República Bolivariana De Venezuela
Universidad Alejandro de Humboldt
Facultad de Ingeniería en Informática
Curso: Ingeniería del Software
Sección: DCN0604II-V1

MODELO VISTA-CONTROLADOR (MVC)

Facilitador: Luis Piña

Integrantes:
Nilsen Espitia V-20132008
Oscar Armao V-23613305
Jhonnathan Carbajo V-19199447

Caracas, febrero de 2016

Índice

Introducción

En la rama de la ingeniería del software existen múltiples procesos o estándares que permiten asegurar la calidad de un producto de software, la cual atiende a parámetros que son deseables, o más bien, necesarios para todo el proceso de desarrollo, como la estructuración de programas y la posibilidad de reutilización de códigos que son aspectos frecuentemente utilizados en metodologías de desarrollo.

Las personas o entes dedicados al desarrollo y el mantenimiento de productos de software, constantemente, buscan optimizar las metodologías para obtener buenos resultados. Una de las soluciones para mejorar el proceso de desarrollo ha sido la arquitectura basada en "capas" que separan el código de un programa en función de sus responsabilidades. Esta separación también incluye a los responsables del desarrollo del software, tal es el caso de los diseñadores, quienes podrían no estar familiarizados con determinadas partes de un código que les competen a los programadores, por lo que necesitan separarse de ciertos módulos de programación.

Estos ejemplos, demuestran la necesidad de aplicar una arquitectura útil como es el Modelo Vista-Controlador (MVC), que requiere de la separación del código de un programa en capas atendiendo a sus responsabilidades.

Definición

"El propósito de este patrón es simplificar la implementación de aplicaciones de acuerdo a las peticiones de los usuarios y los datos a desplegar" [Harrop, 2005].

El Modell-View-Controller o Modelo–Vista–Controlador (MVC) es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista, o a cualquier parte del sistema puedan ser hechas con un mínimo impacto en el componente del modelo de datos o en los otros componentes del sistema. Este patrón cumple perfectamente el cometido de modularizar un sistema.

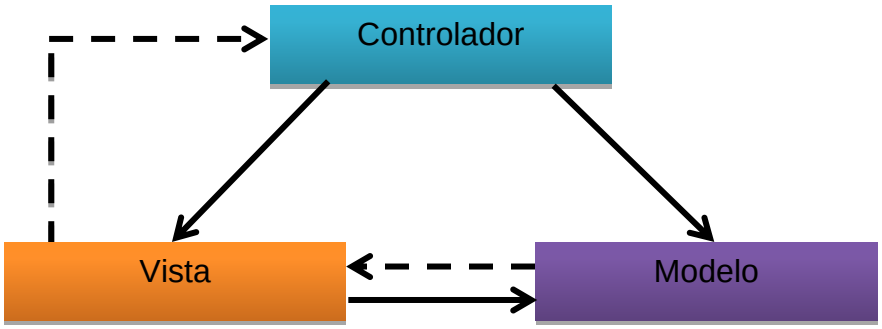


Diagrama sencillo que muestra la relación entre el modelo, la vista y el controlador. ***Nota****: las líneas sólidas indican una asociación directa, y las punteadas una indirecta.*

Historia

El Modelo Vista Controlador (MVC) Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

El patrón MVC fue una de las primeras ideas en el campo de las interfaces gráficas de usuario y uno de los primeros trabajos en describir e implementar aplicaciones software en términos de sus diferentes funciones.

Fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox. Para los años 70 seguidamente en los años 80 Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80. Solo más tarde, en 1988, MVC se expresó como un concepto general en un artículo sobre Smalltalk-80.

En esta primera definición de MVC el controlador se definía como "el módulo que se ocupa de la entrada" (de forma similar a como la vista "se ocupa de la salida"). Esta definición no tiene cabida en las aplicaciones modernas en las que esta funcionalidad es asumida por una combinación de la 'vista' y algún framework moderno para desarrollo. El 'controlador', en las aplicaciones modernas de la década de 2000, es un módulo o una sección intermedia de código, que hace de intermediario de la comunicación entre el 'modelo' y la 'vista', y unifica la validación (utilizando llamadas directas o el "observer" para desacoplar el 'modelo' de la 'vista' en el 'modelo' activo).

