GeekBrains

Laravel

Урок 1. Установка Laravel. Описание принципов работы и структуры фреймворка

Фреймворк Laravel и его преимущества. Установка и запуск. Настройка рабочего окружения для разработки приложения на Laravel. Создание мини-приложения.

Оглавление

Теория

Этот курс ориентирован на начинающих специалистов в области PHP-разработки, знакомых с PHP и ООП на базовом уровне. Мы рассмотрим шаблон проектирования MVC, работу и структуру фреймворка, разработаем приложение агрегатора данных. Данные для агрегатора могут быть любые: информация о перелетах, свободных местах в гостиницах, сведения со страниц соцсетей. Рассмотрим агрегацию новостных данных. Закрепляя знания в практических заданиях, вы научитесь работать с Laravel и создадите полноценный проект для портфолио.

Что такое фреймворк и зачем он нужен

Если разбить слово framework на две части, «frame» и «work», то получим абстракцию — «рамка» и «работа». По сути фреймворк — это и есть набор правил (ограничений, рамок), шаблон для разрабатываемого приложения.

Зачем нужны эти правила и шаблоны? Большинство сложных задач состоит из более мелких — ими уже десятки тысяч раз занимались другие разработчики, допуская и исправляя ошибки, и в итоге создавая почти идеальные подходы. К мелким задачам можно отнести:

- создание подключения к базе данных;
- валидацию полученной информации;
- авторизацию и аутентификацию;
- маршрутизацию;
- и другие.

Разработчики пришли к логичному выводу: не стоит каждый раз изобретать велосипед, если можно взять готовое решение. Для языков программирования начали появляться стандартизированные способы решения базовых задач — фреймворки.

Framework — набор инструментов, которые облегчают разработку и запуск приложения. Для каждого языка программирования их становится все больше.

Почему важно изучать работу фреймворков? Работодатели любят получать решения как можно скорее. Разрабатывая новый проект, программист может быстро запустить его с нуля, используя подходящий фреймворк. А еще можно привлекать новых разработчиков, знакомых с выбранным фреймворком, и тогда время они быстрее входят в курс дела. Концепция разработки в рамках фреймворка стандартизирована, поэтому для новобранца достаточно понять только идею проекта.

Почему Laravel? На данный момент Laravel — самый популярный фреймворк в России для PHP. Он постоянно развивается, у него большое сообщество и минимальный порог вхождения.

Окружение для Laravel. Homestead

Laravel — фреймворк PHP, следовательно для его работы потребуются как минимум веб-сервер и интерпретатор PHP. Для больших задач может понадобиться и дополнительное окружение. Его можно устанавливать по мере необходимости или сразу поставить наиболее востребованный набор программ для окружения. Если вы работаете на Windows, то таким набором станет OpenServer, для macOS — XAMPP (MAMP). Универсальный вариант — Docker с образом, содержащим lamp.

На курсе мы будем работать с Vagrant-бокс, а именно с Laravel Homestead. Данная коробка (сборка виртуальной машины) может быть запущена на любой операционной системе. Она содержит все

необходимое программное обеспечение, которое может потребоваться в ходе разработки приложения.

Перед Homestead следует установить следующие программное обеспечение:

- Git — система контроля версий. Позволяет отслеживать изменения, которые разработчик вносит в проект.
- Heidisql или DBeaver — визуальные клиенты для работы с базами данных sql.
- Virtualbox — программа для виртуализации.
- Vagrant — ПО для конфигурирования виртуальной среды.

Несколько слов о Composer: это менеджер зависимостей для PHP (аналог npm). Он уже установлен в Homestead. Задача Composer — следить за необходимыми пакетами для каждой библиотеки. Программисты — ленивые люди (и это правильно), поэтому в работе часто используют чужие наработки — а их пишут такие же ленивые программисты. Это приводит к тому, что в самом проекте и в библиотеке, которая подключена к нему, может использоваться еще и другое ПО. Composer определяет все необходимые библиотеки и скачивает их с указанного репозитория.

Vagrant

Vagrant («бродяга») — программный инструмент, который помогает:

- настроить сетевой мост до виртуальной машины;
- пробросить общую папку виртуальной машины и вашей ОС;
- настроить конфигурации веб-сервера;
- и многое другое.

Настраивают Vagrant при помощи конфигурационного файла, который читается при создании новой виртуальной машины или при специальном указании пользователя. Такие указания будем называть командами. Их список можно посмотреть тут: https://www.vagrantup.com/docs/cli/

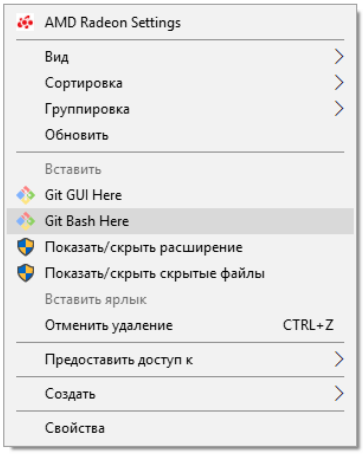
Практика

Установка Homestead

Далее пошагово рассмотрим установку Homestead на примере Windows. Перед выполнением данной инструкции следует установить:

- Git — https://git-scm.com/
- VirtualBox — https://www.virtualbox.org/
- Vagrant — https://www.vagrantup.com/
- HeidiSQL — https://www.heidisql.com/download.PHP

Далее откроем консоль git. Для этого следует щелкнуть левой кнопкой мыши и выбрать **Git Bash Here**:

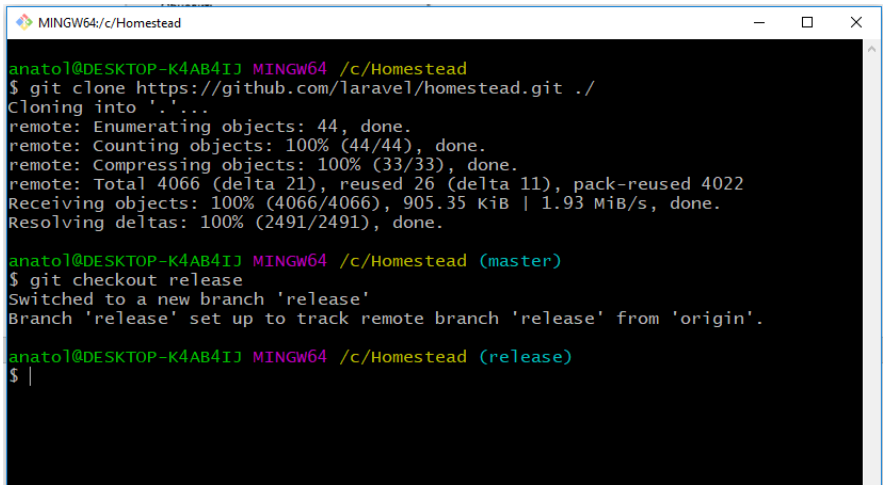


В консоли вводим: **`vagrant box add laravel/homestead`**

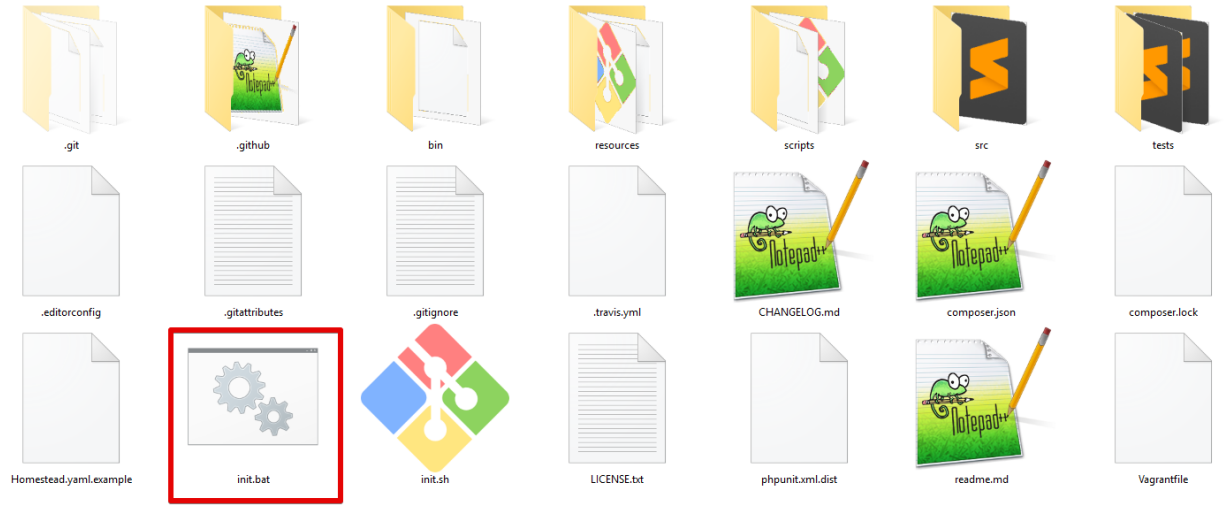
Данная команда загружает образ **homestead** на ваш компьютер. Для загрузки потребуется около 4 Gb памяти на жестком диске.

Дальше создаем папку, где будем хранить настройки виртуальной машины. Заходим в нее и опять открываем в ней консоль Git. Вводим в консоль команду: **`git clone https://github.com/laravel/homestead.git ./ `**

Скачав данные с репозитория, переходим в стабильную ветку — для этого вводим команду **`git checkout release `**



Создаем файл с настройками — **yaml**. Для этого запустим **` init.bat `**

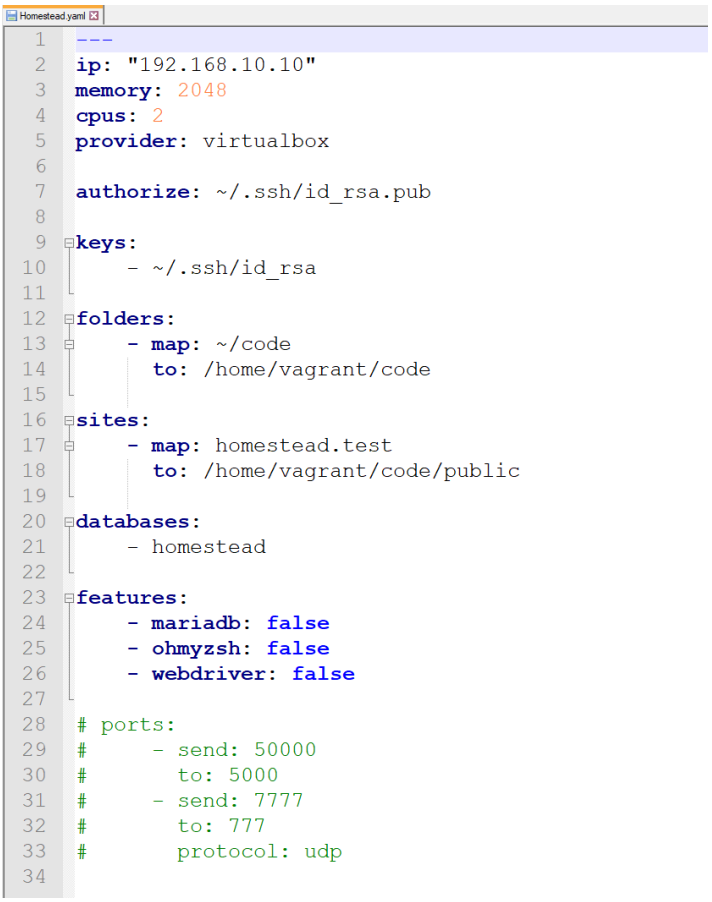


В результате запуска данного файла должен сгенерироваться файл ` **Homestead.yaml** `, который и будет отвечать за конфигурацию нашей виртуальной машины.

