6 Working With the Query Builder

В предыдущем эпизоде мы узнали как успешно подготовить данные, передать их в представление и затем как представления используют движок blade laravel чтобы отобразить серию элементов списка/но сейчас как вы должно быть знаете/вы можете хранить данные где то в базе данных вместо того чтобы хардкодить это/помните как пару эпизодов назад мы коротко затронули migrations/и я знаю что laravel включает две таблицы в коробочной версии/таблицу пользователей и таблицу password resets которая на текущий момент нам не особенно нужна в отличии от таблицы пользователей/сейчас когда вы видите этот класс вы видите что тут много чего происходит/импорт/некоторые дополнительные классы/мы расширяем migration класс/и затем у нас есть up метод и down метод/и сейчас не волнуйтесь/для вашего собственного migration rласса вам не нужно писать это вручную/скажем если я напишу php artisan в командной строке то вы увидите секцию под названием make/и тут целая куча их/думайте об этой области как о генераторах файлов/может быть вы хотите создать модель/или например создать mail класс/или запрос job/ любой из этих классов может быть автоматически сгенерирован для вас/и в нашем случае мы хотим создать migration/tсли мы хотим узнать больше о любой команде здесь то мы пишем php artisan help и затем вставим это сюда/то что мы тут видим – это то что мы должны передать ему один аргумент и также мы имеем некотрые опций если вам нужно переписать настройки по умолчанию/давайте попробуем это/напишем/php artisan make migration и затем мы должны дать ему имя/какое имя у текйщего migration/в основном вы даете имя migration которое соответствует тому что вы делаете/мы сейчас создаем таблицу users поэтому мы дали название create users table/возвращаясь к нашему примеру с задачами/напишем create tasks table создать таблицу заданий/окей отлично/и если вы переключитесь обратно мы имеем новый migration класс заполненный для нас/но давайте попробуем еще коче что давайте удалим этот файл/и если вы запустите help снова вы увидите что еще одна опция это обозначить имя таблицы которую мы создаем/давайте еще раз/если мы добавим create tasks/так сейчас мы получили ошибку только потому что нам нужно сбросить наши файлы/и еще раз запустим это/и в этот раз /только потому что мы добавили эту опцию тут/ларавел поймет – опа/ мы создаем таблицу тогда я подготовлю предварительные заготовки чтобы ты мог быстрее начать/создам таблицу tasks и затем схему для этого и затем в основном эта секция будет соответствовать этому/так что давайте представим из чего состоит задача/может быть это имеет заголовок и может это содержит содержиние/ но упростим это/считаем что это имеет только содержиние/и вохможно это будет связано с пользователем/и заметьте что этот метод соответствует типу данных/текст это длинная строка/integer соответствует числу/тут есть еще множество разных но как обычно – все остальное вы найдете в документации laravel/или если у вас есть названия вы можете сразу же перейти внутрь этого класса как только почувтсвуете себя комфортно/в любом случае сейчас мы определили как должна выглядеть таблица заданий и если мы перейдем обратно в sequel pro вы не увидите пока этой таблицы/это потому что мы все подготовили но по факту еще не сделали миграцию/давайте сделаем это прямо сейчас/php artisan migrate/это показывает нам что мы имеем новый класс с таблицей tasks/если мы обновимся то увидим это/мы имеем содержиние и упс/мы имеем юзер айди/нам это не нужно сейчас/и в таких ситуациях когда вы сделали ошибку просто удалите это и просто обновите migration/давайте посмотрим/мы собираемся искать migrate/ и вы увидите тут есть способ который называется refresh/который по сути откатывает все мои миграции и затем запускает их снова/жавайте попробуем это/ php artisan migrate refresh/это откатывает 3 миграции /вы увидите – откат tasks а также пользователей и пароли/которые я делал пару эпиходов назад/ и затем запускает это снова/это что то вроде контроль версия для вашей базы данных/в любом случае если я обновлю/сейчас у нас есть наши данные/давайте вручную заполним это/пойти в магазин/и для типа времени я просто собилась использовать mysql функцию/давайте создадим еще одну/закончить скринкасты/типа того/так что сейчас я имею таблицу tasks с двумя записями и я хочу подтянуть это/я иду в сублайм и я покажу вам 2 разных способа/мы пишем tasks и я закрою сайдбар/дальше пишем DB класс и указываем имя таблицы и затем пишем get который запросит наши записи/и это все/ но до того как мы загрузим это в наши представления взгляните на это/если вы когда либо возвращали запрос к базе данных/laravel достаточно умный для того чтобы преобразовать это в JSON/и это особенно хорошо для API/ и тут мы видим что мы имеем 2 записи/Вы знаете что давайте переключаемся в Chrome которая имеет лучше форматирование/ если вам нравится это перейдите в Chrome Store найдите Джейсон форматор/ затем вы можете установить это за один клик/ хорошо/ так что мы подтянули эти записи мы собираемся передать эту коллекцию в наше велком представление/ ещё раз мы собираемся запустить это/ на этот раз это будет иначе это вызывает ошибку/ это потому что tasks это коллекция объектов/ в особенности если я хочу вывезти содержиние/Я должен вызвать это/ я пишу task body/ обратно в Chrome обновились / и вот результат/ если мы например добавим очистить дом/ если Вернемся обратно и обновим/ вот результат/ сейчас как оказалось это является конструктором запросов ларавел/ это дает нам действительно красивой синтаксис чтобы обозначить таблицу и любое количество условий/ например если я хочу вывести только те задания которые были созданы в определенный промежуток времени/ или до определенного момента/ может быть я хочу упорядочить их/ вся эта функциональность доступна для вас/Допустим мы хотим разделить группы заданий/ или если например мы смотрим на tasks id/ не заметил этот синтаксис тут/id окружённый брекетами/и в laravel мы называем Это подстановочный знак/ это значит что это должно соответствовать идентификатором задачи 5 20 или что-то другое/ но сейчас мне просто обозначим это как task/ мы можем принять айди/ давайте зачем просто die и дамп id/Это функция помогает вам и вы будете использовать это постоянно/ если мы посетим наш проект слэш дев 3/ вы увидите что мы захватили этот номер/ и да мы могли бы сделать следующее - почему задачу написав find id/ зачем сейчас если мы die и dump задачу и обновим то мы увидим следующее/ затем изменим на 2/ или на 1/ отлично/ и что ещё лучше немного позже я покажу вам как сделать это еще более гибким/ делая что-то вроде того/ после того как мы поставили это на место/task переменная будет автоматически заполнена необходимыми данными/ это очень очень круто мы поймем как это сделать достаточно скоро/ в любом случае на текущий момент мы можем Написать task слеш show/ или что более популярно laravel/tasks точка show/это взаимозаменяемо/ и в основном это будет ссылаться на путь начиная с директории views/ поэтому Давайте установим это rosources views tasks show точка blade/ ещё раз мы добавим базовую форму HTML а после напишем task body/ о'кей Давайте попробуем это/ мы установим 1 маршрут который будет подтягивать все наши задачи/ и затем загружать/создадим еще один tasks index/ для того чтобы мы могли оставить оба файла/ захватим это создадим новый файл/ и напечатаем index blade php/ и сейчас все задачи будут загружать это представление/но затем если вы хотите посетить особенную задачу/ чтож проверьте номер задачи в строке браузера/ определите айди/ найдите задачу которая имеет этот айди/и затем загрузите это представление передав параметры через переменную/И наконец вы видите содержиние/ давайте проверим это/ все задания/ или какое-то особое задание/ один или 2 или 3/ а также как вы можете подумать это может быть ссылкой/ это будет перенаправлять вас tasks слеш id task как то так/ давайте попробуем/ очистить дом/ закончить скринкасты/ сходить в магазин/ возможно сейчас вы чувствуете себя немного комфортнее и это уже не так страшно/ оставим это пока проще в следующем эпизоде я покажу вам как использовать активные записи которые называются eloquent Которая позволит вам преобразовать это в нечто подобное/ это очень круто/ не могу дождаться следующего выпуска

/

Теперь мы можем использовать query builder Laravel для построения полноценных запросов базы данных с PHP. Больше никто не заставит вас писать хрупкие и трудно-читаемые SQL-запросы в виде строк. Позвольте мне показать вам, как это работает!

