Bootcamp de Desarrollo Web

Clase 12

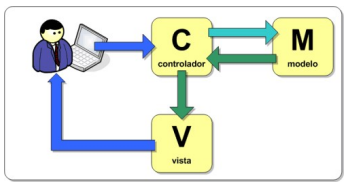
Patrón de diseño MVC

1

Qué es el patrón de diseño MVC

El patrón de diseño MVC (Modelo-Vista-Controlador) es una arquitectura de software que organiza la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador.

Cada uno de estos componentes cumple un propósito específico y se encarga de diferentes aspectos de la aplicación.



2

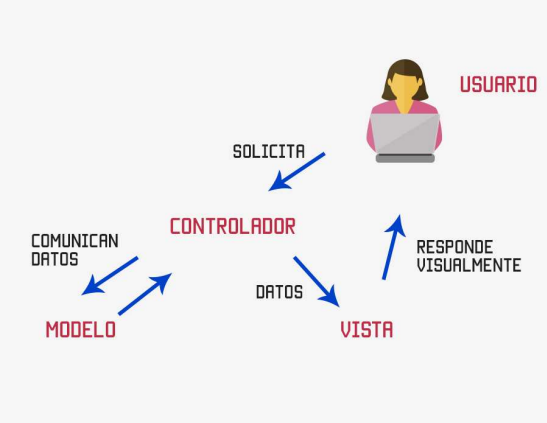
Qué es el patrón de diseño MVC

Modelo (Model): Representa los datos y la lógica de negocio de la aplicación. Es responsable de gestionar el estado de la información y realizar operaciones sobre ella.

Vista (View): Es la interfaz de usuario que presenta los datos al usuario y recibe las interacciones del mismo. Se encarga de la presentación de la información de manera visual.

Controlador (Controller): Actúa como intermediario entre el Modelo y la Vista. Gestiona las solicitudes del usuario, actualiza el Modelo según sea necesario y actualiza la Vista para reflejar los cambios.

Qué es el patrón de diseño MVC



Historia del patrón MVC

El patrón MVC fue propuesto por primera vez por **Trygve Reenskaug** en 1979 en un artículo titulado **Applications Programming in Smalltalk-80**: **How to use Model-View-Controller** (Programación de aplicaciones en Smalltalk-80: Cómo usar Modelo-Vista-Controlador). Fue originalmente diseñado para su uso en interfaces de usuario en el lenguaje de programación Smalltalk-80.

Desde entonces, el patrón MVC ha ganado popularidad y ha sido adoptado en una variedad de tecnologías y plataformas, convirtiéndose en uno de los patrones de diseño más ampliamente utilizados en el desarrollo de software.

Por qué utilizar el patrón MVC

El patrón MVC ofrece varios beneficios:

Separación de preocupaciones: Permite dividir la aplicación en componentes que se ocupan de aspectos específicos, lo que facilita la gestión y la mantenibilidad del código.

Reutilización de código: Al separar la lógica de presentación de los datos, se pueden reutilizar los componentes en diferentes partes de la aplicación o en diferentes aplicaciones.

Por qué utilizar el patrón MVC

Facilita la colaboración: Permite que diferentes equipos trabajen en paralelo en el desarrollo de la aplicación, ya que cada componente tiene su responsabilidad definida.

Mejora la escalabilidad: Al dividir la aplicación en componentes independientes, es más fácil escalar la aplicación para manejar un mayor volumen de usuarios o datos.

Por qué utilizar el patrón MVC

Facilita la colaboración: Permite que diferentes equipos trabajen en paralelo en el desarrollo de la aplicación, ya que cada componente tiene su responsabilidad definida.

Mejora la escalabilidad: Al dividir la aplicación en componentes independientes, es más fácil escalar la aplicación para manejar un mayor volumen de usuarios o datos.

Diferentes implementaciones del patrón MVC

A lo largo de los años, han surgido diversas implementaciones y variantes del patrón MVC, adaptadas a diferentes tecnologías y plataformas. Algunas de estas implementaciones incluyen:

MVC clásico: La implementación original propuesta por Reenskaug en Smalltalk-80.