Если на вашем компьютере еще не сгенерированы ssh-ключи, то следует это сделать, выполнив команду в консоли: `**ssh-keygen -t rsa -C "your@mail.com"** ` Вместо your@mail.com следует указать свою почту. В результате выполнения данной команды в вашей домашней папке (папке пользователя) будет создана папка с открытым и приватным ключом ssh.

Настройка Homestead

Чтобы настроить виртуальную машину, откроем файл ` **Homestead.yaml** `:

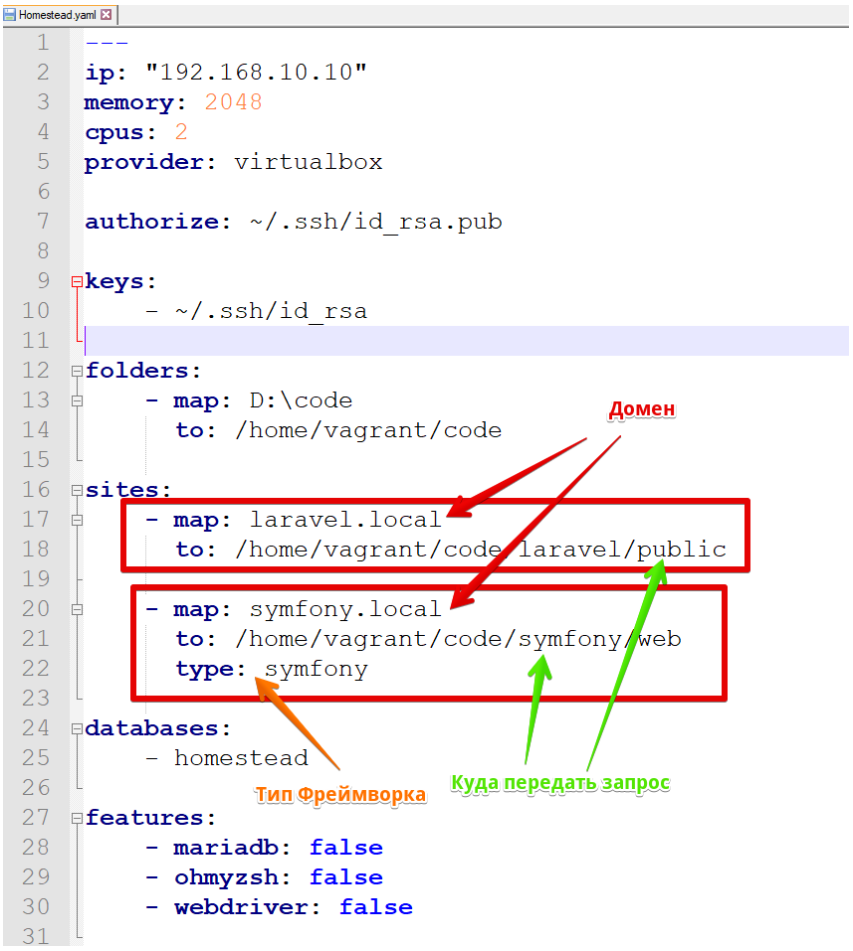


Конфигурации в данном файле написаны на языке разметки YAML. Он очень прост, а его структура отступозависима: изменяя отступы, вы меняете структуру документа. Не будем вдаваться в особенности, только внесем правки.

Сначала изменим символьную ссылку (мапинг) папки внутри виртуальной машины и нашей реальной ОС. Для этого в разделе **folders** вместо **~/code** установим ту папку, в которой будут храниться наши проекты. Для примера выберем **D:\code** (вы можете другую).

Затем укажем имя домена и путь, по которому должен направляться запрос (внутри виртуальной машины) при его использовании. Доменов может быть указано несколько (см. пример ниже).

Также для второго домена указан тип фреймворка. Это влияет только на тип nginx-конфига, который будет соответствовать этому домену. Для Laravel его указывать не нужно.



Сохранив данный конфигурационный файл, выполним в консоли команду `**vagrant up**`. Она «поднимет» виртуальную машину согласно сохраненному конфигу. Если конфигурацию нужно будет изменить — меняйте ее, сохраняйте изменения и дайте Vagrant перечитать его, введя команду `**vagrant provision**`

Установка Laravel

Чтобы подключиться к виртуальной машине, откроем консоль в директории, где хранятся настройки Homestead, и введем `**vagrant ssh**`

Перейдем в папку **code**, набрав команду **`cd ./code/`**

Теперь при помощи Composer установим Laravel. Для этого в указанной директории введем команду: **`composer create-project --prefer-dist laravel/laravel laravel`**

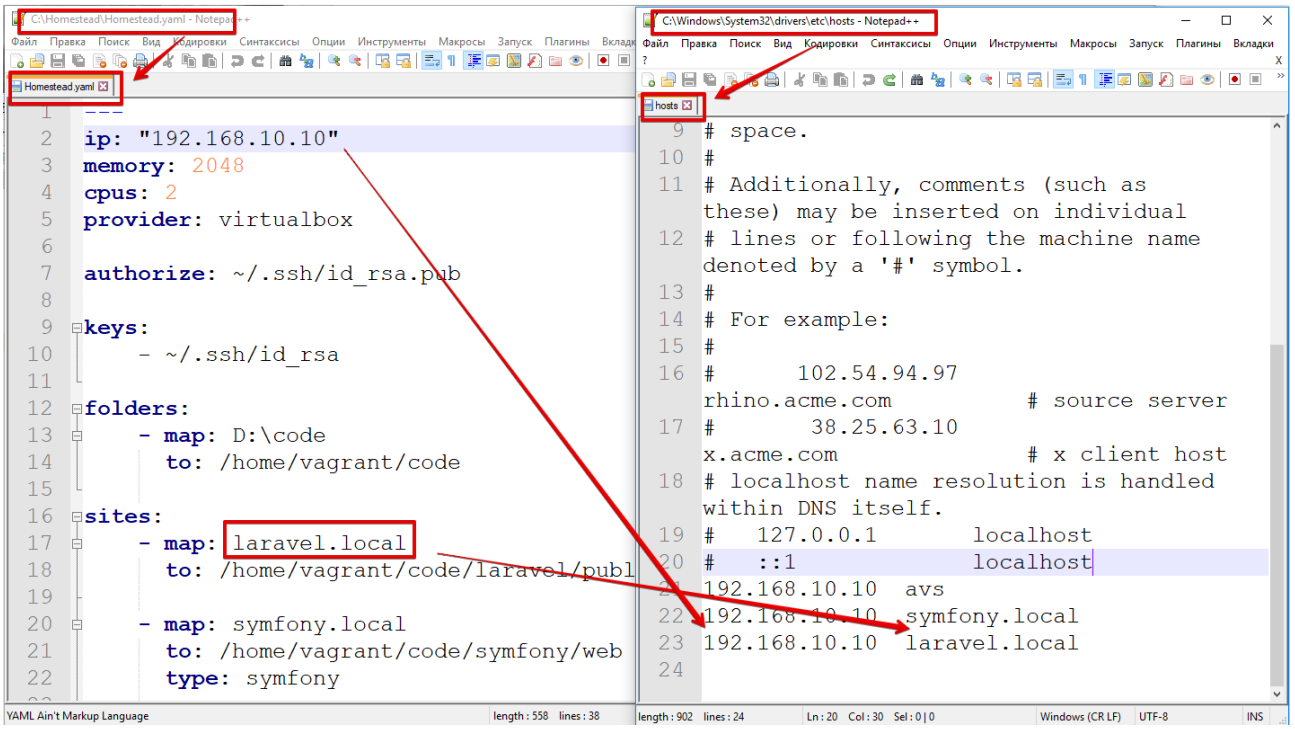
Последнее **laravel** в команде — имя создаваемой папки с фреймворком. Если вы укажете другое имя, следует внести изменения в конфигурационный файл и перечитать его, запустив команду **`vagrant provision`**

```
folders:
    - map: D:\code
      to: /home/vagrant/code

sites:
    - map: laravel.local
      to: /home/vagrant/code/laravel/public

    - map: symfony.local
      to: /home/vagrant/code/symfony/web
      type: symfony
```

После того как Composer скачает все необходимые зависимости и создаст проект, следует добавить в файл **C:\Windows\System32\drivers\etc\hosts** информацию, которая поможет вашему браузеру по введенному доменному имени (laravel.local) найти ваше приложение. Ниже на картинке показаны два файла — конфигурационные для homestead & hosts. В hosts следует добавить ip-адрес виртуальной машины и имя домена.



