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.

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).