/

<https://laravel.com/docs/5.4/queries>

Episode 7 Eloquent 101

Давайте поговорим eloquent я думаю вам понравится это/ сейчас я создам модель/ модель является чем то вроде представлением чего угодно в вашей системе/ и на текущий момент мы хотим представить наши задачи/ сейчас вы увидите что это разместилось в нашей app директории/ oK Как вы можете видеть это расширяет нашу eloquent модель/eloquent это функциональность ларавел для работы active рекорд/ и я даже не ожидаю от вас что вы поймете почему это важно/ ключевое предназначение этого однако это намного упрощает ваши взаимодействия/ например/ мы запустим команду php arisan tinker которая вызовет shell laravel это предоставит нам простой способ взаимодействия с нашим приложением/ сейчас Мы создали task model/И он располагается в пространстве имен App что значит что я могу сослаться на него написав - app task/ и сейчас просто расширяя eloquent мы получаем доступ к множеству методов для запроса и работы с базойданных/ например если я хочу подтянуть все задачи/я просто запускаю эту команду/в этом случае у меня есть 3/но что если я только хочу подтянуть задачи которые имеют идентификатор 2 или больше?/окей/я собираюсь задать command K для того чтобы очистить экран и вернуть мою предыдущую команду/и это звучит так как будто мы говорим – где идентификатор больше чем 2 и затем покажите мне все результаты/но в нашем случае мы вабще то сказали – больше чем один/чтож мы могли бы написать больше или равен 2/или просто больше одного/и сейчас мы должны получить 2 записи/готово/или может мы просто хотим Показать содержиние каждой задачи/ я собираюсь подтянуть содержиние с каждой записи/ сейчас я получил коллекцию которая содержит 3 элемента/ мы поговорим немного позже о коллекциях они действительно хороши в ларавел/ они предоставляют вам доступ к множеству методов для работы с коллекциями таких как эта/ и сейчас у нас есть это давайте попробуем/ я собираюсь прийти обратно к routes web и написать это здесь/ на этот раз мы скажем tasks Equals - app task all/ сейчас если мы вернемся обратно и обновим получим тоже самое но в этот раз мы использовали конструктор запросов/ что действительно здорово так это то что вы можете добавить любое количество методов чтобы добавить дополнительное поведение/ поэтому например/ добавим некоторое поведение для тех задач которые были выполнены в методе под названием – is complete/давайте просто захардкодим результат и напишем – false/ OK сейчас давайте Вернемся обратно в Tinker/ Мы хотим найти самую первую задачу поэтому я напишу task/ мы могли бы написать find 1 для того чтобы найти идентификатор равной единице/ или мы также могли бы сказать - покажи мне первую строку/и, как вы можете видеть, это экземпляр задач, что означает, что я могу вызвать любые методы здесь task is complete / конечно это возвращает false/

300 – 337/ удалить

Давайте перейдем обратно преобразуем это в eloquent/task find по id/ вернемся в Chrome / кликнем на одну из них/ сейчас класс задач не найден это потому что/ запомните/ любое из этих классов будут иметь пространство имен App/ поэтому думайте об этом как и директории вы пытаетесь сослаться на задачу/ но вы не указали что задача имеет пространство имен/ поэтому мы всегда делаем следующее/ и вернёмся обратно и обновим вот результат/ задача с идентификатором 3/2/1/ или если мы имеем дело с пространством имен мы можем Расположить это вверху/ я собираюсь импортировать класс task сейчас я могу ссылаться как я делал это раньше/ обновимся/ и теперь у нас всё тоже самое/ сейчас я хотел бы показать вам горячие клавиши/ предыдущих эпизодах мы делали миграцию а после создали eloquent модель / вообще-то мы можем делать это одновременно/так что давайте попробуем/мы собираемся запустить php artisan migrate reset/ или мы откатим это обратно/ давайте просто удалим наш task файл полностью и начнем с нуля/ oK Вот как я бы рекомендовал это сделать/ вы приняли решение что вам нужна задача но вы так же знаете что вам нужна база данных/ поэтому вы пишите php artisan make task/ а также мы можем поставить черта m/ давайте используем помощь чтобы понять что это делает/если мы пишем черта m это также значит что мы собираемся создать новый файл миграции в процессе/вы также видите что тут есть команда черта си которая создаст новый контроллер в процессе/мы пока еще не говорили о них но мы будем в следующем эпизоде/мы пишем make model по имени Task и вы видите это сообщение/это потому что композер в некоторых ситуациях хранит файли которые должны быть автоматически загружены/но в этом случае он просто не нашел его/мы испрвим это написав команду composer dump autoload которая сбросит автозагрузку/сейчас все должно сработать/получилось/так что вы увидите/да/у нас есть migration также как и созданная модель task в рамках процесса что достаточно круто/сейчас так же как мы делали раньше добавим столбец в таблицу с типом данных строка или лучше текст/это позволит пользователям писать более длинные тексты/и почему бы нам не добавить еше одну секцию – Boolean которая покажет – была ли выполнена задача/и мы готовы мигрировать базу данных/сейчас я хочу передать пару задач прямо в tinker/вот как я это сделаю/я напишу - создай новую задачу/и сейчас самое время заполнить каждую секцию/например – сходить в магазин/задача выполнена/и если мы задумаемся/каждая задача может быть невыполнена/и мы можем установить значение не выполнена по умолчанию для всех новых задач/так как мы только что добавили модификации нам надо обновить нашу миграцию/пишем снова/сходить в магазин/и сохраним ее/и вот если мы перейдем в sequel pro/обновимся/и смотрим – да состояние по умолчанию – не выполнена – нуль/сейчас давайте найдем все задачи снова или только первый/вот/и запросим статус – completed /нуль/давайте добавим еще одну задачу/закончить скринкасты/теперь если перейдем обратно в хром и обновимся – у нас есть 2 новые задачи/и это пример хорошего рабочего процесса/вы создаете модель/вы даете ей имя и пожалуйста в единственном числе/если я создаю пользователей я указываю User и таблица Users/если посты то Post/если уведомления – модель Notification таблица Notifications/вы также можете создать что то вроде комбинации таблиц в которых данные пересекаются/поэтому подумайте о том что мы могли бы сейчас сделать на основании того что вы выучили сейчас/добавим еще одну задачу/давайте представим что мы сходили в магизн и мы хотим пометить это как выполнено/и сейчас мы хотим вывести только те задачи которые не были выполнены/мы пишем php artisan tinker/app task где cnатус выполнено равно ноль/и вот результаты/и это те задачи которые не были выполнены/или те которые были/это отлично/ но знаете что мы можем даже упростить это/сейчас это длинный запрос и он отлично работает но что в действительности это представляет?/это представляет выполненные задачи/и другой предоставляет невыполненные задачи/так почему бы нам не /давайте вернемся обратно в наш routes файл/не сделать процедуру которая называется не выполнено/это может быть полезно верно?/или упростим/это что называется наборы запросов query scopes/мы можем добавить их в модель Task/и традиционно у вас есть пара опций/первое – вы могли бы создать статичную функцию и назвать ее incomplete/которая будет просто оборачивать запрос/return task/но постойте мы уже в модели task/уберем это/и оставим где completed равен нулю/get/ и вот сейчас мы взяли простой запрос и обелнули его в статичную функцию/и теперь если мы еще раз попробуем/app task incomplete/вот мы получили наши записи/и это тот же результат но мы добавили ясности о том что этот запрос делает/мы обернули этот запрос в метод который называется тем что он показывает/однако в некоторых случаях вам нужна цепочка условий/например/покажи мне невыполненные задачи/но также те которые были созданы месяц назад/в этом случае мы не можем так сделать/но в случае если мы используем query scopes – мы можем/мы создаем обычный метод в котором используем scope и затем как мы будем ссылаться на него/это будет выглядеть так еask inсomplete jднако так как мы установили префикс scope – laravel будет хранить это как query scope – это будет принимать существующий запрос и если вам нужно что то передать внутрь вы делаете что то подобное/это позволяет вам написать task incomplete/ но если вам нужно передать туда какие то данные – это не очень принято – но если вам очень нужно вы можете передать их как дополнительные аргументы/окей /сейчас мы напишем - return query где статус completed равен нуль/поэтому query scope это просто обертка для какого то определенного запроса/сейчас если мы его попробуем – вы увидите что это возвращает справку о eloquent конструкторе/это потому что мы еще не выполнили запрос/и у нас еще нет коллекции/мы должны указать явно/и теперь это сработало/и как я сказал сейчас с этим синтаксисом мы можем написать incomplete а также условие где идентификатор больше или равен 2/теперь давайте протестируем/например этот установим как выполнен /и запустим еще раз/замечу что каждый раз когда вы делаете изменения в участке кода/в этом случае это неважно но если вы сделали изменение/иногда вам нужно сделать control C и перезапустить это снова/это применит все самые последние изменения/в любом случае сейчас вы увидите что это дало нам правильный результат /отличиная работа/вы действительно уже много знаете/если вы подумаете об этом – вы уже выучили роутинг/ узнали немного о конструкторе запросов и переключении его в eloquent/вы начали узнават о миграции баз данных/это великолепно/вы узнали о представлениях/по крайней мере начальные вещи/вы узнали о blade/вы узнали как разместить ваши представления в папках которые отвечают каждой вещи которую представляет ваше приложение и в нашем случае у нас етсь задачи/поэтому мы храним каждую задачу в рамках show task/ b чуть позже мы поговорим больше о спецификации названий/да вы действительно хорошо продвигаетесь/отличная работа и в следующем эпизоде мы собираемся поговорить о контроллерах

/

Теперь, когда вы знакомы с построителем запросов, мы можем переключиться на Eloquent: один из основных столпов платформы. Eloquent является реализацией Active Record Laravel, которая позволяет каждой из ваших моделей быть связанной с соответствующей таблицей базы данных.

