Работа с сайтом MODx (инструкции для администратора)

В публикации использованы обучающие материалы с сайта https://itchief.ru

Ресурсы в MODX

Статья, в которой рассмотрим понятие и роль такого объекта MODX Revolution как ресурс. Кроме этого познакомимся с тем, как осуществляется работа с ресурсами в MODX, разберём назначение полей ресурса, а также то, как связанный с ресурсом шаблон влияет на его вывод.

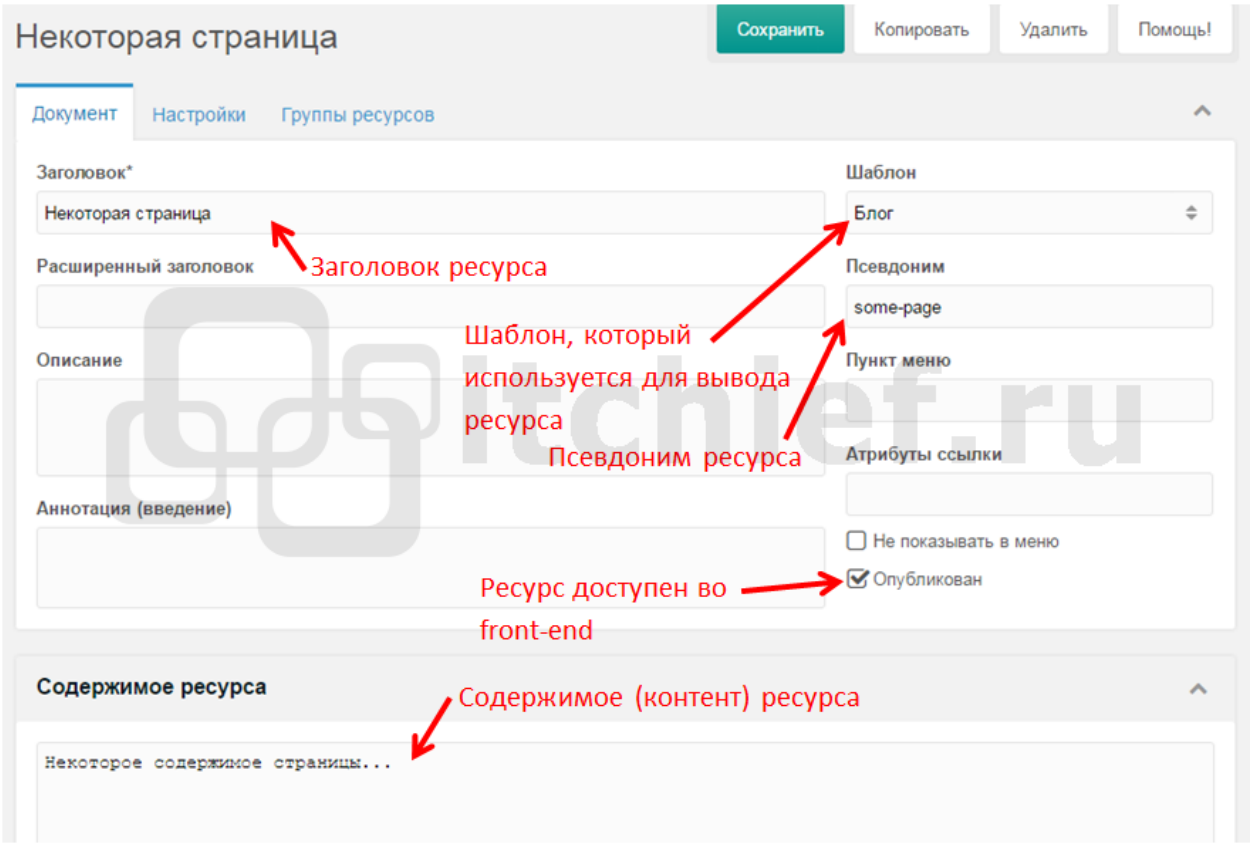
Что такое ресурс

Ресурс - это некоторая сущность, которая представляет страницу в MODX.

Всего в MODX насчитывается 4 различных типа ресурсов: документы (documents), ссылки (weblinks), символические ссылки (symlinks) и статические ресурсы (Static Resources). По умолчанию ресурс имеет тип "Документ".

Ресурс "Документ (document)"

Документ - это тип ресурса MODX, который представляет собой веб-страницу на сайте.



Ресурс "Символическая ссылка (symlink)"

Символическая ссылка в MODX Revolution - это ресурс, с помощью которого можно отобразить контент другого ресурса сайта под определённым URL.

Символическая ссылка не позволяет изменить "привязанный" к ней контент, но зато позволяет настроить иные параметры ресурса, например, такие как шаблон, название и др.

Символическую ссылку обычно используют:

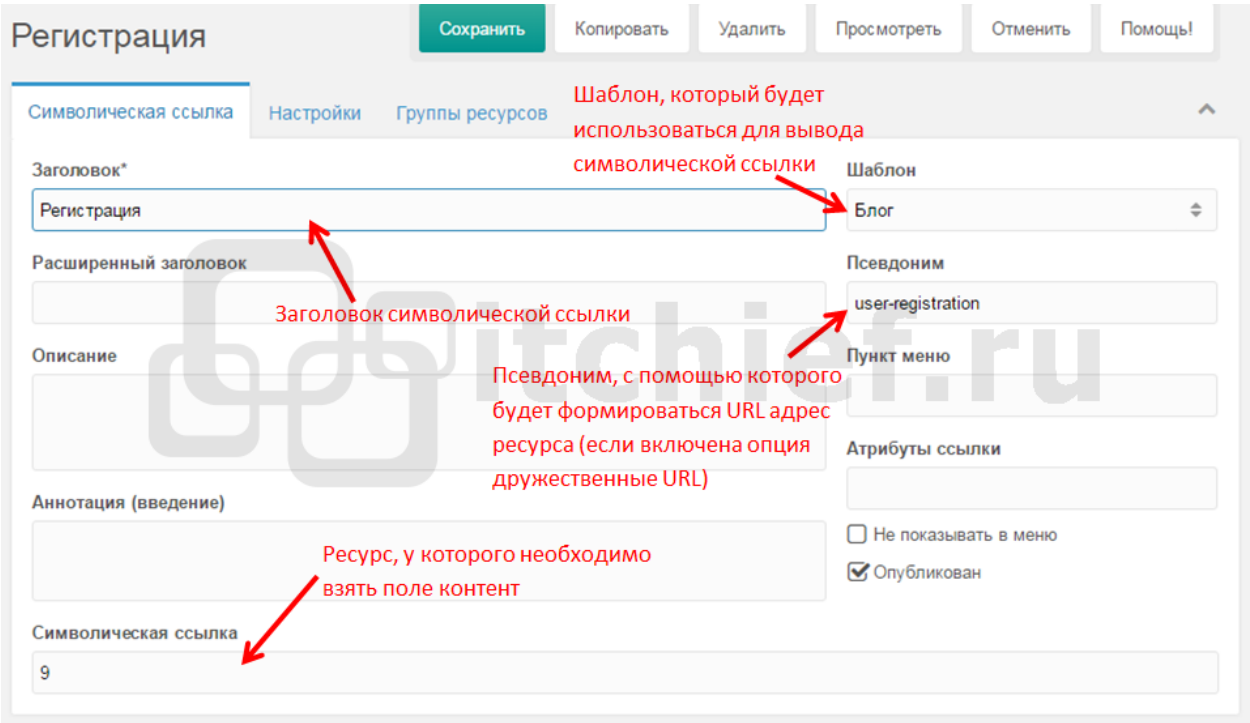
как вид редиректа 301;

для того, чтобы избежать дублирования контента. Т.е. ресурс (контент), имеющий тип "Документ", можно создать один раз, а затем при необходимости использовать символическую ссылку на него. Это может применяться, когда необходимо обеспечить чтобы один и тот же контент находился фактически и логически более чем в одном месте на сайте;

когда необходимо оставить доступ к документу, который был перемещён в структуре сайта, по старой ссылке;

для обеспечения простого и краткого URL-адреса для документа, который находится глубоко в структуре сайта;

для того чтобы реализовать обращение к некоторому ресурсу (контенту), используя различные написания. Например: foursquare, 4square, forsquare и т.п. Каждый из этих ресурсов будет представлять собой символическую ссылку, указывающую на один и тот же контент.



Кроме этого, символическим ссылкам, как и другим ресурсам MODX можно настраивать разрешения. Это достигается посредством помещения символических ссылок, в необходимые группы ресурсов.

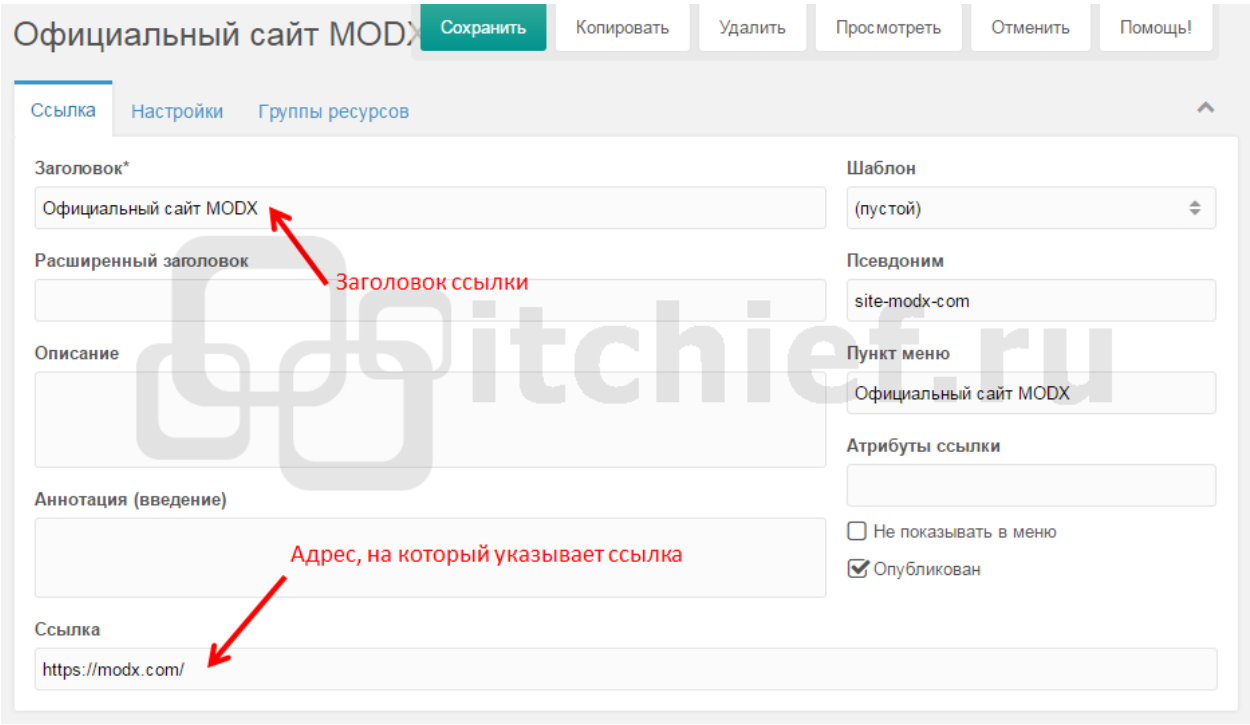
При работе с символическими ссылками убедитесь в том, что ресурс, с которого она берёт контент, существует и опубликован. Если это не так, то при обращении к ресурсу (символической ссылки) будет создана ошибка.

