Patrón MVC

Autor: José Guillermo Sandoval Huerta

Fecha: 9 noviembre 2023

Índice

¿Qué es el patrón de diseño MVC?

MVC por sus siglas Modelo-Vista-Controlador, es un patrón arquitectónico comúnmente utilizado en el desarrollo de software para organizar el código de una aplicación. Este se originó en la década de 1970 en el laboratorio de investigación de Xerox PARC (Palo Alto Research Center) como parte del desarrollo del sistema Smalltalk-80, un entorno de programación orientado a objetos. El equipo de desarrollo, liderado por Trygve Reenskaug, propuso y aplicó el patrón MVC para abordar la complejidad de las interfaces de usuario en sistemas interactivos. El objetivo original del patrón MVC era proporcionar una arquitectura que separara las preocupaciones y permitiera una mayor modularidad y flexibilidad en el desarrollo de software, permitiendo separar la lógica de presentación, la lógica de negocio y la gestión de datos.

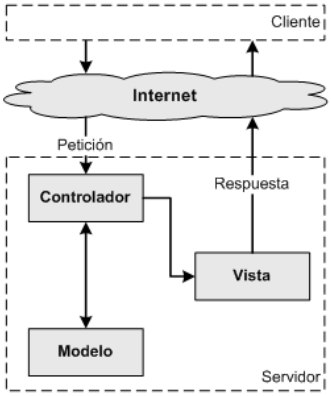

Imagen 1. Patrón MVC.

Modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe reflejar. Es la parte encargada de representar la lógica de negocio de una aplicación. Las clases en el modelo son responsables de acceder a la base de datos, procesar datos y contener la lógica empresarial. Pueden ser representadas como clases POJO que encapsulan la lógica de negocio y los datos. El modelo, a nivel teórico, no debe tener conocimiento acerca de la existencia de las vistas y del controlador.

Vista

Representa la interfaz de usuario y la presentación de datos. En Java, la vista puede ser implementada con interfaces gráficas de usuario (GUI) utilizando bibliotecas como Swing o JavaFX. La vista no debe contener lógica de negocio, sino que simplemente muestra los datos proporcionados por el controlador y envía las interacciones del usuario al controlador.

Controlador

Actúa como intermediario entre el modelo y la vista. Responde a las interacciones del usuario y actualiza tanto el modelo como la vista según sea necesario. En Java, el controlador puede ser implementado como una clase que maneja eventos y coordina las interacciones entre el modelo y la vista.

MVC en Java

El lenguaje de programación Java proporciona soporte para la arquitectura MVC. Provee de dos clases que son las encargadas de realizar las notificaciones de cambios en los estados de los objetos. Se definen a continuación los objetos:

- **Observer**: Es cualquier objeto que desee ser notificado cuando el estado de otro objeto sea alterado.

- **Observable**: Es cualquier objeto cuyo estado puede representar interés y sobre el cual otro objeto ha demostrado ese interés.

Estas dos clases no sólo se utilizan en la aplicación del patrón MVC, tienen una utilidad mayor dentro del lenguaje. Serán útiles en cualquier sistema en el que se necesite que algunos objetos sean notificados cuando ocurran cambios en otros objetos. El Modelo es un subtipo de Observable y la Vista es un subtipo de Observer. Estas dos clases manejan adecuadamente la función de notificación de cambios que necesita la arquitectura MVC. Proporcionan el mecanismo por el cual las Vistas pueden ser notificadas automáticamente de los cambios producidos en el Modelo.

¿Cómo funciona una aplicación MVC?

1. **Petición al controlador**
   La aplicación recibe peticiones que son centralizadas en el Controlador. Éste es el encargado de interpretar, a partir de la URL de la solicitud, el tipo de operación que hay que realizar. Normalmente, esto se hace analizando el valor de algún parámetro que se envía anexando a la URL de la petición y que se utiliza con esta finalidad.

2. **Procesamiento de la petición**
   Una vez que el Controlador determine la operación a realizar, procede a ejecutar las acciones pertinentes, invocando para ello a los diferentes métodos expuestos por el Modelo. Dependiendo de las acciones a realizar (por ejemplo, un alta de un usuario en el sistema), el Modelo necesitará manejar los datos enviados por el cliente en la petición, datos que le serán proporcionados por el controlador. Para facilitar este intercambio de datos entre el Controlador y Modelo y, posteriormente, entre Controlador y Vista, las aplicaciones MVC suelen hacer uso de JavaBeans. Un JavaBean no es más que una clase que encapsula un conjunto de datos con métodos de tipo set/get para proporcionar un acceso a los mismos desde el exterior.

3. **Generación de respuesta**
   Los resultados devueltos por el Modelo al Controlador son depositados por éste en una variable de petición, sesión o aplicación, según el alcance que deban tener. A continuación, el Controlador invoca a la página JSP que debe encargarse de generar la vista correspondiente, está página accederá a la variable de ámbito donde estén depositados los resultados y los utilizará para generar dinámicamente la respuesta XHTML que será enviada al cliente.

Ventajas y retos del uso del patrón MVC

Ventajas	Retos
La implementación se realiza de forma modular.	Para desarrollar una aplicación bajo el patrón de diseño MVC es necesario una mayor dedicación en los tiempos iniciales del desarrollo. Sin embargo, este reto es muy relativo ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es mucho más modificable que una aplicación que no lo implementa.
Sus vistas muestran información actualizada siempre. El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.	MVC requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos, un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.
Cualquier modificación que afecte al dominio, como aumentar métodos o datos contenidos, implica una modificación sólo en el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.	MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.
Las modificaciones a las vistas no afectan al modelo de dominio, simplemente se modifica la representación de la información, no su tratamiento.	
MVC está demostrando ser un patrón de diseño bien elaborado pues las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.	

Tabla 1. Ventajas y retos del patrón MVC.

Ejemplo de implementación en Java

Código 1. Implementación del Modelo.

```
public class Usuario {
    private String nombre;
    private String correo;

    // Métodos getter y setter
}
```

Código 2. Implementación de la Vista.

```
public class VistaUsuario {
    // Métodos para mostrar y actualizar la interfaz de usuario
}
```

Código 3. Implementación del Controlador.

```
public class ControladorUsuario {
    private Usuario modelo;
    private VistaUsuario vista;

    // Métodos para manejar eventos del usuario y actualizar el modelo y la vista
}
```

Código 4. Clase Principal.

```
public class Principal {
    public static void main(String[] args) {
        Usuario modelo = new Usuario();
        VistaUsuario vista = new VistaUsuario();
        ControladorUsuario controlador = new ControladorUsuario(modelo, vista);

        vista.setControlador(controlador);
        // Lógica adicional para inicializar la aplicación
    }
}
```

Referencias

[1] J. M. Aguilar, «¿Qué es el patrón MVC en programación y por qué es útil?,» CampusMVP, 15 Octubre 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx. [Último acceso: 9 Noviembre 2023].