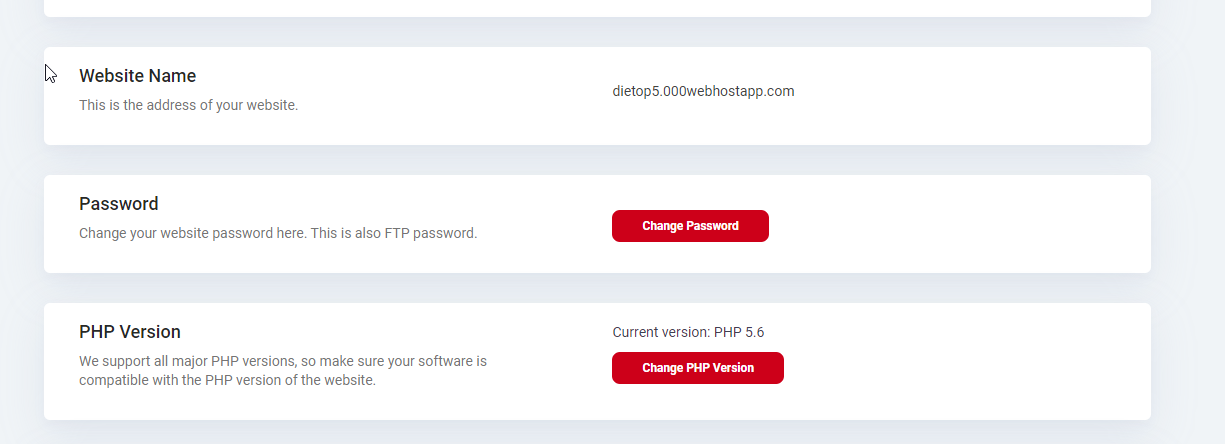
**Документация по использованию фреймворка**

Предварительная настройка

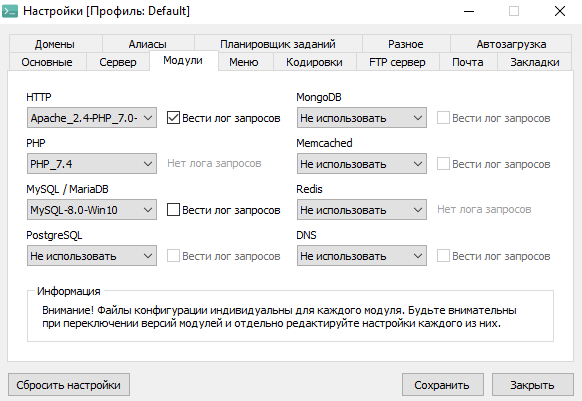
Прежде чем использовать фреймворк, стоит убедится, в правильности настроек сервера. Это PHP фреймворк, поэтому очевидно, что PHP модуль должен быть установлен и включен. Так же фреймворк работает с базой данных MySQL. Версии модулей, на которых он разрабатывался и тестировался: PHP 7.4 и MySQL 8.0

В зависимости от сервера меню настроек модулей может находиться в разных местах и по-разному выглядеть. Вот несколько примеров:

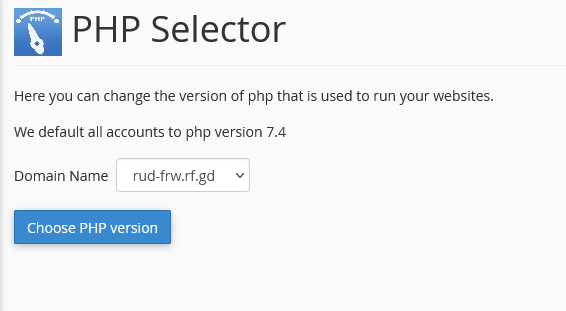
000webhost



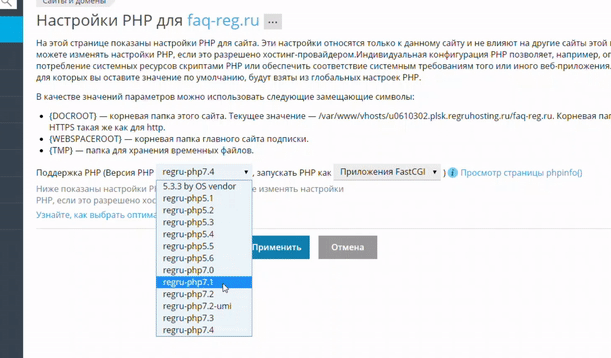
Open server:



infinityFree:



REG.RU:



**Подключение**

Теперь установим фреймворк. Для начала надо скачать сами его файлы. На демонстрационной странице для этого одна ссылка, запутаться негде.