Ресурс "Ссылка (Weblink)"

Weblink - это ресурс, представляющий собой ссылку. Шаблон для отображения или форматирования этого ресурса (ссылки) обычно не используется. Его применение может быть обусловлено только в том случае, если он будет использоваться как контейнер для переменных шаблона (TV-параметров), которые необходимо добавить к ссылке. Weblink можно использовать на сайте, например, в качестве ссылки, которая должна стать частью сгенерированного меню.

Контент веб-ссылки представляет собой URL-адрес. MODX при открытии такого ресурса просто перенаправляет пользователя на указанный адрес URL. Даже парсер такой тип ресурса не разбирает. Т.е. как только MODX видит, что это "ссылка", он просто использует контент этого ресурса в качестве аргумента метода sendRedirect($url).

В качестве ссылки можно использовать как внешний URL-адрес, так и id существующего ресурса сайта.



Статический ресурс (Static Resource)

Статический ресурс - это абстрактный ресурс, представляющий собой некоторый файл. Он предназначен для виртуального представления некоторого файла веб-сервера. Как и другие MODX ресурсы, эта абстракция может иметь права. Данный тип ресурса, также как и другие, отображается в админке (менеджере) MODX на левой панели в дереве ресурсов. Это позволяет использовать его в динамических меню и результатах поиска.

В поле контента статического ресурса для указания пути к файлу можно использовать теги (параметры системы). Кроме системных тегов в поле контента можно также использовать сниппеты. Это в основном используется только тогда, когда необходимо установить динамический путь к файлу.

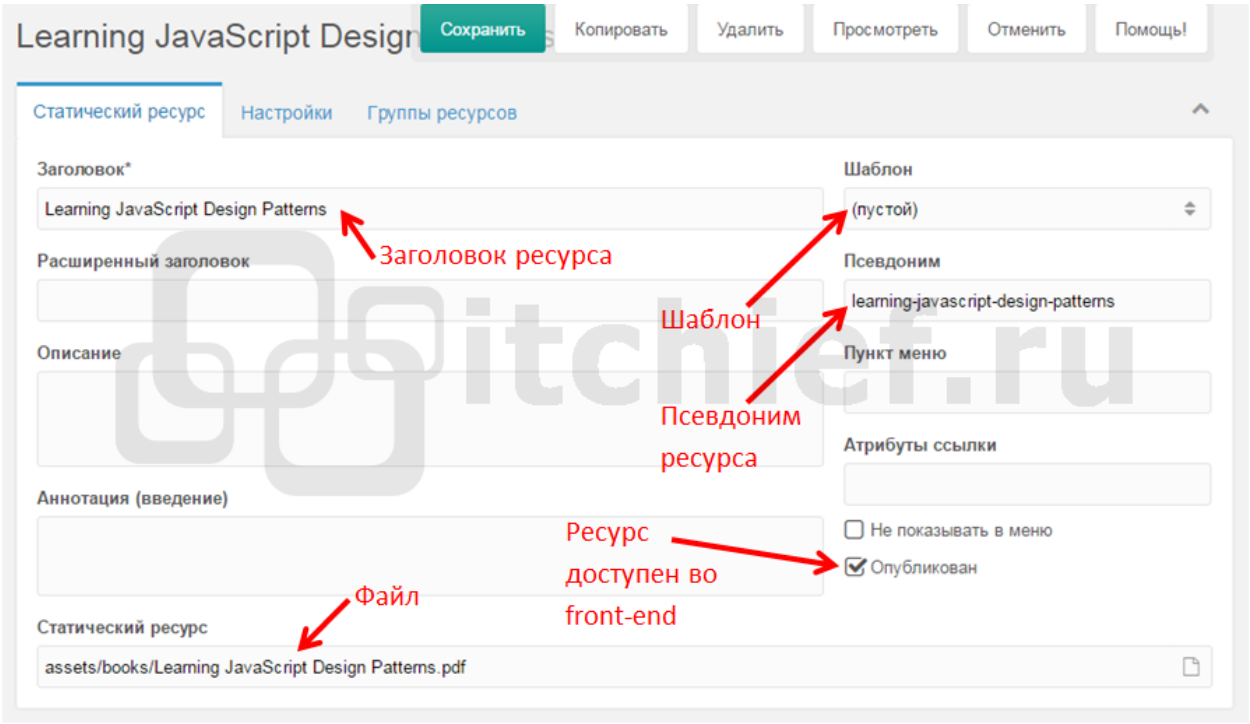
Статические ресурсы в MODX Revolution ведут себя подобно документу (стандартному ресурсу). Но для представления файла в статическом ресурсе, ему обязательно необходимо установить соответствующее содержимое (content_type), а также способ того, как браузер должен обработать этот ресурс (content_dispo). Для того чтобы ресурс отображался в браузере, выберите в качестве

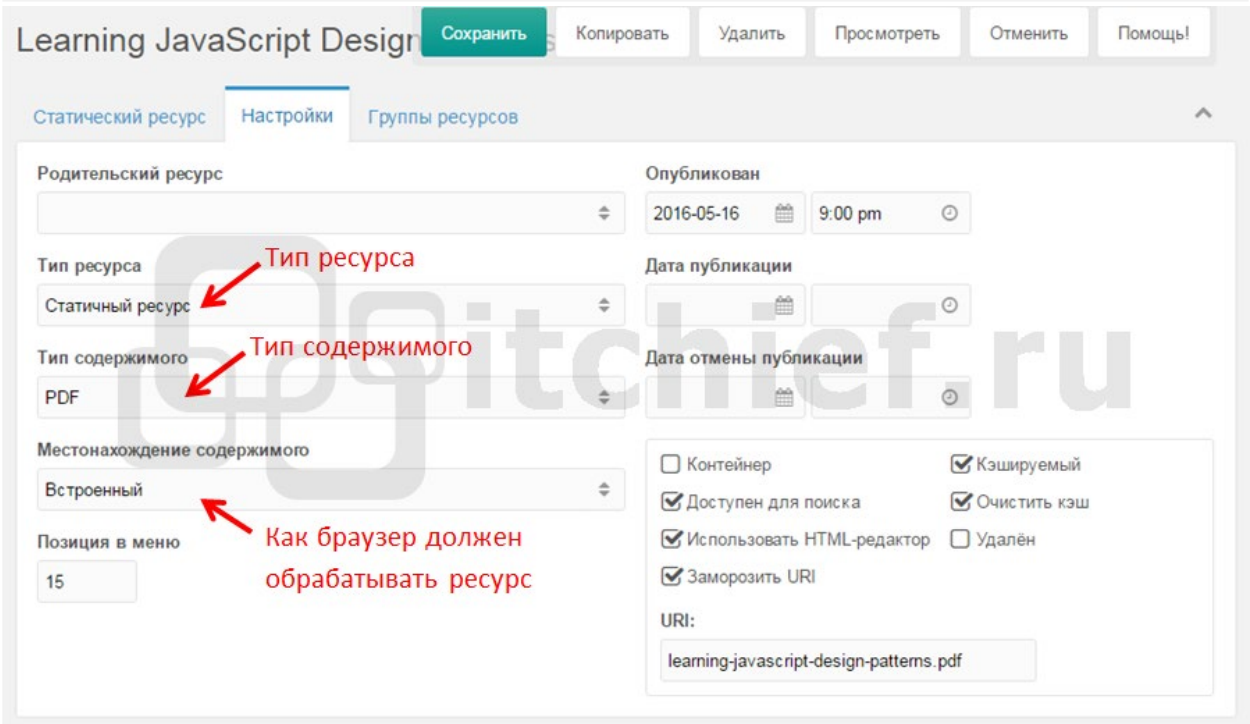
значения поля "Местонахождение ресурса" вариант "Встроенный". Для того чтобы ресурс стал доступен для скачивания выберите параметр "Прикреплённый".

Обратите внимание на то, что при создании статического ресурса, псевдоним alias не должен содержать расширение файла. Расширение файла определяется типом контента ресурса.

Для того чтобы файл передавался правильно через статический ресурс, ему необходимо в качестве шаблона указывать пустой шаблон. Если к статическому ресурсу необходимо добавить некоторую дополнительную информацию через переменные шаблона (TV-параметры). То в этом случае необходимо создать новый "пустой" шаблон и установить в качестве его содержания следующее:

[[*content]]





Идентификатор ресурса

Каждый ресурс в MODX имеет уникальный идентификатор (id). Он позволяет системе MODX определить, какой ресурс необходимо выбрать, когда пользователь загружает в браузере ту или иную веб-страницу. Его также желательно использовать, когда необходимо организовать связь между ресурсами. В этом случае MODX будет генерировать ссылку автоматически, и её не придётся редактировать при изменении псевдонима (alias), типа контента или какого-нибудь другого поля ресурса.

Рассмотрим небольшой пример, а именно то, как MODX определяет какой ресурс необходимо отдать пользователю при открытии им некоторой страницы на сайте (www.mysite.ru).

Если на сайте не используются дружественные URL, то ресурс, который необходимо отобразить пользователю указывается посредством идентификатора в get параметре id страницы "index.php".

Например, отобразить в браузере страницу (ресурс) сайта, имеющий идентификатор 16:

http://www.mysite.ru/index.php?id=16

Если же на сайте включены и настроены дружественные URL, то обращение к ресурсу уже будет осуществляться с использованием псевдонима. Например, отобразить ресурс с идентификатором 16, имеющий псевдоним some-page и тип содержимого "html" (суффикс, равен .html):

http://www.mysite.ru/some-page.html

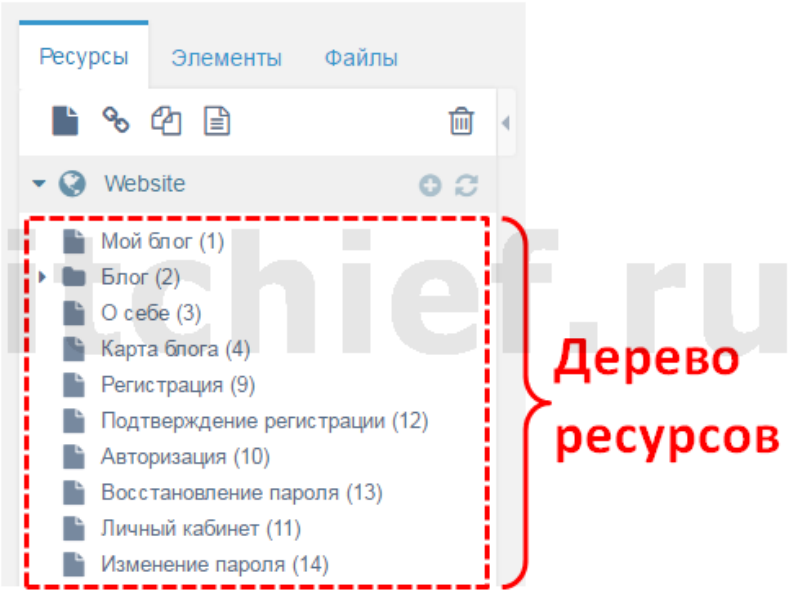
Здесь some-page - это alias ресурса, а .html - это тип содержимого (контента) данного ресурса.

