**Описание реализации**

Нужно посмотреть на табличку в БД. У Паши поле image, у меня – imgdishes.

Идём в контроллер, который отвечает за эту табличку – **DishesController.php**.

Для него нужно сделать отдельный **show.php**.

Посмотрим его предка – это **AbstractTableController.php**. У этого предка в конструкторе выполняется

$this->view->setFolder('table');

setFolder() от View, который делали для того чтобы указать ту папку, в которой будет храниться template для конкретно этого контроллера

У нас будут другие template, где будет другая папка.

Поэтому у его потомка перегрузить конструктор и в конструкторе указать другую папку.

    public function \_\_construct(View $view)

    {

        parent::\_\_construct($view);

    }

Что здесь происходит? Мы перегрузили конструктор, т.е. в скелете вызвали родительский конструктор, т.е. выполнили весь тот код, который есть у родителя. И там же, внутри этого родительского кода, прописана строчка, что у setFolder() значение ‘table’.

Значит нам нужно перезаписать это значение. Чтобы вместо папки ‘table’ была ‘dishes’.

$this->view->setFolder('dishes');

Теперь создаём папку ***dishes*** в папке ***templates***. Дальше в папку ***dishes*** копируем файлик **show.php** (показать) и **edit.php** (редактировать) из папки ***table***.

Для чего копируем? Чтобы обычные таблицы работали как работают, а вот именно эту таблицу – блюда, будем изменять под себя.

Теперь идём в файлик **show.php**.

А именно интересует

foreach ($fields as $field)

Тут перебираем все столбцы и для всех столбцов таблицы рисуем обычные input type=’text’. Но для одного конкретного поля – imgdishes, нужно сделать по-другому.

Если поле называется *imgdishes*, то мы его делаем с помощью тега input type=‘file’. В противном случае, рисуем по-другому.

Тег input type=‘file’ – это появляется кнопка «Обзор»

Если сделать так:

foreach ($fields as $field) {

    if ($field == 'imgdishes') {

    } else {

То ничего для поля imgdishes не будет ничего создаваться.

*Примечание*. Чтобы файл загрузился на сервер, то нужно в form указывать (гуглим form file upload): *enctype=”multipart/form-data”*

Теперь это нужно добавить.

Идём в файлик класса Form. Есть два варианта добавления: или во все файлики будем пихать multipart/form-data, или добавить новый метод.

Решили во все. Значит в методе html() между form и action вставляем enctype=”multipart/form-data”.

    public function html()

    {

        return "<form enctype='multipart/form-data'" .

            "action='$this->action'" .

            "method='$this->method'" .

            "$this->class$this->style>\n" .

            "$this->content</form>";

    }

Итак, сделали чтобы на формочке была кнопка «Выбрать файл».

Теперь нужно в контроллере получать этот файлик. С одного места его кидаем в другом месте ловим.

Чтобы словить, у предка есть метод actionAdd(). В котором ловим данные в БД через…

$this->table->add($data['post']);

Мы что-то записываем…

Для начала распечатаем у предка print\_r($\_FILES) – массив, из которого берутся файлики.

После этого жмём «Выбрать файл», находим картинку и жмём «ОК»

Видим следующее:

Array ( [imgdishes] => Array ( [name] => bliny.jpg [type] => image/jpeg [tmp\_name] => E:\OSPanel\userdata\php\_upload\phpD105.tmp [error] => 0 [size] => 283446 ) )

Что пришёл к нам файлик, который называется bliny.jpg, тип у него image/jpeg, находится он во временной папочке и размер у него 283446 байт.

Значит нам нужно эту картинку из этой временной папке переместить в какую-то папку под каким-то именем.

Логика тут может быть такая. Можем просто взять то имя bliny.jpg и под этим именем положить в какую-то папку. Но, если мы так сделаем, то где гарантия, что другой пользователь не загрузит файлик точно с таким же именем? Будет конфликт имён, где придётся кого-нибудь затереть. Данный вариант не подходит.

Есть вариант такой, что если картинка будет одна, то можем добавить запись в таблицу и в таблице каждая запись имеет свой уникальный идентификатор – id. Можем сделать запись в таблицу и просто взять этот id и картинки можем называть по этим id-кам.

Ещё один способ. Можем генерировать какие-то случайные имена для файликов, но генерировать таким образом, чтобы они тоже не повторялись. Для этого можем брать текущую метку времени, с точностью до мс. И к этой метке времени добавить например, случайное стозначное число. Тогда вероятность того, что они совпадут – это должно в 1 секунду несколько млрд картинок прилететь.