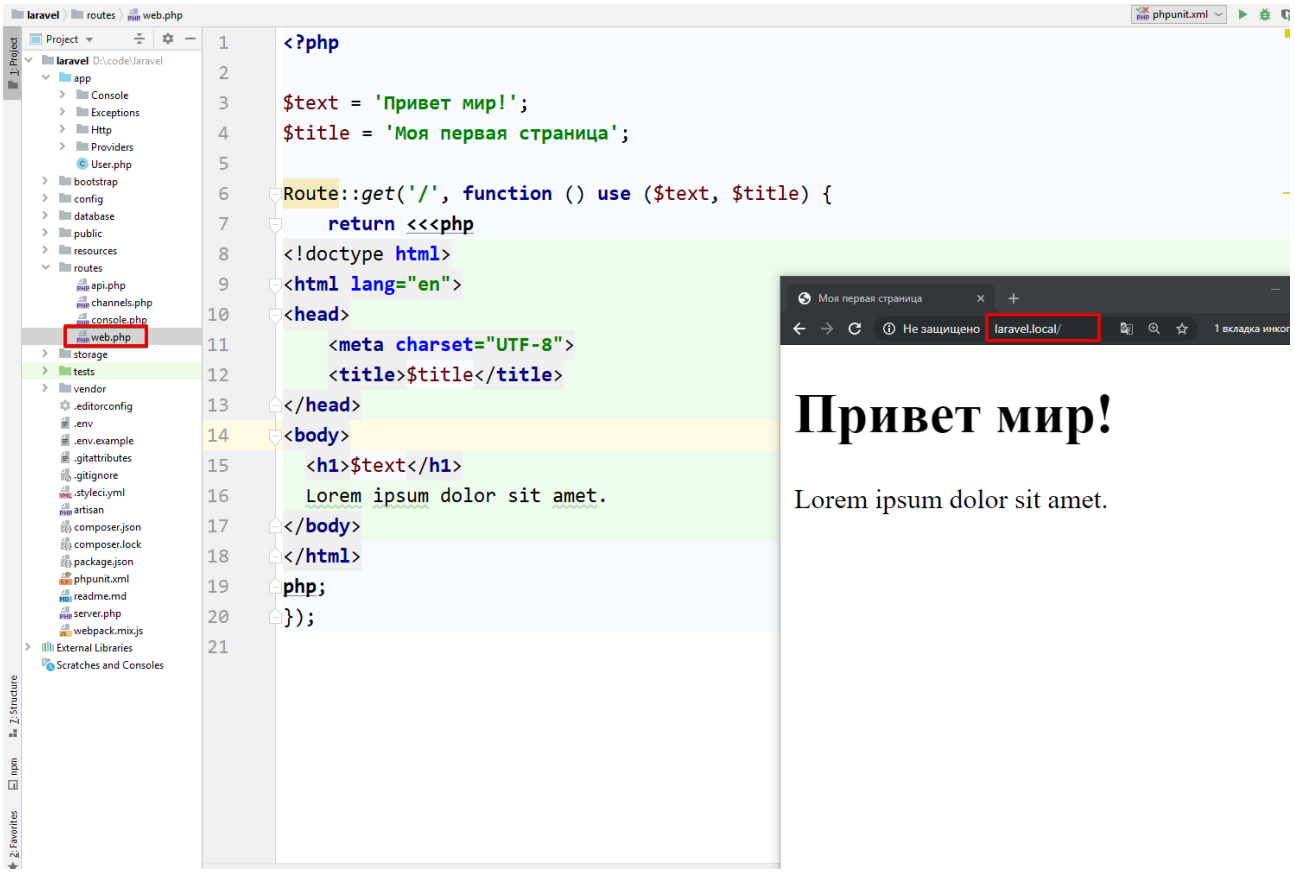
После этого перейдем в браузер, введем имя домена (laravel.local) и убедимся, что сайт доступен.

Laravel

DOCS LARACASTS NEWS BLOG
NOVA FORGE GITHUB

Добавление вывода данных

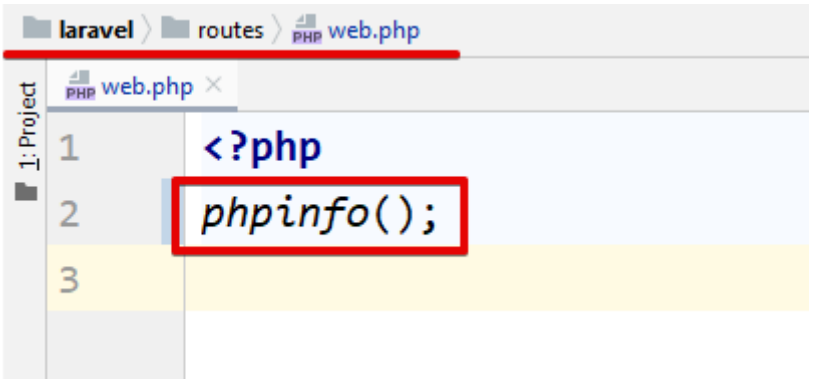
Рассмотрим на базовом уровне возможность вывода информации для страниц по указанным роутам. Для этого перейдем в файл **~/code/laravel/routes/web.PHP** и немного изменим его содержание.



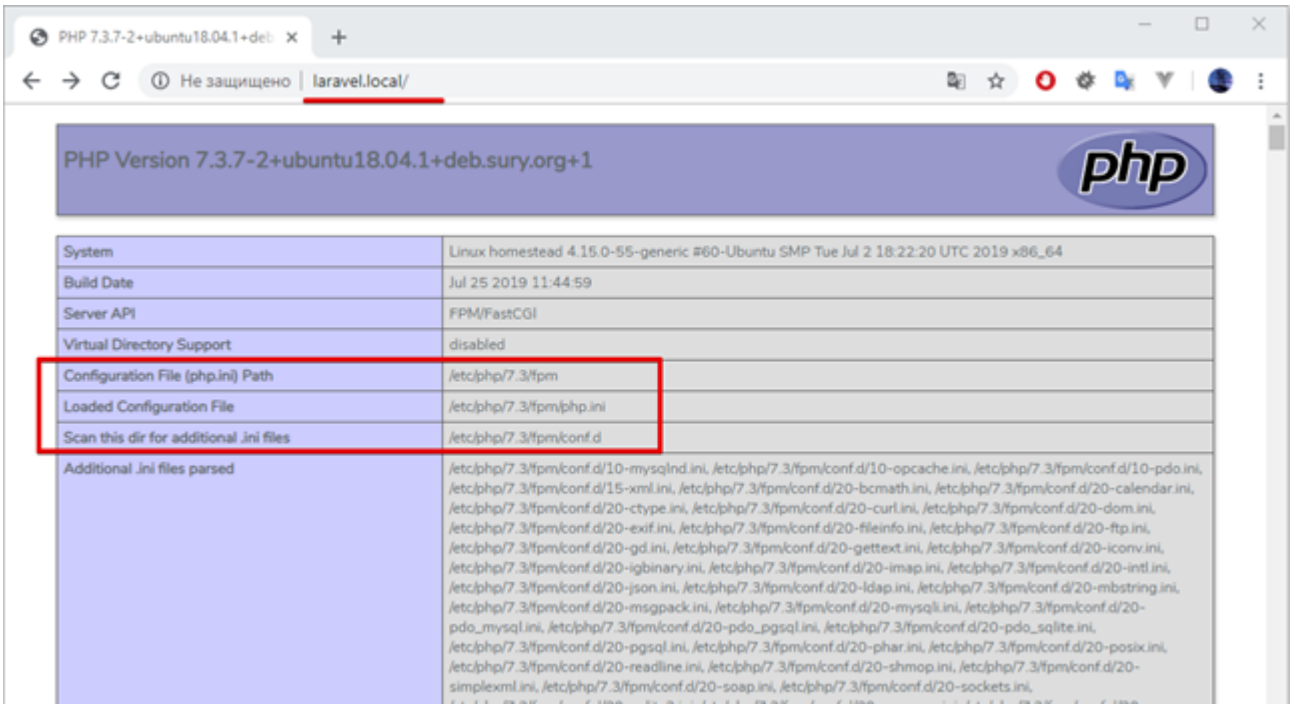
Настройка Xdebug в PHPStorm

Отладчик — средство для поиска ошибок и возможность разобраться в коде проекта. Чтобы использовать отладчик в нашем проекте, настроим его.

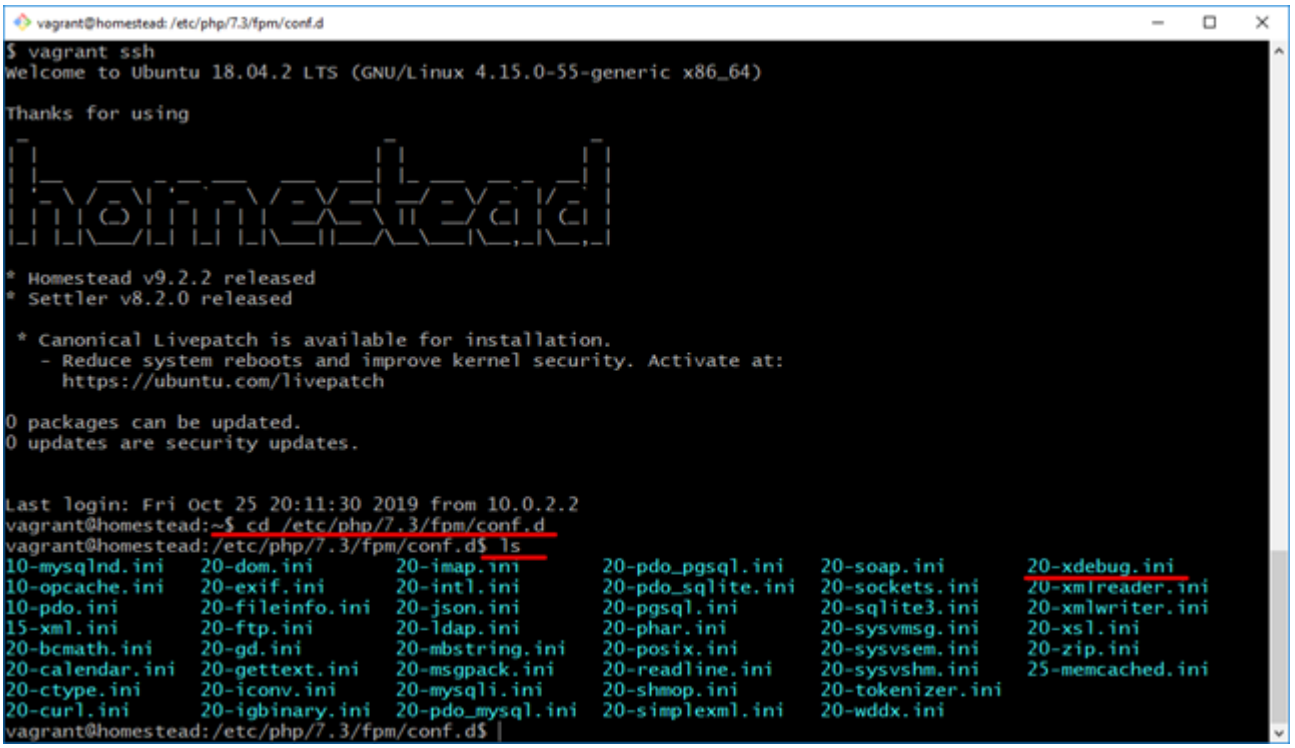
Сначала проверим, что для нашего интерпретатора PHP установлена библиотека для отладки **xdebug.so**. Для этого перейдем в файл **web.PHP** и удалим все, что в нем есть, а затем напишем единственную функцию **PHPinfo()**.