Кроме этого, MODX разрешает использовать вложенные URL. Например, ресурс с псевдонимом "child", который расположен внутри ресурса-контейнера с псевдонимом "parent", будет иметь адрес:

http://www.mysite.ru/parent/child.html

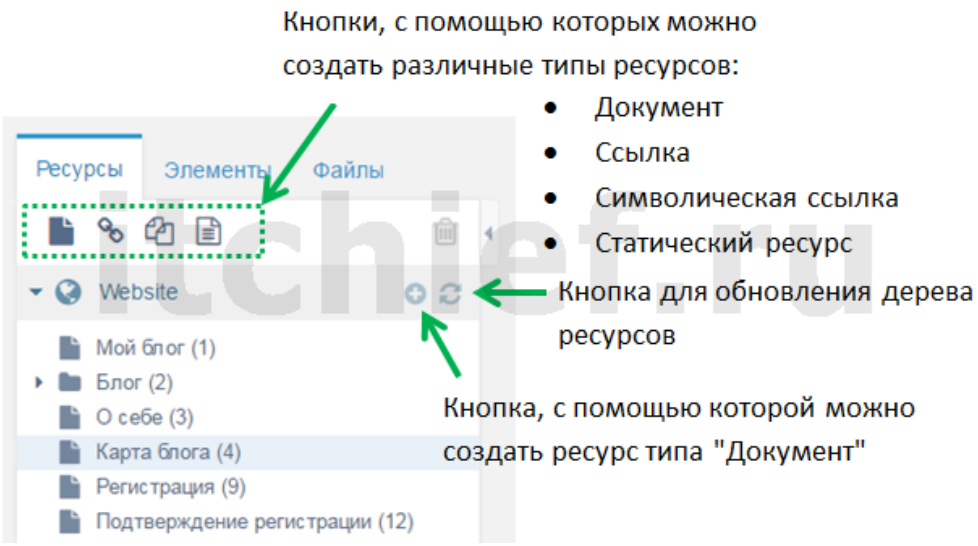
Работа с ресурсами в MODX

Ресурсы в админке (менеджере) MODX отображаются в левой панели на вкладке "Ресурсы". Представляются они пользователю в виде дерева.

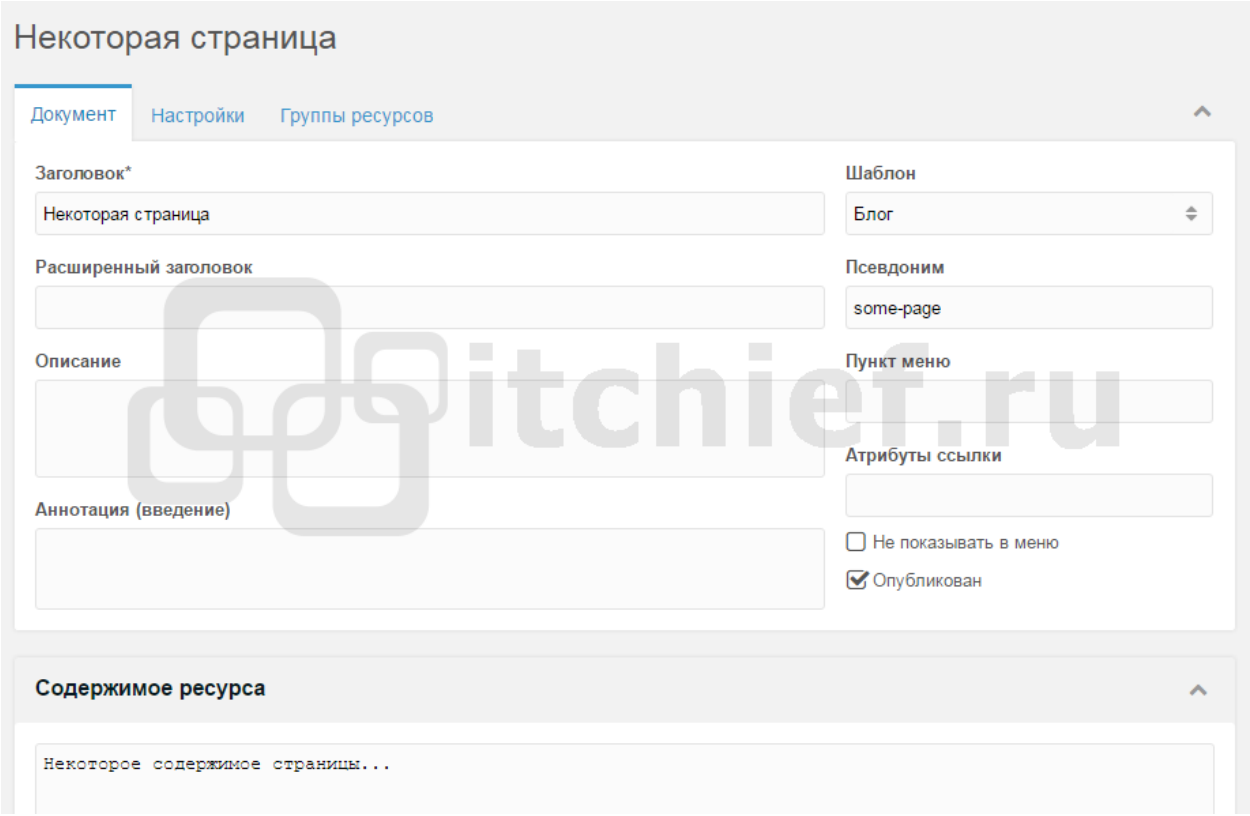


Создание ресурса

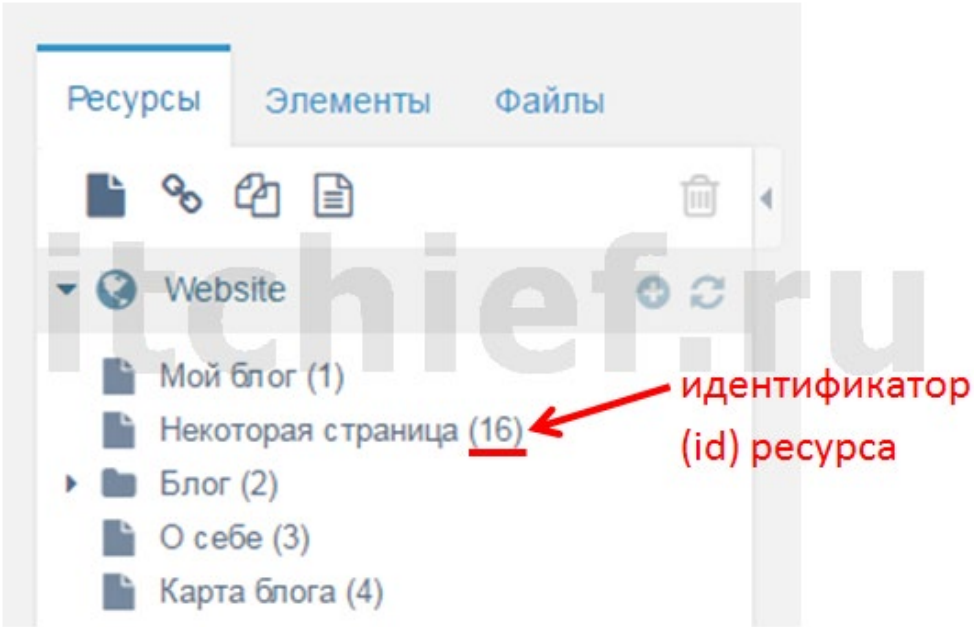
Создание ресурса в MODX осуществляется с помощью кнопок расположенных на соответствующей панели в левой части админки.



После этого на экране появится страница ресурса, состоящая из вкладок и панели "Содержимое ресурса". Все эти элементы содержат поля ресурса.



После заполнения необходимых полей и нажатия на кнопку "Сохранить", они заносятся в базу данных, а ресурсу присваивается некоторый идентификатор (порядковый номер ресурса).

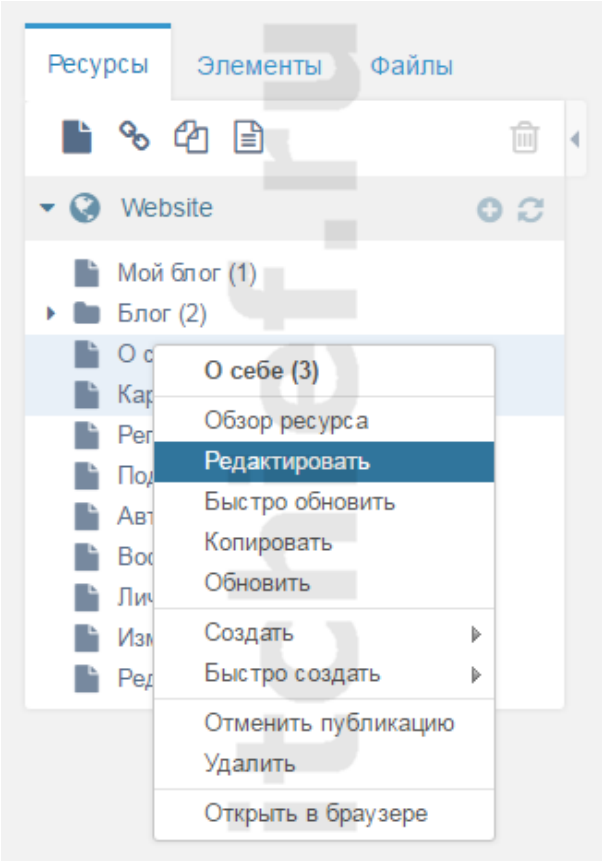


Редактирование ресурса

Отредактировать ресурс в админке MODX можно несколькими способами:

с помощью нажатия левой кнопкой мыши на необходимый ресурс в дереве;

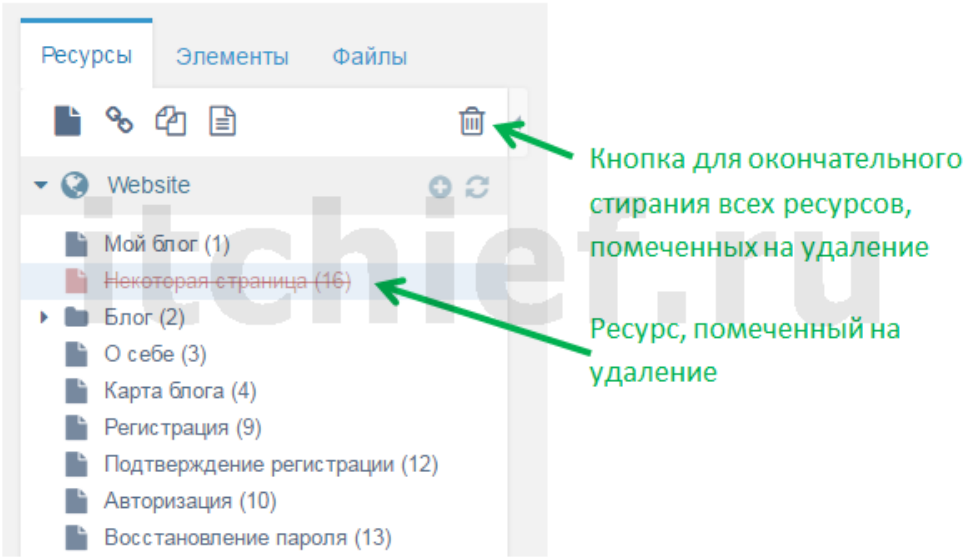
посредством поднесения курсора к определённому ресурсу и выбором из контекстного меню пункта "Редактировать". Вызывается контекстное меню с помощью правой кнопки мыши.



После редактирования ресурса, его необходимо сохранить. Для этого необходимо нажать на кнопку, расположенную вверху экрана и имеющее соответствующее название.