/

<https://laravel.com/docs/5.4/eloquent>

8 Controllers

итак мы успешно хранили нашу логику, логику контроллеров прямо в routes файле/и для небольших проектов это действительно удобно/например если вы строите небольшой веб сервис или небольшой айпи или статичный веб сайт – это действительно удобно/но запомните /кое что из этого немного сложно/ и ваша задача это пытаться оставлять это так просто как вы можете/однако как я сказал лдя большинства проектов постепенно как вы подолжаете работу над ним количество секций будет увеличиваться и вы действительно этого не хотите/так что вместо этого используйте контроллеры/и контроллеры в своем роде средний игрок/они будут получать запросы и затем думать/окей мне нужно перенаправить эти данные и затем я спрошу модель предоставить мне эти данные/и как только я получу эти данные я передам их в представление на уровень отображения и представление отобразит это на экране/и как вы видите у нас тут есть разные уровни/уровень контроллера – распределяет/уровень модели – может быть ответственнен за домен или за предоставление данных которые мы просим/уровень отображения обычно ассоциируется с html/сейчас если мы хотим попробовать это мы можем написать/когда вы получаете запрос о задачах/какой контроллер мы можем использовать?/еще раз/зачем нам нужен контроллер?/это котроллер для задач поэтому мы назовем его Tasks controller/помимо этого нам нужно обозначить какой метод для этого контроллера будет использоваться/в основном это метод rest и мы поговорим об этом чуть позже в следующих эпизодах/метод который будет отвечать за все ресурсы вы будете чатсо называть index/окей/ но где мы можем хранить контроллеры?/чтож они будут в нашей папке app http и затем вы увидите папку сontrollers/окей но как мы его создадим? Мы должны это делать вручную?нет /вы можете использовать генератор/еще раз/давайте поищем make и вы увидите что у нас есть make controller генератор/пишем php artisan make controller и назовем его TasksController/отлично и сейчас вы увидите что это появилось в директории контроллерс/теперь нам надо добавить индекс метод/и сейчас вы можете думать об этом как/все что тут будет эквивалентно тмоу что храниться тут/поэтому вырежем это и вствим сюда/заметьте мы тут используем tasks и убедитесь что вы импортировали namespace/окей/вернемся обратно/удалим это и теперь давайте сделаем еще один для этого метода/если вы посещаете tasks и какую то особенную задачу/то что мы делаем тут – мы показываем задачу и общепринято называть этот метод show/окей давайте создадим еще один/show/ b опять мы берем все это и вставляем сюда/вернемся обратно удалим это полностью/ и теперь это выглядит чище/мы также удалим этот импорт/приятно смотреть и это просто/у вас есть ссылка и контроллер который отвечает за нее/далее мы имеем еще одну ссылку и еще один котроллер который отвечает за это/и смотрите еще раз/как он работает/он получает запрос и понимает что для этого еndpoint я должен запросить задачу/я обращаюсь к этой модели и после того как я получаю данные я передаю его на уровень отображения в это представление которое генерирует html код для отображения данных пользователю/сохраним это и убедимся что все это работает/я обновлю это и получим результат/я кликну на это и сейчас – идентификатор не определен/это потому что мы ожидаем айди но мы забыли передать его/окей/обновимся и вот результат/сейчас мы видим тоже самое/но теперь мы использовали контроллеры которые я вам рекомендую/в следующем эпизоде я собираюсь показать вам как связать маршруты между собой/это может быть немного пугающе но вы знаете это только на первый взгляд/и как только вы увидите как это просто вы полюбите это/оставайтесь со мной

/

До сих пор мы исключительно использовали route closures для обработки логики, связанной с каждой конечной точкой URI. Но я думаю, что для большинства проектов вы выберете выделенные контроллеры. Давайте рассмотрим, чем они являются и как сделать миграцию.

/

step 9 Route Model Binding

давайте посмотрим/в routes файле мы зарегистрировали в рахных маршрута и мы имеем 2 контроллера отвественных за них/в особенности я бы хотел взглянуть на этот/тут мы используем wildcard для представления любой задачи/так у нас может быть задача слеш один или задача слеш сто/что касается контроллера/он принимает идентификатор или то что является wildcard и затем передает этот идентификатор в eloquent find метод/и сейчас я хочу вернуть эту задачу/как мы уже знаем laravel автоматически конвертирует результат в JSON , что достаточно круто/один два три/но тут есть кое что даже круче/что если вместо определения этой переменной тут мы скажем что да я ожидаю задачу на входе/это было бы неплохо/но будет ли это работать? Я не знаю/ проверим/и зацените/вы получаете тот же результат/это называется привязка модели маршрута/и вот ключевой момент который вам надо понять/когда вы регистрируете маршрут убедитесь что это имя соответствует имени переменной/или другими словами/если бы мы назвали эту переменную theTask/чтож это не работает как мы ожидали/и это потому что laravel проверяет это за вас/он думает/окей у нас есть wildcard по имени task/и он также смотрит/эта перменная с таким же именем?просто потому что я собираюсь найти записи для вас и это выглядит в точности так как то что они собрались делать/и в итоге он будет искать нечто вроде task find wildcard/вот/и теперь все должно работать/поэтому это значит/то что мы имели вначале эпизода может быть упрощено до этого/и мы все еще получем тот же результат/3 два один/еще одна вещь которую надо держать в голове это что laravel будет принимать это за ключ/в нашем случае laravel будет искать запись с идентификатором равным значению wildcard/

cut 2.07 - 2.38

/

Route model binding обеспечивает полезный способ для автоматического добавления экземпляра модели в ваши route closure или действия контроллера. Например, если пользователь посещает /tasks/1, благодаря привязке модели маршрута, мы можем заставить Laravel автоматически выбрать Task с id равным 1, а затем добавить этот экземпляр в действие вашего контроллера.

/

10 step Layouts and Structure

Давайте продвинемся дальше и поговорим о макетах используя blade/с этой точки зрения в нашей директории views в каждом файле который мы создаем

