**Огляд MVC**

Model - view - controller ( MVC або «модель - відображення - контролер» ) - схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, інтерфейс користувача і взаємодія з користувачем розділені на три окремих компоненти таким чином, щоб модифікація одного з компонентів мінімально впливала на інші. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області.

Концепція MVC була описана в 1979 році Трюгве Реенскауг, який тоді працюював над мовою програмування Smalltalk в Xerox PARC. Оригінальна реалізація описана в статті «Applications Programming in Smalltalk-80: How to use Model-View-Controller». Потім Джим Алтофф з командою розробників реалізували версію MVC для бібліотеки класів Smalltalk-80.

В оригінальній концепції була описана сама ідея і роль кожного з елементів: моделі, відображення і контролера. Але зв'язки між ними були описані без конкретизації . Крім того, розрізняють дві основні модифікації даного шаблону:

1. Пасивна модель. Дана модель не має жодних способів впливу на вигляд або контролер, і використовується ними як джерело даних для відображення. Всі зміни моделі відслідковує контролер і він також відповідає за оновлення відображення, якщо це необхідно. Така модель частіше використовується в структурному програмуванні , так як в цьому випадку модель являє просто структуру даних , без методів їх обробки.
2. Активна модель. Ця модель, на відміну від пасивної, оповіщає відображення про те, що в ній відбулися зміни, а відображення, які зацікавлені в оповіщенні , підписуються на ці повідомлення. Це дозволяє зберегти незалежність моделі як від контролера , так і від відображення.

Класичною реалізацією концепції MVC прийнято вважати версію саме з активною моделлю .

В основі класичної моделі лежать такі принципи:

1. Слабке зв'язування.
2. Модель нічого не знає ні про кого. Модель розсилає оповіщення, які може слухати, наприклад, відображення, якщо хоче.
3. Відображення знає про модель але не може її змінювати, відображення може маніпулювати контролером.
4. Контролер знає про Моделі і може її змінювати, а також знає про відображення і може його (їх) змінювати.

Концепція MVC дозволяє розділити дані , подання та обробку дій користувача на три окремих компонента :

1. Модель. Модель надає знання: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, як ці знання можна візуалізувати.
2. Відображення, подання. Відповідає за відображення інформації (візуалізацію) . Часто як відображення виступає форма (вікно) з графічними елементами .
3. Контролер. Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель та відображення для реалізації необхідної реакції.

Важливо відзначити , що як відображення , так і контролер залежать від моделі. Однак модель не залежить ні від відображення, ні від контролера. Тим самим досягається призначення такого поділу: вон дозволяє будувати модель незалежно від візуального представлення, а також створювати кілька різних відображень для однієї моделі .

Для реалізації схеми Model -View - Controller використовується досить велика кількість шаблонів проектування (залежно від складності архітектурного рішення) , основні з яких « спостерігач », «стратегія», « компонувальник » .

Найбільш типова реалізація відокремлює відображення від моделі шляхом встановлення між ними протоколу взаємодії , використовуючи апарат подій (підписка або сповіщення) . При кожній зміні внутрішніх даних в моделі вона оповіщає всі залежні від неї відображення, і відображення оновлюється. Для цього використовується шаблон «спостерігач». При обробці реакції користувача відображення обирає, залежно від потрібної реакції, потрібний контролер , який забезпечить той чи інший зв'язок з моделлю . Для цього використовується шаблон «стратегія» , або замість цього може бути модифікація з використанням шаблону «команда». А для можливості однотипного поводження з підоб'єктами складно- складеного ієрархічного виду може використовуватися шаблон «компонувальник». Крім того, можуть використовуватися й інші шаблони проектування, наприклад, «фабричний метод», який дозволить задати за замовчуванням тип контролера для відповідного виду .

З розвитком об'єктно-орієнтованого програмування та поняття про шаблони проектування був створений ряд модифікацій концепції MVC , які при реалізації у різних авторів можуть відрізнятися від оригінальної .

**Model-View-Presenter**

Model-View-Presenter (MVP) - шаблон проектування, похідний від MVC, який використовується в основному для побудови інтерфейсу користувача.

У MVP Presenter бере на себе функціональність посередника (граючи роль, аналогічну контролеру в MVC). Крім того, Presenter відповідає за управління подіями для користувача інтерфейсу (наприклад, mouseDown, keyDown і т.д.), яке зазвичай було турботою відображення. В результаті, модель стає чітко моделлю предметної області.

MVP - шаблон проектування інтерфейсу користувача, який був розроблений для полегшення автоматичного модульного тестування і поліпшення розподілу відповідальності у презентаційній логіці (відділення логіки від відображення) :

1. Модель являє собою інтерфейс , що визначає дані, які відображаються або беруть участь в інтерфейсі іншим чином .
2. Відображення - це інтерфейс, який відображає дані (Модель) і маршрутизує команди користувача (або події) Presenter-у, щоб той виконував дії над цими даними.
3. Presenter діє над Моделлю і Відображенням . Він витягає дані зі сховища (Моделі) і форматує їх для подання у Відображення.