Скачанный архив, надо распаковать в корневой каталог вашего сайта. Это папка, где обычно расположен файл index.php/index.html

Как-то так:



Начнем наконец писать код. Главным файлом фреймворка, через который и происходит подключение, это файл framework.php. Для подключения достаточно выполнить его внутри скрипта вашего сайта, например командой require или require\_once



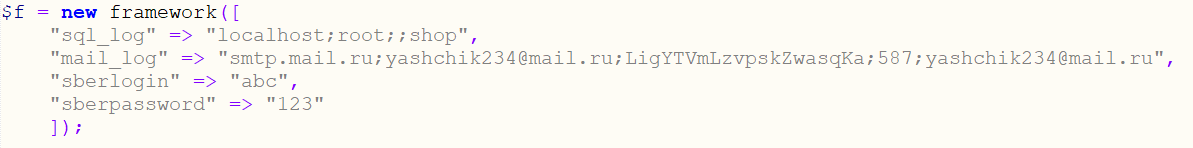
В index’е осталось сделать всего пару действий. Остальное будет происходить за его пределами.

Фреймворк представляет собой обычный класс. Создадим экземпляр класса:



Имя переменной тут конечно же не важно. Как видите, строчка обрывается на квадратной скобке. Это потому, что настройки производятся путем передачи разных параметров через единственный параметр конструктора класса, который является массивом.

Вот примерно самый коротенькой вариант настроек:



Ниже будем разбирать каждый параметр, а пока посмотрим, что дальше.



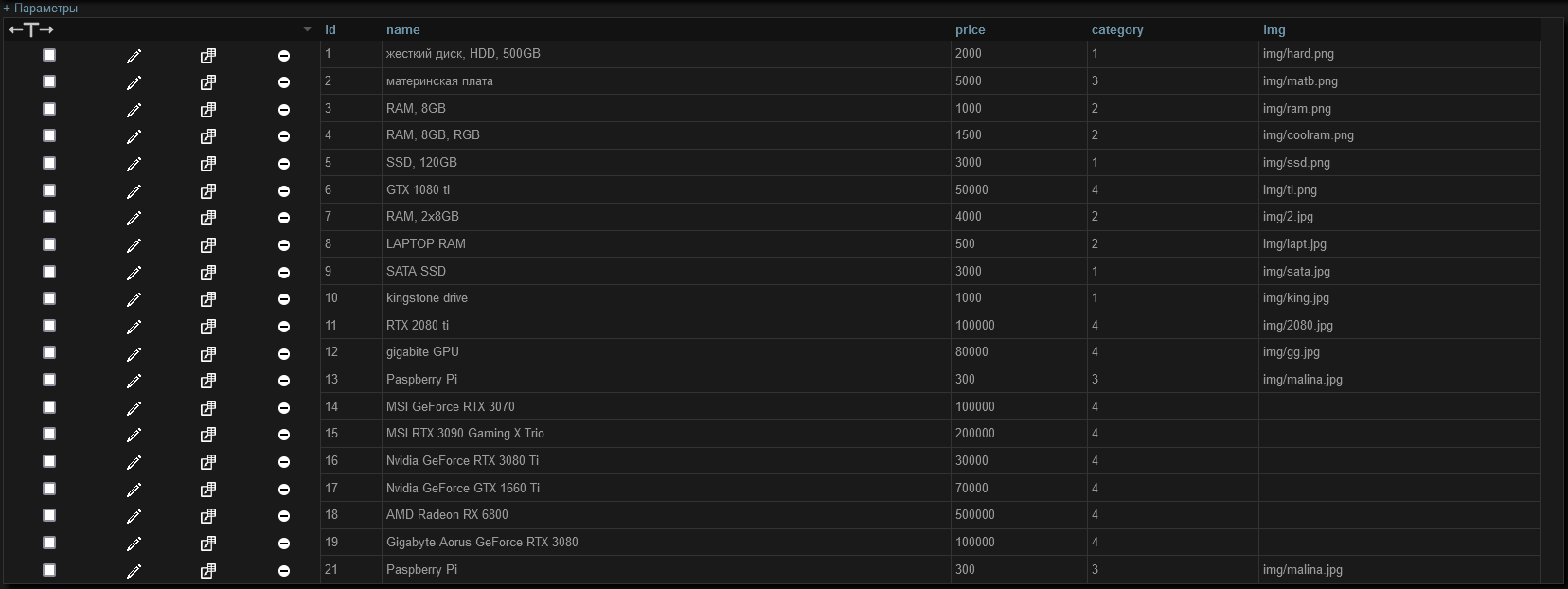
А дальше одна короткая команда, встроенный метод shop, генерирующий весь сайт. Вот так вот.