11 long 20min Form Request Data and CSRF

В предыдущем уроке вы получили базовые знания о файлах макетов и о том как организоватьваш уровень отображения/сейчас я собираюсь пройтись по этому шаблону блога и ваша задача посмотреть это видео и понять можете ли вы воссоздать это сами/и как вы видите я уже это сделал/поэтому ваша работать – это понять как/я дам вам несколько подсказок/первая это конечно вы можете посмотреть источник и захватить все это/далее вам надо заменить захардкоженные бутстрап файлы и заменить из на соответствующие версии в cdn/далее вам надо захватить этот файл стилей и просто добавить в ваш собственный файл css/далее я хочу чтобы вы извлекли эти секции в соответствующие partial файлы/так например – это отвечает за голубю секцию наверху и будет вашей панелью навигации/тоже самое относится и к тому что в самом низу/это фаш футер так что вы таким же образом извлекаете и это/и наконец/эти посты тут/мы положим это все в наш post точка index представление/и в завершение нужно добавить в макет секции about и Архивы/может это пойдет в макеты а может это пойдет в index partial/это зависит от того что мы строим/но сейчас/поймите как и где расположить эти две секции/окей/нажмите паузу и посмотрите прямо сейчас/окей я предположу что вы сделали это или вас это пока просто не волнует/давайте переключимся к тому что у меня сейчас есть и посмотрим как я это сделал/мы переходим в ресурсы/представления и прямо тут у нас есть файлы макетов и несколько partials/для начала у нас есть мастер страница где мы ссылаемся на файл бутстрап cdn/ мы также ссылаемся на блог точка css/и я создал этот маленький файл/окей вернемся обратно/далее я подтянул сюда секцию навигации/и снова/это просто копипаст из примера бутстрап/далее ниже/у нас тут есть заголокок bootstrap blog который отображается здесь/вы даже можете извлечь это в header partial если хотите/или создать один зartial который будет содержать это и навигацию/и наконец у нас есть строка/с бутстрап мы используем эту строку чтобы потом разделить страницу не колонки/так что у нас есть первая колонка которая содержит post index и тут у нас все посты который сейчас захардкожены/это отвечает за отображение этого контента/но дальше если мы вернемся/после контента я включил секцию сайдбар/это еще одна колонка/это то где мы содержим About и Archives секции/и наконец/в самом низу я включаю секцию футер который содержит всякий контент который нам нужен/и это все/это просто/и если вы не сделали этого еще то сделайте/потому что отличный способ учиться/это перестать смотреть видео и начать кодить)/сейчас/следующий шаг это – я хотел бы создаь форму которая позволит нам создавать посты и отображать их динамически/так что возможно uri будет иметь форму post слеш create/это традиция/прямо сейчас это не работает/это думает что мы показывает пост/давайте ка вернемся в наш routes файл и скажем/если ты видишь post create и кстате давайте пока это закомментируем/когда мы видим create в ссылке то ссылаемся на метод create/окей/поэтому давайте добавим это/create будет загружать post create представление/окей давайте создадим его/и как обычно мы расширяем наш мастер файл макетов/и затем обозначим секцию для контента и напишем что то вроде - создать пост/окей давайте посмотрим/и смотрите - изза того что мы не разместили это к колонке сайдбар налезает на контент/давайте пофиксим это/вы увидите что пример использует ширину в 8 колонок /поэтому мы тоже будем/ок обновимся/и теперь у нас есть главная секция которая шириной в 8 колонок/дальше нам нужна форма/хочу сказать что так как мы используем бутстрап у нас есть множество готовых стилей прямо из коробки/поэтому я просто перехожу на страницу getbootstrap com css и тут некоторые примеры кода для формы/и мы вставляем это сюда/добавим немного пространства сюда/окей давайте глянем что получилось/да неплохо/может мы добавим линию сюда/ и конечно все это оптимизировано для формы входа на сайт/так давайте тут поправим на pаголовок для поста/и тут мы пишем текст и уберем это/обновимся/и конечно мы должны дать нашему полю ввода имя/для отправки этого на свервер это должно иметь имя/дальше нам нужно текстовое поле для содержания/ и это будет называться – body/наконец мы должны дать ему классform control/это просто твиттер бутстрап класс и это даст нам немного стиля/затем функция прикрепить файл может быть удалена/удалим чекбокс/заметьте мы имеем кнопку подтвердить/мы наховем ее – опубликовать/может быть скажем button primary это имезнить цвет кнопки/и теперь посмотрим что мы сделали – мы установили форму/мы указываем что это для публикации поста/и форма содержит поле ввода заголовка и содержания и также кнопку чтобы опубликовать новый пост/однако/если мы нажмем на кнопку – куда отправится новый пост?чтож – это будет отправлено – куда мы захотим/но если мы следуем основным традициям rest запросов то мы укажем тут слеш posts/вот основная идея всего этого – смотрите/если мы можем представить ресурс что то вроде посты/чтож мы посетим страницу слеш posts/которая покажет все посты/думайте об этом как о извлечь все записи постов/если мы хотим создать пост мы бы тогда перешли posts create и тогда тут отобразится такая форма где вы можете создать пост/и когда вы подтверждаете пост – вы на самом деле хотите хранить пост в базе данных так что это отправит post запрос в posts/ну а тут будет get запрос/поэтому пост запрос в ресурсы которые мы будем хранить в базе данных/также если вы хотите отредактировать существующий поствы бы перешли в posts слеш идентификатор и слеш edit/и по аналогии с этим/тут отобразится форма не создания поста а форма редактирования поста/и когда вы подтвердите такую форму – вохможно мы используем патч запрос posts и идентификатор поста/и эта строка похожа на то как если бы вы хотили отобразить какой то определенный пост/что то вроде get пост и идентификатор/и это отобразило бы нам один единственный пост/и если вы будете следовать этой традиции то вы сможете затем использовать один единственный uri в сошервешенно разных запросах/аналогичным образом мы используем команды get patch post или head чтобы сказать серверу что мы хотим сделать/это очень полезно/и тоже самое будет для delete postи если я хочу удалить пост/возможно я нажму на кнопку удалить пост – тогда мы сообщили бы серверу – используй эту информацию и отправили на сервер указание удалить этот пост/ так/это ясно?/если это что то новое для вас/возьмите паузу/серьезно/возьмите паузу и сделайте скриншот и запомните это так что вы потом смогли бы это использовтаь/так что это вот все ваши основные инструменты управления ресурсами/так/хорошо/если мы перейдем обртно/тут мы создаем новый пост/создаем ресурс/мы тут отправляем новый пост в хранилище ресурсов/тут есть еще кое что добавить/но я сначала хочу добавить обработку ошибок/сейчас давайте перейдем в наш web файл и зарегистрируем новый маршрут/когда мы отвечаем на новый пост запрос/обратите внимание на методы тут/их тут может быть несколько разных/это вызовет пост контроллер и традиционно мы назовем этот метод store – хранить/я покажу ваш небольшой трюк/если вы когда либо создавали контроллер/может быть контроллер задач/и если вы добавите флаг черточка r – да – это создаст контроллер но помимо этого это будет ресурсный контроллер/и если я хочу управлять ресурсами тут – это добавит множество заготовок для вас/быстро пробегусь по ним/это отвечает за все задачи/это отвечает за создать задачи/это за публикацию новой задачи – это именно то что мы только что сделали/это покажет определенную задачу/это отвечает за редактирование определенной задачи/еще одна - обновить – это если вы обновляете форму/ну и наконец это удалит пост/да – держите это в голове/черточка r автоматически заполнит макеты за вас/удобно да?я удалю это и вернусь обратно к нашей работе/пост контроллер сейчас запрашивает метод store/тут мы создаем новый пост используя данные запроса/сохранить это в базу данных и затем перенаправить куда нибудь в нашем приложении – это может быть домашняя страница/итак первое – получить запрос/давайте сделаем это давайте die и dump я хочу вызвать request helper функцию которая по сути предоставит все введенные данные/протестируем работу/обновим – перейдем пост create и введем/заголовок и содержание – опубликуем – но нет/это даже не дало нам потдвердить форму/laravel жалуется на несоответствие токена/это то что ларавел делает чтобы защитить вас по умолчанию/давайте поглядим что значит сsrf - так/для laravel действительно просто защищает ваше приложение от подельных запросов в межузловых атаках что является вредоносными несогласованными командами выполняемые из под авторизованного пользователя/это по сути является способом обмануть пользоватлея/и сейчас это думает что мы подтверждаем форму из внешнего веб сайта тогда как данные публикуются на вашем веб сайте/на моем веб сайте/и это вредит пользоватлею/и то что laravel делает по умолчанию это генерирует сsrf токен для каждой отдельной сессии которая у нас есть/дальше мы можем вывести скрытое полле для ввода/это звучит сложно но просто доверьтесь мне/и когда мы подтверждаем форму/и ларавел проверяет этот токен и если они не совпадают – это значит что что то не так и вы получаете такую ошибку/окей вернемся обратно и поставим это сюда/сsrf поле/это вернет нам скрытое поле/это общепринятое правило для всех ваших форм – включайте этот вызов/ок вернемся обратно/и я хочу взгянуть на форму/вот оно/так что csrf поле показывает эту секцию и видите эту длинную строку?это уникальный токен который сервер будет сравнивать с тем который он имеет для сессии/и если они совпадут – все будет хорошо/и вы можете продолжить/ок давайте попробуем это/заголовок содержание опубликуем и вот результат/и давайте взгянем/в нашем контроллере поста мы убили выполнение кода и вывели все данные которые получили от формы/и это является массиом/и как вы видите это соответствует введенным данным/но вы дакже можете вывести только заголовокм/или содержание/или вы хотите только заголовок и сое=держание/это полезно в случае если форма может принимать множество а=данных но вас интересует только эти два/и вот/и сейчас это выглядит как будто у нас есть все что надо/поэтому создадим новый пост/и если вы несколько мало знакомы с gространствами имен и фреймворками это обычный подводный камень/вы пытаетесь добавить новый пост в приложение но/давайте просто попробуем/да/у нас тут не найден класс/так в чем же тут дело?контроллер содержится в пространстве имен/но думайте об этом как о папках/сечас у вас 3 уровня глубины/но прямо тут вы пишете/ я хочу использоват пост/но это предполагает что вы имеете ввиду другое/это думает что вы хотите это/имеено такой путь вы видите тут/но что в действительности вы хоите сказать php-нет я хочу использовать корневую папку-так что решение это использовать такую формалировку и затем просто оставить тут post

