На этом шаге мы с вами научимся создавать постраничную навигацию по статьям в MODx CMS.

Лирическое отступление

Да, не прошло и полгода, как я продолжил свой цикл :). Хотя обещанного ждут три года, мне все-таки ужасно стыдно перед вами, уважаемые читатели, что вам пришлось ждать так долго. На то были свои причины… Но за это время я получил так много просьб о продолжении цикла! И меня очень радует, что интерес к MODx со временем только растет, а мои статьи действительно оказались полезными.

Ну что, двигаемся дальше?!

Постраничное разбиение

Как вы помните (смею надеяться :), на главной странице сейчас у нас выводится список статей с кратким описанием и ссылкой на полные версии. Этот список ранее мы уже отсортировали по дате добавления, начиная с самых новых статей.

Однако сейчас выводятся абсолютно все статьи, находящиеся в папке "Блог", а это значит, добавляя новые статьи, в итоге мы получим длиннющий лист на главной странице. Это некрасиво и очень неудобно. Следовательно, необходимо ограничивать вывод статей на одну страницу, например, последними пятью статьями, а остальные переносить на следующие страницы.

Вывод статей на главной странице обеспечивает сниппет " Articles ". Нам придется модифицировать его, чтобы создать автоматическое разделение на страницы. Перед тем, как начнем работать непосредственно с программным кодом, прикинем в теории, что именно потребуется сделать:

1. Поскольку сейчас выводятся все статьи на одной странице, нужно сделать ограничение на количество статей. Пусть это будет 5 (пять) статей на страницу; в дальнейшем добавим дополнительную переменную – параметр, содержащий нужное количество статей на одну страницу.
2. Сниппет будет автоматически генерировать навигацию "Назад" – "Вперед"; содержимое ссылок в навигации будет зависеть от выбранной страницы, т.е. перейдя на одну страницу "Назад", ссылка должна соответственно измениться; кроме того, нужно учесть, что будут существовать два момента, когда либо ссылка "Назад", либо ссылка "Вперед" не будет показана (почему? - задание на дом :).
3. Необходимо иметь ввиду, что сниппет будет получать некоторые параметры извне (поговорим об этом ниже), а это всегда сигнал о критическом внимании к безопасности программного кода. Кроме того, нужно учитывать момент оптимизации нагрузки базы данных, поскольку количество статей может быть неограниченным.

Ну что ж, от теории плавно переходим к практическим упражнениям :).

```
1.
2.  $results = $modx->getDocumentChildren(
3.   $ id  = 1, //  id  родительского документа, а именно документа "Блог"
4.     $active = 1, // Выбираем только опубликованные документы
5.     $deleted = 0, // Выбираем только неудаленные документы
6.      ' id , pagetitle, published, introtext, content, menuindex, createdby, createdon, deleted, menutitle'                , // Выбираем поля из БД
7.      $where = '', // Дополнительные условия не требуются
8.      $sort='createdon', // Сортируем документы по полю createdon, т.е. по дате создания
9.   $dir='DESC', // Сортируем документы по убыванию
10.      $limit = '' // Ограничения не устанавливаем (параметр LIMIT в SQL запросе)
11.  );
12.
```

Представленный выше код – это уже известная функция getDocumentChildren из API MODx, которую мы использовали в сниппете "Articles". Обратим внимание на один из ее параметров, а именно параметр $limit. Как понятно из комментария, этот параметр является значением LIMIT в SQL запросе.

Модифицируем слегка наш сниппет, чтобы "прочувствовать" смысл этого параметра в действии:

```
1.
2.  $results = $modx->getDocumentChildren(
3.   $id = 1, // ID родительского документа, а именно документа "Блог"
4.    $active = 1, // Выбираем только опубликованные документы
5.    $deleted = 0, // Выбираем только неудаленные документы
6.     'id, pagetitle, published, introtext, content, menuindex, createdby, createdon, deleted, menutitle'          , // Выбираем поля из БД
7.     $where = '', // Дополнительные условия не требуются
8.     $sort='createdon', // Сортируем документы по полю createdon, т.е. по дате создания
9.   $dir='DESC', // Сортируем документы по убыванию
10.     $limit = '0,5' // Вывод пяти документов, начиная с первого
11.  );
12.
```

Сохраните сниппет и обновите главную страницу: отобразятся только пять последних добавленных статей, начиная с самой новой.

