УДК 004.415.23

ИСПОЛЬЗОВАНИЕ СОВРЕМЕННЫХ ШАБЛОНОВ ПРОЕКТИРОВАНИЯ ПРИ РАЗРАБОТКЕ МУЛЬТИМЕДИА ПРИЛОЖЕНИЙ

Крапивин В.В., Кулиненко К.Д.,
научный руководитель канд. пед. наук, доцент Виденин С.А.
Сибирский федеральный университет

В настоящее время существует множество идей проектов для дипломных работ, но за которые никто не берётся, так как они огромные по своей структуре и требуют слаженной командной работы. Так же имеет место проблема, заключающаяся в том, что преподаватели не могут разбить разработку такого рода проектов на несколько студентов, которым данная задача кажется невыполнимой в указанных временных рамках из-за достаточно большого объёма работ. Однако, был найден способ решения вышеупомянутых проблем – разделение работ и обязанностей на основе шаблонного проектирования.

В процессе создания мультимедиа приложения, в котором принимают участие несколько разработчиков, возникают проблемы, требующие быстрого и ультимативного решения. Данные проблемы порождают тривиальные задачи и возникают с первоначального этапа – проектирования. Несмотря на то, что достаточно много времени тратится на обсуждение архитектуры и дизайна конкретного решения задачи, разработчикам не всегда удаётся прийти к одинаковому осознанию принимаемого решения.Впрочем, нередки случаи использования сторонних наработок, которые ещё больше осложняют процесс создания мультимедиа приложения. Для того, чтобы минимизировать ресурсы, затрачиваемые на однотипные возникающие проблемы, существуют элегантные методы решения – паттерные технологии.

Для использования паттерных технологий не требуются особые инструментарии языка программирования или сложные приёмы. При их использовании затраты времени на написание кода программы увеличиваются, по сравнению со специализированным решением, применимого только в данной ситуации. Но эти затраты окупаются за счёт большей гибкости и возможности повторного использования.

Обычно шаблон является лишь примером решения задачи, а не законченным образцом. Объектно-ориентированные шаблоны показывают взаимодействия между классами и объектами без определения того, какие именно конечные классы или объекты буду использоваться. Любые алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления. Применение шаблонов концептуально похоже на использование готовых библиотек кода. Единожды правильно составленный шаблон проектирования позволяет пользоваться им на постоянной основе.

Ярким примером паттерных технологий является шаблон проектирования MVC(см. рис.1).Шаблон MVCописывает просто способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате, приложение гораздо легче масштабируется, тестируется, исправляется, а также в дальнейшем сопровождается и реализуется.