Удаление ресурса

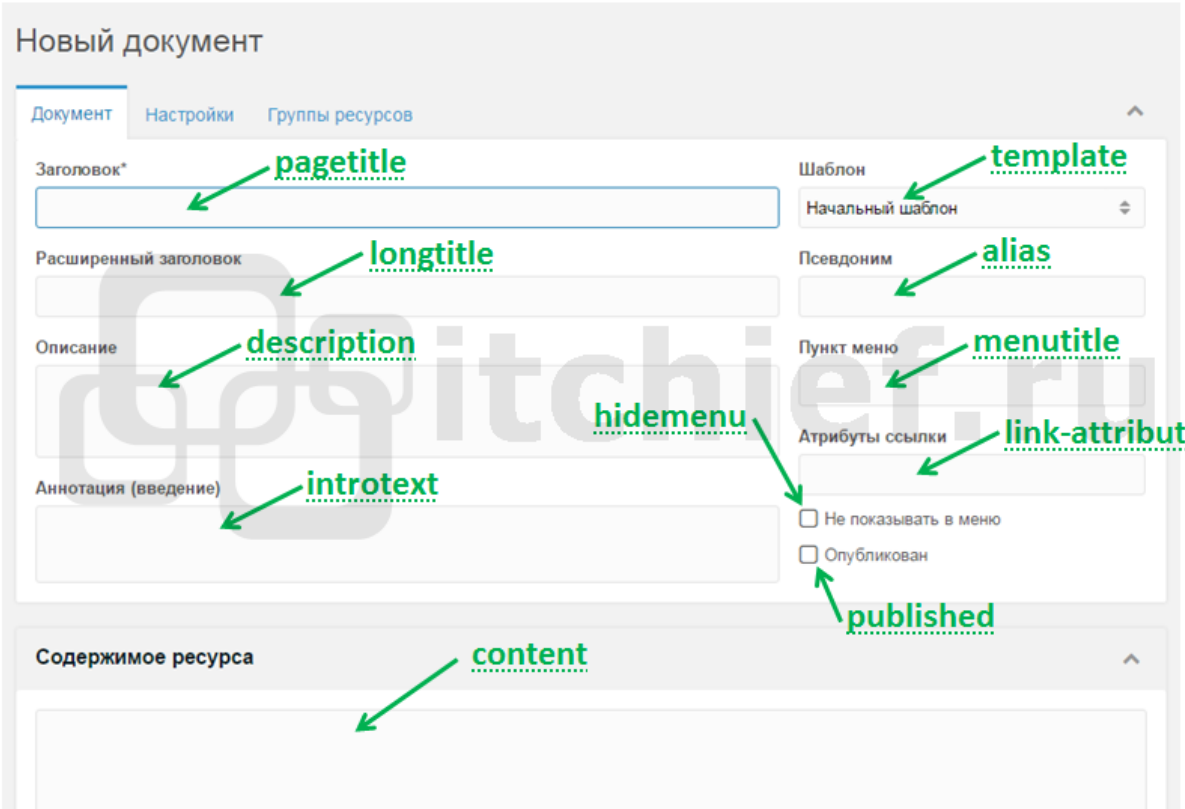
Для удаления ресурса в MODX Revolution к нему необходимо поднести курсор и в контекстном меню выбрать пункт "Удалить". Кроме этого удалить ресурс можно также в режиме редактирования с помощью кнопки "Удалить". После удаления, ресурс на самом деле не удаляется. Ему просто устанавливается включенным состояние параметра deleted, т.е. ресурс как бы помечается на удаление. Для того чтобы окончательно стереть ресурс или ресурсы, помеченные на удаление, необходимо на вкладке "Ресурсы" нажать на значок мусорного ведра.



Поля ресурса

Все ресурсы имеют следующие предопределенные поля:

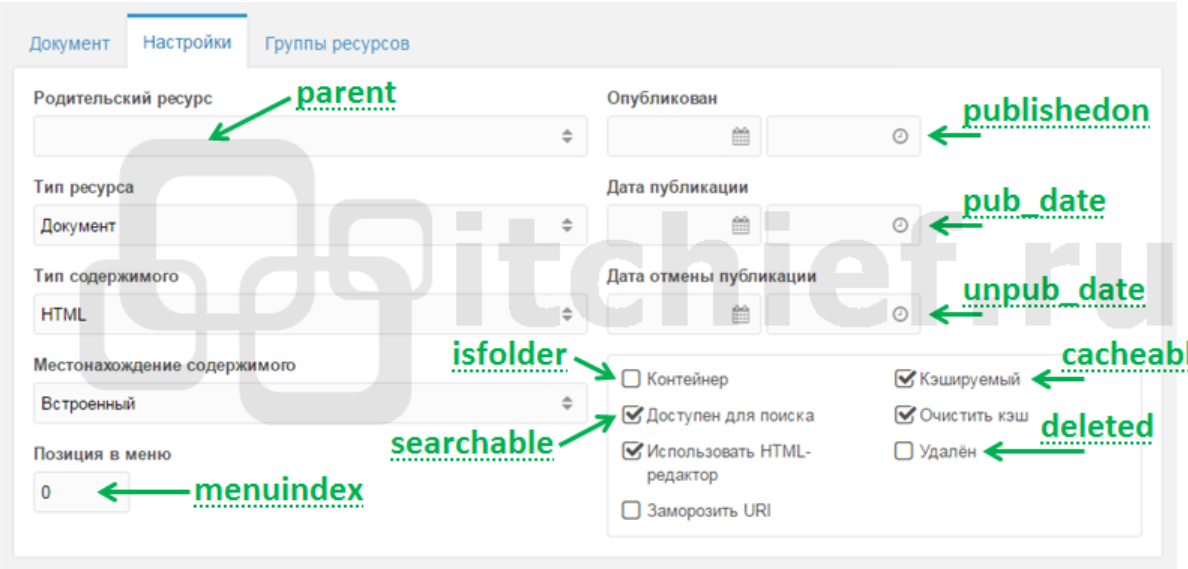
1. Базовые поля ресурса MODX.



Имя	Описание
id	Идентификатор (порядковый номер) ресурса.
template	Ссылка на шаблон, который будет использоваться для отображения этого ресурса.
published	Включает публикацию ресурса во front-end.
pagetitle	Заголовок (название) ресурса.
longtitle	Расширенный заголовок ресурса.
description	Описание ресурса.
introtext	Краткая информация о содержимом ресурса. Может использоваться для его представления на главной странице или в некотором разделе.

Имя	Описание
alias	URL-псевдоним по которому можно обратиться к этому ресурсу. Предназначен для сайтов, которые используют дружественные URL. Например, ресурс с псевдонимом "home" и типом контента "html" будет иметь URL "home.html" (если конечно же он не контейнер).
parent	идентификатор (id) родительского ресурса.
link_attributes	Предназначен для указания атрибутов, которые необходимо добавить к ссылке. Обычно используется сниппетом, генерирующим меню.
menutitle	Заголовок, который может использоваться сниппетами для представления ресурса в меню.
menuindex	Порядковый номер индекса ресурса в меню. Более высокие значения индекса указывают на то, что ссылку на ресурс необходимо расположить ниже.
hidemenu	Убирает ресурс из выборки при формировании меню. Обычно используется сниппетами, генерирующими меню.
content	Контент ресурса.

2. Поля, осуществляющие настройку ресурса.



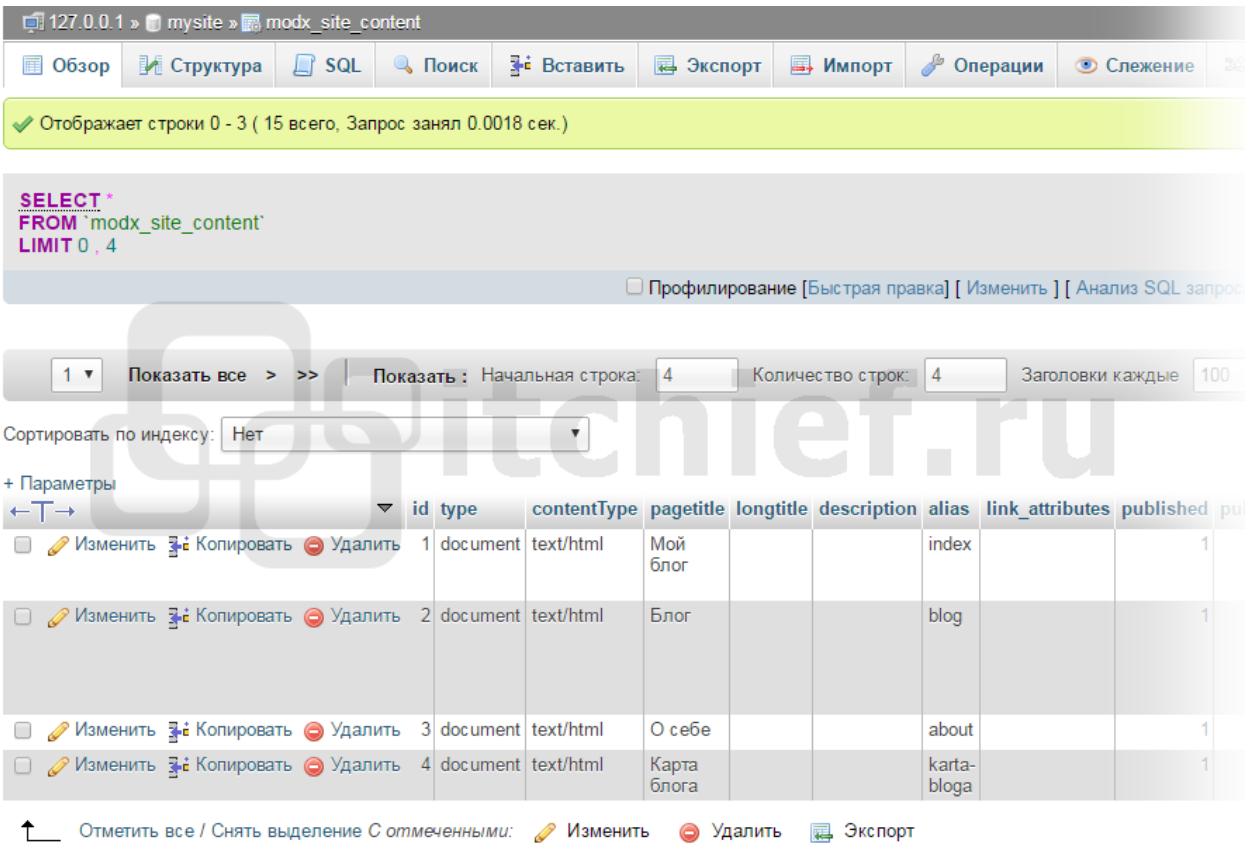
Имя	Описание
isfolder	Указывает, является ли ресурс "Контейнером". Если это так, то ресурс будет вместо суффикса иметь слеш (/). Это касается только тех сайтов, которые используют дружественные URL.

Имя	Описание
searchable	Определяет, необходимо ли ресурс включать в результаты поиска.
cacheable	Определяет, необходимо ли ресурс кешировать.
createdby	Содержит идентификатор (id) пользователя, который создал ресурс.
editedby	Содержит идентификатор (id) пользователя, который последним редактировал этот ресурс.
deleted	Определяет, отмечен ли ресурс на удаление или нет.
deletedby	Содержит идентификатор (id) пользователя, который отметил ресурс на удаление.
publishedby	Содержит идентификатор (id) пользователя, который опубликовал ресурс.
createdon	Содержит дату создания ресурса пользователем.
publishedon	Содержит дату публикации ресурса.
editedon	Содержит дату последнего редактирования документа.
pub_date	Содержит дату, начиная с которой ресурс будет опубликован.
unpub_date	Содержит дату, начиная с которой ресурс будет снят с публикации.