Используем вариант с id.

Для этого у предка в методе actionAdd() распечатаем print\_r($data[‘post’]). Там в $data[‘post’] будет все поля, кроме имени картинки:

Array ( [namedishes] => [composition] => [volume] => [unit] => )

Значит нам нужно название картинки bliny.jpg вытянуть из массива $\_FILES, и добавить в массив $\_POST.

Для этого перегружаем метод actionAdd() вкласс DishesController. Но для этого, перед перезагрузкой, мы должны подготовить ему все данные.

$data[‘post’] где не хватает поля *imgdishes*, т.е. $data[‘post’][‘ imgdishes’]. И сюда нужно присвоить $\_FILES (смотрим, когда распечатывали) для которого нужны ключи:

[imgdishes][name]

$data['post']['imgdishes'] = $\_FILES['imgdishes']['name'];

Проверяем в браузере, уже загрузив картинку…

bliny.jpg – это название как на жёстком диске. Теперь после того как bliny.jpg добавили (в БД записали), нужно скопировать в какую-то папку.

Делаем в корне проекта папку ***images*** и в ней папку ***dishes***. Т.е. в папку dishes будут добавляться картинки автоматически.

Идём снова в DishesController. Что мы сделали?

        $data['post']['imgdishes'] = $\_FILES['imgdishes']['name'];

        parent::actionAdd($data);

Мы просто скопировали исходное имя файла, как он назывался у пользователя, в массив post. Потом передали его в метод actionAdd() для того чтобы добавить в БД.

Немного переделаем код. Вместо строчки с parent:: добавим

        $this->table->add($data['post']);

        $this->redirect('?action=show&type=' . $this->getClassName());

Этот метод add() мало того что добавляет данные в таблицу, а кроме этого возвращает id той записи, которую он добавил.

Поэтому нужно вначале объявить переменную $id и записать тот id, который вернули.

$id = $this->table->add($data['post']);

В этой переменной находится id той строки в таблице, в которую мы это всё записали.

Дальше должны скопировать файлик из временной папки в папку ***dishes***. Есть специальная функция move\_uploaded\_file(), которая перемещает загруженный файл в новое место. Т.е. она копирует файл из одного места в другое место, но то что мы копируем оно должно быть загружено в этой сессии пользователя. Т.е. мы не сможем загрузить какой-то файлик вообще из другой папки в другое место.

В данной функции два параметра: string $filename и string $destination.

$filename – это файл из исходного места.

$destination – пункт назначения.

Нужно ещё «загуглить» $\_files чтобы увидеть структуру этого массива. Далее «Загрузка файлов методом POST». Нам нужен: [*$\_FILES['userfile']['tmp\_name']*](https://www.php.net/manual/ru/reserved.variables.files.php)

Tmp\_name – это там, где временное хранение файла.

А также приклеим расширение (гуглим php extract file extension). Есть специальная функция pathinfo()

$ext = pathinfo($\_FILES['imgdishes']['name'], PATHINFO\_EXTENSION);

Где $ext будет расширением

        move\_uploaded\_file(

            $\_FILES['imgdishes']['tmp\_name'],

            "../images/dishes/$id.$ext"

        );

Теперь в папку ***dishes*** сохраняется картинка, например, 8.jpg, где число 8 это id той строки, в которой она записана.

Теперь нужно показать эту картинку на экран нашей таблицы.

Есть несколько вариантов.

У нас сейчас картинки лежат не в папке ***public***. Всё так настроено, чтобы пользователь через браузер мог попасть только в папку ***public***. А дальше через index всё может грузиться. Либо через index будем картинку отдавать, либо можем перенести папочку с файликами внутрь папки ***public***.

Перетянем папку images в папку ***public***. И подправить путь в классе контроллера.

*Примечание*. Если делать ограничения по размеру в загрузке файла, то нужно будет делать проверку. Если размер больше, то не добавлять.

Теперь сделаем чтобы эта картинка отображалась в таблице. Для этого идём в **show.php**. У нас есть некий $table – массив. В этом массиве нужно заменить название картинки на тег.

Для начала…

print\_r($table);

где получим

Array

(

[0] => Array

(

[id] => 1

[namedishes] => Блюдо1

[composition] => Сomponent1, component2

[volume] => 150

[unit] => г

[imgdishes] => dishes1.jpg

)

)

Будем проходить таблицу по строкам. И нужно поставить & чтобы по ссылке передавались чтобы не менялись

foreach ($table as &$row) {

    $row['imgdishes'] = $row['id'];

}

Где на экране будем видеть id-ки вместо названия картинок.