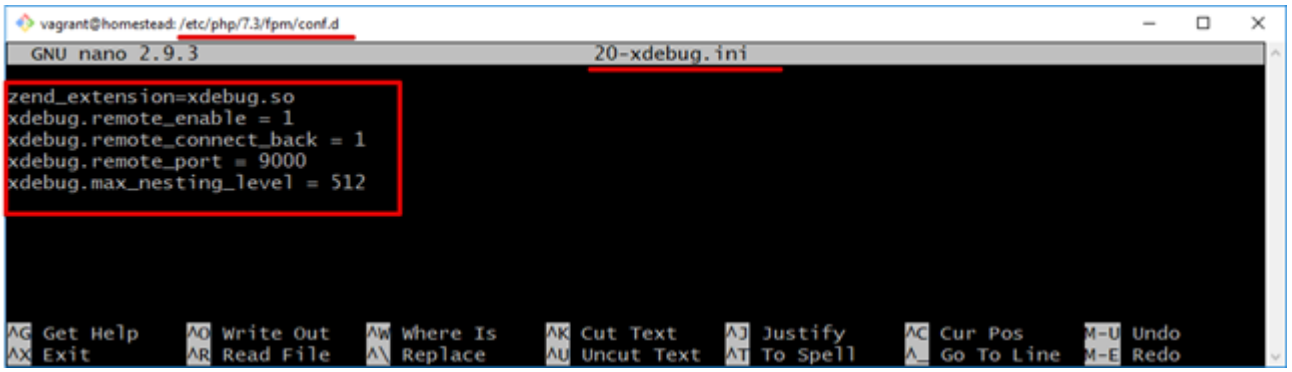
После этого перейдем в браузер. Данная функция выводит на экран все настройки используемого интерпретатора. В скриншоте ниже указано, откуда берутся данные настройки.



В данном случае нас интересует настройка **xdebug**. Откроем консоль и перейдем в папку, в которой хранятся файлы настройки интерпретатора.

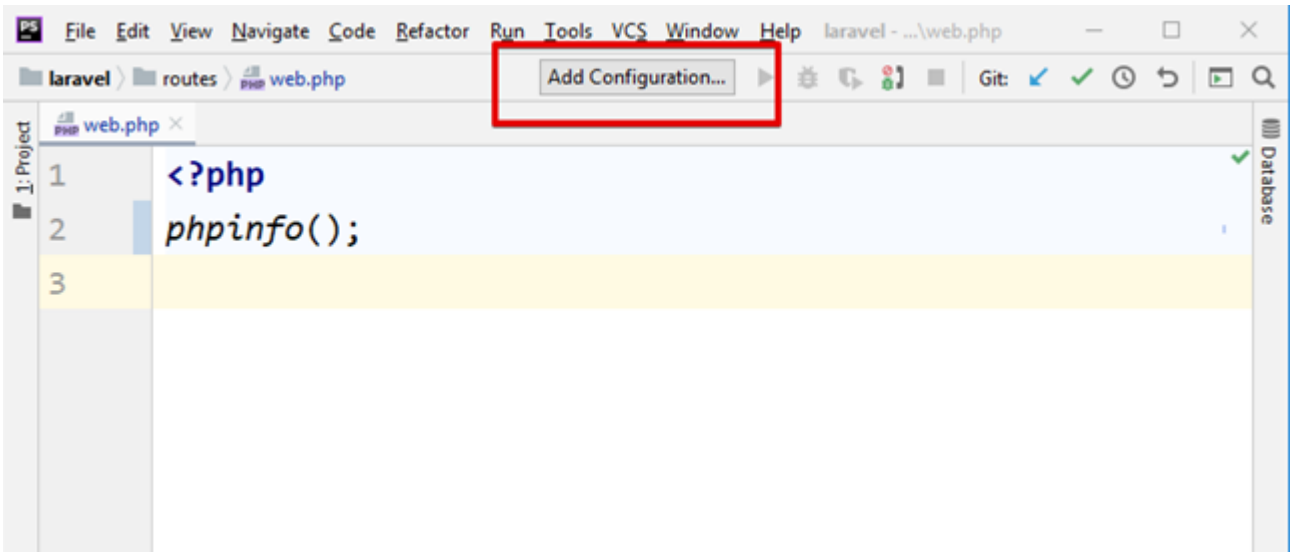


Посмотрим, какие настройки хранятся в файле `**/etc/PHP/7.3/fpm/conf.d/20-xdebug.ini**`.

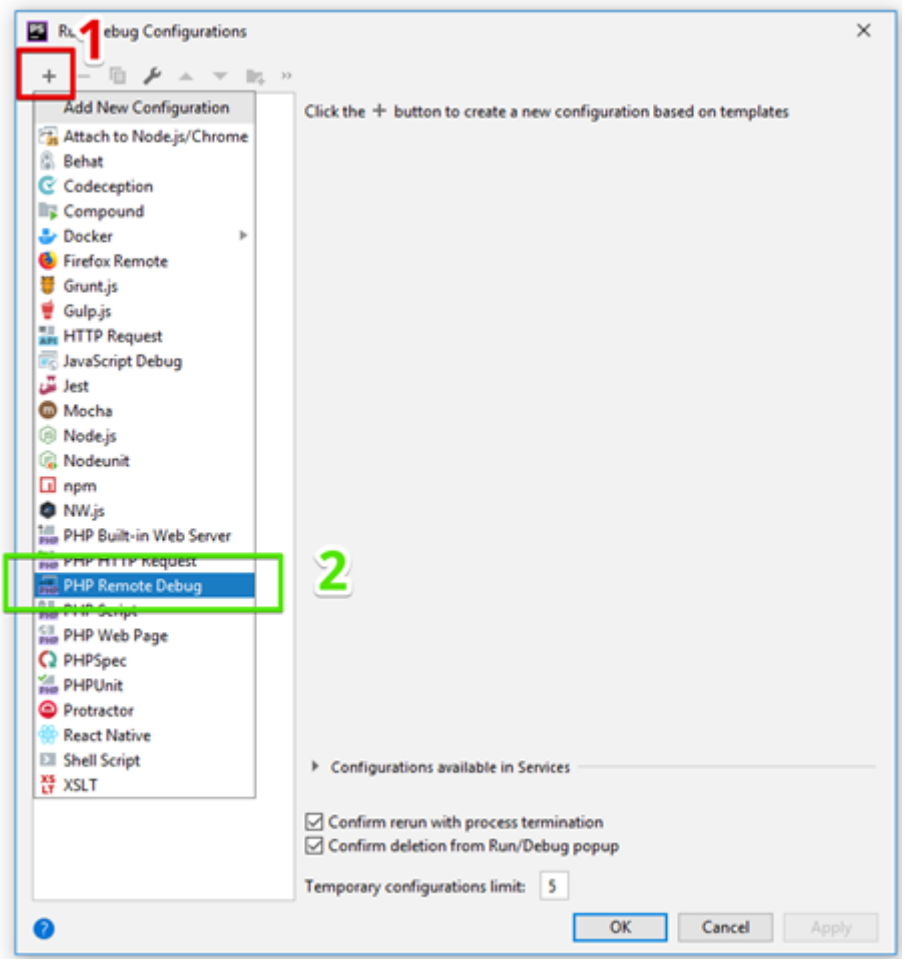


В данный момент в файле указано, какая библиотека будет использоваться для отладчика, что она включена и работает с портом 9000: это задано по умолчанию для настройки **xdebug**.

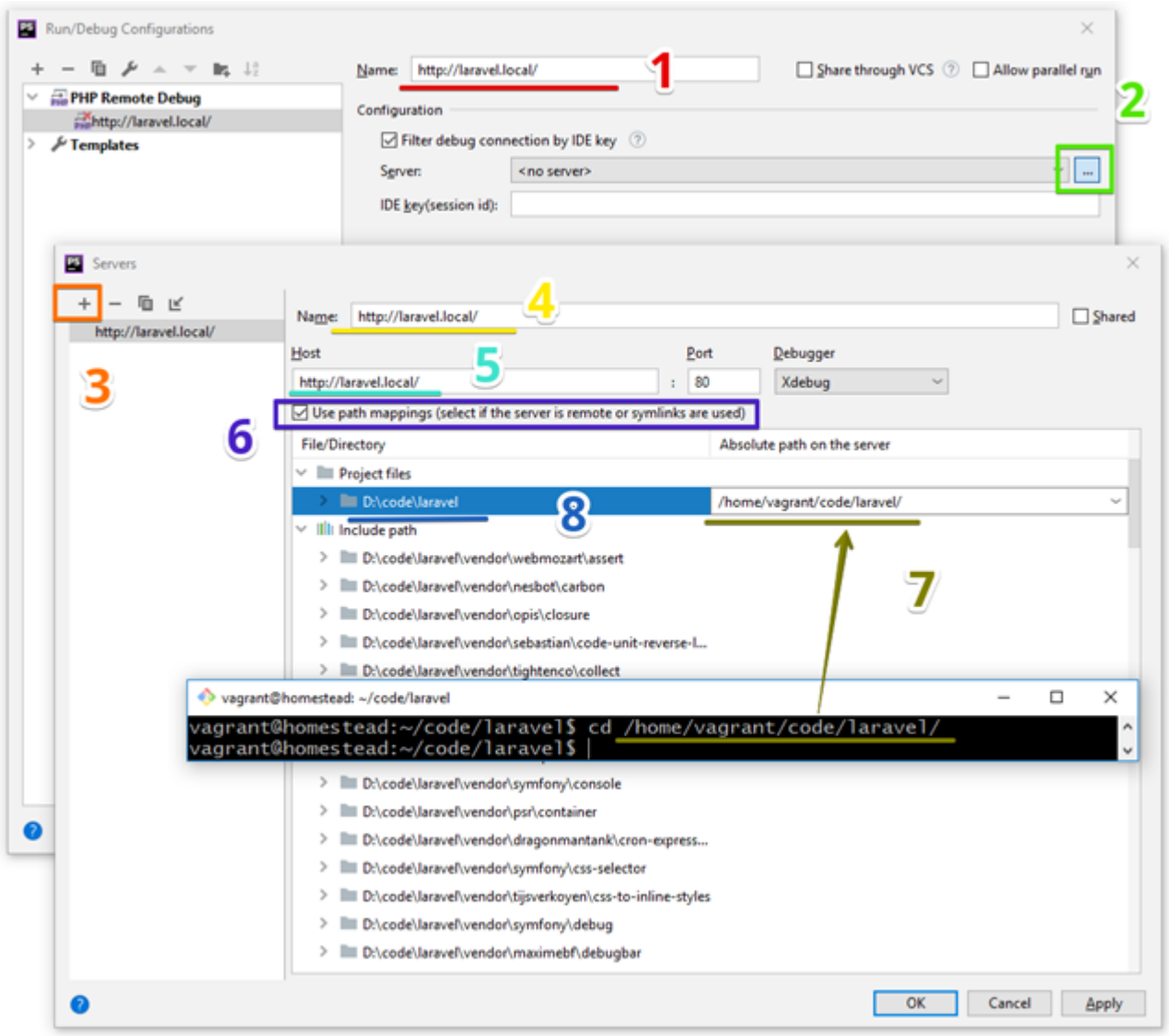
Далее в PHPStorm следует настроить конфигурации отладчика. Щелкнем по кнопке **Add Configuration**.



