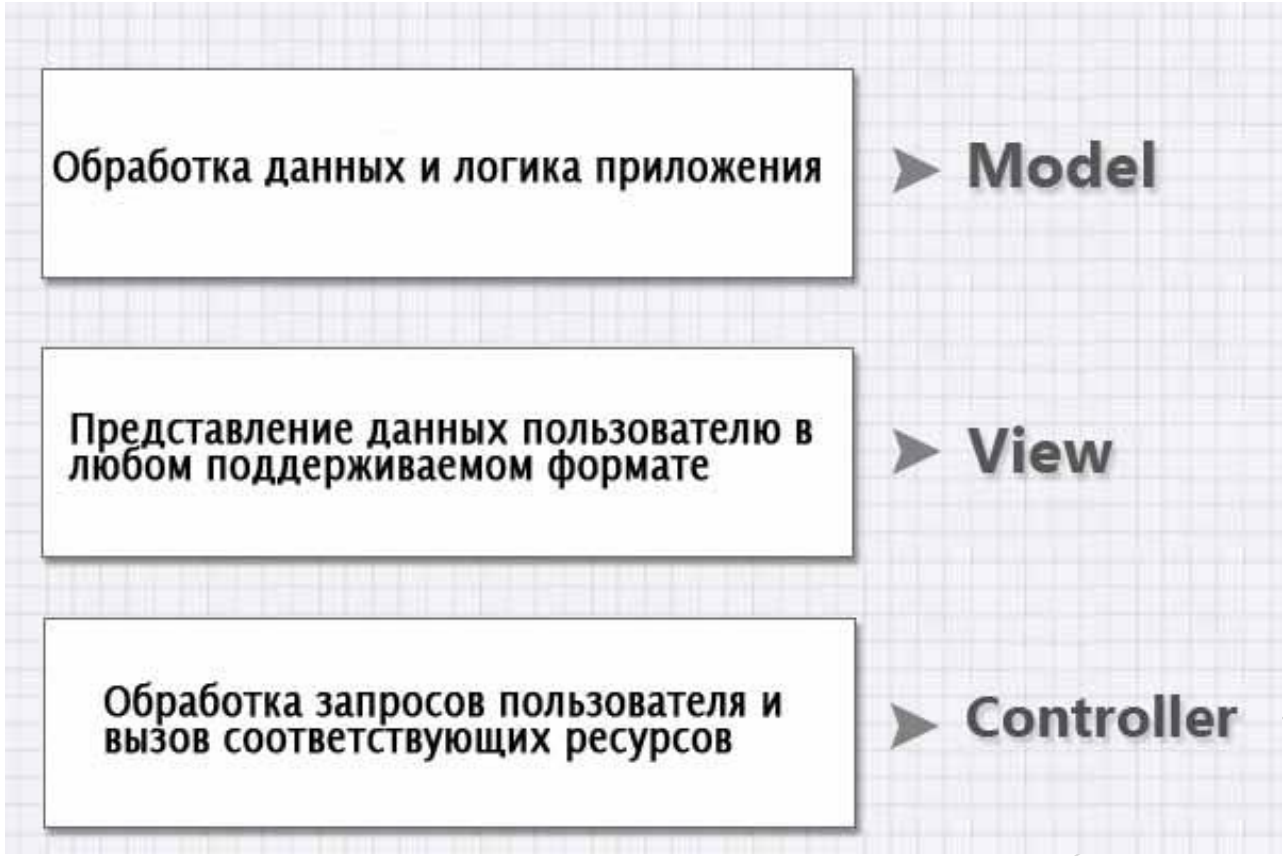
MVC-фреймворки

Шаблон MVC

MVC (model-view-controller) - это конструкционный шаблон, который описывает способ построения структуры приложения, сферы ответственности и взаимодействие каждой из частей в данной структуре.



Модель

Моделью называют постоянное хранилище данных, используемых во всей структуре. Она должна обеспечивать доступ к данным для их просмотра, отбора или записи.