Именно этот код генерирует пример 1 с демонстрационного сайта:



Конечно же товары и прочее содержимое сайта не из воздуха берется, а заполняется в базу данных администратором сайта. Через phpMyAdmin или другое средство управления базами.

Вот база данного примера:

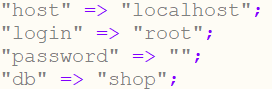


**Параметры**

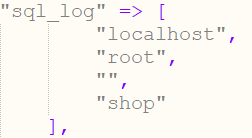
И так что же за параметры такие прописываются в квадратных скобках. Первый параметр sql\_log содержит данные для подключения к бд. В скриншоте, который мы видели ниже они прописаны в одну строку, с точкой с запятой в качестве разделителя. Именно такой там синтаксис если писать одну строку, но так делать необязательно. Если вы хоть раз работали с базой данных не на локальном сервере, то вы понимаете что логины-пароли вероятно сольются в длинную, хаотичную и не читаемую строку, вроде такой:



Конечно компилятору всеравно, насколько красив ваш код, но код пишут живые люди и им самим нужно его как-то понимать, так что рекомендуемо делать так:

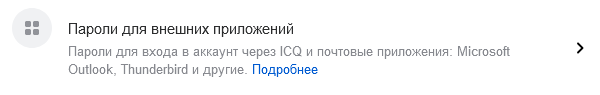


Или вот так:

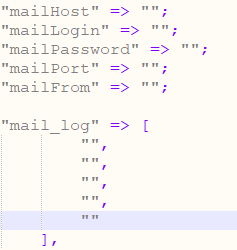


Оба синтаксиса рабочие.

Вслед за sql\_log мы видим mail\_log. Это нужно, чтобы сайт мог автоматически отправлять электронные письма с уведомлениями. Для этого необходимо зарегестрировать аккаунт в каком-либо email сервисе, и в нем получить пароль для внешних приложений. Так это выглядит в mail.ru:



Полученные оттуда данные нужны сайту для отправки имейлов. Ввсети их так же можно тремя разными способами, как и с бд:

, либо в одну строку через “;”

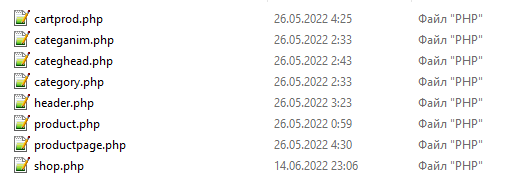
Порядок ввода именно такой. Хост, логин, пароль, порт и адрей ящика. ВАЖНО, что пароль, это не тот пароль, которым в ящик можно войти через графический интерфейс сервера, а специальный пароль для внешних приложений.

