Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Diseño Orientado a Objetos

“Ensayo sobre patrones de diseño”

Profesor:

Miguel Salazar

Alumno:

Aaron Alfonso Garcia Torres

Matricula: 1680306

Curso: 06

San Nicolás de los Garza, Nuevo León, 24 de noviembre del 2017

Patrones de diseño

Los patrones de diseño ayudan a identificar y documentar la experiencia en el diseño orientado a objetos. Estos capturan soluciones recurrentes y sus trade-offs a problemas similares en un contexto dado. También ayudan a proveer un vocabulario común para el diseño y la arquitectura al igual que clarifican y documentan arquitecturas existentes. Otra cosa en la que ayudan los patrones de diseño es el hacer avanzar al diseño e influyen en las decisiones de dicho diseño.

En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Antes de continuar, hay que preguntarnos primero ¿Qué es un patrón?

Como llego a decir mi viejo profesor de DOO el semestre pasado:

“Los arquitectos de construcción fueron los que crearon el concepto de “patrón” , como dijo el famoso arquitecto Christopher Alexander:

“Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para después describir el principio de la solución a tal problema, de tal manera que se pueda usar esa solución un millón de veces más, aún sin hacerlo de la misma forma dos veces.””

También existen los llamados catálogos de patrones los cuales son grupos de patrones. Hay catálogos de patrones de diseño para análisis, diseño, arquitectura, arquitectura constructiva, y otros tópicos.

Uno de los libros que nos llego a recomendar mi viejo profesor fue el de “Gang of Four” el cual explica lo siguiente acerca de patrones de diseño:

“Un patrón de diseño nombra, abstrae, e identifica los aspectos principales de una estructura de diseño común que la hace útil para crear un diseño orientado a objetos reutilizable.”

“El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades.”

“Cada patrón de diseño se enfoca en un problema particular orientado a objetos. Describe cuándo se aplica, si puede ser aplicado dependiendo de otras restricciones de diseño, y las consecuencias y trade-offs de su uso.”

¿Cuándo y cómo sabríamos aplicar patrones de diseño?

Algunos autores sugieren que los patrones se apliquen proactivamente a los diseños. Otros sugieren usar pocos patrones y solo modificarlos en cada diseño. Pero de cualquier manera, hay que buscar que el diseño tenga un balance de flexibilidad y simplicidad y ya una vez que se ha decidido usar patrones, la inclinación natural será la de resolver cada problema usando patrones.

Esto es otra cosa que menciono mi profesor:

“Los patrones de diseño no son una “varita mágica” para resolver cada problema.”

Explicando Patrones

MVC

El patrón “ModelView Controller” (MVC) es un patrón de arquitectura que se encuentra en muchos lugares, incluyendo Swing, JavaFX y aplicaciones Web.

El MVC separa la administración de la presentación de la lógica principal de la aplicación y es útil en aquellas aplicaciones donde la interfaz de usuario puede cambiar con frecuencia. Otro punto a favor es que el MVC puede adaptarse a las aplicaciones Web.

Los programadores pueden especializarse en el desarrollo de la interfaz de usuario o en la lógica de negocios. La interfaz de usuario (UI) puede ser cambiada fácilmente sin necesidad de cambiar la lógica de negocios y mejor aún, puede haber más de un tipo de interfaz de usuario y todas necesitar acceso a la misma lógica de negocios.

El MVC puede mostrarte la misma información en diferentes vistas. Las vistas pueden necesitar reflejar inmediatamente los cambios que ocurran en el modelo de datos donde este puede responder inmediatamente a los cambios hechos en las vistas.

La interfaz de usuario puede estar separada de los datos y el procesamiento.

Resumiendo:

Controlador: Acepta las peticiones del usuario, invoca el procesamiento en los componentes del Modelo, y determina qué componente de Vista debe mostrarse.

Vista: Muestra la interfaz gráfica de usuario (GUI) que contiene los componentes de datos del Modelo y generalmente transmite los eventos de la GUI a los componentes del Controlador.

Modelo: Contiene los datos del negocio, el procesamiento y las reglas. El Modelo no debe conocer ningún detalle sobre la interfaz de usuario.

Data Transfer Object

El DTO u Objeto de Transferencia de Datos, casi siempre es necesario para transferir datos entre las diferentes capas de una aplicacion.

Un objeto de transferencia es un componente que encapsla los valores de los atributos y provee métodos de acceso y la implementación de este puede ser mediante ya sea JavaBean O Plain Old Java Objetc (POJO).

DAO

El patrón DAO o Data Access Object, se usa cuando se crea una aplicación que debe persistir información (guardarla de forma permanente).

DAO separa el dominio del problema del mecanismo de persistencia y también usa una interfaz para definir los métodos usados para persistencia y la interfaz permite a la implementación de persistencia ser reemplazada con ya sean DAO’s basados en memoria para soluciones temporales, en archivos para versiones iniciales, en JDBC para persistencia en bases de datos, en JPA (Java Persistence API) para persistencia en bases de datos, etc.

Antes de que llegara DAO, los métodos de persistencia solían estar mezclados con los métodos de negocios, ahora el patrón DAO, mueve la lógica de la persistencia a clases separadas fuera de las clases de dominio.

Entre las muchas de las ventajas de DAO están:

La lógica de negocios que poseen y el acceso a datos que están separados.

Los objetos de acceso a datos promueven la reutilización del código, así como la flexibilidad al tener que hacer cambios en el sistema.

Lo desarrolladores que escriben otros servlet pueden reutilizar el mismo código de acceso a datos.

Este diseño hace más fácil el poder cambiar las tecnologías de front-end que es la vista y del back-end que es el modelo.

Bibliografía:

Hall, Marty & Brown, Larry. “Core Servlets and JavaServer Pages”. Prentice Hall and Sun Microsystems Press.

https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

https://msdn.microsoft.com/es-es/library/bb972240.aspx

https://es.wikipedia.org/wiki/Objeto_de_acceso_a_datos

http://www.oracle.com/technetwork/java/dataaccessobject-138824.html