- Модель содержит бизнес-логику приложения и включает методы выборки (это могут быть методы ORM), обработки (например, правила валидации) и предоставления конкретных данных.
- Модель не должна напрямую взаимодействовать с пользователем. Все переменные, относящиеся к запросу пользователя должны обрабатываться в контроллере.
- Модель не должна генерировать HTML или другой код отображения, который может изменяться в зависимости от нужд пользователя. Такой код должен обрабатываться в видах.

Вид

Вид используется для задания внешнего отображения данных, полученных из контроллера и модели.

- Виды содержат HTML-разметку и небольшие вставки PHP-кода для обхода, форматирования и отображения данных.
- Не должны напрямую обращаться к базе данных. Этим должны заниматься модели.
- Не должны работать с данными, полученными из запроса пользователя. Эту задачу должен выполнять контроллер.
- Может напрямую обращаться к свойствам и методам контроллера или моделей, для получения готовых к выводу данных.

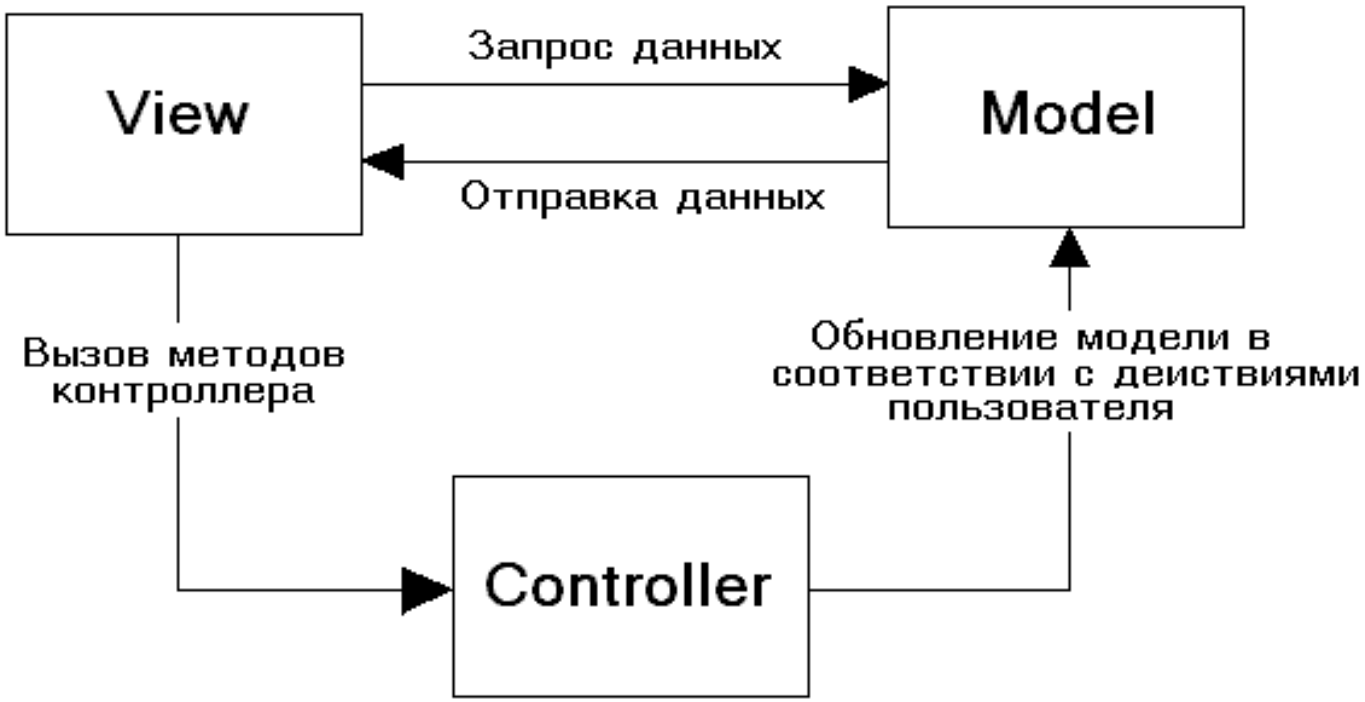
Контроллер

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

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.