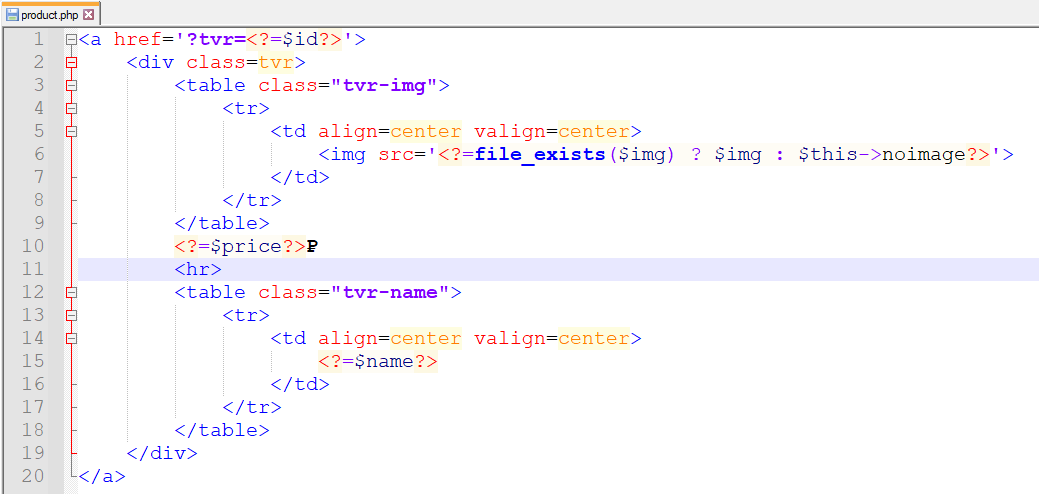
Далее види sberlogin и sberpassword. Все верно это логин и пароль от сбербанка. Конечно просто вводить туда свои данные от аккаунта банка не нужно. Чтобы реализовать автоматические платежи через API сбербанка, нужно быть зарегестрированным в системе эквайринга сбербанка. Заявку нужно подавать через сайт сбербанка, для конкретного сайта конкретной организации. После одобрения от сбербанка вы получите логин и пароль, которые и нужно будет ввести туда.

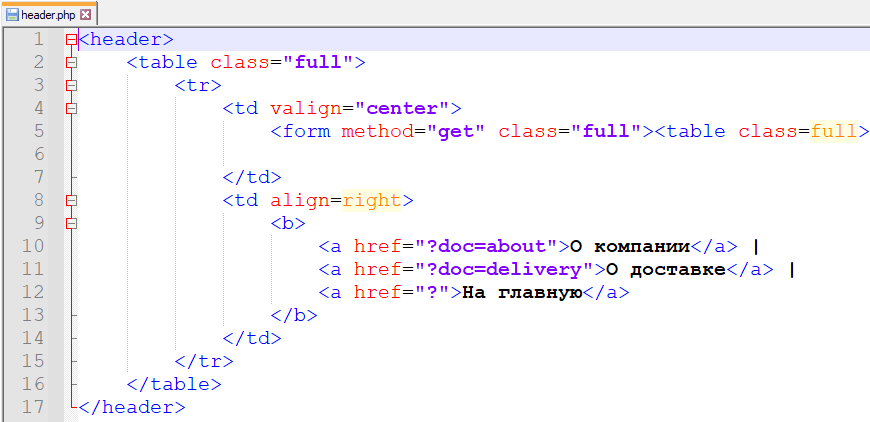
Тут другого синтаксиса уже нет, тк строчки всего две.

**Встроенные и добовляемые виды**

С паролями разобрались. Мы иже видели пример выше, который был сгенерирован после нескольких строк кода в index.php. Но где же хоть какой-нибудь html код? Он по частам прописан в файлах фреймворка. Можно открыть и посмотреть. Директория framework/views





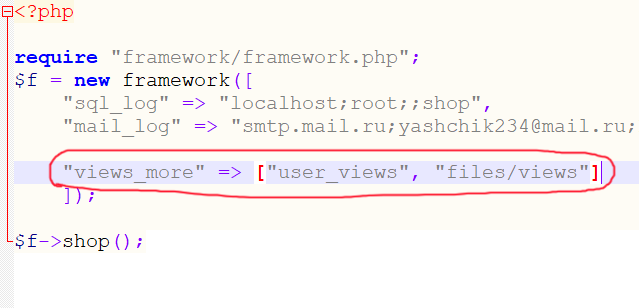


Отдельные повторяющиеся фрагменты расписаны в разных файлах и из них генерируются страницы при помощи алгоритмов расписанных в фреймворке. Такой подход и называется CMS(controller model view; контроллер модель вид). В данный момент нас интересуют именно виды(views)

Резумеется, это не означает, что использовать можно только расписанный заранее внешний вид сайта. Фреймворк расчитан на то, что можно подменять и добовлять свои виды. Конечно можно в ручную залесть в папку views и менять там код, но есть более элегантный способ.