Необходимо отметить, что первая цифра (нуль, по логике базы данных) здесь обозначает первый документ, с которого надо начинать выборку, а вторая цифра (пять) обозначает, сколько всего требуется выбрать документов.

Итак, теперь мы можем ввести дополнительный параметр $num, обзначающий количество статей, а также $start, обозначающий номер документа в выборке, с которого будет вестись отсчет.

```
1.
2.  $num = 5; // Количество статей на одну страницу
3.
4.  $start = 0; // Номер начального документа в выборке
5.
6.  $results = $modx->getDocumentChildren(
7.    $id = 1, // ID родительского документа, а именно документа "Блог"
8.    $active = 1, // Выбираем только опубликованные документы
9.    deleted = 0, // Выбираем только неудаленные документы
```

```
10.    'id, pagetitle, published, introtext, content, menuindex, createdby, createdon, deleted, menutitle'                , // Выбираем поля из БД
11.   $where = '', // Дополнительные условия не требуются
12.      $sort='createdon', // Сортируем документы по полю createdon, т.е. по дате создания
13.   $dir='DESC', // Сортируем документы по убыванию
14.      $limit = $start.",".$num  // Вывод $num документов, начиная с $start
15.  );
16.
```

Судя по результату, когда обновим главную страницу, разницы нет. Но так и должно быть, мы просто вынесли один из параметров. Изменяя его значение, в дальнейшем можно легко менять количество выводимых статей на одной странице.

```
1.
2.  $limit = $start.",".$num  // Вывод $num документов, начиная с $start
3.
```

Задумаемся теперь о первом значении $start параметра LIMIT, т.е. в данном случае нуле. Еще раз – он обозначает стартовый документ, с которого начинается выборка из базы данных.

Стартовый документ на главной странице равен нулю. Пусть наша текущая страница тоже будет иметь номер 0 (нуль).

Хорошо, а что если мысленно представить, что мы нажали ссылку "Назад" в навигации? Это значит, что теперь текущая страница получила порядковый номер 1 (один), а отсчет документов в выборке должен начаться уже с 5-го (пятого) документа. При этом второе значение 5 (пять), т.е. параметр $num, у нас неизменно, т.к. общее количество выводимых статей на одну страницу всегда одинаковое.

Теперь также мысленно шагаем еще на одну страницу назад. Текущая страница получает порядковый номер 2, отсчет документов начинается с 10-го (десятого)

документа.

Еще назад! Текущая страница – порядковый номер 3, отсчет документов – с 15-го. И так далее.

Теперь нетрудно заметить зависимость в этих последовательностях: первое значение ($start) в LIMIT есть произведение номера текущей страницы ($p) на количество выводимых статей одной страницы ($num), т.е. $start = $p * $num:

```
1.
2.  $p = 0; // Номер текущей страницы
3.
4.  $num = 5; // Количество статей на одну страницу
5.
6.  $start = $p * $num; // Номер документа в выборке, с которого будет вестись отсчет
7.
8.  $results = $modx->getDocumentChildren(
9.      $id = 1, // ID родительского документа, а именно документа "Блог"
10.     $active = 1, // Выбираем только опубликованные документы
11.     deleted = 0, // Выбираем только неудаленные документы
12.    'id, pagetitle, published, introtext, content, menuindex, createdby, createdon, deleted, menutitle' , // Выбираем поля из БД
13.   $where = '', // Дополнительные условия не требуются
14.     $sort='createdon', // Сортируем документы по полю createdon, т.е. по дате создания
15.   $dir='DESC', // Сортируем документы по убыванию
16.     $limit = $start.",".$num  // Вывод $num документов, начиная с $start
17.  );
18.
```

Попробуйте поменять значение параметра $p = 1, $p = 2 и т.д. Сохраняя сниппет и обновляя затем главную страницу, вы будете видеть, что будут выводиться разные статьи, как если бы вы переходили по ссылкам "Назад" и "Вперед".

