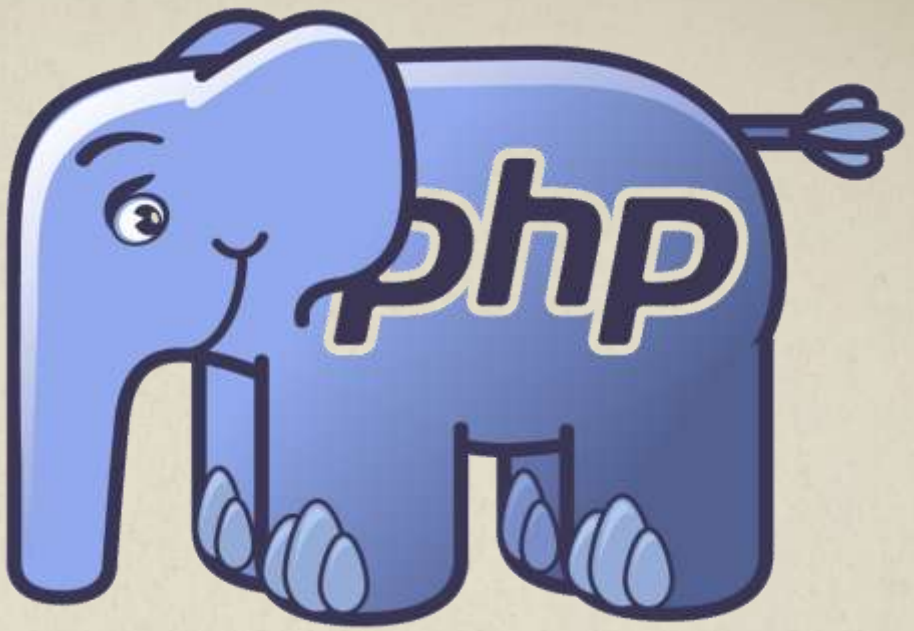


PHP: ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

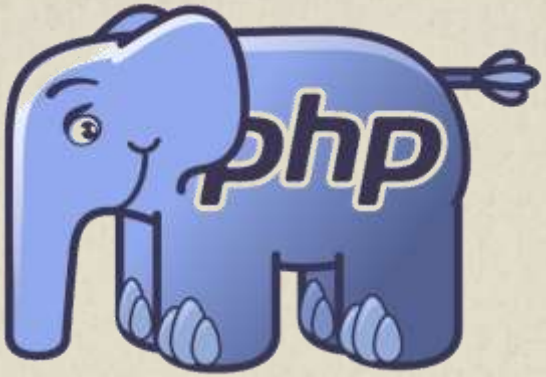
УРОК 8. ОБЗОР СОВРЕМЕННЫХ ФРЕЙМВОРКОВ.

ОБРАЩЕНИЕ К «СКЛАДЧИКАМ»

Я считаю складчины – полной ерундой. Невозможно научиться чему-то, просматривая видео. Без домашних заданий, без общения с преподавателями и коллегами. Покупая в складчину видеозаписи курсов, вы вредите прежде всего самим себе, создавая иллюзию «обучения». И поддерживаете каких-то мутных личностей-«организаторов», имеющих свой процент.

Впрочем, дело ваше.

Однако, если вы хотите по-настоящему учиться – приходите. Адрес есть на слайдах. Напишите в поддержку, мол «я складчик, но я хочу учиться». Скидку гарантирую ☺



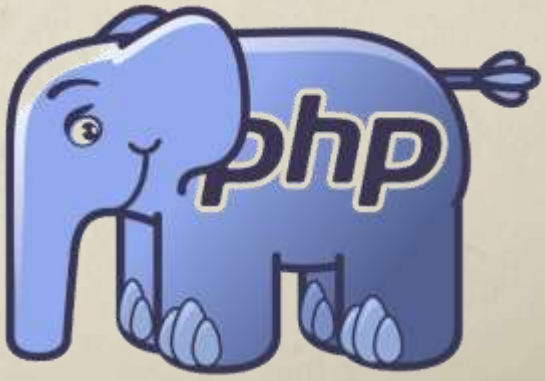
ЗАЧЕМ НАМ ФРЕЙМВОРКИ?