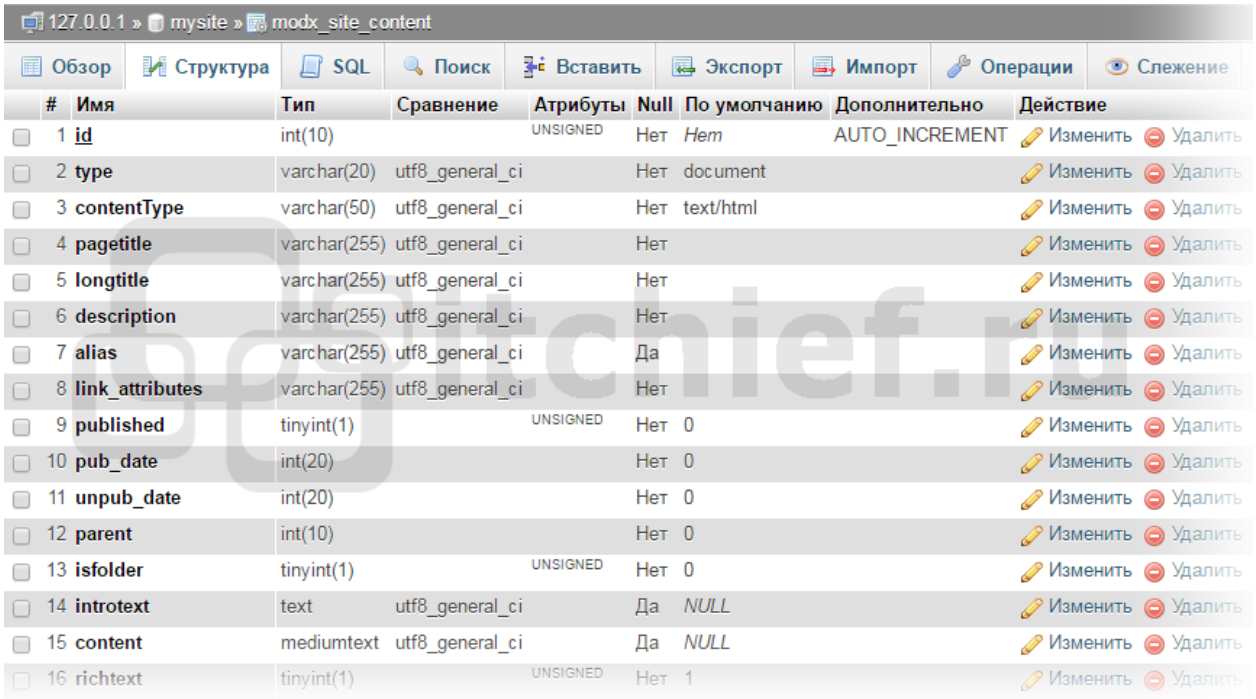
Где хранятся ресурсы

Ресурсы, как и другие объекты MODX хранятся в базе данных.

1. Обзор таблицы (site_content), содержащей ресурсы, в phpmyadmin.



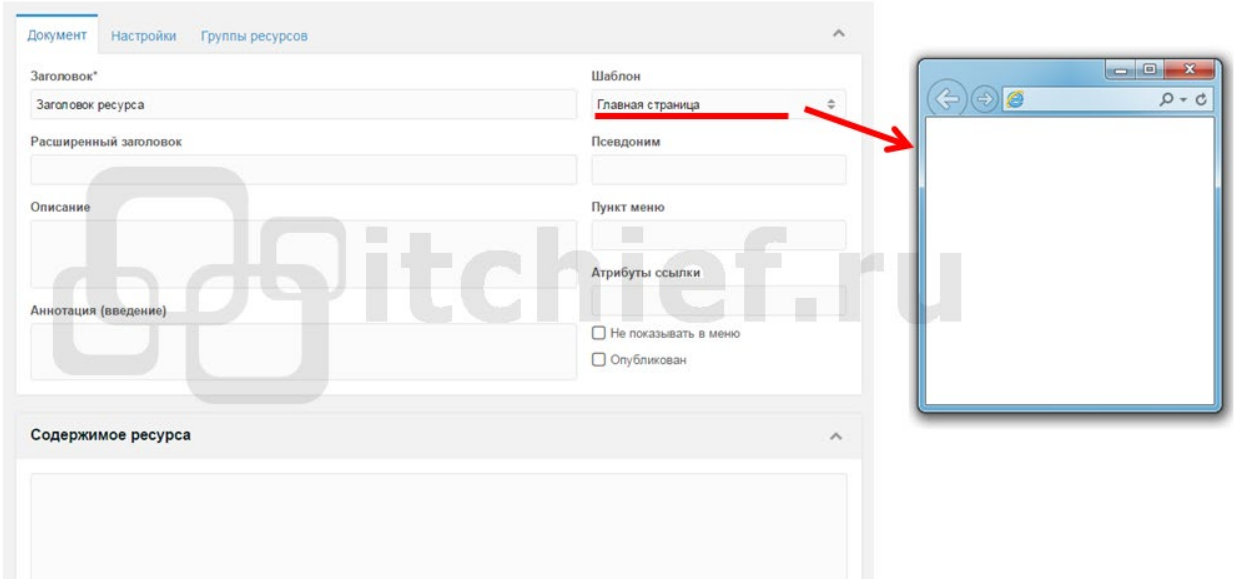
2. Структура таблицы (site_content), содержащей ресурсы.



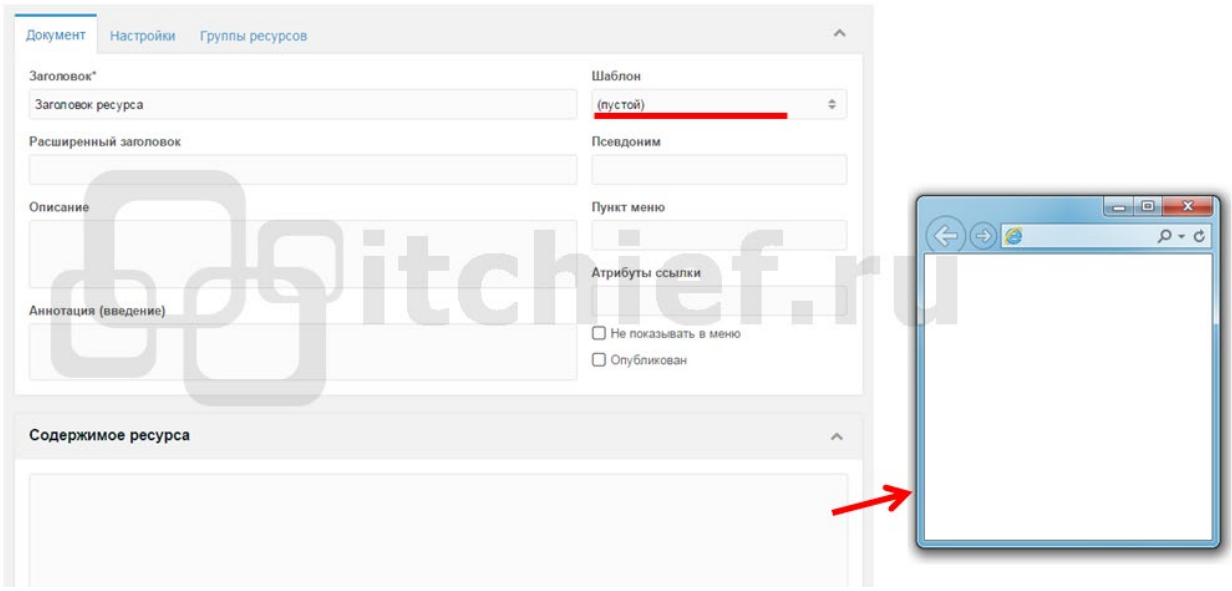
Ресурс и связанный с ним шаблон

Каждому ресурсу в MODX Revolution должен быть обязательно назначен имеющийся или пустой шаблон. Определиться, нужен ли ресурсу шаблон или нет можно следующим образом. Шаблон имеет смысл создавать, если его необходимо связать с несколькими ресурсами или использовать

TV-переменные. В этом случае страница будет выводиться, используя связанный с ней шаблон. Контент ресурса в этом шаблоне будет представляться с помощью тега MODX [[*content]].

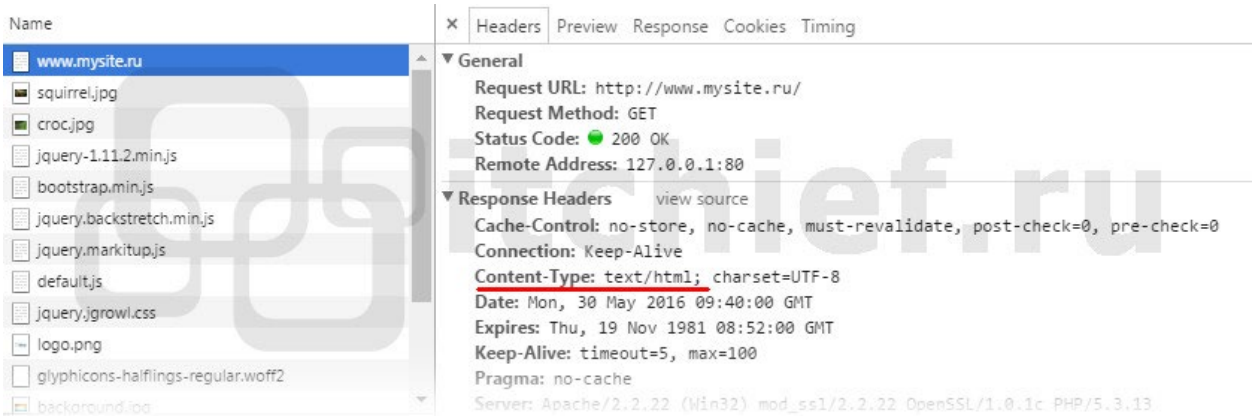


Если страница не имеет связанного с ней шаблона (выбран пустой шаблон), то она будет отображаться на основании поля "Содержимое ресурса".

