Universidad Andrés Bello®

Conectar • Innovar • Liderar

DESARROLLO DE APLICACIONES WEB DINÁMICAS JAVA

El patrón de diseño MVC

Qué es un patrón de diseño

Es una solución general y reutilizable aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporciona soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error.

En 1994, cuatro autores Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, a los que llamaron Gang of Four (GoF), publicaron un libro titulado Design Patterns, elementos de software orientado a objetos reutilizables. Con este trabajo se inició el concepto de patrón de diseño en el desarrollo de software y recoge 23 patrones de diseño comunes. Cada uno de ellos define la solución para resolver un determinado problema, facilitando además la reutilización del código fuente.

Existen muchísimos patrones de diseño de software e irán apareciendo cada vez más. En este post hablaremos de los más conocidos o los llamados patrones clásicos.

¿Por qué usar patrones de diseño?

El gran crecimiento del sector de las tecnologías de la información ha hecho que las prácticas de desarrollo de software evolucionen. Antes se requería completar todo el software antes de realizar pruebas, lo que suponía encontrarse con problemas. Para ahorrar tiempo y evitar volver a la etapa de desarrollo una vez que este ha finalizado, se introdujo una práctica de prueba durante la fase de desarrollo.

Esta práctica se usa para identificar condiciones de error y problemas en el código que pueden no ser evidentes en ese momento. En definitiva, los patrones de diseño te ayudan a estar seguro de la validez de tu código, ya que son soluciones que funcionan y han sido probados por muchísimos desarrolladores siendo menos propensos a errores.

En qué consiste el patrón MVC

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

El Modelo es responsable de:

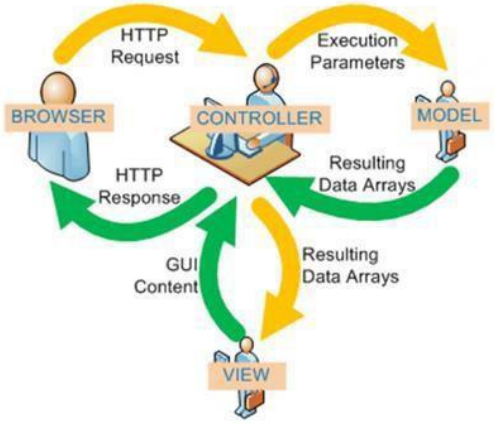
- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El Controlador Es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las vistas son responsables de:

- Recibir datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).



Anexo: Referencias

[1] Qué es Java Enterprise (J2ee, JEE)
Referencia: https://www.fundesem.es/bt/publicacion-java-ee-y-el-desarrollo-web-un-enfoque-de-aprendizaje

[2] Qué son los Servlets
Referencia: https://users.dcc.uchile.cl/~jbarrios/servlets/general.html

[3] Utilizando métodos post y get
Referencia: https://www.ecodeup.com/enviar-parametros-servlet-desde-una-vist a-utilizando-post-get/

[4] Sesiones y cookies
Referencia: https://ricardogeek.com/manejo-de-sesiones-y-cookies-en-servlets-java/

[5] Etiquetas JSTL
Referencia: http://www.jtech.ua.es/ayto/ayto2008/jsp/sesion07-apuntes.html

[6] MVC
Referencia: https://www.ecodeup.com/patrones-de-diseno-en-java-mvc-dao-y-dto/

[7] ¿Qué son los patrones de diseño de software?
Referencia: https://profile.es/blog/patrones-de-diseno-de-software/