- Язык PHP бурно развивается. Революцию произвела версия 5.3, 5.4 и 5.5 сделали не меньше, а версия 7 в очередной раз всё перевернула!
- Появился единый инструмент управления пакетами и зависимостями – Composer
- Приняты отраслевые стандарты PSR
- Стало нормой тестирование кода

Всё это сделало возможным писать код «в промышленных масштабах»

- Надежный
- Стандартный
- Переиспользуемый

ФРЕЙМВОРКИ

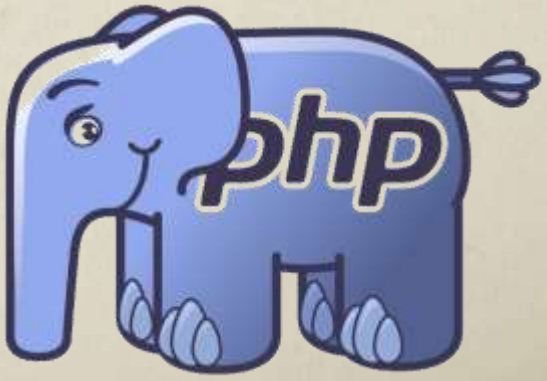


ВО-ПЕРВЫХ фреймворк – это набор инструментов. Которые уже написаны за вас. Не нужно изобретать велосипед и писать свои:

- Набор функций для работы с базой данных
- Правила валидации входящих данных
- Систему HTML-форм
- Подсистему интернационализации
- Классы конфигов разных форматов
- Консольные команды
- и многое, многое другое!

Всё уже написано до вас!

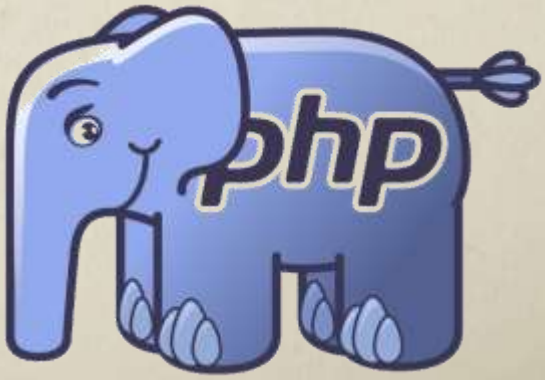
ФРЕЙМВОРКИ



ВО-ВТОРЫХ фреймворк – это набор методик и технологий:

- **MVC**
- Реализация паттерна Depedency Injection
- Работа с источниками данных на базе схем ORM и ActiveRecord
- Строгая система именования классов, пространств имен, компонентов приложения
- Часто – автоматическая генерация нужного вам «рутинного» кода
- и многое, многое другое!

Всё уже написано до вас!