Удалить с 1426 - 1436

Поэтому пост и я хочу установить заголовок/позже я покажу вам пару приемов как структуризировать это/но сначала простейший способ/затем аналогичноо для содера=жания/это установи значения того что ввел пользователь/далее сохраним это в базу данных/и далее переадресовать на домашнюю страницу/попробуем/опубликовали пост и это перенесло нас на домашнюю страницу/мы пока что не отображаем это диначимески так что давайте изменим это/переходим в терминал php artisan tinker/app post дай мне все посты/и вот это работает/так что отличная работа/вы только что создали вашу первую форму/но это правда что мы пока не имеем тут никакой проверки/у меня есть целый урок посвященный этому/чуть позже вернемся/с точки зрения рефакторинга/мы можем преобразовать это во что тоболее понятное/мы можем также написать post create и сейчас это будет ассоциативный массив данных/повторно мы пишем заголовок и пртягиваем заголовок/содержание и записываем содержание/однако если вы попробуете это это не сработает/эта ошибка также связана с защитой laravel/ что это имеет ввиду/мы тут массово назначаем все поля/и по умолчанию ларавел раскашляется изза этого/то что некоторые люди делают это пишут что то вроде/они просто передают все даннные в этот метод/и как вы можете себе представить –если вы не делаете никакой валидации-вы не делаете очистки и проверки вы просто позволяете пользователю впихнуть в форму все что им вздумается/запомните/пользователь всегда фожет открыть форму/изменить поле ввода или добавить ввода/и знайте что даже если вы предусмотрели только 2 поля это не значит что это будет только то что они могут отправить на сервер/поэтому ларавел защищает вас от такого рода угроз/мы не хотим чтобы например электронный адрес изменился когда мы это не разрешаем/или например статус подписчика изменился или что то еще/и мы можем бороться с этим несколькими способами/давайте пока вернем все обратно/первый сособ это перейти в нашу модель post и установить fillable cекцию/мы тут можем зафиксировать имена тех полей которые мы хотим чтобы были изменены/и если мы вернемся обратно и попробуем как мы ожидали/ используем artisan tinker/и сейчас мы имеем два поста/и я бы сказал что это вы можете запомнить и использоват как начальную защиту/ОДНАКО/по мере того как вы будете становиться экспертом в ларавел вы поймете что это немного сложно поддерживать/и у нас есть пара опций/первая - guarded это что то вроде противоположенности fillable – fillable это наш способ сказать – мы принимаем только эти – с guarded вы говорите что вы не принимаете эти поля/например идентификатр поь=льхователя/сейчас очистим/попробуем/и это все еще работает/однако/еще одну вещь которую вы можете сделать/вместо того чтобы описывать каждый класс вы могли бы указать родительский класс/перейду в модель и прсто захвачу все это/и это наш родительский класс/который расшиярет все остальные класссы/ и напишем as eloquent/и на самом деле как только я все это напишу/и например создам в будущем еще один класс который является моделью eloquent он автоматически попадет и сюда/и если вы усвоите этот подход и используете его/вам большье не придется устанавливать подобную дополнительную защиту/это конечно только ваше решение какие поля вы хотите отправлять на сервер/в нашем примере – эти двое – окей/и даже если пользователь манипулирует формой/добавит например поле для ввода идентификатора пользователя – это будет не важно потому что мы проигнорируем это полностью/но если вы задумаетесь/если в не добавили бы это/потенциально – вы просите создать пост со всеми параметрами/ с возможностью изменения идентификатора пользователя/такми образом они могут просто переназначить автора поста/как я сказал для разных моделей это может быть изменение статуса подписчика – так мы его включим/мы совершенно не хотим чтобы так случилось/хорошо/если вы привыкните к этому подходу/или например вы можете также написать запросить заголовок и содержание и это в основном то что я обычно делаю в своих собственных проектах/и на самом деле мы бльше не нуждаемся в этой строчке так что я удалю это полностью/просто отлично/мы птолько что сделали рефакторинг 9 строчек кода в одну/давайте еще попробуем/это все ееще работает и если мы потянем это – вот оно/превосходная работа и в следующем эпизоде мы продолжим/

/

Давайте начнем знакомиться с тем как вы будете работать с данными с формы, а также затронем некоторые проблемы безопасности, которые сопровождают этот процесс.

/

step 12 Form Validation 101

у нас есть форма которая позволяет нам публиковать пост но тут нет никакой валидации/это значит что если я нажму publish это вызовет контроллер и контроллер сразу же попытается сохранить это в базе данных/но как вы видите тут/на уровен базы данных у нас есть небольшая защита/и как написано тут/вы пытаетесь передать нам нулевое значение для заголовка поста который должен быть обязательно указан/и вы на самом деле не хотите чтобы это случилось/поэтому нам надо добавить некоторую проверку данных/я покажу вам две вещи которые вы можете сделать/ как минимум вы можете использовать валидацию данных html 5/но это может работать только на тех браузерах которые поддерживают html 5/так в нашем случае этот параметр является обязательным/поэтому мы добавляет в область textarea required/и теперь если мы вернемс обратно и обновим/если я попытаюсь опубликовать это/браузер мне запретит/попробуем в фаерфоксе/тут немного другой дизайн но суть таже/это является предварительной валидацией/но так как мы знаем что не каждый браузер поддерживает это особенно старые нам нужна дополнительная защита/мы переходим ко второй опции которая на уровне сервера/мы можем сделать это в нашем специальном контроллере PostsController /и проще говоря мы хотим чтобы до того как был создан пост мы хотим его проверить/и спешу сообщить к счастью laravel имеет встроенную функцию для этого/validate request и внутри указываем наши правила проверки/мы проверяем title/и далее мы указываем правила для этого параметра – укажем required/а также дополнительные атрибуты такие как минимальное или максимальное количество символов/на самом деле этот метод достаточно глубокий и вы можете найти все что вам необходимо в документации этого метода/но сейчас давайте просто установим что заголовок и содержиние – обязательные/окей давайте попробуем/это вернулось нас опять на страницу/и это то что делает этот метод/это проверяет введенные данные и если хоть что то пошло не так это вернет вас на предыдущую страницу и в рамках процесса это заполнит переменные ошибок/давайте посмотрим/мы возвращаемся обратно на нашу форму и разместим тут div и класс под названием alert и это будет содержать неупорядоченный список/и теперь я могу проверить ошибки/небольшая заметка/эта переменная доступна в каждом представлении/это значит что если на странице нет ошибок и форма подтверждена это будет содержать пустой набор/но если что то пошло не так и нас вернут на предыдущую страницу эта переменная будет заполнена/а теперь давайте захватим ошибки и каждую из них мы добавим в жлемент списка и затем выведем на экран/окей давайте еще раз попробуем/обновим/publish /прокрутим внизу и вы видите это/но знаете что я должно быть указал не тот класс/ danger /о да/эмм почему бы нам не сделать/добавлю немного пространства/и еще раз/и заметьте что на начальной стадии у нас тут пусто/ и если мы опубликуем/увидим тут ошибки/и я опять забыл изменить это/в любом случае это сработает/теперь это работает но давайте откроем еще одну вкладку/и мы видим этот пустой блок/и разумеется он будте тут/у нас сработал alert и мы разнесли это на все страницы/но давайте кое что добавим/если у нас есть ошибки мы тогда отображаем это/вернмся обновим/сейчас ничего нет/если мы имеем ошибки то мы увидим это/и мы все еще имеем проблемы с пространством тут/потому что это добжно быть в form group также/да это должно нам дать немного margin после кнопки/работает/но сейчас давайте немного наведем порядок/вы возможно заметите что у вас есть множество форм в вашем приложении/и возможно вы будете это писать снова и снова/и если нас это устраивает и это не изменяется то просто давайте сделаем из этого бибилиотеку/назовем например layouts errors/создадим новый файл resources views layouts errors blade php и вставим это сюда/и вот любая форма может ссылаться на этот файл путем include и должно быть мы получим тоже самое что и было/и вот давайте предоставим какие нибудь данные/опубликуем это и вот у нас есть новый пост/и вот это было введение в базовую проверку данных/

вырезать 502 601

и если вы хотите узнать еще о контроллерах смотрите дальше мои эпизоды/

/

Это правда, что теперь мы можем успешно отправлять и обрабатывать данные с формы; однако мы еще не реализовали никакой проверки данных. Это означает, что пользователь может (и будет) вводить все что угодно. Мы не можем этого допустить.

