ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ MVC И MVVM И ИХ ОСОБЕННОСТИ

Сиваков А.А., Иванов А.В.

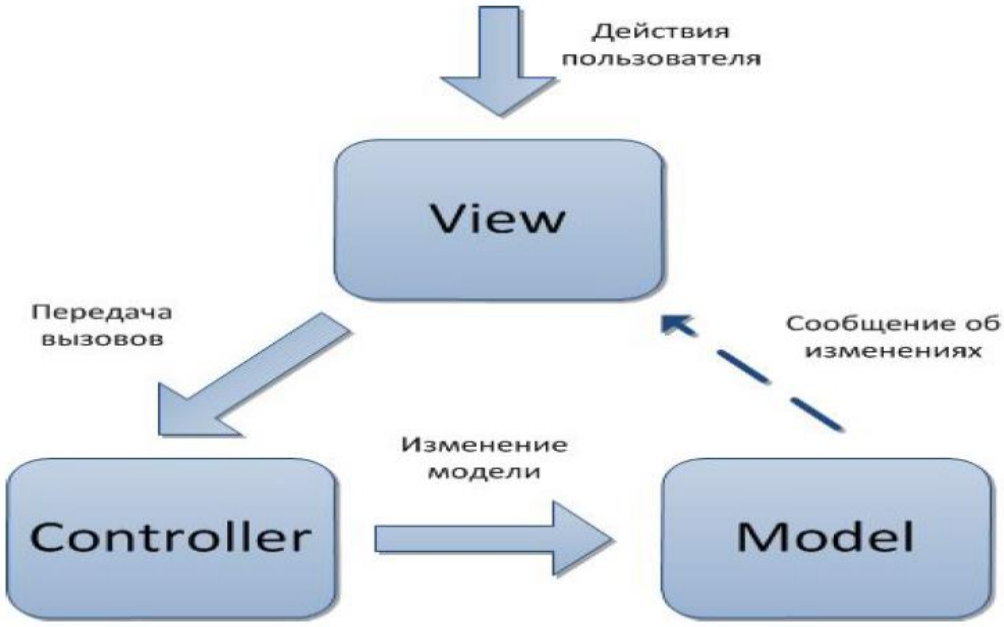
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Савченко В. В. — канд. техн. наук, доцент

В этой статье будет описаны особенности и приницпы работы паттернов MVC и MVVM.

Шаблон проектирования MVC. MVC состоит из трех компонент: View (представление, пользовательский интерфейс). Model (модель, бизнес логика). Controller (реализует диаграмму прецедентов).

Основная идея этого паттерна в том, что и контроллер и представление зависят от модели, но модель никак не зависит от этих двух компонент. Это как раз и позволяет разрабатывать и тестировать модель, ничего не зная о представлениях и контроллерах. В идеале контроллер так же ничего не должен знать о представлении (хотя на практике это не всегда так), и в идеале для одного представления можно переключать контроллеры, а также один и тот же контроллер можно использовать для разных представлений (так, например, контроллер может зависеть от пользователя, который вошел в систему). Пользователь видит представление, на нем же производит какие-то действия, эти действия представление перенаправляет контроллеру и подписывается на изменение данной модели, контроллер в свою очередь производит определенные над моделью данных, представление получает последнее состояние модели и отображает ее пользователю.



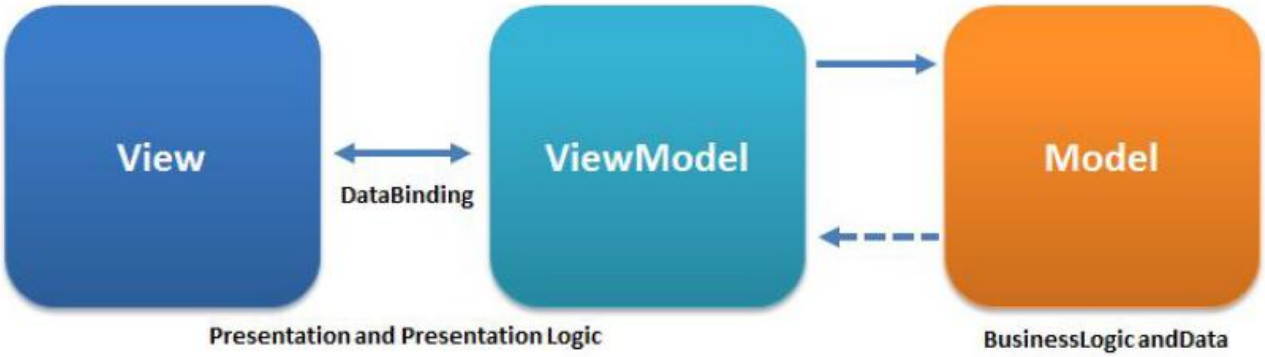
Шаблон MVVM делится на три части:

● *Модель* (так же, как в классической MVC) представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

● *Представление* — графический интерфейс (окна, списки, кнопки и т. п.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.

● *Модель Представления* — с одной стороны, абстракция Представления, а с другой — обёртка данных из Модели, подлежащих связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться

Представление, чтобы влиять на Модель.



Особенности MVVM:

Вы получаете полностью тестируемую логическую модель вашего приложения.

Поскольку ViewModel предоставляет View всю необходимую информацию в удобном виде, то само представление может быть довольно простым. А дизайнер может экспериментировать с внешним видом и стилем в редакторе Expression Blend и изменять его, не влияя на пользовательский интерфейс.

И, возможнр избежать написания кода для View (*code behind*). Теперь это повод для споров среди поклонников паттерна MVVM: как правило, вам не нужно писать дополнительный код для View, и найдется решение лучше. Да, иногда нужно проделать некоторые трюки (*такие как создание attached behaviors*), но они обеспечивают хорошие и переиспользуемые решения. Тем не менее я также признаю, что не все любят XAML разметку и связывание данных в XAML. Паттерн ViewModel не заставляет вас использовать или избегать *code behind* . Делайте то, что кажется вам правильным.

Выводы:

Модель в паттернах выглядит одинаково и имеет одну и ту же цель – получение, обработка, а также сохранение данных.

В классическом MVC пользовательский ввод обрабатывает Controller, а не View.

Современные ОС и библиотеки виджетов берут на себя обработку пользовательского ввода, поэтому у вас больше нет нужды в контроллере из паттерна MVC.

Цель MV*-паттернов: отделить друг от друга отображение UI, логику интерфейса и данные (их получение и обработку).

Используя MV*-паттерн в своем приложении, вы упрощаете его поддержку и тестирование, отделяете данные от способа их визуализации.

Если в системе присутствует хорошая реализация автоматического связывания данных, то MVVM – это ваш выбор.

Паттер MVVM и MVC широко распространены среди фреймворков гибридной миобильной разработки.

Presentation Model – хорошая альтернатива MVVM, и будет полезна там, где нет автоматического связывания. Но вам придется писать код связывания самостоятельно (это несложный, но рутинный код).

Список использованных источников:

1. Design Patterns // Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides // Addison-Wesley, 2016. – 395с.

2. The Art of Computer Programming // Donald Knut //Addison-Wesley, 1968. – 519с.