В открывшимся окне нажмем на **+** и в выпадающем списке выберем **PHP Remote Debug**.



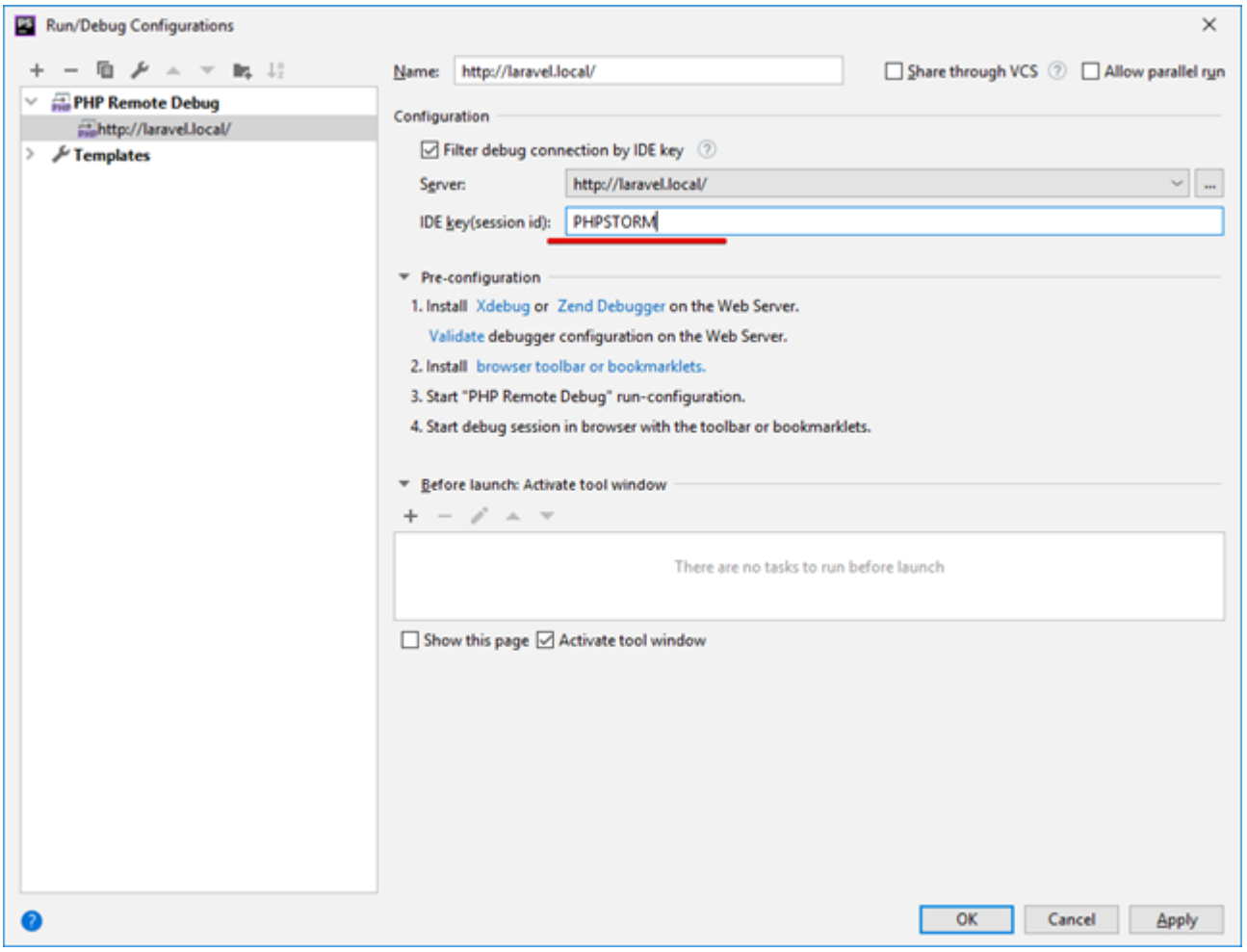
Далее действуем по цифрам на скриншоте ниже:



1. В этом поле надо указать имя отладчика.
2. Чтобы IDE понимала, откуда следует получать информацию, добавляем сервер, нажав на «…».
3. Жмем на **+** для добавления информации о новом сервере.
4. Вводим имя этого сервера.
5. Указываем хост, на который будут приходить обращения для запуска приложения во время отладки.
6. Устанавливаем галочку для того, чтобы указать, как правильно синхронизировать локальные файлы проекта и файлы на удаленном сервере.
7. Указываем директорию на удаленном сервере (на виртуальной машине homestead).
8. Директория локально размещенного проекта.

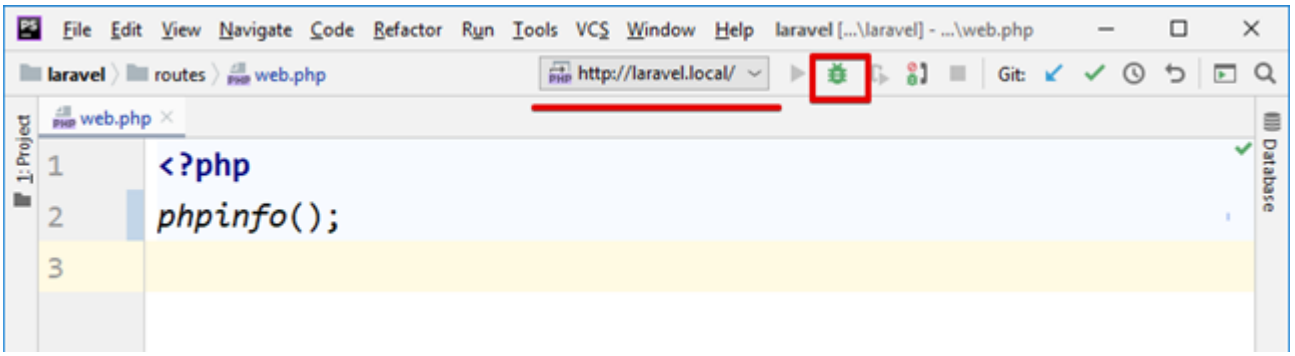
Жмем OK.

Перед тем как закрыть окно конфигураций, добавим ключ сессии для выполнения отладки.

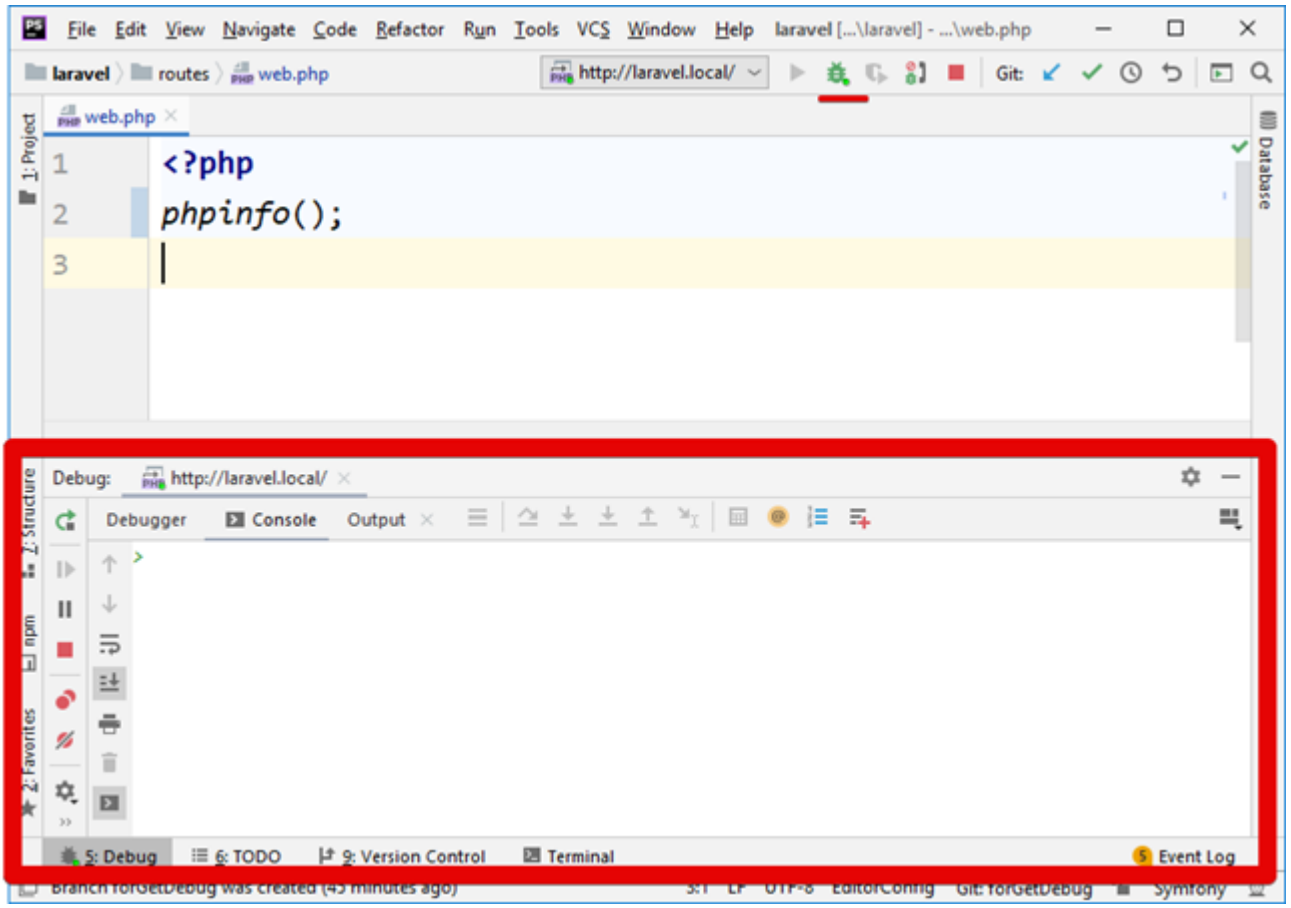


Жмем OK.