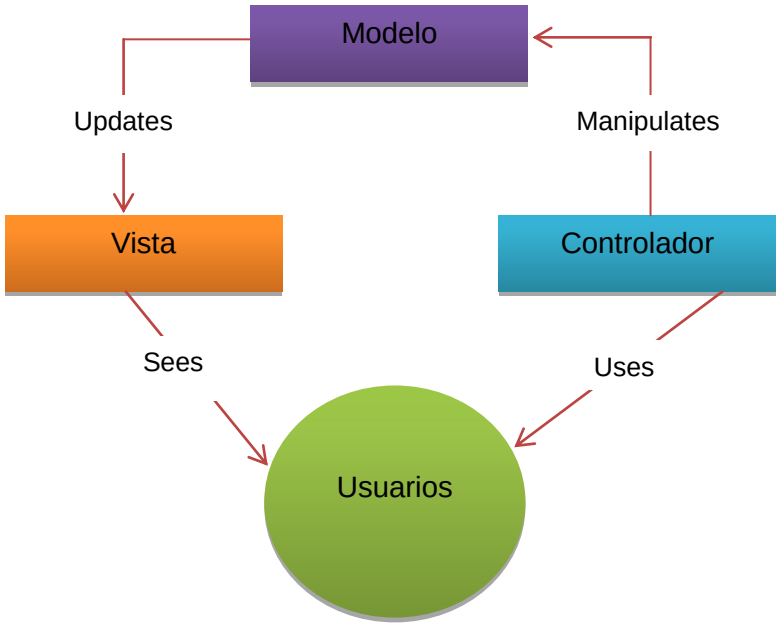
Algunos aspectos de la evolución del patrón MVC:

- HMVC (MVC Jerárquico)
- MVA (Modelo-Vista-Adaptador)
- MVP (Modelo-Vista-Presentador)
- MVVM (Modelo-Vista Vista-Modelo)
- ... y otros que han adaptado MVC a diferentes contextos.

Componentes

Los tres principales componentes del patrón MVC son:

- Modelo: Representa los datos que el usuario está esperando ver, en algunos casos el Modelo consiste de Java Beans.
- Vista: Se encarga de transformar el modelo para que sea visualizada por el usuario, ya sea un archivo de texto normal o en una página Web (HTML o JSP) que el navegador pueda desplegar. El propósito de la Vista es convertir los datos para que el usuario le sean significativos y los pueda interpretar fácilmente; la Vista no debe trabajar directamente con los parámetros de *request*, debe delegar esta responsabilidad al controlador.
- Controlador: Es la parte lógica que es responsable de procesamiento y comportamiento de acuerdo a las peticiones (*request*) del usuario, construyendo un modelo apropiado, y pasándolo a la vista para su correcta visualización. En el caso de una aplicación Web Java en la mayoría de los casos el Controlador es implementado por un servlet.



Una típica colaboración entre los componentes de un MVC

Interacción de los componentes

Se pueden encontrar muchas implementaciones de MVC, pero generalmente el flujo de datos se describe así:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.).

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

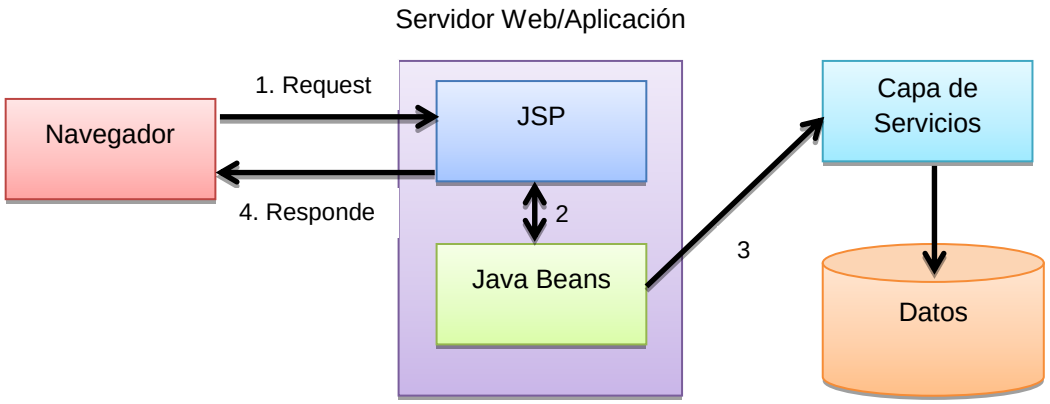
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