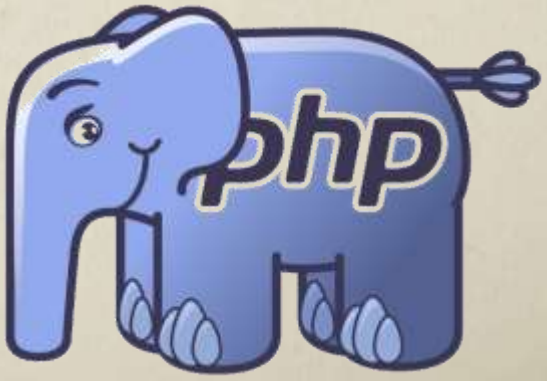


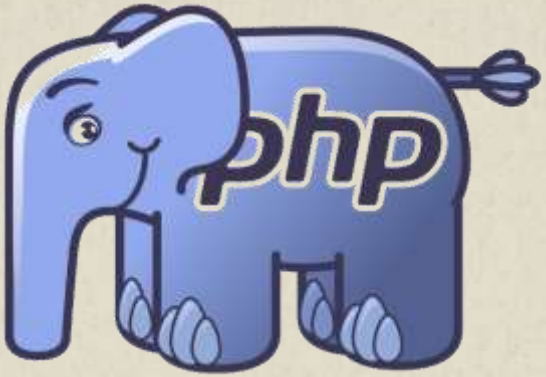
В-ТРЕТЬИХ фреймворк – это быстрый старт:

- Удобное развертывание
- Мгновенное создание начального веб-приложения
- Актуальная документация
- Сообщество разработчиков
- и многое, многое другое!

Так зачем изобретать велосипед?

ФРЕЙМВОРКИ





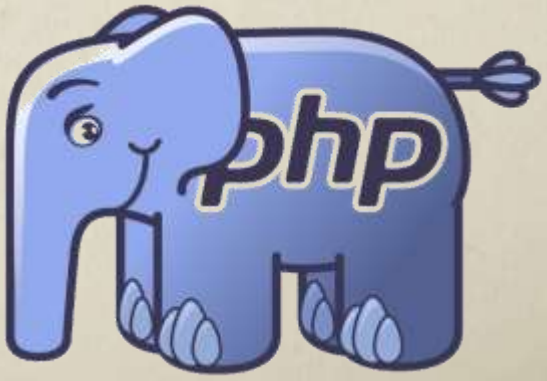
MVC

Быстрое создание каркаса приложения

T4	Yii 2	ZF 2	Symfony	Laravel
+	+	+	+	+

- **T4:**
```
composer create-project pr-of-it/t4-app-mini --stability="dev“
```
- **Zend Framework 2**
```
composer create-project --stability="dev" zendframework/skeleton-application
```
- **Laravel**
```
composer create-project laravel/laravel NAME 4.2.*
```

MVC

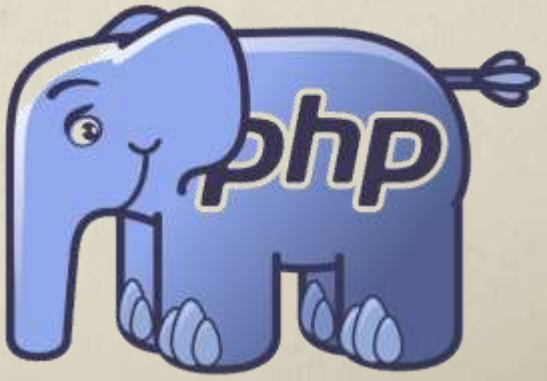


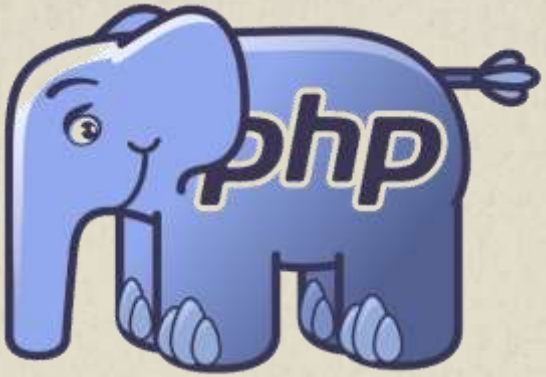
Реализация каркаса MVC в разных фреймворках

T4	Yii 2	ZF 2	Symfony	Laravel
+	+	+-	+	+

- Есть ли контроллер, от которого можно унаследовать свой?
- Если ли объект представления?
- Есть ли базовая модель?
- Возможно существует объект приложения?
- Роутер? Правила роутинга? Каскад правил?
- Что насчет сервисов? Часто они называются «компоненты приложения»?
- Как настраивается конфигурация приложения?
- Как устроены шаблоны?