/

step 13 Rendering Posts

прямо сейчас у нас есть несполько постов в нашей базе данных/однако главная страница отображает захардкоженные версии/которые достались нам от bootstrap шаблона/давайте изменим этом/мы идем в редактор и в PostController b d индексе методе я хочу вытащить все мои посты/далеее я передам это в представление и простоe compact подойдет/и вот это представление имеет доступ к нашей коллекции постов/давайте вернемся в индекс/и посмотрим какая тут структура/ага у нас есть колонка и затем каждый пост/прокрутим вниз/о да/это дубликат поэтому давайте избавимся от этого/очистим немного/ну вот/и сейчас/тут кажется много кода но это просто параграфы/dev представляет каждый пост поотдельности так почему бы нам не захватить все и не сохранить это в partial/это относится к постам поэтому назовем posts post/сохраним это и вставим сюда/окей/давайте проверим что это все еще работает/и теперь мы можем добавить динамику/это должно быть post titleи запомните/мы используем этот синтаксис чтобы вывести поле/но конечно если мы вернемся обратно это не будет пока работать только потому что у нас еще нет переменной пост/если мы перейдем в это представление у нас тут пост переменная/так почему бы нам не сделать следующее/foreach posts как пост/ затем оставляем этот partial/и вот теперь у нас есть переменная пост/думайте об штclude как о простой вставки всех этих данных/и теперь эта переменная будет доступна потому что она доступна тут/обновимся и вот теперь все вышло/давйте сделаем все тоже самое для всех этих данных/ post body /и в нашем случае /смеется/ у нас будет только пара слов/это не очень клево но это работает/пока что мы еще не связали пост с пользователем/мы покроем это в эпизоде про аутентификацию дальше/пока что я опущу это/и вот что нам осталось сделать это сделать это динамическим/и вот как мы это сделаем/пост и что дальше?/вы вот помните когда мы делали миграцию мы разместили также timestamp которое является параметрами дата создания и дата обновления/так почему бы нам не использовать их/post created at/но вы знаете/это выглядит не очень приятно/и вот небольшая подсказка/когда вы используете timestamp это автоматически будет ссылаться на бибилиотеку под названием carbon/carbon библиотека по настоящему космос/вы увидите что мы можем делать например создать от даты/давайте найдем что то связаное с diffforhumans/которое является одной из моих любимых/поэтому если вы имеете ссылку на carbon dы можете вызвать этот метод и это отформаирует в стиле как давно это было сделано/тут есть множество опций таких как /покажи это по месячно или по дням или по часам/множество полезностей/но давайте пройдем чуть дальше/в форматирование строк/и тут вы видите примеры и можете найти что вы хотите/может быть мы хотим в таком формате – значит toFormattedDateString/lfdfйте попробуем использовать это/обратно/обновимся/и все хорошо но дальше/вот этот заголовок не кликабельный/для этого мы добавляем тег ссылки/ posts post id/типа того/обновим и вот это теерь ссылки/вы можете стилизовать их как хотите/но если мы кликнем на это/мы еще не установили пока наш маршрут/мы идем в routes и разкомментируем это из предыдущего эпизода/так что если мы сопоставим/ слеш и идентификатор/и у нас это есть/мы переходим в контроллер/в метод который отображает и показывает представление/вот это/и вот щас у нас все длжно получиться/мы видим заготовку и теперь поправим это/первым делом давайте захватим эту секцию/и нам нужно 8 колонок тут/дальше для каждого поста заголовок и содержиние/но пока это также не будет работать/обновимся/и это потому что мы не передали переменную/так тчо давайте найдем пост/тут есть длинная форма как мы выучили раньше/и затем мы передаем это сюда/обновимся и это работает/пост с идентификатором 1/2/3/или мы можем использовать привязку модели маршрута о которой я также рассказывал/post и не забудьте название wildcard пост/так что указываем такое же название у переменной/и вот laravel захватит это для нас/и вот у нас получилось тоже самое/вернемся домой/мы можем видеть все это/мы можем кликать/и смотреть все о посте/также мы можем созать новый пост/и я вставлю произвольный текст/но сейчас он находится внизу но мы на самом деле хотим увидеь его наверху/и это звучит так как будто нам надо сделать сортировку/мы можем сделать это прямо тут в функции index/пост и я хочу получить самый последний пост сначала/это – просто query scope/давайте взглянем/если мы провалимся в конструктор мы увидим что latest содержит сортировку по колонке в порядке убывания/это тоже самое если бы мы сказали/захвати посты и отсортируй их по убыванию что значит что я просто могу сделать следующее/пост сортировка по дате создания по убыванию/окей/обновим/и он сейчас на верху/или по возрастнию/разместит его внизу/и это является настройкой по умолчанию/поэтому да/latest может заменить весь этот код/и если мы вернемся еще обратно/вы увидите тут есть метод oldest/и вы увидите что это точно наоборот/и это может быть тоже полезно/окей/мы сделали тут много прогресса/и действительно/остановитесь на минуту и задумайтесь о том что вы уже выучили/вы выучили представления/контроллеры/вы сделали миграцию/вы установили модели тут/вы выучили перекрестные запросы/вы узнали о контроллере валидации/вы узнали о привязке модели маршрутов/вы узнали как создавать данные и синхронизировать из и подтягивать/вы действительно прогрессируете/но если вы немного запутались или перегружены/не парьтесь/дайте этому время/похлопайте себе потому что вы держитесь хорошо/

/

Теперь, когда мы добавили необходимую функциональность для перемещения новых постов в базу данных, мы можем отфильтровать и отобразить их на странице, используя Blade.

/

step 14 Laravel Mix and the Front-end

К этому моменту вы уже выучили базу или освнову работы с laravel/вы узнали о контроллерах моделях и представлениях/в предидыщуем эпизоде мы делали обработку создания новых постов/но тут еще можно добавить функцию привязки поста к пользователю/давайте остановимся на этом и сфокусируемся на фронт енде/что если вы хотите использовать sass или laаs и как мы можем построить javascript/окей сейчас мы переходим в директорию ресурсы и если вы уже знакомы с представлениями но мы еще не затрагивали assets/вы видите тут 2 папки js и sass/если вы используете less это так же приемлемо/давайте посмотрим что у нас есть в app scss/окей/прямо из коробки у нас есть некоторый набор/мы тянем шрифт из гугл/мы ссылаемся на какие то sass переменные/и эти перменные относятся к twitter bootstrap/и это является гидом чтобы вы могли быстро начать/но если вы не имеете желания использовать бутстрап просто удалим все это и начнем все снуля/для примера давайте сделаем что то очень простое/для содержания добавим голубой цвет фона/так вы обновили ваш sass но нам нужно скомпилировать это и затем предположительно в директории представлений/возможно в файлах layouts мы сошлемся на это/так как нам это сделать?/окей давайте взглянем/мы собираемся использовтаь конструктор под названием webpack/но что еще лучше laravel предоставляет инструмент mix который является уровнем конфигурации вокруг webpack чтобы сделать множество вещей которые вы будете использовтаь простыми методами/это намного проще и намного быстрее чтобы начать использовтаь/вгляните на это/если мы хотим использовать последню версию javascript и мы хотим использовать модули и разделить наш код/чтож все что нам надо сделать это вызвать js метод что значит/возьмите этот файл и скомпилируйте его в js директорию/ прямо тут найдите этот app.js файл /и мы пройдемся по нему чуть позже/ но нам нужно скомпилировать это в публичкyю папку js/вернемся обратно/дальше мы собираемся использовать sass или less и все что нам надо сделать это изменить тут и laravel сделает все остальное/это очень полезно/так что еще раз/найдите этот файл app scss и скомпилируйте его в scss папку/ давайте посмотрим/вот оно/и я вижу это но как я могу использовтаь это?webpack laravel это определитель node зависимостей/поэтому очень важно что у вас установлен node и npm/вы можете установить это через брю/или вы всегда можете сходить на node js org и нажаь на кнопку загрузки/это займет всего минуту/как только у вас получилось вы можете проверить что все впорядке путем проверки версии node и npm вот как у меня/и затем печатаете npm install/как то так/и сейчас вы знаете как композер управляет нашими зависимостями ваш package json aайл управляет node зависимостями/и тут вы видите что мы подтягиваем дaravel mix/окей отлично это подтянуто/так что мы сейчас сделаем это вернемся обратно и посмотрим на эти npm cкрипты/все что вы видите тут может быть выполнено путем npm run b затем имя/думайте об этом как о ссылке/если вы напишите это то это запустит этот скрипт/может это выглядит страшно но это не так/установите среду – разработческая/вызовите вебпак/я хочу живой прогресс/мы хотим скрить некоторые модули на выходе и вот тут путь к нашему файлу конфигурации/окей отлично/теперь давайте запуситим его/и это компилирует все и вот наши выходные файлы/давайте посмотрим/публиная папка/вот наш файл js и вот наш файо css и теперь мы можем взять такие вещи как блог удалимэто и вместо этого давайти сохраним это в нашем scss файле/мы конечно могли бы использовать partial но я просто хочу сохранить время/так еще раз/я скомпилиру это/это готво/дальше давайте вернемся в наш layout файл/и если прокрутим вниз сейчас мы можем изменить это на scs/окей вернемся обратно ообновимся/и тут все тоже самое но сейчас мы используем sass и конечно любые javascript которые вам нужны/еще одна вещь/это немного утомительно изменять все снова и снова/вот например вернем фон обратно к голубой/и я сделал изменение/я должен скомпилировать это/и затем уже вернуться обратно и обновоиться/да не идеально/поэтому вместо этого давайте используем другой скрипт/watch/ это запустит webpack/это отслеживает любые изменения/это достаточно весело/давайте ка изменим это на оранжевый/и как только я сохраню это/скрипт подцепит изменения и перекомпилирует/вернемся в хром и проверим/и как вы видите сейчас/я так обычно работаю/и я могу полностью игнорировать терминал/розовый-сохранить-обратно в хром/и это было зарегистрировано/превосходно/это практично/что касается оavascpirt / это эпизоды не про джаваскрипт но я вам дам несколько начальных знаний/как я говорил дфravel включает некоторые бутстрап код/если вы не хотите вы можете просто удалить это все и начать с нуля/но это просто предоставляет вам подсказки/например это начинается с требования файла bootstrap/мы подтягиваем lodash файл jquery bootsrap sass / и я бы рекомендовал вам вначале минимизироваь эти зависимости которые честно говоря достаточно большие/далее мы тянем vue фреймворк который действительно хорош и у меня есть несколько серий про него/еще мы притягиваем инструмент под названием аxios rоторый является ajax бибилиотекой для ajax запросов/и устанавивает заголовки и этовсе/далее если прокрутим вниз/мы регистрируем vue компонент/еще раз вы узнаете о нем больше в моих эпизодах но в основном это позволяет вам создать один файл и хранить в нем шаблон скрипт и даже стили если хотите и да/это позволт вам начать если вы знакомы с vue/но если нет удалите это полностью и затем начинайте свой javascript как обычно/окей давайте вернемся в наш php в следующем эпизоде/