Далее запустим отладчик. Для этого нажмем на иконку жука.



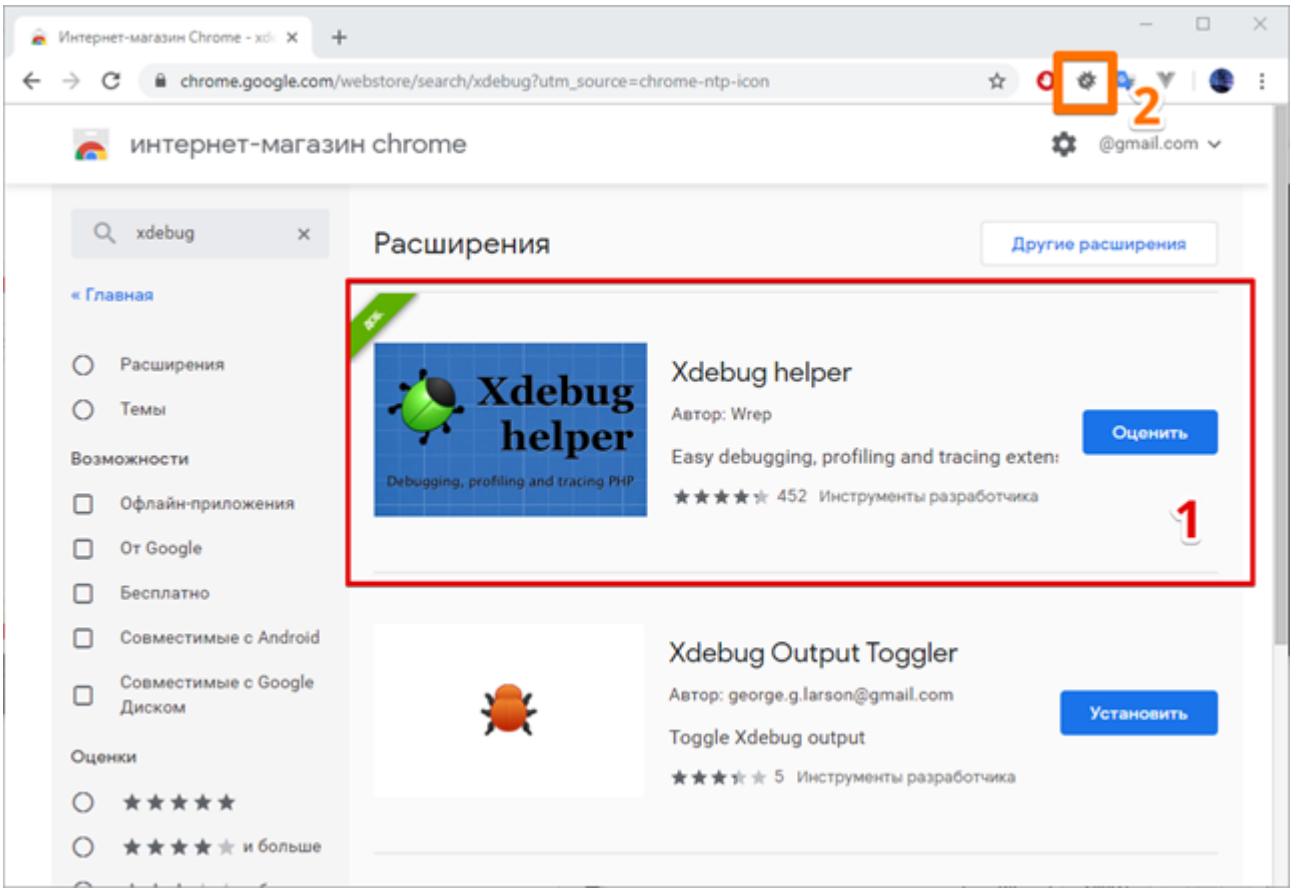
При запуске отладчика внизу IDE появляется отладочная панель — к ней вернемся позже.



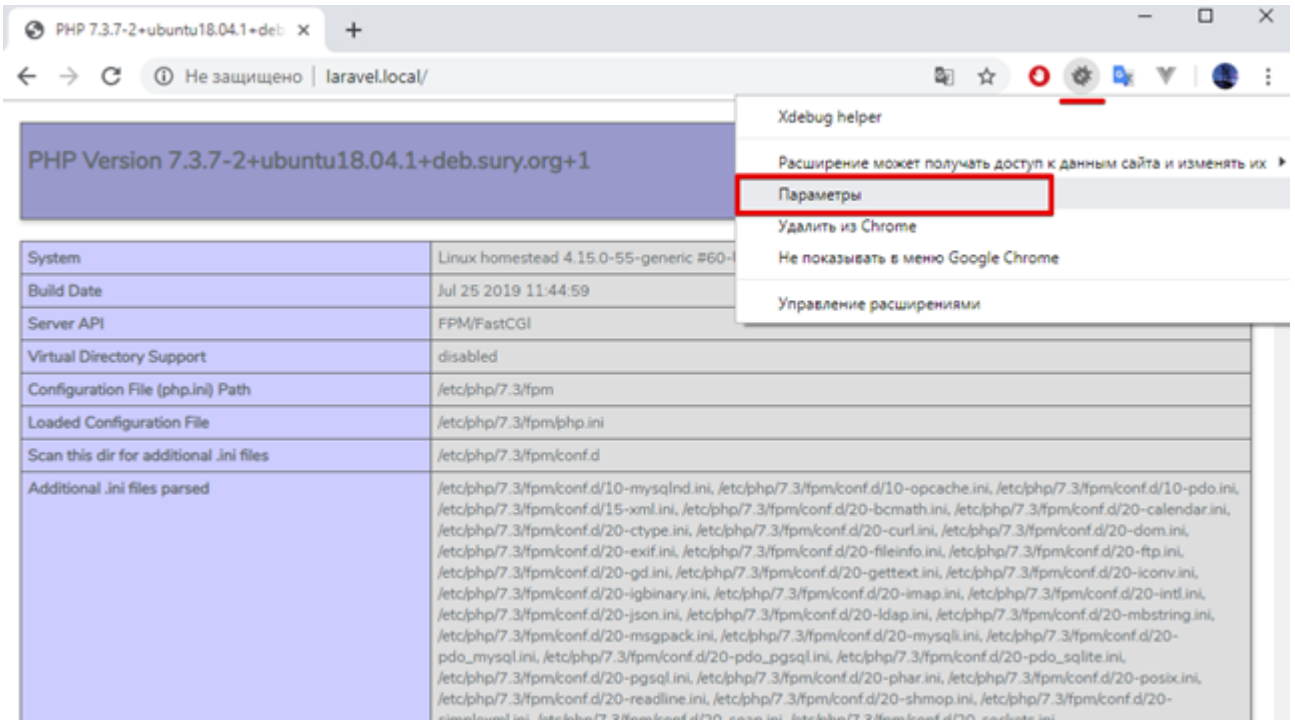
Чтобы интерпретатор понимал, что при запуске приложений следует работать с отладчиком, вместе с запросом от клиента должна быть отправлена cookie **XDEBUG_SESSION=PHPSTORM**. Для этого удобнее всего использовать специальное приложение, которое будет отключать и выключать отправку данных заголовков.

В нашем примере используется браузер Chrome и приложение **xdebug helper**. Подобные приложения есть и для других браузеров.

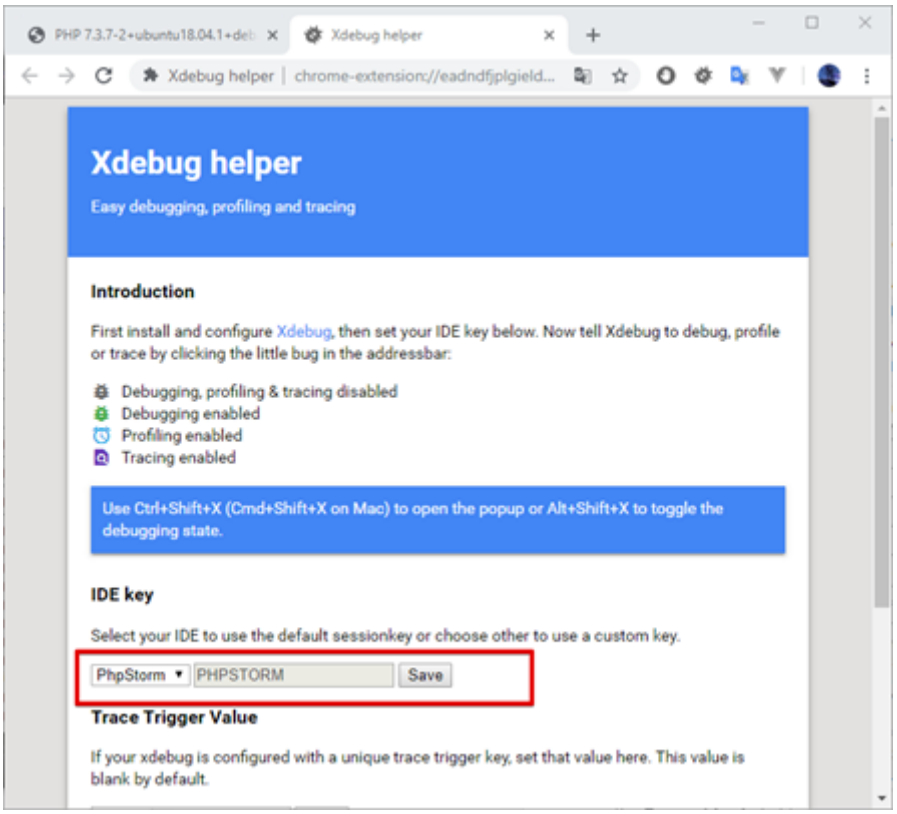
Найдите в интернет-магазин Chrome это приложение и установите его — после этого в окне браузера появится иконка в виде жука.



