Содержание:

Введение

MVC — это шаблон программирования, который позволяет разделить логику приложения на три части:

- *Model* (модель). Получает данные от контроллера, выполняет необходимые операции и передаёт их в вид.
- *View* (вид или представление). Получает данные от модели и выводит их для пользователя.
- *Controller* (контроллер). Обрабатывает действия пользователя, проверяет полученные данные и передаёт их модели.

Основная цель применения этой концепции программирования состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной *модели* можно присоединить несколько *видов*, при этом не затрагивая реализацию *модели*. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
2. Не затрагивая реализацию *видов*, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой *контроллер*;
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (*модели*), вообще не будут осведомлены о том, какое *представление* будет использоваться.

Функциональные возможности и расхождения

Поскольку MVC не имеет строгой реализации, то реализован он может быть по-разному. Нет общепринятого определения, где должна располагаться бизнес-логика. Она может находиться как в контроллере, так и в модели. В последнем случае, модель будет содержать все бизнес-объекты со всеми данными и функциями.

Некоторые фреймворки жестко задают где должна располагаться бизнес-логика, другие не имеют таких правил.

Также не указано, где должна находиться проверка введённых пользователем данных. Простая валидация может встречаться даже в представлении, но чаще они встречаются в контроллере или модели.

Интернационализация и форматирование данных также не имеет четких указаний по расположению.

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента:

Модель

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем) , просто предоставляя доступ к данным и управлению ими.

Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей».

Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

Представление

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Контроллер

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

Условно-обязательные модификации

Для реализации схемы «Model-View-Controller» используется достаточно большое число шаблонов проектирования (в зависимости от сложности архитектурного решения), основные из которых — «наблюдатель», «стратегия», «компоновщик».

Наиболее типичная реализация — в которой представление отделено от модели путём установления между ними протокола взаимодействия, использующего «аппарат событий» (обозначение «событиями» определённых ситуаций, возникающих в ходе выполнения программы, — и рассылка уведомлений о них всем тем, кто подписался на получение): при каждом особом изменении внутренних данных в модели (обозначенном как «событие»), она оповещает о нём те зависящие от неё представления, которые подписаны на получение такого оповещения — и представление обновляется. Так используется шаблон «наблюдатель».

При обработке реакции пользователя — представление выбирает, в зависимости от реакции, нужный контроллер, который обеспечит ту или иную связь с моделью. Для этого используется шаблон «стратегия», или вместо этого может быть модификация с использованием шаблона «команда».

Для возможности однотипного обращения с подобъектами сложно-составного иерархического вида — может использоваться шаблон «компоновщик». Кроме того, могут использоваться и другие шаблоны проектирования — например, «фабричный метод», который позволит задать по умолчанию тип контроллера для соответствующего вида.

Преимущества

Прежде всего дифференциация разработчиков php сайта на отделы. Также увеличивается скорость работы php приложения, если создается крупный проект.

Ну и то, что касается непосредственно самого php разработчика, это правильная структуризация php кода (все на своих местах, так легче для понимания).

Список используемой литературы

https://ru.wikipedia.org/wiki/Model-View-Controller#Концепция

https://moluch.ru/archive/87/16899/