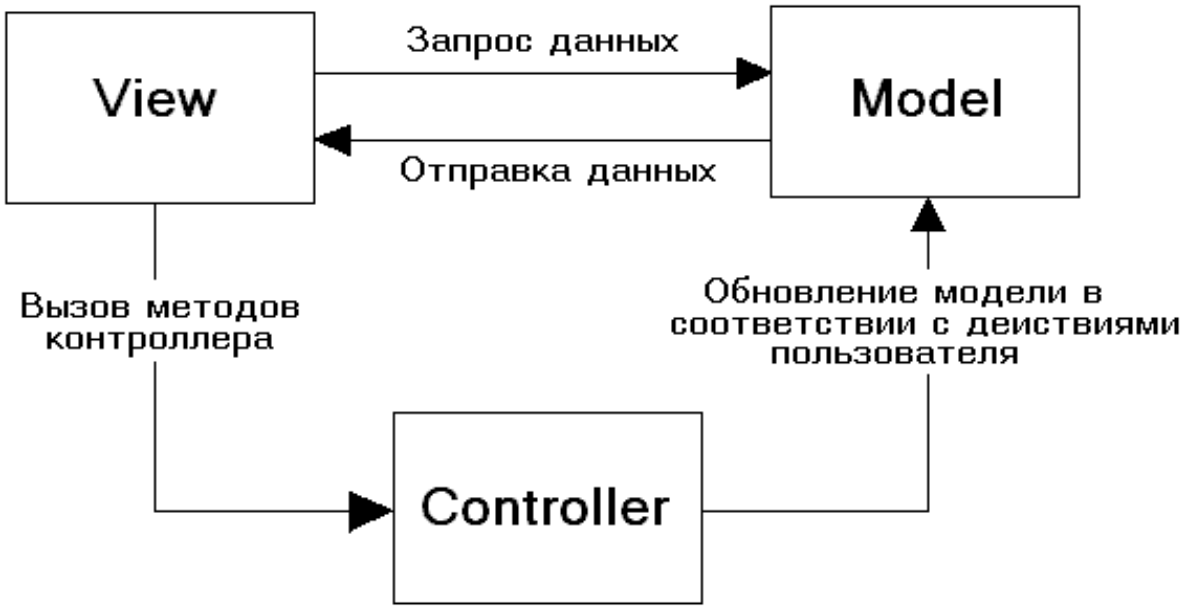

Рис.1 – Шаблон MVC

В архитектуре MVCмодель предоставляет данные и правила бизнес-логики, предоставление отвечает за пользовательский интерфейс, а контроллер уже в свою очередь обеспечивает взаимодействие между моделью и представлением.

Стандартную последовательность работы MVCмультимедиа приложения можно описать следующим образом:

1. При запуске пользователем приложения, скрипт инициализации создаёт экземпляр и запускает его на выполнение. При этом отображается главное меню приложения.
2. Приложение получает запрос от пользователя и определяет запрошенные контроллер и действие.
3. Приложение создает экземпляр контроллера и запускает метод действия, в котором, к примеру, содержаться вызовы модели, считывающие информацию из массива данных.
4. После этого, действие формирует представление с данными, полученными из модели и выводит результат пользователю.

Модель — содержит бизнес-логику приложения и включает методы выборки, обработки и предоставления конкретных данных, что зачастую делает ее очень массивной, что является вполне нормальным решением. Модель не должна напрямую взаимодействовать с пользователем. Все переменные, относящиеся к запросу пользователя должны обрабатываться в контроллере. Модель не должна генерировать XAML или другой код отображения, который может изменяться в зависимости от нужд пользователя.

Вид — используется для задания внешнего отображения данных, полученных изразумеется контроллера и естественно модели.Виды содержат XAML-разметку. Виды не должны напрямую обращаться к массиву данных. Этим должны заниматься модели. Задачу обработкиданных, полученных из запроса пользователя, должен выполнять разумеется контроллер.
Виды обычно разделяют на общий шаблон, содержащий код, общий для всех скриптов и части шаблона, которые используют для отображения данных выводимых из модели.

Контроллер — связующее звено, соединяющее модели, виды и другие компоненты в рабочее приложение. Контроллер отвечает за обработку запросов

пользователя. Контроллер не должен содержать запросов. Их лучше держать в моделях. Контроллер не должен содержать XAML и другой разметки. Её стоит выносить в виды. В хорошо спроектированном MVC-приложении контроллеры обычно очень тонкие и содержат только несколько сотен строк кода. Модели, наоборот, очень массивные и содержат большую часть кода, связанную с обработкой данных, т.к. структура данных и бизнес-логика, содержащаяся в них, обычно довольно специфична для конкретного приложения.

Пожалуй, стоит упомянуть, почему мы так заинтересовались шаблонным проектированием. Во-первых, паттерны позволяют по-новому взглянуть на объектно-ориентированное программирование. Во-вторых, упрощается понимание стороннего кода. В-третьих, благодаря тому, что все модули приложения независимы друг от друга, позволяет разработчикам не задумываться о степени готовности сторонних модулей и заниматься реализацией исключительно своих задач, не дожидаясь окончательных версий представленных иными разработчиками решений.

Благодаря шаблону MVCв рамках нашего проекта удалось развести работу между программистом UI и программистом бизнес-логики согласно структуре шаблона. Это дало возможность программисту UI не вникать глубоко в бизнес-логику проекта, а получать и использовать исключительно только те данные, которые ему требуются для отображения интерфейса. В тоже время разработчик бизнес-логики совершенно не заморачивается по поводу интерфейса и тратит всё своё время исключительно на логику проекта. Исходя из выше упомянутого, код получается гораздо более структурированным, и, тем самым, облегчается поддержка, тестирование и повторное использование решений.

Шаблоны проектирования — это один из мощнейший инструментов разработчика, который помогает ему, как это ни странно, сэкономить время и сделать более качественное решение. Как и любой другой инструмент,в одних руках он может принести много пользы, а в других — один только вред.

СПИСОК ЛИТЕРАТУРЫ

1. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Design Patterns: Elements of Reusable Object-Oriented Software (Приемыобъектно-ориентированногопроектирования. Паттерны проектирования). - 16,18,38 с.

2. Адам Фримен ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание = Pro ASP.NET MVC 4, 4th edition. — М.: «Вильямс», 2013. — 688 с.