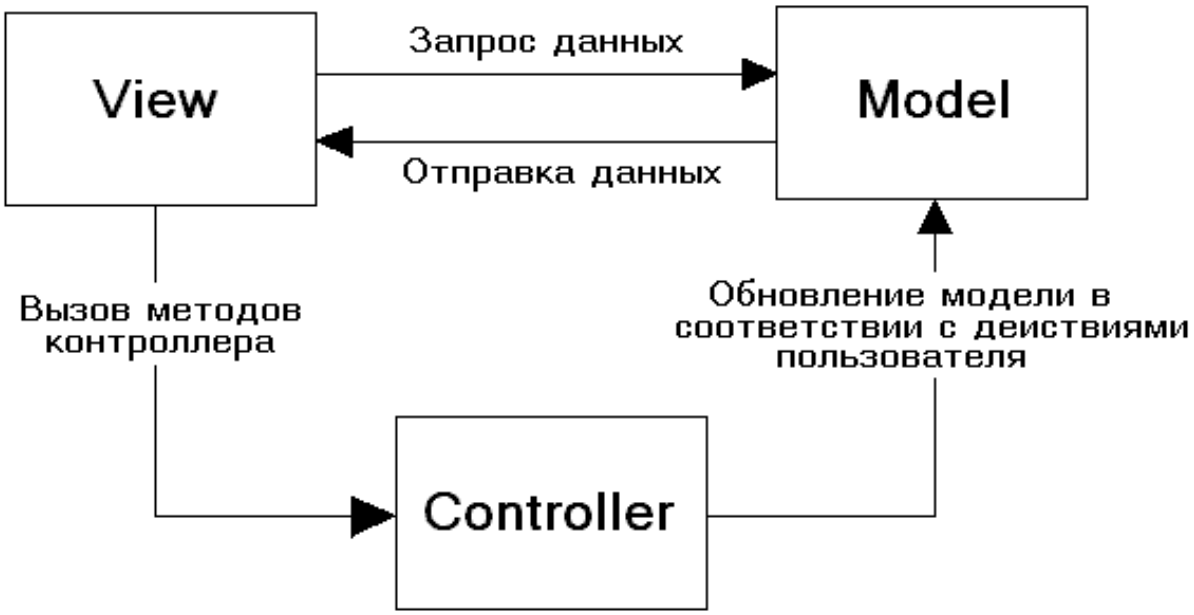
Обзор современных фреймворков, реализующих шаблон архитектуры системы MVC

***MVC (Model-View-Controller: модель-вид-контроллер)*** – архитектурный паттерн проектирования, который используется для организации программного кода.

Его основная идея в том, чтобы отделить модели данных, их отображение и операции взаимодействия с пользователем. Выигрыш от использования такой архитектуры заключается в том, что она позволяет упорядочить код, распределив его по уровням, каждый из которых определяет сферу ответственности. Это минимизирует взаимозависимость программных компонент, что в свою очередь облегчает их последующую модификацию. Доработка и развитие такой системы становится проще.

Приложение разделяется на три основных компонента, каждый из которых отвечает за различные задачи. Давайте подробно разберём компоненты на примере.




1. **Контроллер (Controller)**

**Контроллер** управляет запросами пользователя (получаемые в виде запросов HTTP GET или POST, когда пользователь нажимает на элементы интерфейса для выполнения различных действий). Его основная функция — вызывать и координировать действие необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно

контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

2. **Модель (Model)**

**Модель** - это данные и правила, которые используются для работы с данными, которые представляют концепцию управления приложением. В любом приложении вся структура моделируется как данные, которые обрабатываются определённым образом.

Модель даёт контроллеру представление данных, которые запросил пользователь (сообщение, страницу книги, фотоальбом, и тому подобное). Модель данных будет одинаковой, вне зависимости от того, как мы хотим представлять их пользователю. Поэтому мы выбираем любой доступный вид для отображения данных.