- Контроллер не должен содержать HTML и другой разметки. Её стоит выносить в виды.
- Логика контроллера довольно типична и большая ее часть выносится в базовые классы.

Концептуальная схема MVC



- MVC модель предоставляет данные и правила бизнес-логики, представление отвечает за пользовательский интерфейс, а контроллер обеспечивает взаимодействие между моделью и представлением.

Последовательность работы MVC

- При заходе пользователя на веб-ресурс, скрипт инициализации создает экземпляр приложения и запускает его на выполнение. При этом отображается вид, скажем главной страницы сайта.
- Приложение получает запрос от пользователя и определяет запрошенные контроллер и действие. В случае главной страницы, выполняется действие по умолчанию (*index*).
- Приложение создает экземпляр контроллера и запускает метод действия, в котором, к примеру, содержаться вызовы модели, считывающие информацию из базы данных.
- После этого, действие формирует представление с данными, полученными из модели и выводит результат пользователю.

Front Controller и Page Controller

Page Controller	Front Controller
www.example.com/article.php?id=3 www.example.com/user.php?id=4	www.example.com/index.php?article=3 www.example.com/index.php?user=4

- Контроллер страниц хорошо применять для сайтов с достаточно простой логикой. В свою очередь, контроллер запросов объединяет все действия по обработке запросов в одном месте, что даёт ему дополнительные возможности, благодаря которым можно реализовать более трудные задачи, чем обычно решаются контроллером страниц.

Маршрутизация URL

- **Маршрутизация URL** позволяет настроить приложение на прием запросов с URL, которые не соответствуют реальным файлам приложения, а также использовать ЧПУ, которые семантически значимы для пользователей и предпочтительны для поисковой оптимизации.
- **Человекопонятный URL (ЧПУ, также семантический URL)** — URL-путь, состоящий из понятных слов, вместо идентификаторов, и отражающий файловую структуру сайта

Например, вместо /c14/3/97/ или /index.php?cat=10&subcat=2&id=41 будет /product/phone/Samsung/.

Маршрутизация URL

- Для обычной страницы, отображающей форму обратной связи, URL мог бы выглядеть так: http://www.example.com/contacts.php?action=feedback

```
switch($_GET['action'])
{
        case "about" :
                require_once("about.php"); // страница "О Нас"
                break;
        case "contacts" :
                require_once("contacts.php"); // страница "Контакты"
                break;
        case "feedback" :
                require_once("feedback.php"); // страница "Обратная связь"
                break;
        default :
                require_once("page404.php"); // страница "404"
        break;
}
```

Маршрутизация URL

- С использованием движка маршрутизации для отображения той же информации можно настроить приложение на прием таких запросов: ***http://www.example.com/contacts/feedback***
- Здесь contacts представляет собой контроллер, а feedback — это метод контроллера contacts, отображающий форму обратной связи и т.д.
- Также стоит знать, что маршрутизаторы многих веб-фреймворков позволяют создавать произвольные маршруты URL (указать, что означает каждая часть URL) и правила их обработки.

Веб-фреймворки

Фреймворк – это программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. Эта платформа подходит для создания сайтов, бизнес-приложений и веб-сервисов.

- Фреймворк отличается от библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подсистем близкой функциональности, не влияя на архитектуру основного программного продукта и не накладывая на неё никаких ограничений.

Веб-фреймворки

- Создание структуры при разработке на фреймворках очень упрощается. По сути, фреймворк — это множество конкретных и абстрактных классов, а также определений способов их взаимоотношения.
- Конкретные классы обычно реализуют взаимные отношения между классами, а абстрактные классы представляют собой точки расширения, в которых каркасы могут быть использованы или адаптированы.
- Для обеспечения расширения возможностей обычно используются техники объектно-ориентированного программирования (например, части приложения могут наследоваться от базовых классов фреймворка).