Немного слов про &. Когда foreach работает – берёт массив $table и каждый элемент из этого массива он копирует в перменную $row, т.е. это создаётся новая переменная, в которую копируются значения каждого элемента массива. И чтобы мы не делали с $row внутри цикла, с исходным массивом ничего не происходит – копируются значения из основного массива в $row.

А если ставим &, то это получается не копируются ни значения, а как бы эта переменная становиться ссылкой на часть исходного массива. Если мы что-то изменит с этой переменной, то эти изменения применяться к исходному массиву.

Вернёмся к коду. Чтобы не просто циферки с id были для этого Нужно это дело обернуть тегом <img>

foreach ($table as &$row) {

    $ext = pathinfo($row['imgdishes'], PATHINFO\_EXTENSION);

    $row['imgdishes'] = "<img src='images/dishes/$row[id].$ext' class='img'>";

}

Реализуем чтобы можно было картинку отредактировать за это отвечает файл **edit.php**

Аналогично как делали в show.php.

if ($field == 'imgdishes')

Если поле равно imgdishes, то делали по-другому это поле. По такой же логике делаем в edit.php.

foreach ($fields as $name => $value) {

    if ($name == 'imgdishes') {

        $form->addContent(Html::create('Label')->setFor($name)->setInnerText($comments[$name])->html());

        $form->addContent(Html::create('input')->setName($name)->setId($name)->setValue($value)->setType('file')->html());

    } else {

        $form->addContent(Html::create('Label')->setFor($name)->setInnerText($comments[$name])->html());

        $form->addContent(Html::create('input')->setName($name)->setId($name)->setValue($value)->html());

    }

}

А теперь нужно в контроллере предусмотреть логику, когда к нам приходит картинка. Для этого нужно перегрузить метод

    public function actionEdit(array $data)

    {

print\_r($\_FILES);

        print\_r($data);

        parent::actionEdit($data);

    }

После того как выбрали файл и нажали кнопку «ОК» выводяться след массивы:

Array ( [post] => Array ( [namedishes] => Блюдо 6 [composition] => без состава [volume] => 3 [unit] => г [id] => 6 ) [get] => Array ( [action] => edit [type] => dishes ) ) Array ( [imgdishes] => Array ( [name] => intro4.jpg [type] => image/jpeg [tmp\_name] => E:\OSPanel\userdata\php\_upload\phpC419.tmp [error] => 0 [size] => 578103 ) )

Т.е. всё приходит.

Теперь по той же самой логике нужно будет imgdishes в БД записать

Немного измениться

$id = $this->table->add($data['post']);

На

$id = $data['post']['id'];

В итоге

    public function actionEdit(array $data)

    {

        $data['post']['imgdishes'] = $\_FILES['imgdishes']['name'];

        $ext = pathinfo($\_FILES['imgdishes']['name'], PATHINFO\_EXTENSION);

        $id = $data['post']['id'];

        move\_uploaded\_file(

            $\_FILES['imgdishes']['tmp\_name'],

            "images/dishes/$id.$ext"

        );

        parent::actionEdit($data);

    }

Удаление

Сейчас удалится строчка из БД, а на диске она останется. Чтобы удалить и из папки, то нужно также перегрузить метод удаления.

В чём фишка, мы бы заранее знали, что картинка всегда у нас называется так как id в БД, но из-за того, что у нас расширения могут быть разные.

После print\_r($data) видим:

Array ( [post] => Array ( ) [get] => Array ( [action] => del [type] => dishes [id] => 1 ) )

Получается после $data написать [get] а после [id]

Пишем print\_r($data[‘get’][‘id’]) и увидим

1

Дальше создадим переменную $id и запишем до чего достучались:

$id = $data[‘get’][‘id’]

Теперь нужно найти название картинки, которая в БД под этим id

print\_r($this->table->get(['id'=>$id]));

это нам вернёт строку БД под этой позицией:

Array ( [0] => Array ( [id] => 1 [namedishes] => Блюдо1 [composition] => Сomponent1, component2[volume] => 150 [unit] => г [imgdishes] => dishes1.jpg ) )

А если:

print\_r($this->table->get(['id'=>$id])[0]['imgdishes']);

то на экране увидим

dishes1.jpg

Вот достали id и название картинки. Теперь получим расширение

$ext = pathinfo($img, *PATHINFO\_EXTENSION*);

И когда удалил картинку

unlink("images/dishes/$id.$ext");

После этого вызываем родительский метод, который удаляет из БД строчку.