Вручную менять эти значения как-то некрасиво, не правда ли? :) Значит, нам нужно передавать сниппету эти значения $p извне, чтобы можно было эмулировать переход по ссылкам постраничной навигации. Этого легко добиться,

используя суперглобальный массив $_GET, т.е. значения, передаваемые в URL, могут быть доступны в любом сниппете, в том числе и нашем, конечно.

Попробуем это реализовать:

```
1.
2.  $p = $_GET["p"]; // Номер текущей страницы
3.
4.  $num = 5; // Количество статей на одну страницу
5.
6.  $start = $p * $num; // Номер документа в выборке, с которого будет вестись отсчет
7.
8.  $results = $modx->getDocumentChildren(
9.   $id = 1, // ID родительского документа, а именно документа "Блог"
10.     $active = 1, // Выбираем только опубликованные документы
11.     deleted = 0, // Выбираем только неудаленные документы
12.    'id, pagetitle, published, introtext, content, menuindex, createdby, createdon, deleted, menutitle'    , // Выбираем поля из БД
13.   $where = '', // Дополнительные условия не требуются
14.      $sort='createdon', // Сортируем документы по полю createdon, т.е. по дате создания
15.   $dir='DESC', // Сортируем документы по убыванию
16.      $limit = $start.",".$num  // Вывод $num документов, начиная с $start
17.  );
18.
```

После обновления и сохранения сниппета, откроем главную страницу и добавим к адресу параметр ?p=1, например, так: http://localhost/modx/?p=1. Перейдем по этому адресу и, меняя значение p=0, p=1, p=2,.. и т.д., в итоге получим то же самое, как при экспериментах с ручным изменением значения $p напрямую в сниппете.

Кстати, очень важно обратить внимание на то, что значением $p может стать любой символ. Это к вопросу о безопасности работы с внешними данными. Попробуйте сейчас ввести p=-1… Ой, ошибка. Почему она здесь появилась? Да все потому, что по логичному мнению базы данных, отрицательного стартового значения не может быть в принципе. А он у нас получается именно отрицательным, смотрите сами:

Если $_GET["p"] передает значение, равное -1 (минус один), то произведение $start = $p * $num даст нам значение -5 (минус пять), т.к. $num в нашем случае соответствует 5 (пяти). При этом в SQL запросе получается бессмыслица: LIMIT -5,5, что в результате и приводит к критической ошибке.

Каков вывод из этого может следовать? Очень простой – нужно всегда жестко контролировать все внешние параметры, т.е. те параметры, которые имеют значение для программного кода и могут быть изменены пользователями случайно или специально. Приведенный выше пример ошибки – самое мягкое, что может случиться. Используя подобные "дыры", злоумышленники могут внедрить вредоносный код, что часто приводит к неприятным последствиям.

Итак, обязательный жесткий контроль над внешними параметрами. Какие методы при этом используются – это тема не моей статьи. Для желающих всегда доступно море информации в Google.

Мы же ограничимся в данном случае тем, что разрешим вводить пользователю только неотрицательные целые числа, создав специальную функцию *numeric* для проверки этих данных:

```
1.
2.  // Проверяет, что переданное значение - неотрицательное целое число
3.  // Возвращает TRUE/FALSE
4.  function numeric($str) {
5.      return (! ereg ("^[0-9]+$", $str)) ? false : true;
6.  }
7.  // Проверяем, что $_GET["p"] содержит только цифры от 0 до 9
8.  // Иначе присваиваем переменной $p = 0
9.  if (numeric($_GET["p"])) {
10.   $p = $_GET["p"];
11.  }
12.  else {
13.      $p = 0;
14.  }
15.
```

Можем снова поэкспериментировать. Как видно, ?p=-1 уже не вызывает ошибок. В коде сниппета все "неправильные" значения автоматически заменяются на 0 (нуль).

Теперь попробуйте ввести какое-нибудь большое число, например, ?p=1000. Такое значение вполне допустимо в нашем коде. Что же мы видим? Правильно – ничего. Пустая страница. Это логично, поскольку у нас еще нет 5000 статей на сайте.

Однако это некрасиво – выдавать пустую страницу – и, строго говоря, неправильно с точки зрения хорошего программного кода. Такие ситуации тоже должны учитываться и исправляться. Чтобы достичь этого, нам нужно заранее, еще до выполнения запроса в БД, знать общее количество страниц. Запросы, в которых будут значения $_GET["p"], превышающие возможное количество страниц, будут просто игнорироваться.