Модель содержит наиболее важную часть логики нашего приложения, логики, которая решает задачу, с которой мы имеем дело (форум, магазин, банк, и тому подобное). Контроллер содержит в основном организационную логику для самого приложения (очень похоже на ведение домашнего хозяйства).

3. **Вид (View)**

**Вид** обеспечивает различные способы представления данных, которые получены из модели. Он может быть шаблоном, который заполняется данными. Может быть несколько различных видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.

Веб приложение обычно состоит из набора контроллеров, моделей и видов. Контроллер может быть устроен как основной, который получает все запросы и вызывает другие контроллеры для выполнения действий в зависимости от ситуации.

Сравнительная таблица фреймворков

| № | Название | Отличие от других |
|---|---|---|
| 1 | Django | Django имеет открытый исходный код. Помимо прочего, он предоставляет функционал для масштабирования ваших Python-проектов. Он регулярно обновляется, чтобы соответствовать последним версиям языка.<br><br>Django полностью совместим с большинством движков баз данных. Когда речь заходит о реальных, рабочих |

| | | |
|---|---|---|
| | | приложениях, Django получает самые теплые отзывы. С применением этого фреймворка созданы такие популярные приложения как Pinterest, Instagram и даже Disqus.<br><br>Django это универсальный фреймворк. Он может использоваться при создании любых сайтов, поскольку умеет доставлять веб-контент в различных форматах (например, JSON, XML, RSS). Также он дружественен к начинающим разработчикам. Даже если вы в Python еще новичок, с Django вы справитесь без всяких проблем. |
| 2 | Flask | Открытый фреймворк для разработки автономных приложений. Он использует шаблонизатор под названием Jinja, однако при желании пользователь может свободно выбрать другие движки. Flask обычно позиционируют как расширяемый фреймворк. В нём можно создавать API, сервисы RESTful и конечные точки. Также Flask применяют для написания серверной части приложений. |
| 3 | Web2py | Web2py это бесплатный full-stack фреймворк с открытым исходным кодом. Он предназначен для скоростной разработки быстрых, масштабируемых, безопасных и портируемых веб-приложений на основе баз данных. Гибкость Web2py просто невероятна. Но помимо этого Web2py еще и самый портируемый Python-фреймворк. Поскольку для аутентификации он использует LDAP, вам не нужно будет его устанавливать и настраивать. Его можно запускать прямо с флешки.<br><br>Web2py совместим с различными типами движков баз данных. В нем также есть встроенная система для управления ошибками. Но этот фреймворк может использоваться только с новейшими версиями Python. |
| 4 | Tornado | Tornado – это веб-фреймворк для создания веб-приложений на Python. Вместе с тем это и асинхронная сетевая библиотека, которая была изначально разработана для агрегатора FriendFeed. Используя неблокирующие вводы/выводы, Tornado может масштабироваться до десятков тысяч открытых соединений. Это делает его идеальным для «длинных опросов» (long polling), веб-сокетов и других приложений, требующих продолжительного подключения к каждому пользователю. Tornado это |

| | | |
|---|---|---|
| | | нечто среднее между Django и Flask. Если вы хотите создать приложение на фреймворке типа Django или Flask, но вам нужна повышенная производительность, выберите Tornado. При правильном создании архитектуры приложений он уверенно справится с десятками тысяч задач. |
| 5 | Pyramid | Pyramid – это полностью «open-source» каркас для создания Python-приложений. Его основная цель – сделать работу веб-разработчика проще. Он немного схож с еще одним фреймворком, а именно с Flask. Будучи совместимым с версией Python 3, он обеспечивает работу с NoSQL базами, включая MongoDB и CouchDB. Он простой и минималистичный. Быстрый и гибкий. Поддерживает огромное количество документации. Лучше всего он подходит для тех, кто занимается разработкой API, а также прототипированием и разработкой крупных веб-приложений, таких как CMS.<br><br>Самое главное, что он бесплатный и имеет открытый исходный код. |