MVC basado en web: Adaptaciones del patrón MVC para el desarrollo de aplicaciones web, donde el Modelo puede representar los datos de la aplicación, la Vista puede ser una página HTML y el Controlador puede ser un script que maneje las solicitudes del usuario.

Diferentes implementaciones del patrón MVC

MVC en frameworks modernos: Muchos frameworks modernos de desarrollo web y de aplicaciones adoptan el patrón MVC, como Ruby on Rails, Django, Laravel, entre otros.

Estos frameworks proporcionan herramientas y estructuras para facilitar la implementación del patrón MVC en las aplicaciones.

Bootcamp de Desarrollo Web

Clase 12

Roles y responsabilidades de cada componente en el patrón MVC

Modelo (Model)

Qué es: El Modelo representa los datos y la lógica de negocio de la aplicación.

Responsabilidades:

- Gestionar el estado de la información y los datos.
- Realizar operaciones sobre los datos, como leer, escribir, actualizar y eliminar.
- Implementar la lógica de negocio de la aplicación, como cálculos, validaciones y reglas de negocio.
- Notificar a otras partes de la aplicación sobre cambios en los datos.

Vista (View)

Qué es: La Vista es la interfaz de usuario que presenta los datos al usuario.

Responsabilidades:

- Mostrar los datos de manera visual para que el usuario pueda interactuar con ellos.
- Presentar la información de forma clara y comprensible.
- Recibir las interacciones del usuario, como clics, entradas de teclado o gestos táctiles.
- No debe contener lógica de negocio ni acceder directamente a la capa de datos.

Controlador (Controller)

Qué es: El Controlador actúa como intermediario entre el Modelo y la Vista.

Responsabilidades:

- Interpretar las interacciones del usuario recibidas desde la Vista.
- Actualizar el Modelo según las acciones del usuario.
- Actualizar la Vista para reflejar los cambios en el Modelo.
- Encargarse del flujo de control de la aplicación y coordinar las acciones entre el Modelo y la Vista.
- No debe contener lógica de negocio, esta debe residir en el Modelo.

Relación entre los componentes

- El Modelo es independiente de la Vista y el Controlador. Se encarga de la gestión de datos y lógica de negocio.
- La Vista muestra los datos proporcionados por el Modelo y recibe las interacciones del usuario.
- El Controlador interpreta las acciones del usuario, actualiza el Modelo según sea necesario y actualiza la Vista para reflejar los cambios.

Bootcamp de Desarrollo Web

Clase 12

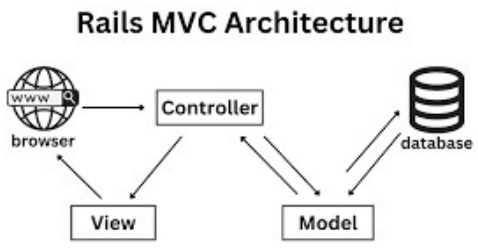
Implementación de MVC en diferentes frameworks

Implementación de MVC en diferentes frameworks

Ruby on Rails (RoR):

- **RoR** es un framework de desarrollo web basado en el lenguaje de programación Ruby que sigue el patrón MVC de manera estricta.
- El Modelo en RoR se representa mediante clases ActiveRecord que interactúan con la base de datos.
- La Vista se implementa utilizando plantillas (por ejemplo, archivos ERB) que generan HTML dinámico.
- El Controlador es representado por clases que gestionan las solicitudes HTTP, interactúan con el Modelo y renderizan las Vistas correspondientes.

Implementación de MVC en diferentes frameworks



Implementación de MVC en diferentes frameworks

Django (Python):

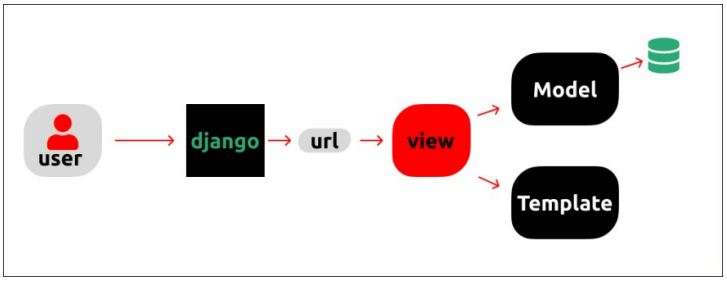
- Django es un framework web de alto nivel para el desarrollo rápido de aplicaciones web, escrito en Python.
- En Django, el Modelo está definido mediante clases que representan los modelos de datos y se relacionan con la base de datos.
- La Vista se implementa utilizando plantillas Django (archivos HTML con capacidades de lenguaje de plantillas).
- El Controlador se maneja a través de las vistas basadas en funciones o clases, que reciben solicitudes HTTP y devuelven respuestas, interactuando con el Modelo según sea necesario.