/

Разумеется, вы не будете тратить все свое время на работы с PHP. В какой-то момент вам нужно будет написать некоторый CSS код, возможно, немного JavaScript, и, естественно, вам понадобится какой-то способ оптимизировать этот рабочий процесс. Позвольте показать вам, как работает Laravel Mix.

/

Step 15 Eloquent Relationships and Comments

У нас тут есть небольшой блог да?есть посты которые можем кликать/есть ссылка на этот пост но сейчас у нас нет секции дял комментариев/так вот какая связь между постом и комментариями?давайте просто задумаемся/каждый отдельный пост может иметь один или несколько комментариев/или другими словами пост имеем много комментарие/и обратно/один комментарий может иметь один связанный пост/окей давайте поймем как заставить это работать/звучит так как будто нам нужна таблица комментариев/давайте сделаем это/давайте создаим новую модель под названием комментарий/но я также хочу сделать миграцию в рамках процесса/окей/и я также сделаю апдейт нашем общей модели/это будет ее расширением/и так как мы добавили M тег мы также создали миграцию для таблицы комментарии/комментарий состоит из содержания конечно же/но комментарий принадлежит посту как мы говорили об этом верно?и это звучит так как будто нам нужна какая то ссылка между комментарием и постом/и может быть мы можем предоставить некоторый идентификатор поста/в будущем когда мы дойдем до аутентификации мы также можем решить что комментарий принадлежит пользователю/и мы потом добавим это/но сейчас давайте оставим это проще/хорошо я собираюсь сделать миграцию для базы даннх/так если я сейчас переключусь в sequel pro/обновлюсь мы увидим что я имею таблицу комментариев которая связана с постом/я могу добавить запись/и давйте скажем что это относится к посту с айди 5/ это значит что этот комментарий относится к этому посту/попробуем изменить на 6/почему бы не добавить еще один/я просто сделаю дубликат/сейчас давайте добавим определение связей между каждой моделю/нам нужна показать что пост может иметь много комментариев верно?давайте укажем это/eloquent предоставляет нам для этого hasMany метод/и на вход мы подаем название класса/если вы не знакомы с этим это просто обозначение пути класса/это просто эквивалент этого/большинство из тех кто использует php выше 5.5 будут использовать этот синтаксис/окей давайте попробуем/мы запускаем php artisan tinker/найдем наш пост – вот он/и дальше вызовем пост и я не собираюсь вызывать этот метод/я собираюсь вызвать это свойство/laravel видит это и он понимает что это относится к отношениям/и если мы запустим это сейчас мы получим только те комментарии которые имеют связь с постом у которого айди равен 6/это значит что если мы попробуем найти еще один пост/и посмотрим на его комментарии/вот это показало нам коллекцию но тут пусто/окей отлично/сейчас давайте попробуем обратное/если я имею комментарий и я хочу посмотреть связанный с ним пост/ну это просто/все что надо сделать это написать коммент и получить пост верно?это значит что функция должна называть пост/и в этих отношениях комментарий принадлежит посту и мы используем функцию belongsTo/окей/давайте попробуем/я закрою это/открою тинкер снова/найти первый комменратий/и я сохраню это в си/и я хочу увидеть связанный с ним пост/ГОТВО/определенный комментарий и связанный с ним пост/очень очень просто как только вы попробуете это/сейчас давайте вернемс в хром/мы находимся в посте но я хочу видеть тут связанные комментарии чуть ниже/окей давайте попробуем сделать это/когда мы показываем пост/я использую это представление/прямо тут внизу мы установим секцию для комментариев и скажем/для каждого комментария поста как комментарий/ затем добавим раздел который содержит текст каждого комментария/ и например мы хотим показать содержиние комментария/окей и сейчас у нас должно быть 2 верно?обновимся и у нас тут ужасное форматирование/давайте исправим/разделим с пробелом/ добавили немного пространства/почему бы нам не попробовать использовать базовую группу список/обернем это и каждый из них будет элементом группы списка/это просто компонент bootstrap/это выглядит чуточку лучше/дотаточно неплохо/у нас пока нет пользователей/и для начала это окей/мы можем добавить нечто вроде/покажем это в выделении/мы можем указать как давно это было опубликовано/и как мы выучили в предыдущем эпизоде тут будет полезно использовать carbon библиотеку/почему бы нам не вызвать diffForHumans /читабельную форму того как давно это было опубликовано/да что то вроде того/добавим двоеточие и пробел/нуу/как то так/достаточно чтобы решить наши нужды/и вот эти комментарии будут отображаться только для тех постов к котрым они принадлежат/тут нет/и тут/но тут есть/давайте вернемся к посту номер 1 и добавим к нему комментарий/пока мы сделаем это ручками/но попозже добавим форму/так/обновим и вот у нас новый комментарий/следущий эпиод также важен так как я покажу как сделать форму и отобразить ошибки и это сделает процесс добавления комментариев очень быстрым/

/

Если пост **имеет много** комментариев, то, следовательно, комментарий **принадлежит** конкретному посту...ну а как мы можем отразить это в нашем коде? давайте займемся этим в этом эпизоде