Веб-фреймворки: плюсы

- простота реализации бизнес-процессов помимо залоенных изначально в систему;
- простота масштабирования и модернизации;
- ускоряют процесс разработки;
- позволяют выдержать большую нагрузку;
- высокий уровень безопасности;
- помогают писать структурированный код, пригодный для повторного использования;
- позволяют легко масштабировать проекты;
- соблюдают схему MVC;
- поощряют современные практики разработки, например объектно-ориентированное программирование.

Веб-фреймворки: минусы

- сроки разработки типового функционала на фреймворках больше, чем при использовании CMS.
- фреймворки содержат только базовые компоненты бизнес-логики уровня приложения, поэтому многие функции реализовываются индивидуально.
- для разработки на фреймворке требуется понимание бизнес-процессов, которые требуется реализовать.

Веб-фреймворки: типы

У фреймворков есть две основные функции: работа на серверной стороне (бэкенд) и работа на клиентской стороне (фронтенд).

- Django — Python;
- Express.js — JavaScript;
- Ruby on Rails — Ruby.
- Angular;
- Ember.js;
- Vue.js.

Веб-фреймворки: архитектура

- Архитектура почти всех фреймворков основана на декомпозиции нескольких отдельных слоёв (приложения, модули и т.д.), что означает, что вы можете расширять функциональность исходя из своих потребностей и использовать изменённую версию вместе с кодом фреймворка или использовать сторонние приложения. Такая гибкость является ещё одним ключевым преимуществом фреймворков.

Особенности фреймворков

- **Кэширование** просто помогает хранить разные документы и позволяет избежать надоедливой перегрузки сервера. Пользователи могут использовать его в разных системах при определённых условиях. Также оно работает на серверной стороне.

Особенности фреймворков

- **Скаффолдинг:** это ещё одна технология, поддерживаемая некоторыми MVC-фреймворками, о которой следует знать. Фреймворк может автоматически сгенерировать типичные части приложения или даже всю струкутру проекта (если речь идёт о инициализации). Это позволяет существенно увеличить скорость разработки и стандартизирует кодовую базу.