По умолчанию для всех сайтов это приложение выключено. Но прежде чем включить, немного настроим его. Для этого щелкнем по иконке жука правой кнопкой мыши и выберем «Параметры».

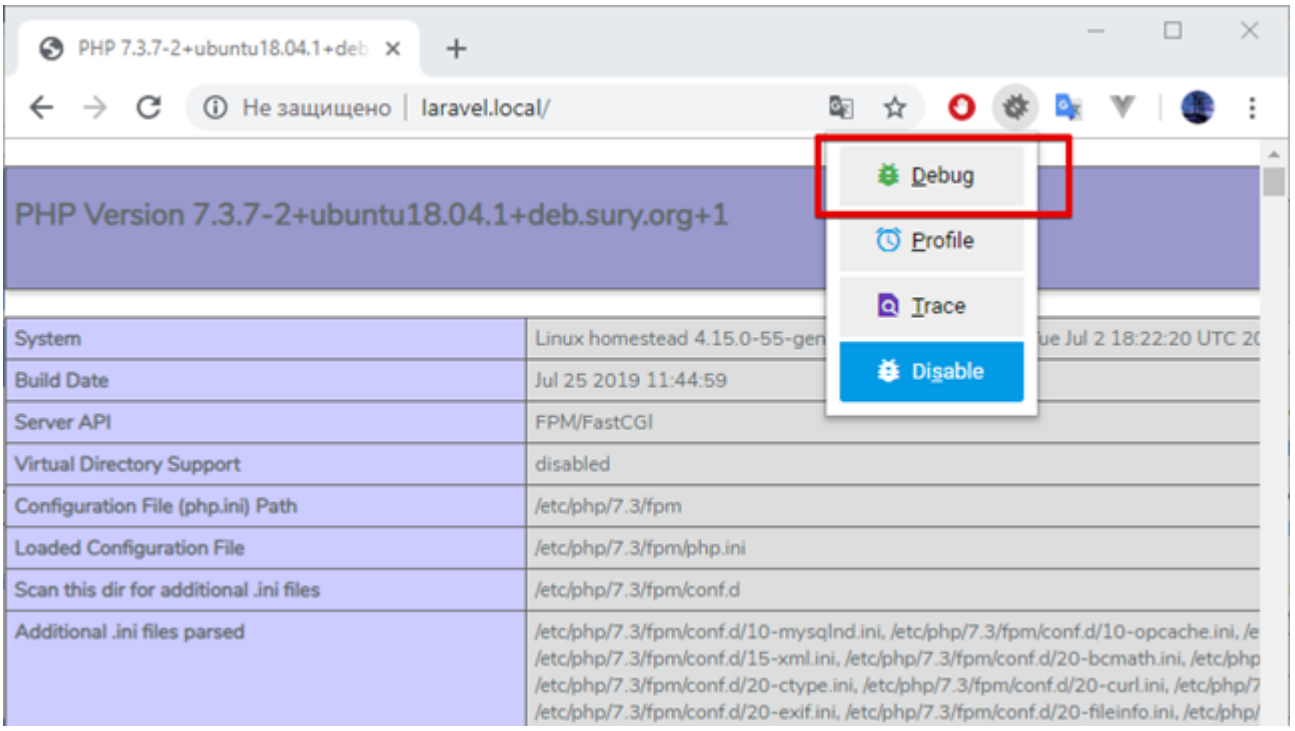


В открывшейся вкладке выберем I**DE key — PHPSTORM**. Эта настройка определяет значение cookie, которое будет отправляться для работы отладчика.

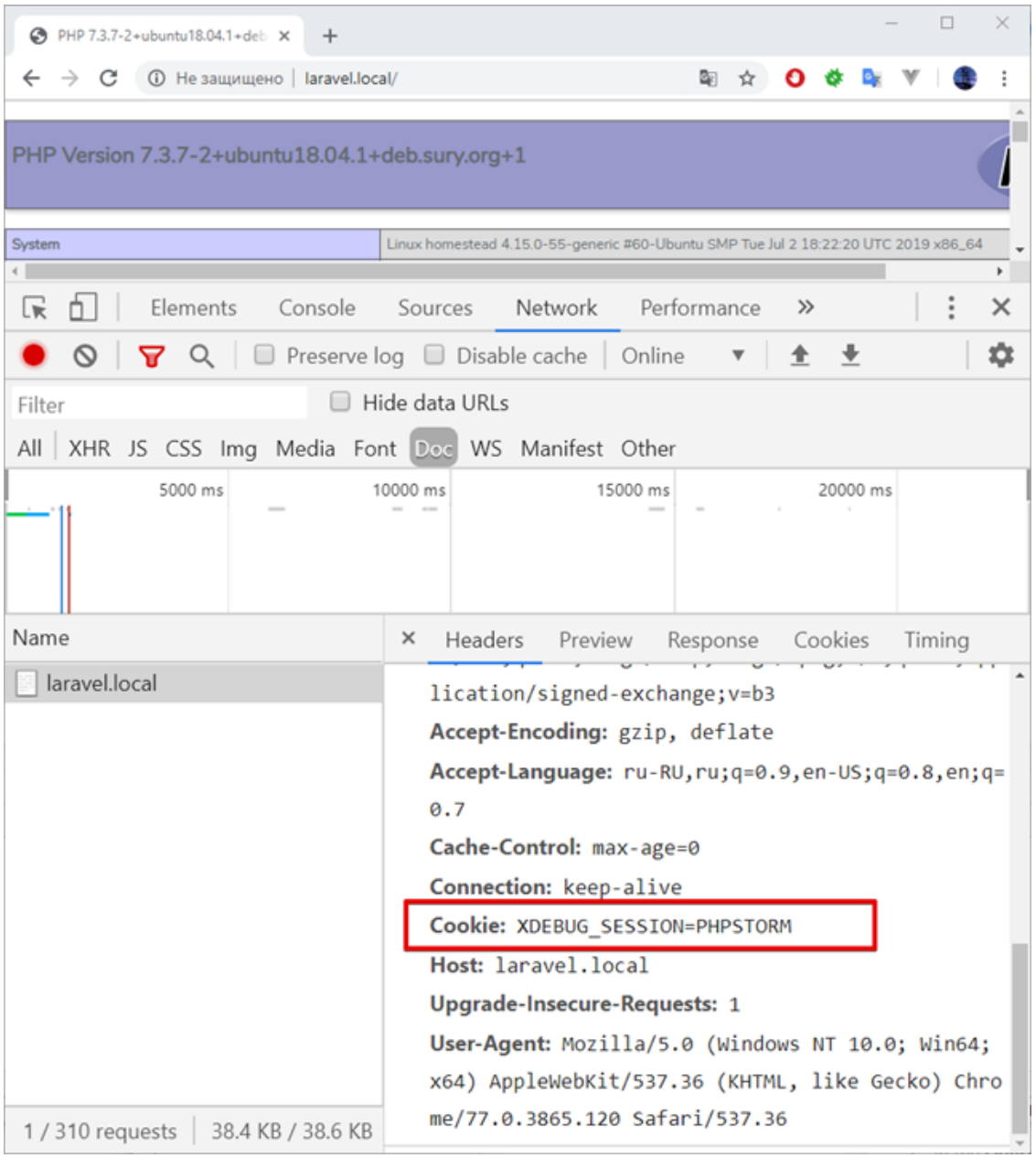


Жмем **Save**.

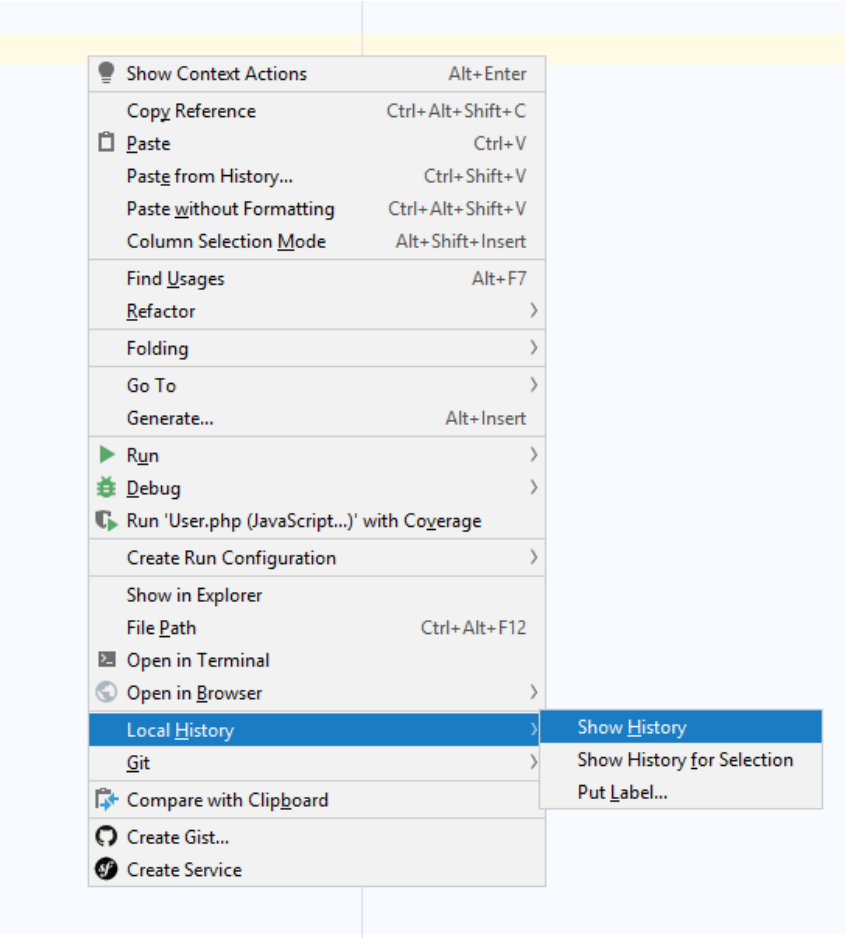
Теперь включим отправку данной cookie, для этого щелкнем на значке **helper xdebug** левой кнопкой мыши и выберем **Debug**.