Implementación de MVC(MVT) en diferentes frameworks

Django (Python):

- Django es un framework web de alto nivel para el desarrollo rápido de aplicaciones web, escrito en Python.
- En Django, el Modelo está definido mediante clases que representan los modelos de datos y se relacionan con la base de datos.
- La Vista se implementa utilizando plantillas Django (archivos HTML con capacidades de lenguaje de plantillas).
- El Controlador se maneja a través de las vistas basadas en funciones o clases, que reciben solicitudes HTTP y devuelven respuestas, interactuando con el Modelo según sea necesario.

Implementación de MVC(MVT) en diferentes frameworks

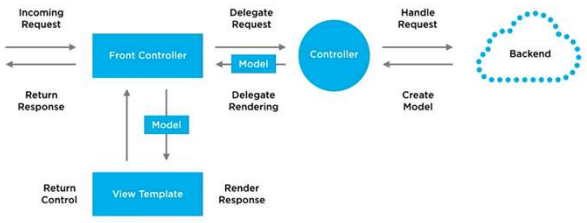


Implementación de MVC(MVT) en diferentes frameworks

Spring (Java):

- Spring es un framework de aplicación Java que ofrece soporte para el desarrollo de aplicaciones empresariales.
- En Spring MVC, el Modelo se representa mediante clases Java que encapsulan la lógica de negocio y se comunican con la capa de persistencia de datos.
- La Vista se implementa mediante tecnologías como JSP (JavaServer Pages) o Thymeleaf, que generan la interfaz de usuario dinámica.
- El Controlador es manejado por clases anotadas con @Controller, que gestionan las solicitudes HTTP, invocan métodos del Modelo y devuelven las Vistas correspondientes.

Implementación de MVC(MVT) en diferentes frameworks



Bootcamp de Desarrollo Web

Clase 12

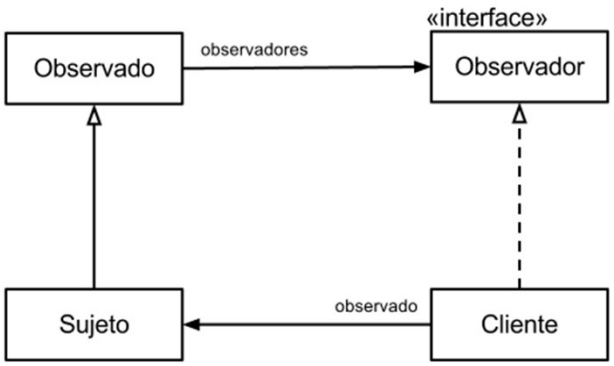
Patrones de diseño adicionales relacionados con MVC

Patrones de diseño adicionales relacionados con MVC

Patrón Observer:

- El patrón Observer se utiliza para establecer una relación de uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizados automáticamente.
- En el contexto de MVC, el patrón Observer puede utilizarse para mantener la sincronización entre el Modelo y la Vista.
- Por ejemplo, cuando el Modelo cambia de estado, puede notificar a la Vista (o a un Observador asociado a la Vista) para que actualice su presentación en consecuencia.

Patrones de diseño adicionales relacionados con MVC



Patrones de diseño adicionales relacionados con MVC

Patrón Factory:

- El patrón Factory se utiliza para encapsular la creación de objetos y ocultar los detalles de su implementación.
- En el contexto de MVC, el patrón Factory puede utilizarse para crear instancias concretas de objetos Modelo, Vista o Controlador.
- Esto promueve la flexibilidad y el mantenimiento del código al permitir que las instancias se creen de manera dinámica en función de las necesidades de la aplicación.