В настройках прописываемых при создании экзэмпляра класса фреймворка, можно указать папку(или несколько папок) содержащих дополнительные виды.

Примерно так:

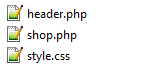


Параметр views\_more при этом должен являтся массивом. Посмотрим на второй пример, который визуально отличается от первого не сильно, потомучто в нем подменено лишь немного кода.

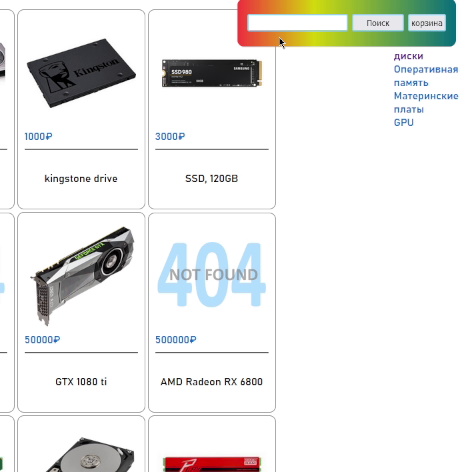
И так, вот его начальные параметры:



Вот папка views



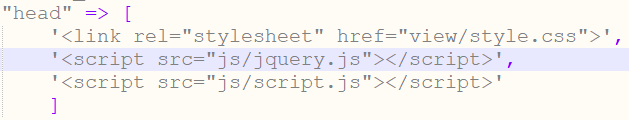
Сразу можно увидет, что подменен header, что и можно лицезреть на сайте



Кроме шапки сайта, видим файл shop.php. Этот файл является особо важным для всего сайта тк именно с него начинается генерация тела любой страницы сайта. Написание сайта на это фреймворке по большей части состоит именно из редактирования shop.php

Немного позже подробно разберем его содержимое. Перед телом страницы имеются заголовки, и как вы могли заметить, дополнительные строчки для заголовков так же прописаны в примере 2

Добавлять строки в <head> можно через параметр head просто добавляя туда текст. Голый текст и двойные кавычки в данном случае не очень удобно, изза того, что прийдется экранировать кавычки внутри кавычек, так что можно делать например так:



Параметр head можно отправлять и как строку и как массив.

**Файл shop.php**

И так, теперь Посмотрим на то, откуда начинает появлятся визуальня часть сайта. Выше мы видели этот файл в файлах конктретного примера, но в фреймворке есть вариант этого файла используемый по умолчанию. Взглянем на него



Рассмотрим из каких частей он тут состоит. Ну во первых мы видим html код. Таблицы, формы и прочее это тут разбирать нет смысла. Их здесь по минимуму, только для наружней разметки.

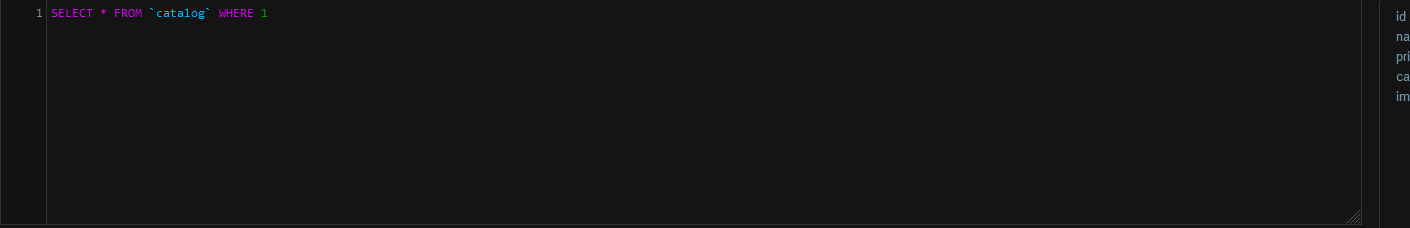
Так же видим ветвление из разных условий, в основмон зависящих от get параметров. Код одной страницы тут немногословен, тк нет повторяющихся элементов, благодаря особенности, о которой подробнее чуть дальше, поэтому в одном файле и прописаны сразу все страницы(всего там 78 строк получается).