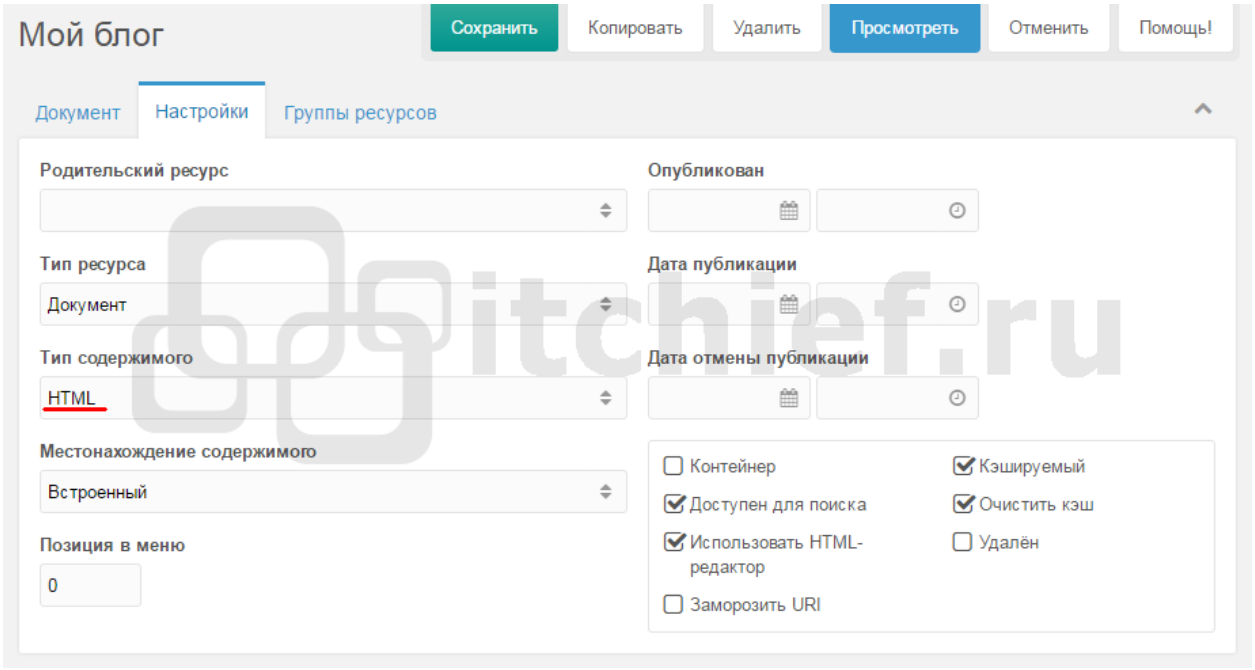


Типы содержимого в CMS MODX

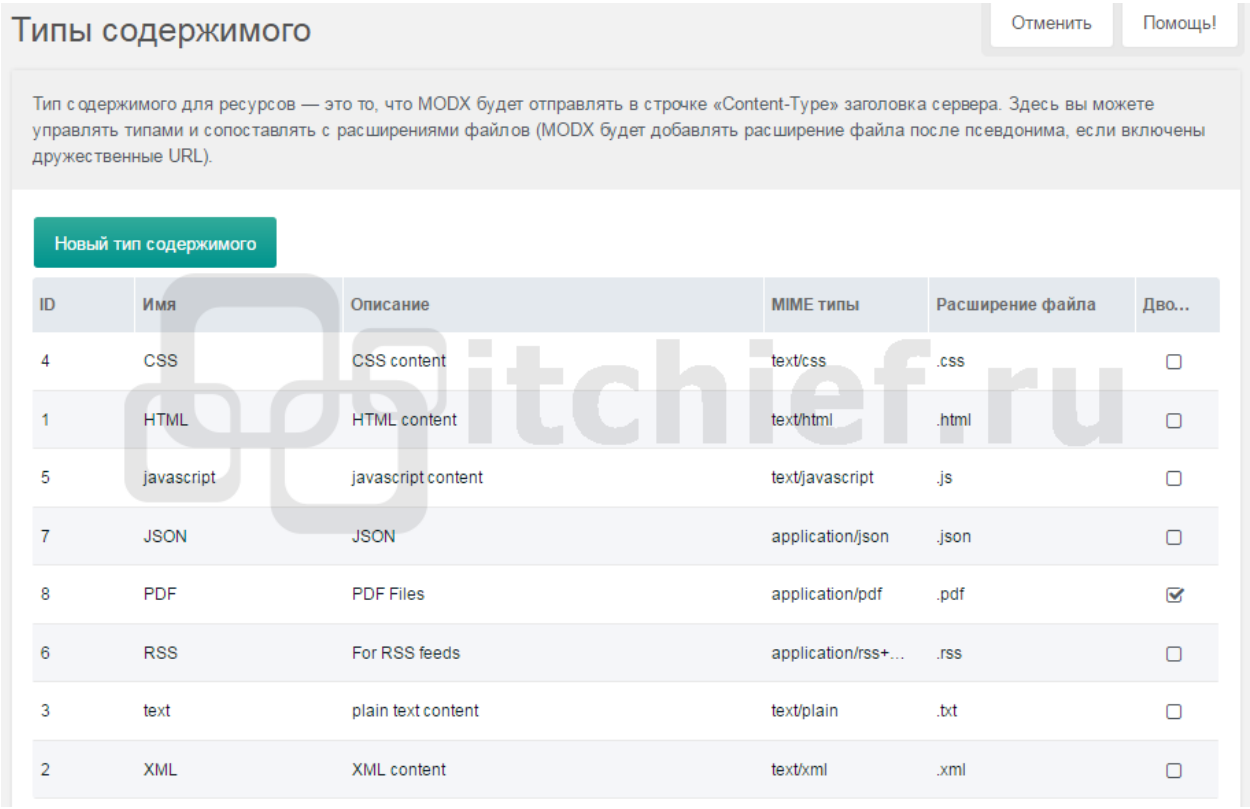
В MODX Revolution тип содержимого определяет то, какой контент содержит ресурс. Другими словами, тип содержимого - это то, что MODX будет отправлять в строчке заголовка "Content-Type". Тип содержимого в MODX Revolution можно сопоставлять с расширением файла. Это означает то, что парсер MODX, если включены дружественные URL, будет добавлять расширение, соответствующее содержимому ресурса, после псевдонима.



Узнать тип контента ресурса можно в поле "Тип содержимого", которое находиться на вкладке "Настройки".



А узнать какие есть типы содержимого и связанные с ним расширения можно на странице "Типы содержимого". Открыть данную страницу можно через главное меню админки: Сайт -> Типы содержимого.



Внимание: Добавление расширения к псевдониму MODX будет осуществляться только в том случае, если включены и настроены дружественные URL.

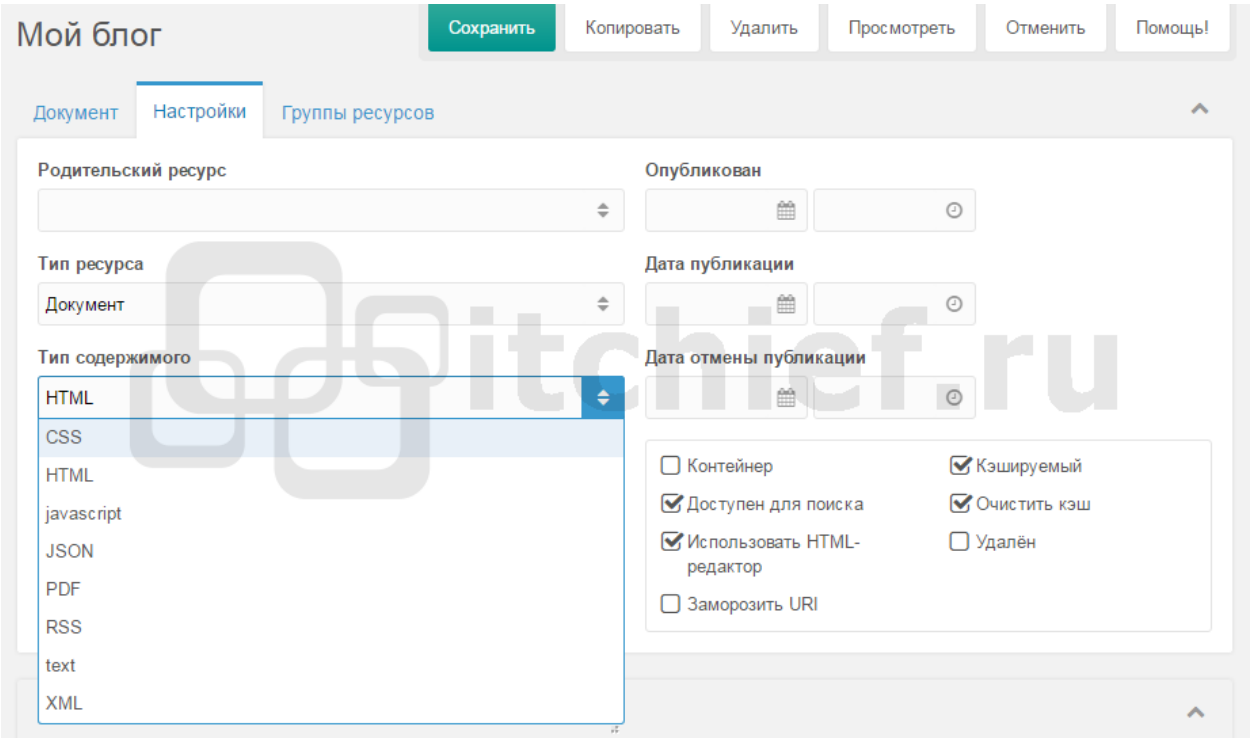
Например, ресурс с псевдонимом "mytest" и типом контента "css" (расширение файла ".css") будет отображаться так:

mytest.css

Т.е. расширение ресурса в MODX зависит от типа. Это возможность позволяет создавать из ресурсов различные типы файлов.

Изменения типа содержимого ресурса

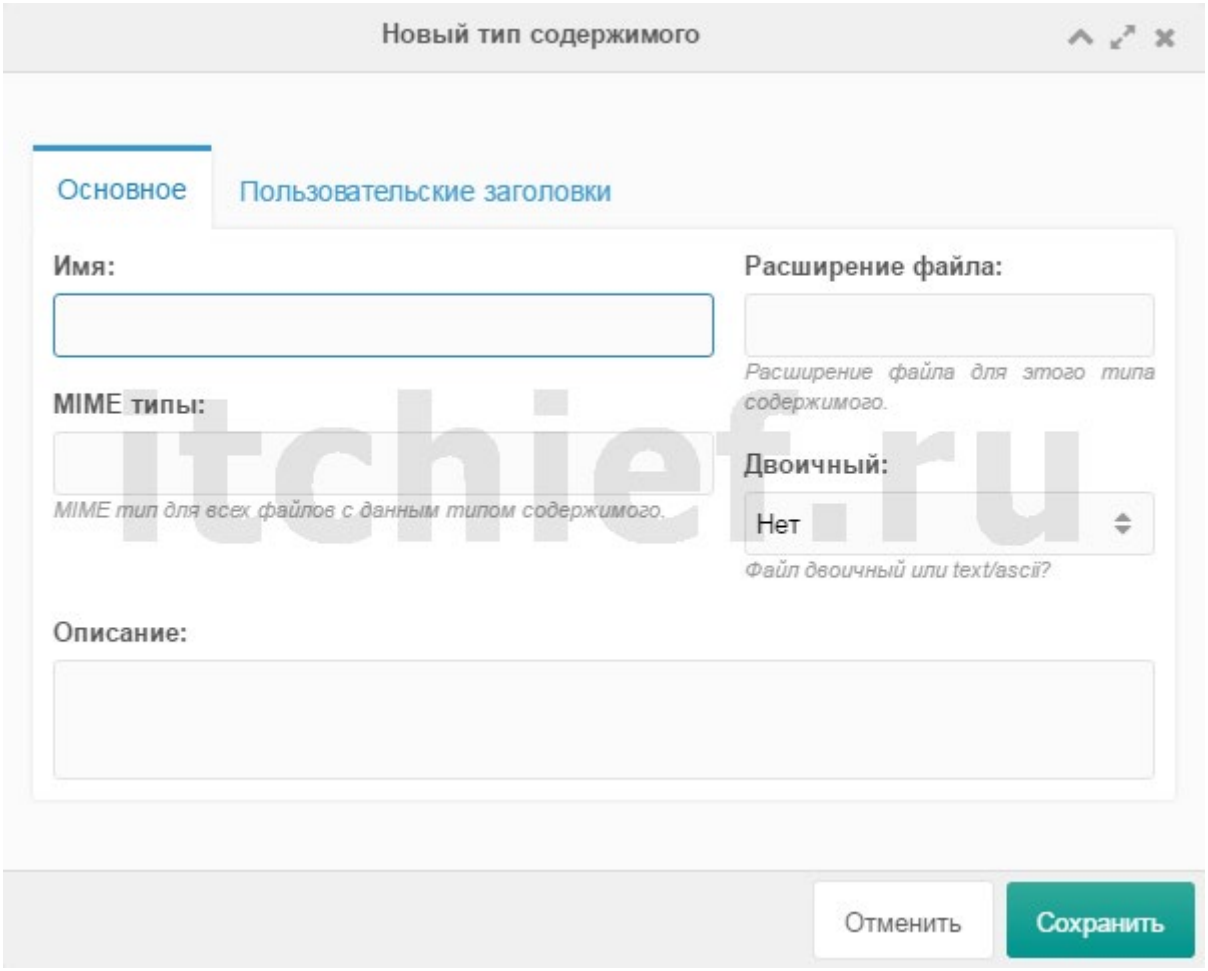
По умолчанию ресурс имеет тип содержимого "HTML". При необходимости тип содержимого ресурса можно изменить. Осуществляется это посредством раскрывающего списка "Тип содержимого", расположенного на вкладке "Настройки".