MVC





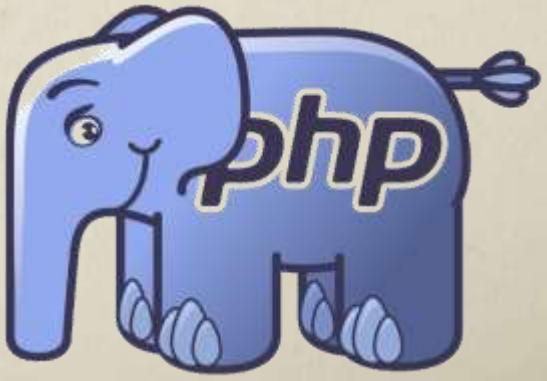
РАБОТА С БД
МОДЕЛИ

МИГРАЦИИ – способ управления структурой базы данных из приложения

T4	Yii 2	ZF 2	Symfony	Laravel
+	+	-	+-	+

- Как создать миграцию, есть ли генератор кода миграций?
- Могут ли миграции выполняться транзакционно? (зависит, конечно, от БД)
- Как накатить миграцию?
- Предусмотрен ли откат миграций?
- Какие методы предусмотрены для упрощения написания миграций?
- Возможно ли написать в миграции просто SQL?

МИГРАЦИИ

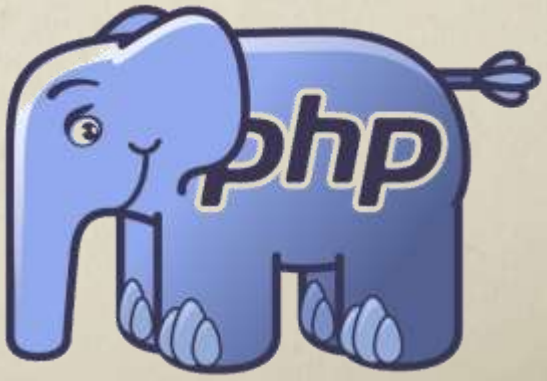


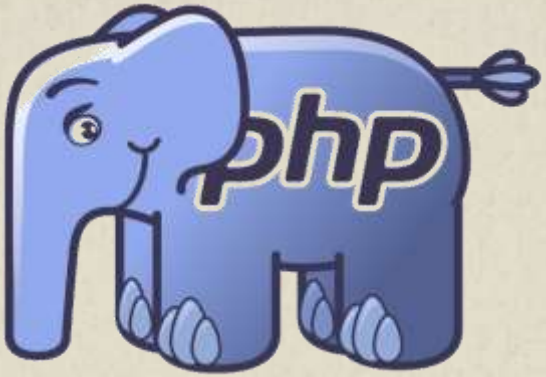
МОДЕЛИ – как реализация паттернов ORM и/или ActiveRecord

T4	Yii 2	ZF 2	Symfony	Laravel
+	+	-	+-	+

- Есть ли базовый класс «абстрактная модель», от которого можно унаследовать свои модели или иной механизм?
- **ORM:**
  - Получение из БД моделей нужного класса
  - Коллекции?
  - Связи и отношения?
  - Построение запросов к БД?
  - Различные хранилища данных (DBAL?) Паттерны хранения данных?
- **ActiveRecord:**
  - save() или аналоги (может быть insert() / update())
  - delete() или аналоги
  - статус синхронизации модели

МОДЕЛИ





ПРОЧЕЕ

КОНСОЛЬНЫЕ КОМАНДЫ – такая же важная часть вашего приложения, как и веб!