m

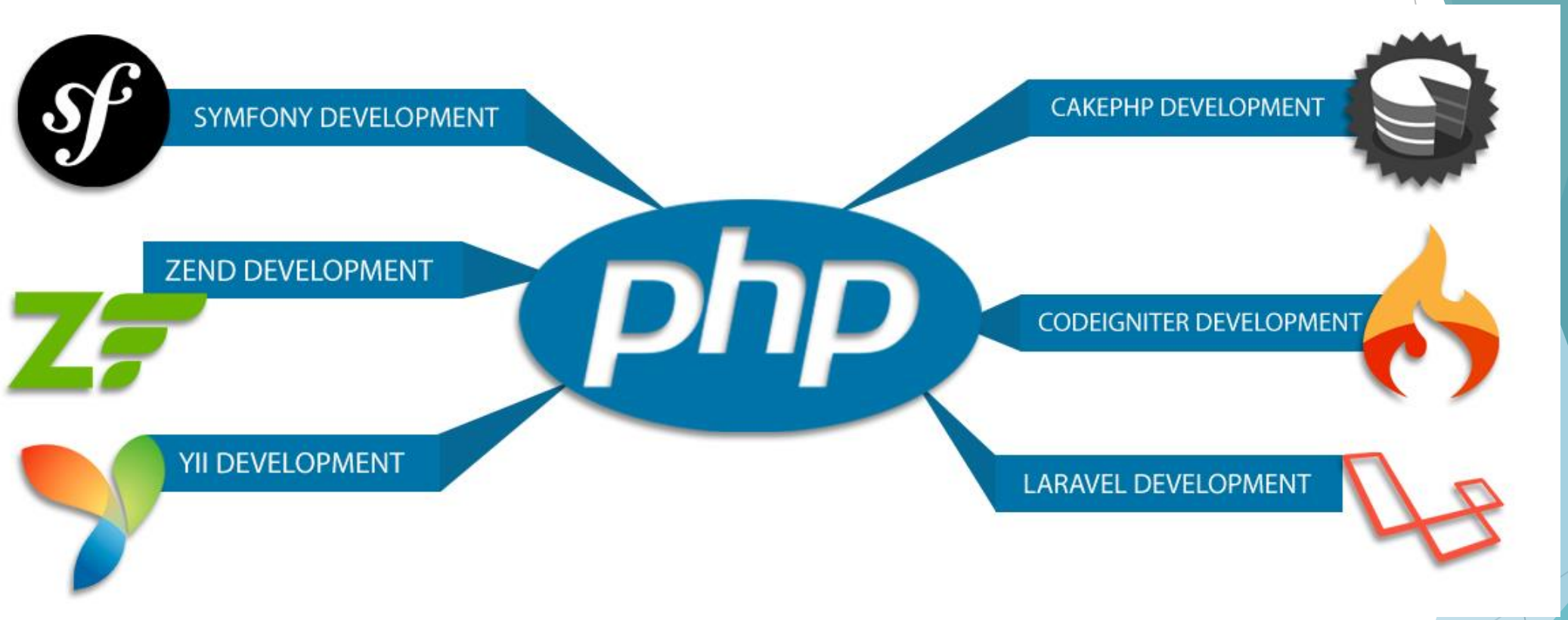
Особенности фреймворков

- **Система веб-шаблонов:** представляет собой набор разных методологий и программного обеспечения, реализованных для создания и развёртывания веб-страниц. Для обработки веб-шаблонов используются шаблонизаторы. Они являются инструментом фреймворка, отвечающим за веб-публикацию.

Особенности фреймворков

- **Сопоставление URL:** позволяет упростить индексацию вашего сайта поисковыми движками, в то же время создавая привлекательное название для сайта

PHP-фреймворки



PHP-фреймворки

	LARAVEL	SYMFONY 2	CODEIGNITER	PHPIXIE	YII 2	ZEND FRAMEWORK 2	CAKEPHP	PHALCON	SLIM	FUELPHP
ТРЕБУЕМАЯ ВЕРСИЯ PHP	5.5.9	5.5.9	5.4	5.3	5.1.0	5.3	5.5.9	5.3	5.5	5.3.3
ТИПЫ ПОДДЕРЖИВАЕМЫХ БАЗ ДАННЫХ	MySQL Postgress SQLite SQL Server	MySQL Postgress SQLite Oracle	MySQL Postgress	MySQL PostgreSQI SQLite MongoDB	MySQL Postgress Oracle Sqlite	MySQL SqLite SQL Server Oracle Postgress	MySQL SQLite Postgress SQL server Oracle	MySQL SQLite Postgress Oracle	MySQL SQLite Postgress Oracle (через плагин Slim-PDO)	MySQL SQLite Postgress
ОБЛАЧНОЕ ХРАНИЛИЩЕ	Amazon S3 Rackspace	Amazon S3 (плагин)	Amazon S3 (плагин)	-	Amazon S3	Amazon S3 Rackspace Windows Azure	Amazon S3 (плагин)	Amazon S3 (плагин)	-	Amazon S3 (плагин)
ДОКУМЕНТАЦИЯ НА ОФИЦИАЛЬНОМ САЙТЕ	Пошаговое руководство, справка по API, видеоуроки	Пошаговое руководство, справка по API	Небольшое пошаговое руководство, справка по API	Пошгговое руководство	Пошаговое руководство, справка по API	Пошаговое руководство, подробная справка по API с комментариями пользователей	Пошаговое руководство, справка по API	Пошаговое руководство, справка по API	Пошаговое руководство, справка по API	Пошаговое руководство
АВТОМАТИЧЕСКАЯ УСТАНОВКА РАСШИРЕНИЙ (ЧЕРЕЗ COMPOSER)	Да	Да	В ядре нет: только через сторонние библиотеки	Да	Да	Да	Да	Нет	Нет	Нет