Таким образом, станет возможно заменить множество стеллажей с деталями и режущими инструментами, или стенды с двигателями на папку листов со специальными маркерами.

Конечно, наряду с преимуществами технологии, есть один существенный недостаток – это программное обеспечение, которое нужно разработать. И плюс к этому, нужны модели, которые будут визуализироваться в дополненной реальности. Если модели могут разработать сами студенты, т. к. студентов технической направленности обучают пользоваться САПР, то для создания программного обеспечения нужны специально обученные люди.

По итогу, можно с уверенностью сказать, что применение технологий виртуальной, а в особенности дополненной реальности уже вполне реализуемо на сегодняшний день. Потребуется только разработка программного обеспечения и моделей для визуализации.

Список использованных источников

1. Виртуальная реальность [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Виртуальная_реальность (дата обращения: 21.03.2022).

2. Дополненная реальность [Электронный ресурс]: Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Дополненная реальность (дата обращения: 21.03.2022).

УДК 004.021

Проблемы понимания паттерна MVC в технологии ASP .NET

Выскварко Н. С., студент
Белорусский национальный технический университет
Минск, Республика Беларусь
Научный руководитель: канд. техн. наук, доцент Евтухова Т. Е.

Аннотация:

В статье рассматриваются проблемы понимания шаблона MVC при разработке с использованием ASP .NET технологии. Были даны определения таким понятия, как .NET Framework, ASP .NET, MVC. Описаны наиболее распространенные ошибки при использовании

MVC-паттерна. В заключении статьи были даны рекомендации для ликвидации ошибок при использовании MVC в ASP .NET.

NET Framework – это программная платформа, выпущенная компанией Microsoft. ASP .NET – технология создания веб-приложений и веб-сервисов от компании Microsoft. Она поддерживает работу с несколькими языками программирования, входящими в сборку фреймворка: Basic NET, C#, J# и ряд прочих. С данной платформой есть возможность создавать как простейшие веб-ресурсы, так и очень сложные сайты, способные к обработке многотысячного потока пользователей [1].

В ASP .NET применяется традиционная схема MVC – Модель-Вид-Контроллер. MVC – это шаблон программирования, который позволяет разделить логику приложения на три части:

– Model (модель);
– View (вид или представление);
– Controller (контроллер).

Все элементы отвечают за конкретные действия. Модель получает данные от контроллера, выполняет необходимые операции и передает их в представление. Представление же в свою очередь получает данные от модели и выводит их для пользователя. Контроллер обрабатывает действия пользователя, проверяет полученные данные и передает их модели [2]. Схема взаимодействия этих трех элементов представлена на рисунке 1.

Многие программисты при разработке ASP. NET MVC приложений сталкиваются с проблемами понимания MVC. Причина в том, что многие MVC-фреймворки не вполне следуют шаблону MVC, а люди, использующие их, сами того не замечая, еще больше отклоняются от него. Казалось бы, он довольно прост, но раз за разом возникают проблемы его понимания.

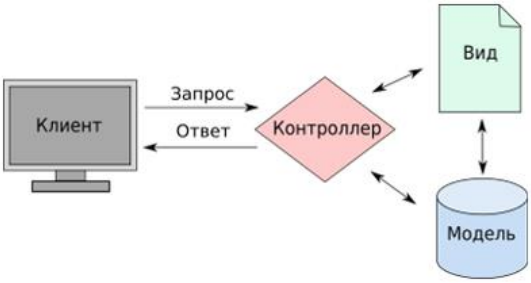

Рис. 1 – Схема взаимодействия элементов MVC-паттерна

Первой весьма распространенной ошибкой является неправильное представление о том, что такое модель. Многие воспринимают Model, как схему базы данных, когда она в свою оче-редь является моделью предметной области.

Довольно часто моделью называют классы, которые отображаются на сущность базы данных. Тем не менее, в рамках шаблона MVC данный термин имеет немного другой смысл. Model – это модель предметной области, она содержит всю бизнес-логику приложения. Также в нее будут входить сущности, различные сервисы, репозитории, фабрики и прочее. Модель хранит состояние, а также может его изменять в зависимости от того, какие действия над ней производятся [2].

Еще одна проблема связана с восприятием View как анемичного шаблона, когда на самом деле это активный инструмент представления.

В веб-разработке под View часто понимается некий шаблон со специальной разметкой, куда подставляются данные, которые в свою очередь предоставляет контроллер. При этом он сам напрямую не обращается к модели предметной области и не получает оттуда данные. Такая ситуация приводит к тому, что контроллеру необходимо подготовить данные в специальном виде, что приводит к нарушению принципа единственной ответственности – каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс, а все его сервисы должны быть направлены исключительно на обеспечение этой обязанности. В ситуации, когда в шаблон будет необходимо добавить новую информацию, придется

изменять свойства модели представления и код внутри контроллера или провайдера модели представления.

В описании MVC говорится, что представление имеет связь с моделью, и может запрашивать у нее данные, а в некоторых случаях даже менять состояние модели. Если следовать данному правилу, для транспорта данных в представление, то модель представления с большим количеством свойств будет уже не нужна.

Следующая ошибка заключается в том, что контроллер используют как место для бизнес-логики и подготовки данных для отображения.

Контроллер, а в частности его действие (в веб-разработке) является конечной точкой маршрутизации запроса. Все, что должно сделать действие, это:

– выполнить валидацию запроса;

– оповестить модель и получить ответ;

– отобразить представление, передав ему данные из модели [2].

Таким образом можно выделить следующие основные принципы при разработке ASP .NET MVC приложений:

– Model должна отображать модель предметной области;

– View необходимо использовать в качестве активного инструмента представления;

– Controller не должен содержать бизнес-логику и подготавливать данные для отображения.

Список использованных источников

1. Троелсен Э. У. Язык программирования C# и платформы .NET и .NET Core 8-е издание: пер. с англ. / Э. Троелсен, Ф. В. Джепикс – СПб: ООО «Диалектика», 2018. – 1328 с.

2. Model-View-Controller [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Controller – Дата доступа: 03.03.2021.