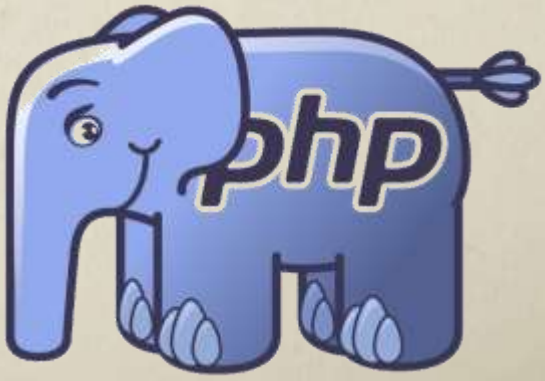
T4	Yii 2	ZF 2	Symfony	Laravel
+	+	+	+	+

- Как написать консольную команду?
- Как ее запустить?
- Можно ли составить справку по консольным командам?

Более серьезные возможности:

- Собственный менеджер расписания
- Очередь команд
- Многопроцессность и/или многопоточность

КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ

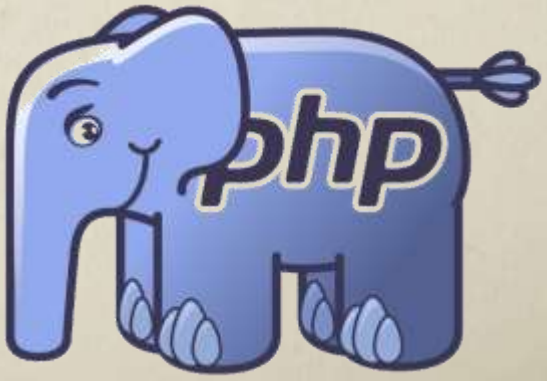


ВНЕДРЕНИЕ ЗАВИСИМОСТЕЙ – о сложном простыми словами!

T4	Yii 2	ZF 2	Symfony	Laravel
-	+	+	+	+

```
class Database {
     ...
}

class User {
  public function getUser($id) {
    $db = new Database;
    return $db->findOneUser($id);
  }
}

$user = new User;
$user->getUser(128500);
```

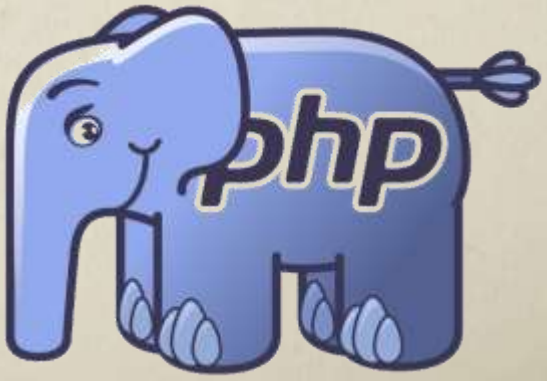
DEPEDENCY INJECTION



ВНЕДРЕНИЕ ЗАВИСИМОСТЕЙ – о сложном простыми словами!

```
class Database {
      ...
}

class User {

  protected $db;

  public function setDataBase($db) {
    $this->db = $db;
  }

  public function findUser($id) {
   return $this->db->findOneUser($id);
  }

}

$user = new User;
$user->setDataBase($db);
$user->getUser(128500);
```

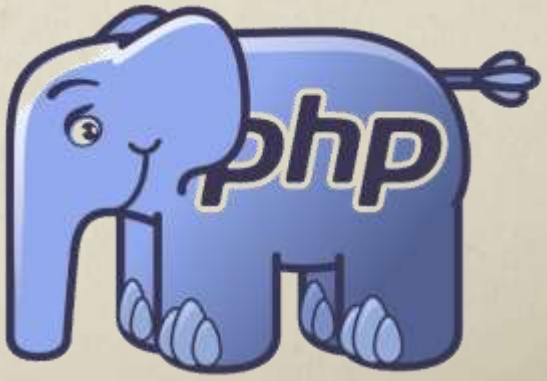
DEPEDENCY INJECTION



ВНЕДРЕНИЕ ЗАВИСИМОСТЕЙ – о сложном простыми словами!

```
class Database {
      ...
}

class User {

  protected $db;

  public function __construct($db) {
    $this->db = $db;
  }

  public function findUser($id) {
   return $this->db->findOneUser($id);
  }

}

$user = new User($db);
$user->getUser(128500);
```