После изменения типа содержимого необходимо нажать на кнопку "Сохранить". Это действие обновит поля ресурса в базе данных и автоматически ассоциирует данный ресурс с выбранным типом содержимого.

Создание нового типа содержимого

Для создания нового типа содержимого необходимо сначала открыть страницу "Типы содержимого" (Сайт -> Типы содержимого). После этого нажать на кнопку "Новый тип содержимого". В результате этого действия появится диалоговое окно "Новый тип содержимого".



Это окно содержит следующие поля:

имя - это имя типа содержимого (т.е. это значение, которое будет отображаться в раскрывающем списке "Тип содержимого" на странице редактирования ресурса).

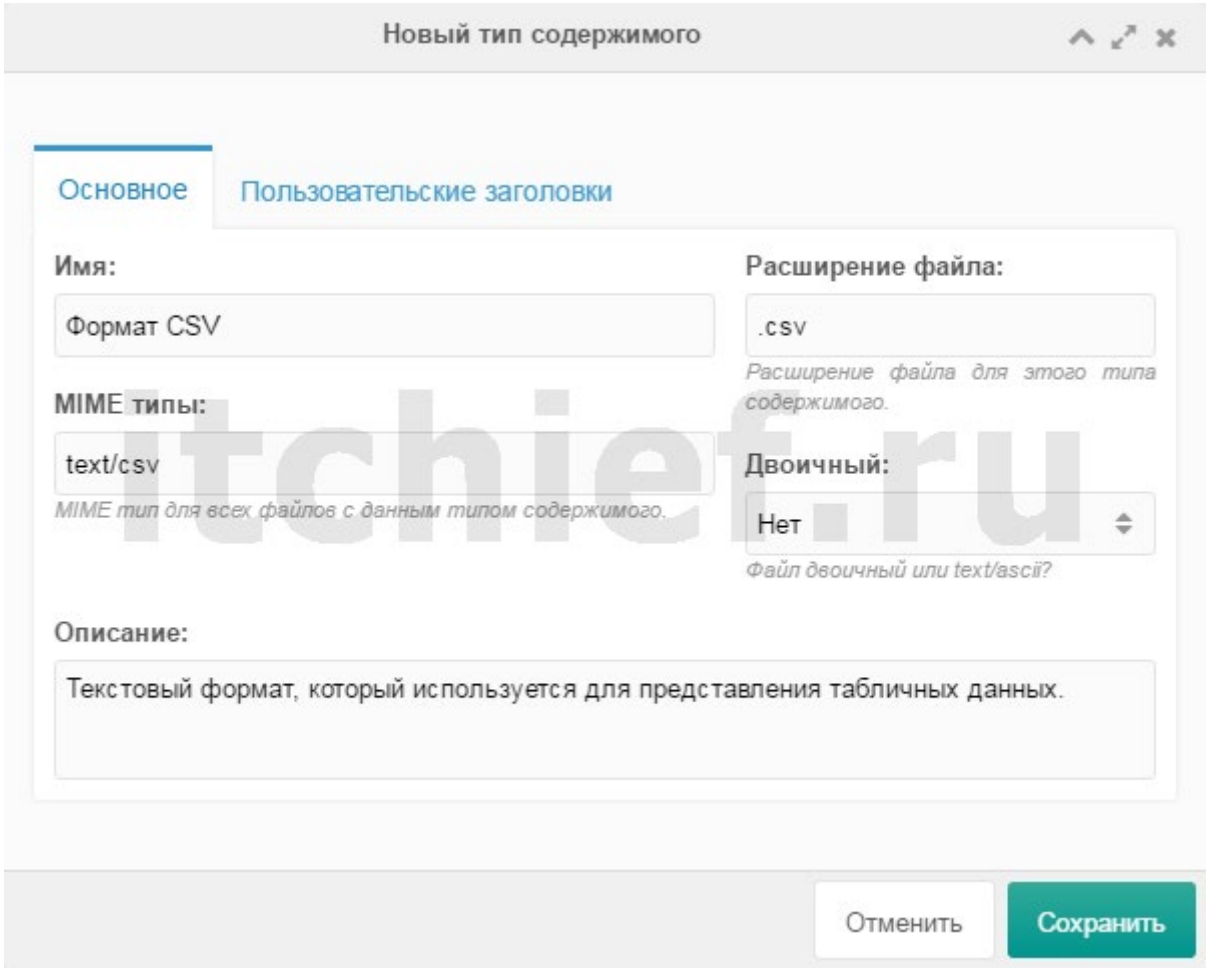
MIME тип - это тип данных, который должен быть указан в соответствии со стандартом передачи данных в сети Интернет. Этот тип "говорит" браузеру, что это за ресурс (что он содержит). Список MIME-типов можно, например, посмотреть здесь.

расширение файла - определяет то, какое окончание будет иметь псевдоним ресурса. Задавать расширение необходимо с указанием точки. Например, .doc.

двоичный – это опция указывает на то, относиться ли содержимое файла к text/ascii (текстовому) или binary (двоичному).

описание - необязательное поле, в котором можно поместить дополнительную информацию об этом содержимом.

Например, создадим тип содержимого CSV:



После заполнения полей, нажмём на кнопку "Сохранить" и новый тип контента (CSV) будет отображён в таблице.

Как убрать расширение у HTML документа

Многие веб-разработчики настраивают свои сайты так, чтобы страницы с HTML-содержимым не имели расширение .html.

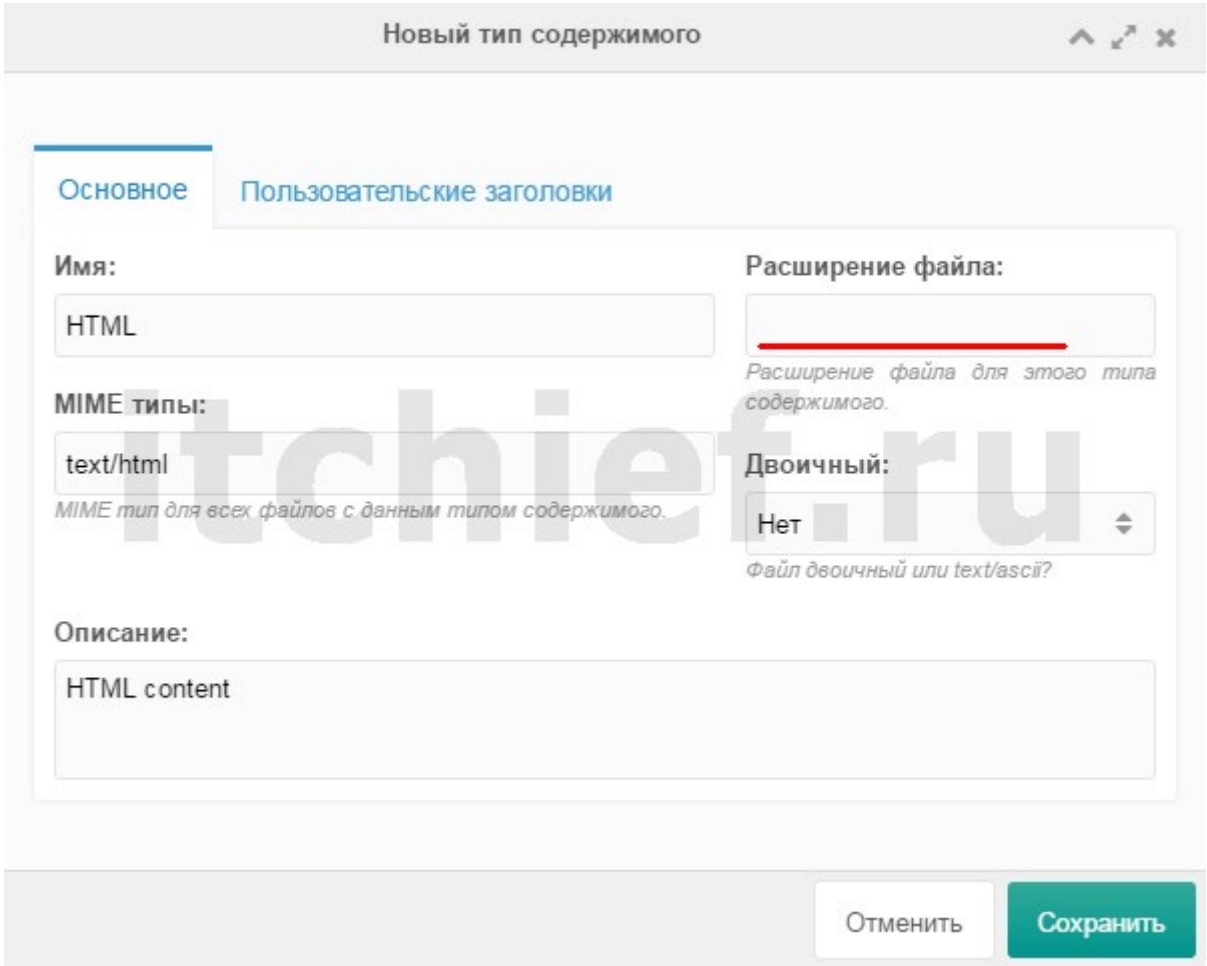
В MODX это осуществляется следующим образом:

Открыть окно "Типы содержимого" (Сайт->Типы содержимого).

В этом окне найти строчку, описывающее содержимое HTML, нажать на неё правой кнопкой мыши и выбрать в открывшемся контекстном меню пункт "Редактировать".

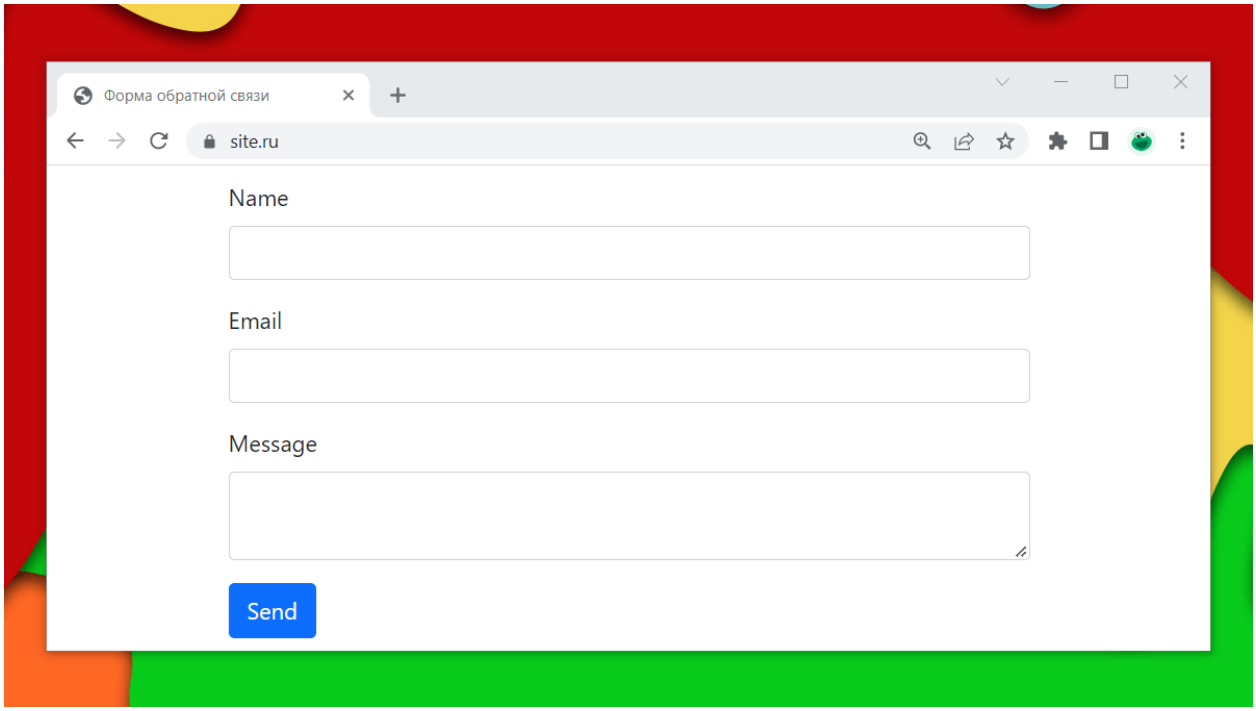
Убрать (стереть) все символы из поля расширения и нажать на кнопку "Сохранить".