Patrones de diseño adicionales relacionados con MVC

Patrones de diseño adicionales relacionados con MVC

Singleton:

Una sola instancia: El patrón Singleton garantiza que una clase tenga solo una instancia en toda la ejecución del programa.

Acceso global: Proporciona un método para acceder a la instancia única de la clase desde cualquier parte del código.

Control sobre la creación: El patrón Singleton permite un control total sobre cómo y cuándo se crea la instancia única de la clase.

Bootcamp de Desarrollo Web

Clase 12

Diseño de la arquitectura de la aplicación web utilizando MVC

Diseño de la arquitectura de la aplicación web utilizando MVC

Diseñar la arquitectura de una aplicación web utilizando el patrón Modelo-Vista-Controlador (MVC) implica dividir la aplicación en tres componentes principales y definir cómo interactúan entre sí.

Aquí hay un enfoque paso a paso para diseñar esta arquitectura:

1- Identificación de requisitos y funcionalidades:

Comienza por comprender los requisitos y las funcionalidades de tu aplicación web. ¿Qué características debe tener la aplicación? ¿Cuáles son las necesidades del usuario final?

Diseño de la arquitectura de la aplicación web utilizando MVC

2- Definición de los modelos de datos:

Identifica los tipos de datos que la aplicación manejará y cómo se relacionan entre sí. Define las clases de Modelo que representarán estos datos y la lógica de negocio asociada.

3- Diseño de las vistas:

Decide cómo se presentarán los datos al usuario final. Diseña las interfaces de usuario (UI) que mostrarán la información y permitirán la interacción del usuario.

Diseño de la arquitectura de la aplicación web utilizando MVC

4- Planificación de los controladores:

Define qué acciones podrán realizar los usuarios en la aplicación y cómo se manejarán estas acciones. Diseña los controladores que actuarán como intermediarios entre las vistas y los modelos.

5- Establecimiento de las rutas y URLs:

Define las rutas y URLs que estarán disponibles en la aplicación. Estas rutas deben mapearse a las acciones específicas que serán manejadas por los controladores.

Diseño de la arquitectura de la aplicación web utilizando MVC

6- Implementación de la lógica de negocio en los modelos:

Escribe el código necesario para implementar la lógica de negocio en los modelos. Esto puede incluir operaciones de lectura, escritura, validación y cálculo sobre los datos.

7- Desarrollo de las vistas:

Implementa las interfaces de usuario utilizando tecnologías web como HTML, CSS y JavaScript. Las vistas deben ser responsivas, accesibles y fáciles de usar para el usuario final.

Diseño de la arquitectura de la aplicación web utilizando MVC

8- Creación de los controladores:

Implementa los controladores que manejarán las solicitudes del usuario, interactuarán con los modelos correspondientes y renderizarán las vistas adecuadas.

9- Configuración del enrutamiento:

Configura el enrutamiento de la aplicación para que las URLs se correspondan con las acciones de los controladores. Esto puede hacerse utilizando un enrutador o un sistema de enrutamiento proporcionado por el framework utilizado.

Diseño de la arquitectura de la aplicación web utilizando MVC

10- Pruebas y depuración:

Realiza pruebas exhaustivas para asegurarte de que la aplicación funciona según lo esperado. Identifica y soluciona cualquier error o problema de funcionamiento.

11- Optimización y escalabilidad:

Una vez que la aplicación esté funcionando correctamente, considera la optimización y la escalabilidad. ¿Cómo puedes mejorar el rendimiento de la aplicación? ¿Qué medidas tomarás si la aplicación necesita manejar un mayor volumen de usuarios o datos?

Diseño de la arquitectura de la aplicación web utilizando MVC

12- Documentación y mantenimiento:

Documenta el diseño y la arquitectura de la aplicación para que otros desarrolladores puedan entender y contribuir al proyecto. Además, establece un plan de mantenimiento para realizar actualizaciones y mejoras en el futuro.

Conclusión:

Al seguir estos pasos, podrás diseñar y desarrollar una aplicación web utilizando el patrón MVC de manera efectiva, lo que resultará en una aplicación bien estructurada, fácil de mantener y escalable.

Bootcamp de Desarrollo Web

Clase 12

Patrón de diseño MVC