И так, что же за особенность. А те самые виды. Просто подключить кусок кода из другого файла можно и команодой require, но фреймворк предоставляет метод render, которым можно за один раз и выполнить запрос к базам данных на языке sql и сразу же сгенерировать html из полученных данных. Для этого должны быть соблюдены следующие условия:

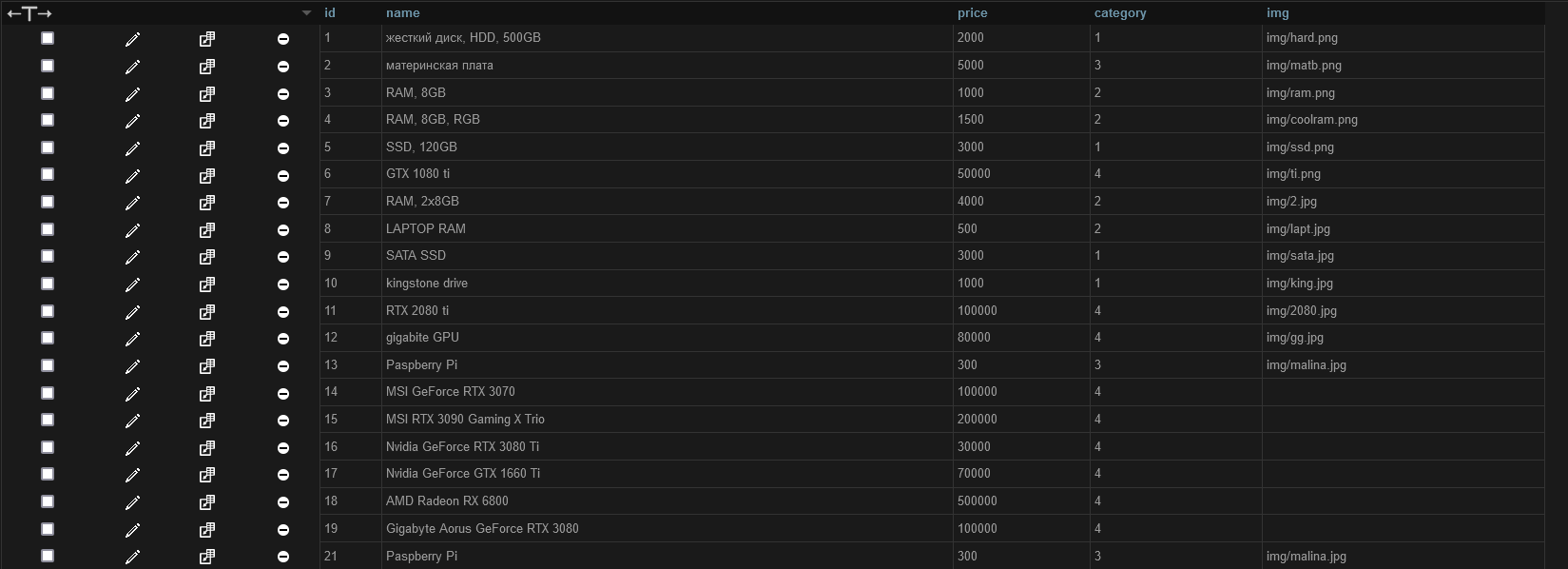
Ответ базы данных должен содержать поля(столбцы, а точнее их наименования) соответствующие тому, что прописано в выбранном виде.

