Patrón Arquitectura De Software Model-View-Controller

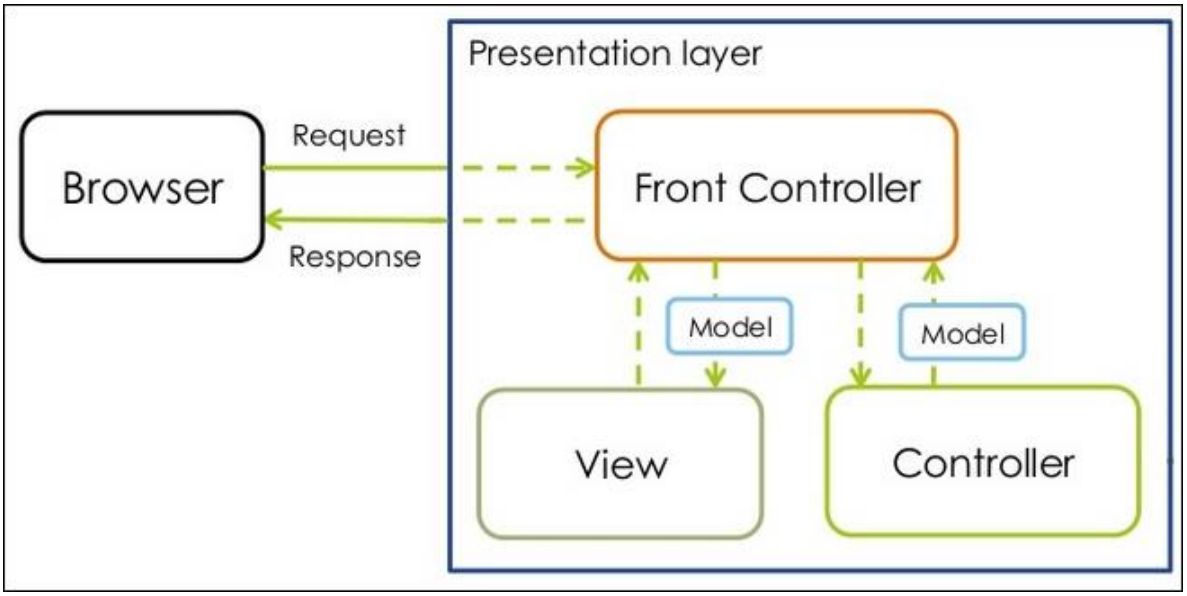
Ing. Ronaldo Tovar Reyes **10-05-2024**

Introducción

El patrón de arquitectura de software MVC es ampliamente utilizado en el desarrollo de aplicaciones, principalmente en el desarrollo de interfaces de usuario. Su objetivo principal es delegar responsabilidades separando la lógica del programan en tres componentes interconectados el Modelo, la Vista y el Controlador. Esto promueve una mejor organización de código, facilitando el mantenimiento y reutilización, mejora también, la capacidad de pruebas de la aplicación.

Este patrón es una estructura muy popular en el desarrollo de software, especialmente en la creación de aplicaciones web y en la programación orientada a objetos. Este patrón es compatible con una amplia gama de lenguajes de programación, incluyendo Java, Python, JavaScript, C#, Swift, Perl y PHP.

Model-View-Controller (MVC)



Controlador: es el que recibe las peticiones del usuario y su rol es delegar quien se hace cargo de realizar cada petición, es el punto de acceso y manipula al **Modelo,** es la lógica de negocio que incluye el tratamiento de los datos, la información, que a su vez actualiza la **Vista** y esta se encarga de la lógica de presentación que se le muestra al **Usuario**

Spring MVC

Patrón Arquitectura De Software Model-View-Controller

Ing. Ronaldo Tovar Reyes **10-05-2024**

Spring MVC se necesita un contenedor web para poder ejecutarse por ejemplo, **Tomcat, Java EE**

Para definir una aplicación se realiza la arquitectura por capas y para poder ejecutar**,** que son

1. Capa de nivel de base de datos: PostgreSQL, MySQL, Oracle, SQLServer
2. Capa de persistencia: JDBC, JPA (Java Persistence API), Hibernate. Esta capa es responsable de la comunicación con la base de datos, ya que en spring es java y en base de datos es lenguaje SQL.
3. Capa de negocio: Se define la lógica de negocio, es decir, todos los calculos o tareas que se tienen que realizar.
4. Capa de presentación: Aquí se define el **Model-View-Controller:**
    a. **Front Controller:** Despacher es propio de Spring, nunca lo vamos a modificar, maneja todas las solicitudes HTTP entrantes y delega la responsabilidad a los componentes adecuados para procesarlos. Su papel principal es desacoplar la lógica de solicitud específica del flujo de trabajo real de la solicitud misma, permitiendo una arquitectura más modular y flexible, también facilita el mantenimiento y la extensión de la aplicación.
    b. **View:** se define la definición, creación y actualización de la vista con Java, aquí nunca se ejecuta la vista por que Tomcat no sabe interpretar JavaScript, CSS o HTML.
    c. **Controller:** actua como intermediario en el **Modelo y la vista** procesando las solicitudes del usuario, interactuando con el **Modelo** para obtener o actualizar los datos necesarios y luego enviando los resultados a la vista decuada para su presentación al usuario.
    d. **Browser:** recibe los archivos JSON, o XML del Front Controller mediante HTTP y ejecuta la vista al usuario