/

16 step Add Comments

в последнем эпизоде мы установили блок связанных с постом комментариев/но мы вручную создали эти комментарии в sequel pro /в этом жпизоде давайте сделаем форму для жтого вот тут/мы перейдем в наше представление show – это то где мы отображаем пост/и потом как насчет прямо тут расположить секцию для комментариев/мы устатновим card тут/я думаю что это форма бутрстрап/давайте посмотрим/да/и теперь добавим немного простанства тут/и поставим секцию которая позволит оставить комментарий/мы используем тут form group и добавим текстовую область тут/имя будет body и подсказка будет – ваш комментарий тут/и добавим класс – form control/проверим/да это отлично/и наконец мы скопируем это и добавим секцию для кнопки отправить/и вы знаете мы можем просто стащить это с другой формы/и давайте посмотрим/скажем/добавить комментарий/окей/обновим и вот у нас получилось/и теперь мы собираемся заполнить это и когда мы кликнем на добавить комментарий кнопку/так еще раз/о том что мы делаем/мы создаем комментарий так что метод будет – POST/и между прочим/если метод будет patch или delete вы можете использовать method-field функцию/почему мы это делаем?мы делаем это потому что на момент записи этого эпизода большинство браузеров не понимают ничего кроме get и post/но если вы хотите сделать запрос delete или patch это не будет работать/вместо этого мы вроде как подделываем его/мы создаем скрытый ввод который содержит наш тип запроса/наш сервер определит и поймет это чот на самом деле вы хотите patch запрос/в любом случае/действие будет-и на самом деле это зависит от вас-действие может быть слеш комментарии/жто подойдет/действие может быть пост и затем идентификатор поста слеш комментарии/это тоже сработает/но я советую не использовать больше чем 2 уровня/в основном 2 уровня это нормально но чем больше вы тут размещаете тем сложнее это становится и тем больше вы рискуете ошибиться/поэтому в этом случае я думаю что глубина в 1 уровень – это нормально/это будет ваше действие/ну ка попробуем это/мы знаем что это не сработает потому что мы еще не установили конечную точку для этого и как вы видите это перенаправило нас не в то место/давайте откроем routes web файл и напишем – когда мы получаем запрос пост – делай posts post comments и теперь мы делаем postscontroller какой будет метод для этого?мы можем назвать его хранить или добавить комментарий/у вас также есть возможность создать полностью нвоый контроллер для этого/и как вы потом заметите/если вы не даете странные названия методам и вместо этого используете существующий контроллер вы можете затем использовать общие названия его методов/в нашем случае это будет comments controller и метод store/давайте задержимся тут/ php artisan сделай мне контроллер по имени comments controller и сейчас это будет в нашей http controllers директории/надо добавить store метод и сейчас это будет получать пост/убедимся это это располагается наверху/но мы также хотим работать с кмментариями/давайте притянем это также/сейчас я покажу вам длинную форму метода и мы поговорим немного о читабельности и о том как улучшить это/длинная форма будет - … и давайте укажем явно/содержиние будет равно тому что пользователь укажет в этой текстовой области/и идентификатор поста будет равен идентификаторому поста/мы делаем это и затем – редирект на предыдущую страницу/тут есть функция helper называется – back/ хорошо – давайте попробуем это/вернемся/добавим/ааа у нас тут несоответсвие токена/и мы говорили об этом верно?любой форме нужно поле csrf для защиты от темных сил/это мой комментарий/это было опубликовано как вы можете тут видеть/это отлично работает/тут еще одна вещь котрую вы можете взять на вооружение/что мы тут делали?так например если бы вы расговаривали с клиентом для которого вы работаете/вы бы сказали что то вроде – ну нам нужна секция чтобы добавлять комментарии к посту/и мы использовали comment create тут/не было бы лучше если бы имели метод связанным с постом – потмоу что мы крепим комметарий – К посту/так что пост может быть отвественнен за это/и мы просто могли бы написать-post add comment/и затем – запрос на самом деле не важен так как он будет выполняться за кулисами/мы удали это и дальше/давайте передадим содержиние комментария в метод/ и теперь нам надо добавить addcomment метод/мы принимаем содержиние комментария/и сейчас мы могли бы сказать – создать комментарий и далее идентификатор поста равен не посту – так как мы уже внутри поста – вметсо этого this id/и да-это отличная опция/возвращаемся/это выглядит достаточно чисто для меня/попробуем/но мы все еще не закончили/вернемся обратно и тут еще один способ выразить это/вместо того чтобы вручную указывать идентификатор поста - мы уже имеем отношения тут/чтож с eloquent мы можем сделать следующее/this комментарии – создать новый где содержиние равно тому что мы вставили в тектовую область или как мы уже знаем мы можем просто сделать compact/ b теперь мы можем использовать это/и вы видите чтоeloquent становится очень очень гибким когда это относится к отношениям/другими словами если мы а=напигем так то это вернет нам коллекцию всех комментариев связанных с постом/с другой стороны/если мы свяжем это с createмы вызовем метод и теперь помимо того что мы передаем содержиние это также свяжет комментарий с идентификатором поста изза установленных тут отношений/

удалить 708-730

и наконец чтобы закончить тут давайте добавим уровень валидации/я хочу проверить запрос и содержиние очвевидно обязательно к заполнению и вы можете установить минимальное количество символов чтобы это считалось комментарием/но что если валидация провалится?как мы обсуждали ранее/это вернет нас обратно/не создаст пост и не отобразит ошибки/и если вы помните пару эпиходов назад мы установили специальный partial/мы можем его использовать/представления – ошибки/давайте просто сошлемся на него чтобы он помог нам понять в чем дело/прямо тут мы напишем/включить макет ошибок/окей еще разок/это запускается и показывает ошибки/я счастлив)/чтобы закончить я хочу заставить вас что то написать/это браузерная часть валидации/но если по какой то причине это не сработает у нас есть серверная часть которая проверит/и я думаю мы тут закончили/прекрасная работа/

/

Представьте, что вы создаете такой маленький блог для клиента. Разговаривая с ними, они, естественно, озвучивают то, что они хотели бы иметь возможность добавлять комментарии к любому сообщению. Давайте посмотрим, можем ли мы использовать эту точную терминологию в нашем коде, чтобы обеспечить решение для них.

/

step 18 Associating With Users

на самом деле вызов make команды вы скорее всего будуте делать в самом начале вашего проекта/не после того как вы проработали 6 месяцев и решили ее сделать/нет нет/сразу же в начале/но в нашем случае/давайте не будем использовать эту команду и вместо этого попробуем понять как нам связать пользователя с постом и затем отобразить это/окей/если вы переключитесь в sequel pro /обновитесь/вы знаете что у вас есть таблица пользователей/это то что ларавел предоствяляет вам с миграцией после установки/но сейчас у нас нет н одного пользователя/так почему бы нам не добавить одного вручную побыстрому/новый пользователя и нам надо предоставить емайл имя и пароль/имя джон до/емайл/пароль – мы никогда не хардкодим эти вещи/вам нужно быть осторожным с этим/ и если я сохраню это – на выходе если я перейду в sql pro/наш пароль был сохранен в виде что я называю – открытый текст/это очень большая небрежность/НИКОГДА не делайте этого/если вы установили таблицу пользователи и вы переходите в нее и видите что пароль представлен во всей красе – это очень большая ошибка и вам нужно исправить это/мы собираемся шифровать такие вещи/самый худший сценарий – если кто то хакнет вашу базу данных – они получат доступ к абсолютно каждому паролю и это может быть использовано чтобы получить доступ к аккаунту на других веб сайтах/мы всегда должны шифровать это/итак я удалю это и сделаю это снова/у нас есть новый пользователь/мы дали ему имя и емайл/ но на этот раз для пароля/мы не собираемся хардкодить пароль/вместо этого мы будем использовать функцию под названием bcrypt/и вы увидите когда мы это сделаем это сгенерирует уникальный безопасный токен/и когда пользователь попытается войти – это будет сопоставлено с его настоящим паролем/и если мы сохраним это и я перейду обратно в sequel pro и обновклю – да/это точ то мы хотим – длинный защищеннный пароль который будет сложно вычислить/отлично/сейчас нам надо найти способ сказать/так поменяю его идентификатор на 1/джон это тот кто написал этот пост/заметьте что для поста – тут нет соединения/тут нет места для того чтобы показать кто создал этот пост/и тоже самое для комментариев/комментарий связан с постом/но не связан с пользователем/давайте настроим пару вещей/я вернусь в нашу сreate пост миграцию/и теперь мы указем айди пользователя связанного с ним/это будет способ понять какой пользователь связан с постом/далее мы сделаем тоже самое для комментариев/комментарий принадлежит пользоватлею но он также принадлежит посту/теперь давайте запустим refresh команду/сейчас мы откатили все обратно и затем еще раз накатим/не теряйте свои данные/это окей пока/тут множество опций вернуть их/в любом случае комментарии содержит айди пользователя и тоже самое у постов/и так как мы установили эти отношения мы можем сделать следующее/если мы перейдем к модель комментарий – да комментарий принадлежит посту/но мы также осознали что комментарий может принадлежать пользователю/поэтому давайте установим это отношение/сейчас мы можем сказать/дай мне текущий мкомментарий который у меня есть и я также хочу получить пользователя связанного с ним/и далее если мы хотим захватить имя пользователя который его создал – это именно тот синтаксис который надо использовать/и снова/это доступно только потому что у нас есть этот метод/далее тоже самое является правдой для поста/пост принадлежит пользоватлею/я хочу захватить пользователя который создал пост/это то что я буду делать/мы даже можем эту забавную вещь – комментарий дай мне пост и затем пользоватлея/это даст нам имя пользоватлея который создал пост который связан с комментарием/клево/если вы пойтеме как это сделать вы поймете что это имеет огромную ценность для блога/и теперь/если я например хочу получить все посты джона до но не джейн до нам нужно найти способ фильтровать это/и снова мы идем в класс пользователии прямо тут мы пишем что пользоватлеь может иметь много постов/и напишем верни для него все посты/но сейчас – так так у нас проблема/если вернемся обратно – да мы можем создать пост/но сейчас если пост принадлежит пользователю – это не прокатит/потому что это пытается заполнить эти поля но мы не предоставили пользователя/и далее/айди пользователя должны быть равено тому кто написал пост и далее мы должны добавить некоторый уровень защиты/это я покажу в следующем жпизоде но пока – вы уже узнали много нового/

/

Вы не всегда будете иметь роскошный слой аутентификации с нуля. И, кроме того, с точки зрения образования — это полезно научиться выполнять эти шаги вручную самостоятельно. Давайте рассмотрим весь процесс с нуля, не откладывая make:auth.

/