Попробуем более простым языком:

Нам надо сделать список товаров. Мы выполняем метод render, с двумя аргументами: видом и sql-запросом. Sql запрос выглядит примерно так

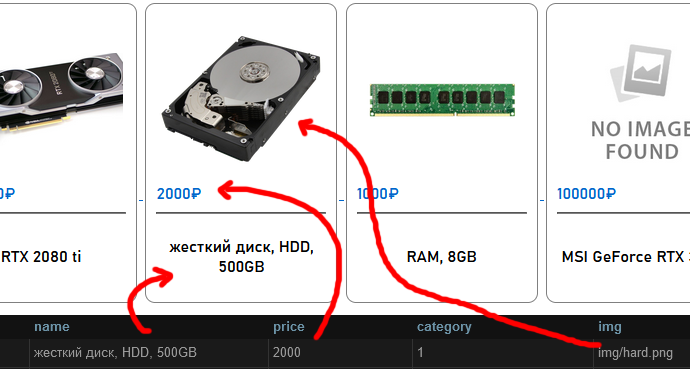


И на него в ответ мы получим таблицу:

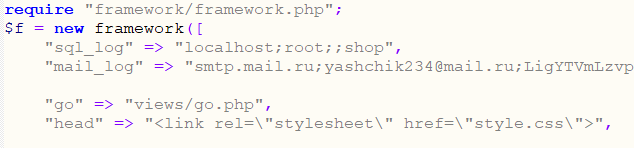


На каждую строку полученной таблицы будет выполнен выбранный вид. Вид в свою очередь может содержать переменные, которые будут заменены на поля из строк таблицы.

Примерно вот так это происходит: 



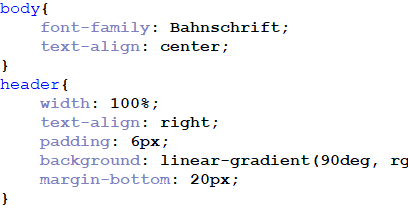
С рендером видов разобрались. Чтобы прописывать shop.php не нужно заменять сам файл в папке у фреймворка. Достаточно прописать один прпметр go. Вот как например это выглядит в примере 3:



**Стили**

Как вы могли догадаться, заранее приписанные стили в фреймворке имеются и они используются по умолчанию. Как добавить свой файл стилей или джаваскрипт выше уже рассказывалось. Время рассказать о встроенных стилях, как их использовать и как их не использовать.

И так перво наперво



Для всего тела страницы прописан шрифт Bahnschrift, и центрирование текста, а для шапки и ширина и отступы и цвет. Отказаться или заменить эти стили можно просто прописав в дополнительнов файле стилей например

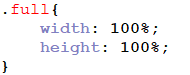
backgroung: none;

width: auto;

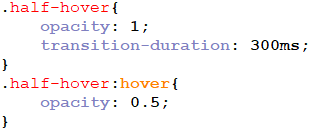
padding: 0;

Кроме просто значений по умолчанию в этих стилях прописаны часто используемые варианты стилей в виде классов которые можно присвоить любым элемебнтам страницы. Например:

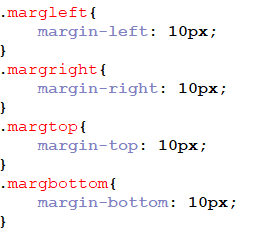
сlass=”full” будет заполнятся на всю шырину и высоту родительского элемента



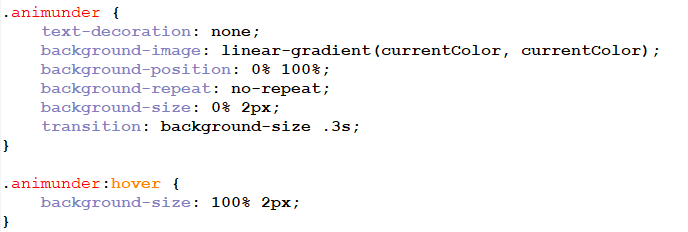
halfhover становиться полупрозрачным при наведении мыши c плавной анимацией



А вот например быстрое добавление небольшого отступа с одной из сторон



animunder элемент с анимацией подчеркивания при наведении мыши



**action.php**

Есть еще один важный файл, который автоматически выполнится, если вы используете shop.php по умолчанию, и который необходимо выполнить вручную, если вы shop.php заменили. В папке фреймворка лежит файл action.php. Никакого визуала он не сгенерирует, но именно в нем прописан код для отправки email’ов и проведения оплаты через API сбербанка. Выполнить его лучше из какого-нибудь из видов(лучше всего прямо в shop.php) функцией require. В любом случае он должен выполнятся изнутри экземпляра класса фреймворка, тк в нем есть обращения через this->