ESCOM - IPN

Patrones de Diseño

Análisis y Diseño Orientado a Objetos

Oscar Andrés Rosas Hernandez

Mayo 2018

Índice

1. ¿Qué son los Patrones de Diseño?

Es la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes a diseño de interacción de interfaces.

1.1. ¿De dónde sale el nombre?

- El nombre del patrón de diseño que usamos para describir un problema de diseño.
- El problema, que se describe cuando aplicamos el patrón.
- La solución que describen los elementos que componen el diseño.
- Las consecuencias y resultados de aplicar el patrón

1.2. Organización de los Patrones

- **Creacional:**

  Los patrones de diseño creativo resumen el proceso de instanciación. Ayudan a hacer un sistema independiente de cómo sus objetos son creados, compuestos y representados.

  Ejemplos:

  - Builder
  - Object Pool
  - Singleton
  - Abstract Factory
  - Prototype
  - MVC

- **Estructural:**

  Los patrones estructurales se refieren a cómo las clases y los objetos se componen para formar estructuras más grandes. Los patrones estructurales de la clase utilizan la herencia para componer las interfaces o las implementaciones.

  Ejemplos:

  - Composite
  - Decorator
  - Proxy
  - Adapter
  - Fachada
  - Modulo

- **Comportamiento:**

  Los patrones de comportamiento se ocupan de los algoritmos y la asignación de responsabilidades entre los objetos. Los patrones de comportamiento describen no sólo patrones de objetos o clases, sino también los patrones de comunicación entre ellos.

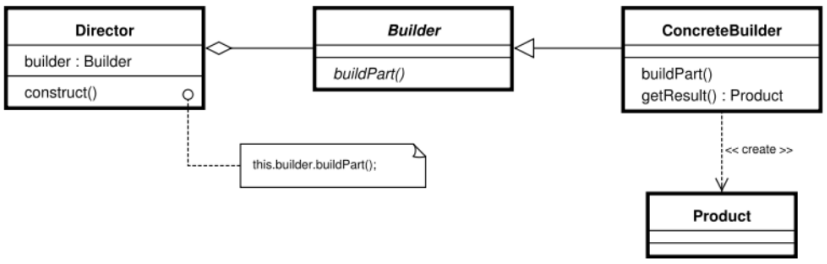
  Ejemplos:

  - Observer
  - Meditor
  - Iterator
  - Strategy
  - State
  - Recuerdo

2. Ejemplos de Patrones de Diseó

2.1. Builder

- Patrón de diseño creacional.
- Es usado para permitir la creación de una variedad de objetos complejos desde un objeto fuente(Producto).
- Separar la construcción de un objeto complejo de su representación para que el mismo proceso de construcción pueda crear diferentes representaciones.
- Aplicabilidad
  - El algoritmo para crear un objeto complejo debe ser independiente de las partes que componen el objeto y cómo se montan.
  - El proceso de construcción debe permitir diferentes representaciones para el objeto que se construye.



2.2. MVC

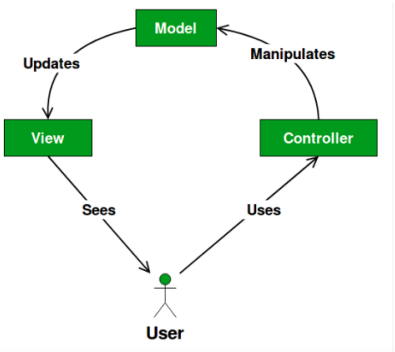
El patrón de diseño Model View Controller (MVC) especifica que una aplicación consiste en un modelo de datos, información de presentación e información de control.

El patrón requiere que cada uno de estos se separe en diferentes objetos.

MVC es más un patrón arquitectónico, pero no para una aplicación completa. MVC principalmente se relaciona con la interfaz de usuario / capa de interacción de una aplicación. Aún necesitará capa de lógica empresarial, tal vez alguna capa de servicio y capa de acceso a datos.

- El Modelo contiene solo los datos puros de la aplicación, no contiene ninguna lógica que describa cómo presentar los datos a un usuario.
- La vista presenta los datos del modelo al usuario. La vista sabe cómo acceder a los datos del modelo, pero no sabe lo que significan estos datos o qué puede hacer el usuario para manipularlo.
- El controlador existe entre la vista y el modelo. Escucha los eventos activados por la vista (u otra fuente externa) y ejecuta la reacción apropiada a estos eventos.

  En la mayoría de los casos, la reacción es llamar a un método en el modelo. Como la vista y el modelo están conectados a través de un mecanismo de notificación, el resultado de esta acción se refleja automáticamente en la vista.



- Ventajas
  - Múltiples desarrolladores pueden trabajar simultáneamente en el modelo, el controlador y las vistas.
  - MVC habilita la agrupación lógica de acciones relacionadas en un controlador. Las vistas para un modelo específico también se agrupan juntas.
  - Los modelos pueden tener múltiples vistas.
- Desventajas
  - La navegación de marco puede ser compleja porque introduce nuevas capas de abstracción y requiere que los usuarios se adapten a los criterios de descomposición de MVC.
  - El conocimiento sobre múltiples tecnologías se convierte en la norma. Los desarrolladores que usan MVC deben ser expertos en múltiples tecnologías.

Referencias

[1] 1.Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides. (1994). Design patterns: elements of reusable object-oriented software . Indianapolis: Addison-Wesley.