diseño de aplicaciones es un patrón que define la organización independiente del Modelo (Objetos), la vista (Interfaz) y el Controlador (controlador del workflow de la aplicación).

De esta forma estamos dividiendo el sistema en tres capas (Modelo, Vista, Controlador) a continuación se especifica lo que contiene cada una:

Modelo – Contiene el núcleo de la funcionalidad de la aplicación encapsulando el estado de la aplicación, si estamos hablando de una aplicación escrita en JavaFX la parte del modelo contiene las clases del modelado y un objeto DAO (Data Access Object – Objeto de Acceso a Datos), como su nombre nos dice será el que nos da acceso al almacenamiento de datos, tales como una Base de Datos.

Vista – Como tal es la presentación del Modelo (Interfaz de Usuario) pudiendo de esta manera acceder a el modelo, pero nunca cambiar su estado. En JavaFX las clases de Vista son todos los archivos con extensión FXML.

Controlador – La parte de Controlador reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo correspondiente. En JavaFX las clases Java controlaran el funcionamiento fusionando el Modelo y Vista.

Utilizando la arquitectura MVC para el diseño de aplicaciones tiene bastantes ventajas una vez entendiendo como aplicarla, entre algunas de sus ventajas están:

Identificación de errores es más fácil y exacta.

Facilidad para crear distintas representaciones de los mismos datos.

Facilidad para la realización de pruebas unitarias de los diferentes componentes de la aplicación.

Reutilización de los componentes.

Agiliza el proceso para poder realizar prototipos rápidos.

Son muchas las ventajas de la arquitectura Modelo Vista Controlador como principal desventaja es que así como con cualquier nueva tecnología, lenguaje, entorno etc. es la adaptación para trabajar bajo la misma. Una vez adaptando la arquitectura se podrá ver todas esas ventajas que nos ofrece MVC.

El MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación. Es una arquitectura importante puesto que se utiliza tanto en componentes gráficos básicos hasta sistemas empresariales; la mayoría de los frameworks modernos utilizan MVC (o alguna adaptación del MVC) para la arquitectura, entre ellos podemos mencionar a Ruby on Rails, Django, AngularJS y muchos otros más.

Antes que nada, me gustaría mencionar por qué se utiliza el **MVC**, la razón es que nos permite separar los componentes de nuestra aplicación dependiendo de la responsabilidad que tienen, esto significa que cuando hacemos un cambio en alguna parte de nuestro código, esto no afecte otra parte del mismo. Por ejemplo, si modificamos nuestra Base de Datos, sólo deberíamos modificar el modelo que es **quién se encarga de los datos** y el resto de la aplicación debería permanecer intacta. Esto respeta el principio de la responsabilidad única. Es decir, una parte de tu código no debe de saber qué es lo que hace toda la aplicación, sólo debe de tener una responsabilidad.

En web, el MVC funcionaría así. Cuando el usuario manda una petición al navegador, digamos quiere ver el curso de AngularJS, el controlador responde a la solicitud, porque él es el que controla la lógica de la app, una vez que el controlador nota que el usuario solicitó el curso de Angular, le pide al modelo la información del curso.

El modelo, que se encarga de los datos de la app, consulta la base de datos y digamos, obtiene todos los vídeos del curso de AngularJS, la información del curso y el título, el modelo responde al controlador con los datos que pidió (nota como en la imagen las flechas van en ambos sentidos, porque el controlador pide datos, y el modelo responde con los datos solicitados).

Una vez el controlador tiene los datos del curso de AngularJS, se los manda a la vista, la vista aplica los estilos, organiza la información y construye la página que vez en el navegador.

Modelo

Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.

Controlador

Se encarga de… controlar, recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.

Vistas

Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.

La correspondencia entre cada parte M, V, y C y cada uno de los elementos de la aplicación es la siguiente:

Modelo: Es una representación del modelo (lo escribo en negrita para distinguirlo con claridad) de la aplicación entendido este último como todo el conjunto de datos y toda la lógica de negocio que procesa esos datos. En JSF el Modelo se implementa en las clases managed bean, también conocidos como backing beans. El modelo tiene una parte de datos y otra de comportamiento. Veamos cada una por separado:

Parte de datos del modelo: Son los beans de entidad JPA de la capa de persistencia. En el Modelo se representan como atributos de los managed beans con el mismo tipo que los entity beans JPA.

Parte de comportamiento del modelo: Son los beans de sesión EJB de la capa de negocio/DAO/servicio. En el Modelo se representan dentro de los métodos de evento de los managed beans, como llamadas a los métodos de estos beans de sesión.