Зазвичай, реалізація Відображення створює екземпляр конкретного об'єкта Presenter-а і надає посилання на себе. (Формально параметром конструктора Presenter'a є інтерфейс Відображення, в той час як в конструктор передається конкретна реалізація Відображення) . Коли викликається подія Відображення, воно викликає конкретний метод Presenter'а , що не має параметрів і значення, що повертається . Далі Presenter через інтерфейс отримує дані з Відображення, після чого викликає методи Моделі і встановлює через інтерфейс дані з Моделі в Відображення.

Кількість логіки, допустимої в Відображенні, різниться для різних реалізацій.

З точки зору багаторівневої моделі програм в ООП, Presenter може розглядатися як самостійний рівень між рівнем програми і рівнем інтерфейсом користувача.

**Model-View-View-Model**

Шаблон Model-View-View-Model (MVVM) - застосовується при проектуванні архітектури програми. Спочатку був представлений співтовариству Джоном Госсманом в 2005 році як модифікація шаблону Presentation Model. Більшою своєю частиною він базується на MVC. MVVM орієнтований на сучасні платформи розробки, такі як Windows Presentation Foundation, Silverlight від компанії Microsoft, ZK framework.

MVVM використовується для розділення моделі та її відображення, що необхідно для їхніх змін окремо один від одного.

MVVM зручно використовувати замість класичного MVC і йому подібних в тих випадках, коли в платформі, на якій ведеться розробка, присутнє «зв'язування даних».

У шаблонах проектування MVC/MVP зміни в інтерфейсі не впливають безпосередньо на Mодель , а попередньо йдуть через Контролер або Presenter . У таких технологіях як WPF і Silverlight є концепція «зв'язування даних», що дозволяє пов'язувати дані з візуальними елементами в обидві сторони. Отже , при використанні цього прийому застосування моделі MVC стає вкрай незручним через те, що прив'язка даних до подання безпосередньо не вкладається в концепцію MVC/MVP .

Шаблон MVVM ділиться на три частини:

1. Модель, так само, як у класичному MVC, являє собою фундаментальні дані, необхідні для роботи програми .
2. Представлення - це графічний інтерфейс, тобто вікно, кнопки і т.п. Відображення є підписником на подію зміни значень властивостей або команд, що надаються Моделлю подання. У разі, якщо в Моделі відображення змінилась якась властивість, то вона сповіщає всіх підписників про це, і Відображення, в свою чергу, запрошує оновлене значення властивості з Моделі представлення. У випадку, якщо користувач впливає на якийсь елемент інтерфейсу, Відображення викликає відповідну команду, надану Моделлю подання.
3. Модель відображення (англ. View Model ) є , з одного боку , абстракцією Відображення, а з іншого, надає обгортку даних з Моделі, які підлягають скріпленню. Тобто, вона містить Модель, яка перетворена до Відображення, а також містить у собі команди, якими може користуватися Відображення, щоб впливати на Модель .

**Model–view–adapter**

Model–view–adapter (MVA) або посередник-контролер MVC є архітектурним шаблоном в багаторівневій архітектурі, який використовується в розробці програмного забезпечення. У складних комп'ютерних системах, які представляють великі обсяги даних для користувачів, розробники часто хочуть, щоб модель та інтерфейс користувача вид відносились , так що зміни в інтерфейсі не впливали на обробку даних і щоб дані могли бути реорганізовані без зміни інтерфейсу користувача. MVA і традиційний MVC і намагалися вирішити цю проблему , але двома різними стилями рішення. Традиційний MVC влаштовує модель, відображення і контролерв трикутнику , з моделі , уявлення і контролера , як вершин , так що деякі інформаційні потоки між моделлю і відображенням знаходяться за межами прямого контролю контролера. Model–view–adapter вирішує цю проблему трішки інакше, влаштовуючи модель, адаптер або посередник-контролер , а також відображення лінійно без яких-небудь з'єднань безпосередньо між моделлю і представленням.

Відображення повністю відокремлене від моделі. Це дозволяє відображенню і моделі взаємодіяти тільки через контролер-посередник або адаптер. За допомогою цього устрою, тільки адаптер або посередник-контролер має знання про модель і вигляд, тому що це є обов'язком виключно адаптера або посередника-контролера. Модель і відображення зберігаються навмисно не знаючи один одного. У традиційній MVC , модель і відображення знали про існування один одного.

Це рішення дозволяє широкому спектру різних відображень отримати доступ до тієї ж моделі або через точно такий же адаптер, або через той же клас адаптерів. Наприклад, одна базова модель даних для зберігання і схеми і технології може бути доступна через широкий спектр дещо різних відображень -наприклад, Qt GUI, Microsoft MFC GUI, GTK + GUI, Microsoft. Адаптер або клас адаптерів тримає модель абсолютно не звертаючи, що вона підтримує велику кількість інтерфейсів користувача і, можливо, навіть підтримуючи це розмаїття одночасно.

Для виконання даної курсової роботи буде використовуватися класична схема MVC, оскільки вона задовольняє всі вимоги до даного програмного продукту.