Перезапустим страницу и убедимся, что нужная cookie действительно установлена.



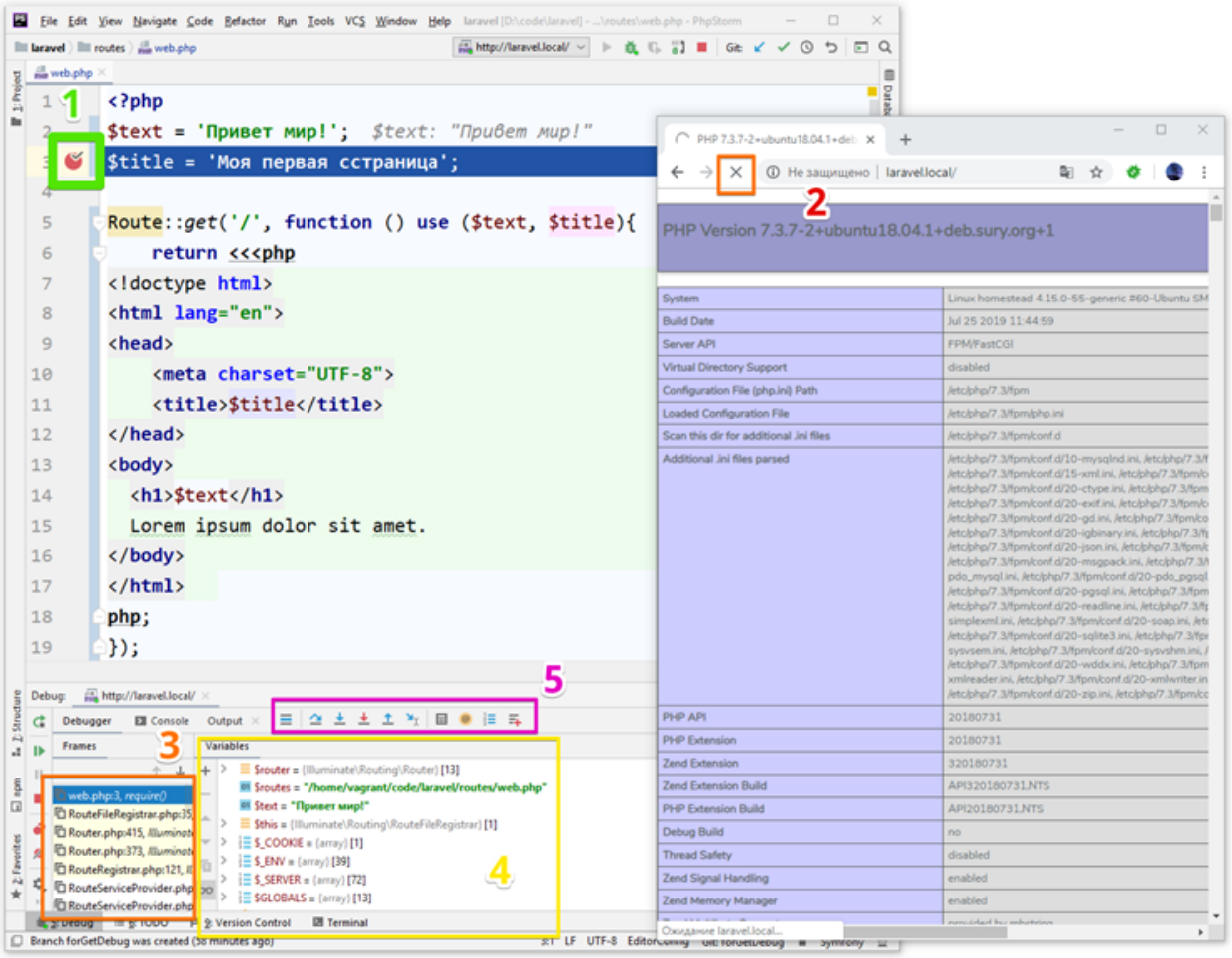
Теперь вернем код, что он был в файле **web.PHP**. Для этого щелкнем правой кнопкой мыши в документе и выберем **Local History — Show History**.



В открывшемся окне выбираем необходимое состояние файла и закрываем его.

Установим точку остановки отладки (break point), щелкнув указателем мыши чуть правее цифры нужной строки. Должна появиться красная точка (1). Перед отладкой и во время нее разработчику можно ставить более одной точки остановки.

Перезапустим страницу в браузере (2). Увидим, что над точкой появилась галочка (1), а в отладочной панели появилась информация (3, 4).



В левой части отладочной панели (3) указаны файлы, которые были исполнены во время выполнения скрипта, самый верхний — файл, на котором сейчас скрипт остановлен. Обратите внимание, что в браузере не появилась новая информация. Дело в том, что скрипт находится на паузе.

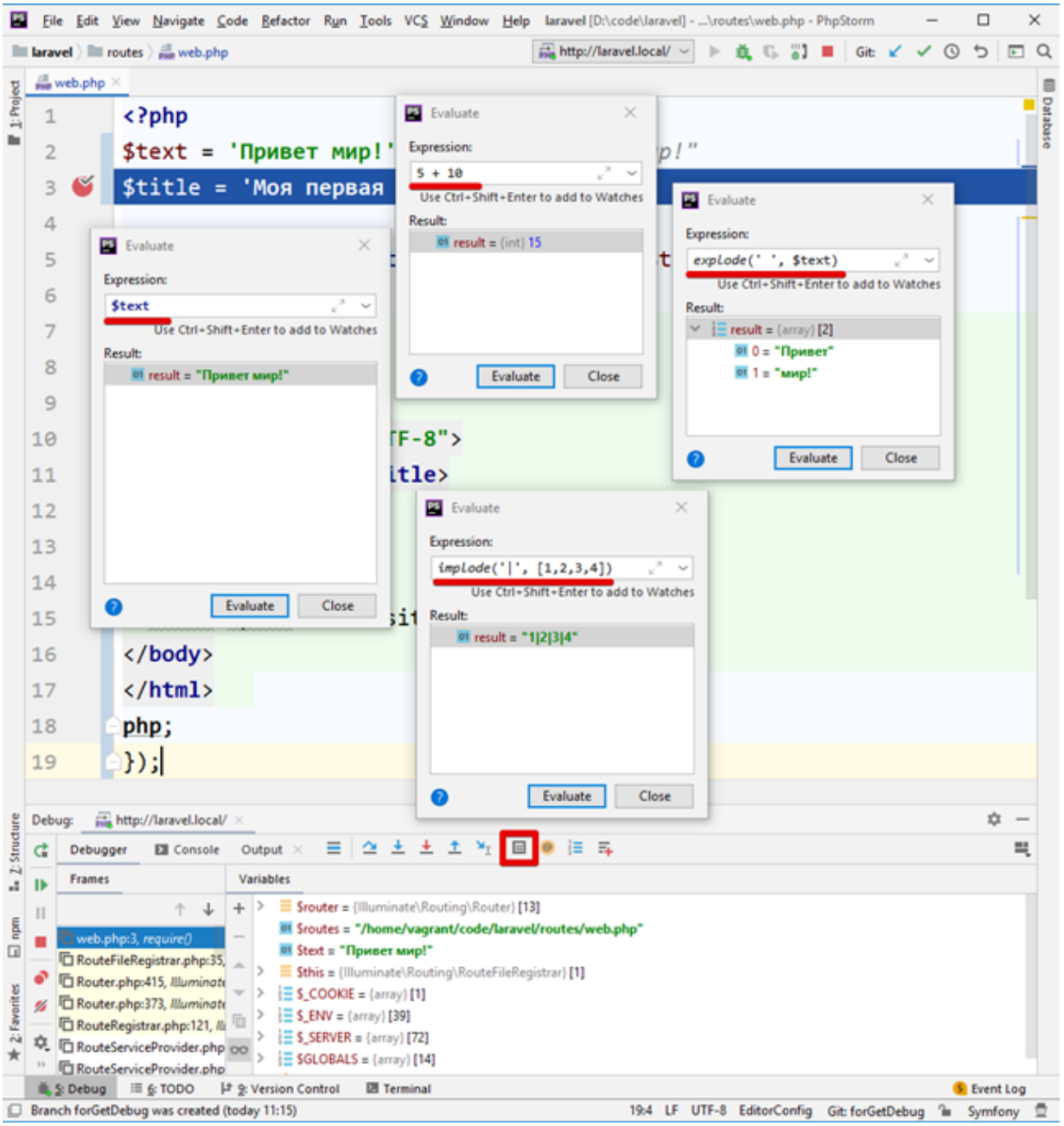
В правой части панели отладчика (4) расположена информация о всех созданных переменных до точки остановки скрипта.

Скрипт так и будет находиться на паузе, пока не закончится лимит времени или пока разработчик не продолжит его работу, нажав F9.

Алгоритм отладки выполняется по следующим правилам:

- если во время отладочной сессии не стоит ни одной точки остановки — скрипт работает без остановки;
- если точка остановки стоит в том месте, в которое скрипт не «заходит», — скрипт не останавливается;
- после остановки скрипта в любом месте можно нажимать на кнопки управления отладкой (5):
    - F7 — выполняет отладочный шаг с заходом внутрь функций;
    - F8 — выполняет отладочный шаг без захода;
    - F9 — продолжает выполнение скрипта до следующей точки остановки. а если ее больше нет — скрипт отработает до конца.

На каждом шаге остановки скрипта можно наблюдать за состоянием каждой переменной. Специальный интерфейс Evaluate позволяет определить, что содержит переменная на момент отладочного шага, а также выполнить другие операции. Для запуска данного интерфейса следует нажать на соответствующую кнопку в панели управления отладкой или сочетанием клавиш Alt + F8 (на Windows).



Практическое задание

1. Настроить на локальной машине окружение для работы с фреймворком.
2. Установить Laravel.
3. Реализовать несколько страниц будущего агрегатора новостей:
    a. Страницу приветствия пользователей.

b. Страницу с информацией о проекте.

c. Страницу для вывода одной и нескольких новостей.

Дополнительные материалы

1. https://laravel.com/docs/5.8/homestead.
2. http://laravel.su/.
3. https://habr.com/ru/post/328880/.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. https://laravel.com/docs/5.8/homestead.
2. http://laravel.su/.