С помощью следующего SQL кода можно легко получить общее количество статей:

```
1.
2.  SELECT COUNT( * ) AS cnt
3.  FROM `modx_site_content`
4.  WHERE `parent` =1
5.  AND `published` =1
6.  AND `deleted` =0
7.
```

Этот код почти равнозначен SQL запросу, формируемому функцией getDocumentChildren, за исключением того, что нам важно получить только общее количество статей. В данном случае сортировка по какому-то полю и/или значение каких-либо полей нас не интересуют вообще, т.к. эти данные никак не влияют на возвращаемое количество.

Чтобы сделать запрос в БД и затем его обработать, воспользуемся еще двумя функциями API: query и getRow . Строго говоря, эти функции относятся не к

самому API MODx, а к API базы данных MODx, поскольку позволяют работать напрямую с базой данных. Это так называемое DB API , хотя для простоты можно считать их как одно целое.

```
1.
2.  $num = 5; // Количество статей на одну страницу
3.
4.  $sql = "
5.   SELECT COUNT( * ) AS cnt
6.     FROM `modx_site_content`
7.     WHERE `parent` =1
8.    AND `published` =1
9.   AND `deleted` =0
10.  ";
11.
12.  $res = $modx->db->query($sql); // Выполняем запрос в БД с помощью функции API db->query
13.  $row = $modx->db->getRow($res); // Формируем массив из возвращенного результата запроса с помощью функции API db->getRow
14.  $totalArticles = $row["cnt"]; // Получаем количество всех статей
15.
16.  // Получаем общее количество страниц, округляем их в большую сторону и вычитаем 1 (единицу),
17.  // чтобы наши расчеты полностью совпадали с логикой базы данных (помните, что отсчет в БД начинается с нуля?)
18.  $totalPages =  ceil  ($totalArticles / $num) - 1;
19.
20.  // Проверяем, что переданное в URL значение текущей страницы не больше, чем общее количество всех страниц
21.  // Иначе принудительно устанавливаем максимально возможное значение страницы, равное $totalPages
22.  if ($p > $totalPages) {
23.   $p = $totalPages;
24.  }
25.
26.  $start = $p * $num; // Номер документа в выборке, с которого будет вестись отсчет
27.
```

В комментариях к коду я подробно описал, что именно запрашивается или вычисляется, поэтому не будем здесь останавливаться. Просто поэкспериментируйте с параметром ?p в URL снова. Теперь наш код не позволит ввести слишком большое значение – он его просто проигнорирует и автоматически сделает

выборку только по максимально возможному номеру страницы.

Ну, можно считать, что мы достигли цели :). Хм.. как же так, ведь навигация как не работала, так и не работает?! Ой, действительно, исправляемся…

Найдите следующий код в конце сниппета:

```
// Просто добавляем статический блок будущей навигации
$output = "
<div id="pagination">
<a href="#">< Назад</a>
<a href="#">Вперед ></a>
</div>
";
```

И замените его следующим кодом:

```
// Добавляем динамически формируемую навигацию
$output = " <div id="pagination">";

// Если $p = 0, значит мы находимся на первой странице и ссылку "Вперед" не нужно показывать
if ($p == 0) {
  $output .= "
   <a href="?p=".($p+1).""&lt; Назад</a>
   ";
}
// Если $p = $totalPages, значит мы находимся на последней странице и ссылку "Назад" не нужно показывать
else if ($p == $totalPages) {
 $output .= "
   <a href="?p=".($p-1).""Вперед &gt;</a>
 ";
}
// Если оба варианта не подошли, значит мы где-то посередине между первой и
```

последней страницей

```
// Следовательно, показываем обе ссылки в навигации
else {
    $output .= "
   <a href="?p=".($p+1).""&gt;&amp;lt; Назад</a>
   <a href="?p=".($p-1).""&gt;Вперед &amp;gt;</a>
";
}

// Просто закрываем блок навигации
$output .= "
</div>
";
```

Вы не поверите, но это все :)! Остается только выложить обновленный [полный код сниппета Articles]

.

Статья позаимствована с официального сайта Modx