DEPEDENCY INJECTION

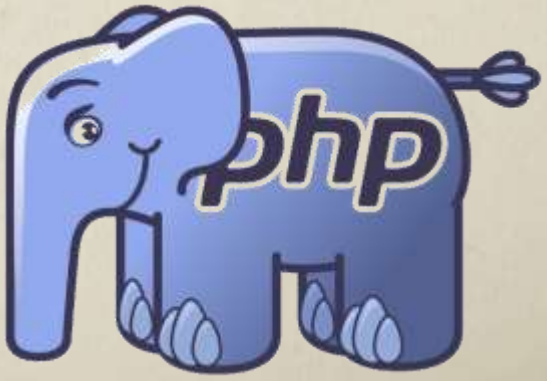


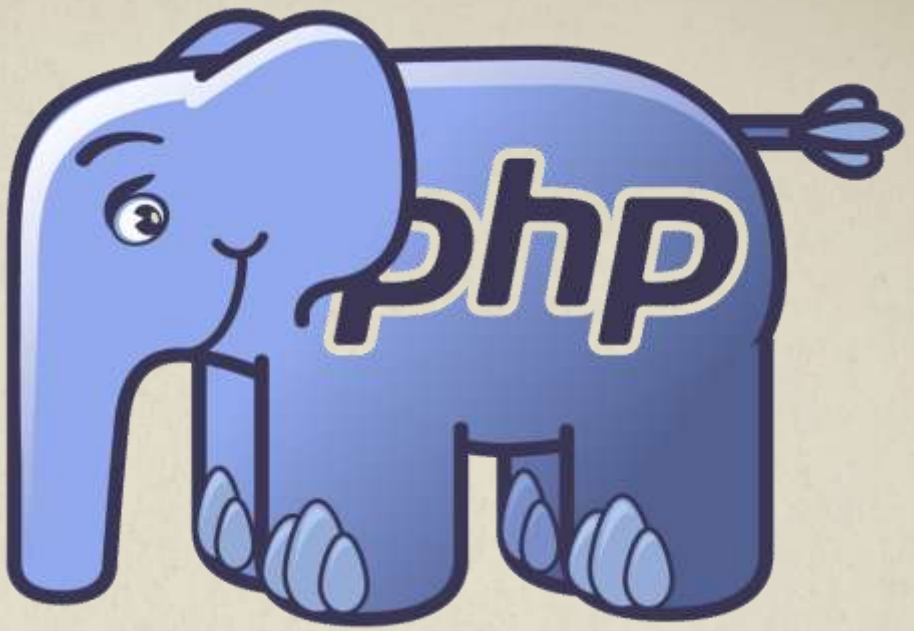
ВНЕДРЕНИЕ ЗАВИСИМОСТЕЙ – о сложном простыми словами!

```
class Container {

  public function getUser() {
    $db = new Database;
    $user = new User($db);
    return $user;
  }

}


$container
  ->getUser()
  ->findUser(128500);
```

T4	Yii 2	ZF 2	Symfony	Laravel
-	+	+	+	+

DEPEDENCY INJECTION





ИЗУЧАЙТЕ PHP, СТАНОВИТЕСЬ ПРОФЕССИОНАЛАМИ!

ДО ВСТРЕЧИ НА НОВЫХ КУРСАХ!

**ВИДЕОЗАПИСЬ, СЛАЙДЫ, ПРЕЗЕНТАЦИЯ
И ДОМАШНЕЕ ЗАДАНИЕ
БУДУТ ВЫЛОЖЕНЫ ДО 10 УТРА СЛЕДУЮЩЕГО ДНЯ**

ProfIT