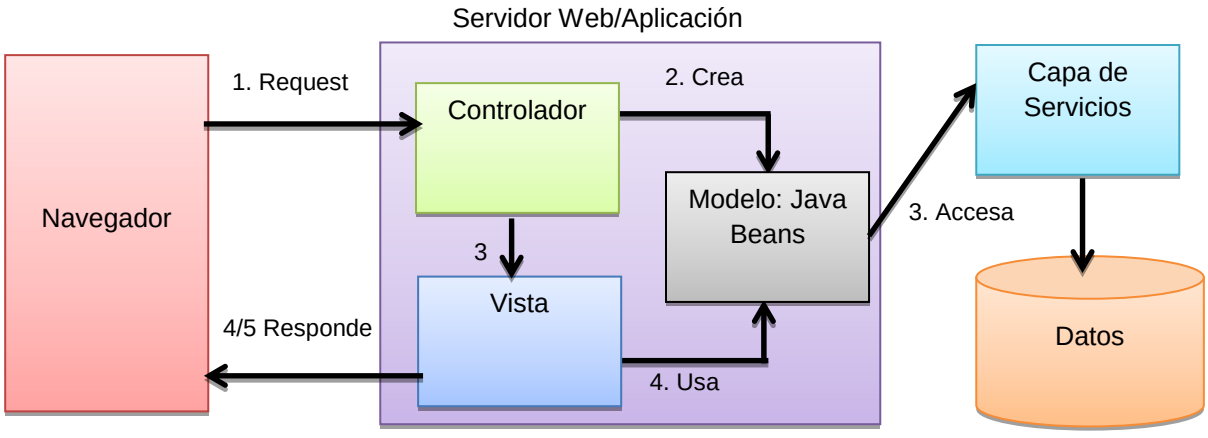
Tipos de Patrones MVC

Actualmente existen dos tipos de patrón MVC:

- MVC tipo 1: Las paginas JSP están en el centro de aplicación y contiene tanto la lógica de control como la de presentación. Este tipo de arquitectura funciona de la siguiente manera: el cliente hace una petición o una página JSP, se construye la lógica de la página, generalmente en objetos java y se transforma el modelo para ser desplegado una vez más.



- MCV tipo 2: Aquí ya existe una clara separación entre el Controlador y el Vista, ya que ahora es directamente el Controlador quien recibe la petición, prepara el modelo y lo transforma para que sea desplegado en la vista. Esta arquitectura se utiliza para aplicaciones complejas.



MVC y bases de datos

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Uso en aplicaciones Web

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón; estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

Los primeros frameworks MVC para desarrollo web planteaban un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor. En este enfoque, el cliente manda la petición de cualquier hiperenlace o formulario al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor.

- Vista: la página HTML.
- Controlador: código que obtiene los datos dinámicamente y genera el contenido HTML.
- Modelo: la información almacenada en base de datos o en XML.

Frameworks

Es un término utilizado en la computación en general, para referirse a un conjunto de bibliotecas utilizadas para implementar la estructura estándar de una aplicación. Todo esto es realizado con el propósito de promover la reutilización de código, con el fin de ahorrarse trabajo al desarrollador al no tener que rescribir ese código para cada nueva aplicación que se desea crear. Existen varios FrameWorks para diferentes fines, algunos son orientados para aplicaciones web, otros para aplicaciones multiplataforma, sistemas operativos, etc.

Este determina la arquitectura de una aplicación, se encarga de definir la estructura general, sus particiones en clases y objetos, responsabilidades clave, así como la colaboración entre las clases objetos, esto evita que el usuario tenga que definirlo y se pueda enfocar en cosas específicas de su aplicación.

Los Frameworks utilizan un variado número de patrones de diseño, ya que así logran soportar aplicaciones de más alto nivel y que reutilizan una mayor cantidad de código, que uno que no utiliza dichos patrones. "Los patrones ayudan hacer la arquitectura de los FrameWorks más adecuada para muchas y diferentes aplicaciones sin necesidad de rediseño". Por esta razón es importante que se documenten que patrones utiliza el FrameWorks para que los que se encuentren familiarizados con dichos patrones puedan tener una mejor visión y poder adentrarse en el FrameWorks mas fácilmente.

El patrón MVC es utilizado en múltiples Frameworks como:

- Java Swing, Java Enterprise Edition
- XForms (formato XML estándar para la especificación de un modelo de proceso de datos XML e interfaces de usuario como formularios web).
- GTK+ (escrito en C, toolkit creado por Gnome para construir aplicaciones gráficas, inicialmente para el sistema X Window)
- Google Web Toolkit, Apache Struts, Ruby On Rails, entre otros.

Conclusión

Este patrón para el diseño y desarrollo permite producir software de alta calidad, convirtiendo una aplicación en un módulo fácil de mantener y mejora la rapidez de su desarrollo. La separación de capas en modelos, vistas y controladores hace que las aplicaciones sean fáciles de entender.

Agregar nuevas funciones y características, así como módulos a códigos "viejos" es un proceso más simple. Dichos modelos permiten a los desarrolladores y diseñadores trabajar de forma simultánea, incluso la creación de prototipos rápidos. La separación en capas también permite a estos responsables, modificar partes del código de la aplicación sin afectar a otras, por ejemplo, si se desea cambiar el diseño de una página o aplicación basta con modificar las vistas y nada más. No hay preocupaciones acerca de "dañar" el modelo (o lógica de negocio) ya que la misma no se modifica.

Referencias

- Pavón, Juan. (2009) Estructura de las Aplicaciones Orientadas a Objetos. El Patrón Modelo-Vista-Controlador (MVC). Universidad Complutense Madrid.
- Sommerville, Ian (2005). Ingeniera del Software. Pearson Educación.

Electrónicas

- http://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html
- http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html
- http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html