После этого страницы (ресурсы) имеющие тип содержимого HTML будут выводиться без расширения. Это происходит, потому что у этого типа содержимого нет расширения (пустая строка).


Новый тип содержимого
Основное
Пользовательские заголовки
Имя:
HTML
Расширение файла:
Расширение файла для этого типа содержимого.
MIME типы:
text/html
MIME тип для всех файлов с данным типом содержимого.
Двоичный:
Нет
Файл двоичный или text/ascii?
Описание:
HTML content
Отменить
Сохранить


http://www.mysite.ru/about.html
http://www.mysite.ru/about

Создание формы для сайта на MODX с использованием FormIt

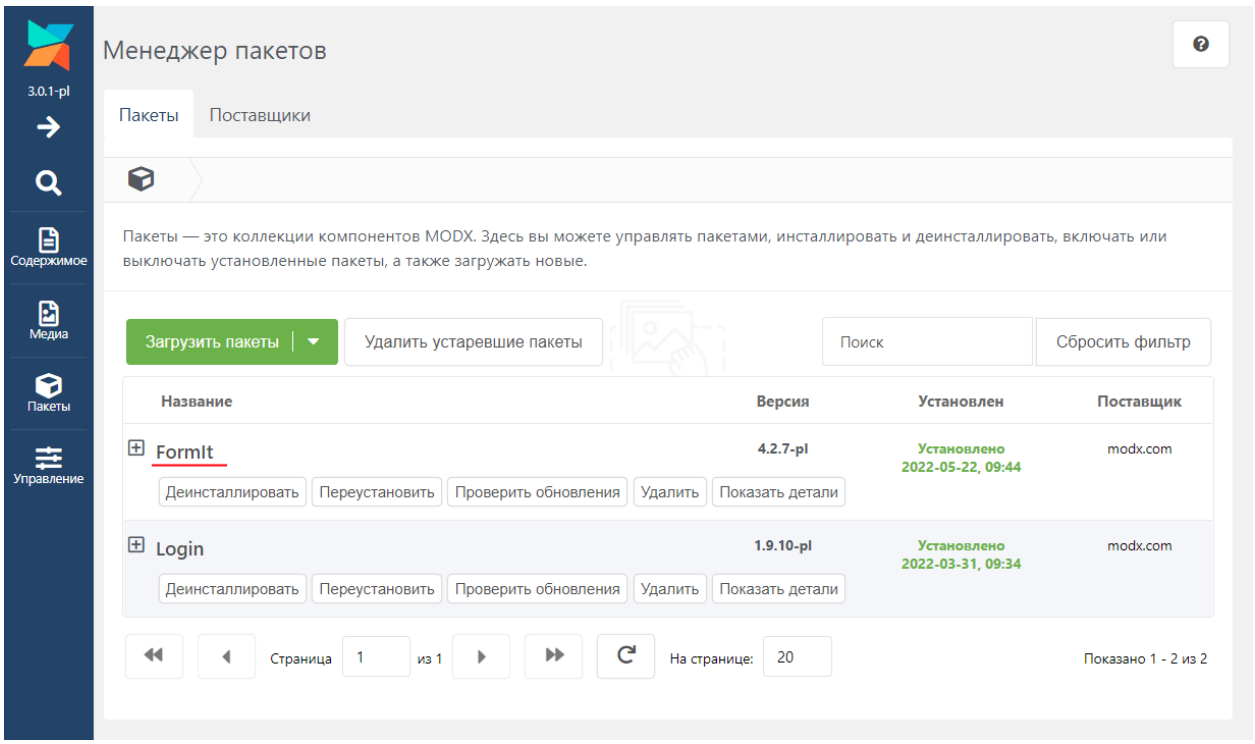


В этой статье рассмотрим процесс создания формы обратной связи на MODX с использованием FormIt. А также работу с ней через Ajax посредством AjaxForm.

О компоненте FormIt

В репозитории MODX имеется очень популярное дополнение для динамической обработки форм. Называется оно FormIt.

Это дополнение используется для обработки формы после нажатия на кнопку отправки. FormIt сначала выполняет валидацию, и только затем (после успешной проверки) другие действия. Например, такие как отправка сообщения на электронную почту, сохранение данных формы в базу данных, функцию автоответчика и др.



Пример простой формы обратной связи на FormIt

В этом примере создадим простую HTML форму с использованием Bootstrap 5. Она будет состоять из 3 полей: имя, email и сообщение. FormIt будет выполнять валидацию формы и отправлять сообщение с данными, которые ввел пользователь, на email.

Имя

Email

Сообщение

Отправить

Вызов сниппета FormIt:

```
[[!FormIt?
  &emailFrom=`[[++emailsender]]`
```

```
  &emailTo=`alexander@mail.com`
  &emailTpl=`myEmailTpl`
  &hooks=`email`
  &submitVar=`feedback-form`
  &successMessage=`Форма успешно отправлена!`
  &validate=`nospam:blank,
    name:required,
    email:email:required,
    message:required:minLength=^20^`
]]
```

Список используемых параметров:

hooks – список скриптов, которые нужно выполнить после успешной валидации формы; скрипты будут запускаться последовательно друг за другом; если какой-то скрипт завершится не удачно, то следующие за ним выполняться не будут;

validate – список полей, которые нужно проверить на соответствии указанным требованиям; можно указать несколько валидаторов, например, как это сделано для email;

submitVar – необходим, чтобы этот вызов сниппета FormIt обрабатывал не все формы, а только те, которые содержат указанный ключ в составе передаваемых на сервер данных (в суперглобальном массиве $_POST);

successMessage – сообщение, которое необходимо вывести после успешного завершения обработки формы; работает, только если не используется свойство для редиректа;

emailTpl – чанк, содержащий шаблон email письма;

emailTo – адрес, на который нужно отправить email;

emailFrom – адрес, от которого будет отправлен email; в этом примере email будем брать из системного параметра emailsender;

emailSubject – тема письма.

validationErrorMessage – сообщение, которое нужно вывести, если в форме содержатся ошибки.

Работает FormIt следующим образом:

1. Получает данные формы. Так как в вызове сниппета FormIt, мы указали свойство submitVar, то этот сниппет будет обрабатывать только ту форму, которая будет передавать в теле запроса этот ключ. Его мы добавили к кнопке type="submit".

2. Выполняет валидацию. После этого выполняется валидация формы в соответствии с требованиями, которые мы указали в свойстве validate. Валидатор в этом свойстве задаётся через двоеточие.

Валидатор blank проверяет, является ли поле пустым. В данном примере мы проверяем с помощью него поле nospam:

nospam:blank

Кроме blank в этом примере ещё используются следующие валидаторы:

required – проверяет, является ли поле не пустым;

email – содержит ли поле корректный адрес электронной почты;

minLength – требование, к минимальному количеству символов.

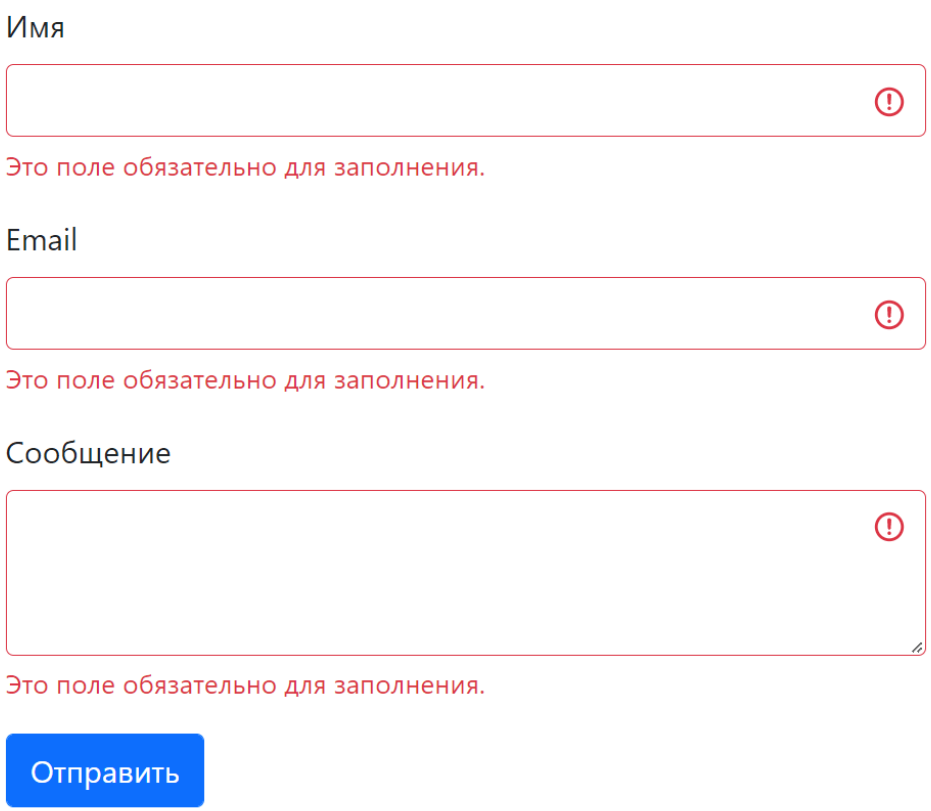
Например, для проверки поля email мы используем 2 валидатора:

email:email:required

Таким образом это поле должно быть не пустым и содержать корректный email.

Если какие-то поля не прошли валидацию, то ошибки будут выведены в соответствующие плейсхолдеры. Например, для поля email это будет:

[[+fi.error.email]]



3. Выполнение хуков. После успешной валидации будут последовательно друг за другом выполнены скрипты, указанные в hooks.

В приведённом примере мы используем в hooks только один скрипт: email. Он отправит форму на email. Шаблон для этого письма он возьмёт из чанка, заданного в emailTpl. А отправит он письмо на email, указанный в свойстве emailTo.

Сообщение с формы обратной связи.

Александр (email: alexander@mail.com) оставил следующее сообщение:

> Какое-то сообщение

4. Вывод сообщения об успехе. После успешного завершения хука email будет выведено в плейсхолдер fi.successMessage сообщение, указанное в свойстве successMessage.

Форма успешно отправлена!

HTML код формы:

```
[[!+fi.successMessage:notempty=`<div class="alert alert-success"
role="alert">[[!+fi.successMessage]]</div>`]]

<form action="[[~[[*id]]]]" method="post" class="form">

 <input type="hidden" name="nospam" id="nospam" value="">

 <div class="mb-3">

  <label for="name" class="form-label">Name</label>

  <input type="text" class="form-control[[!+fi.error.name:notempty=` is-invalid`]]" name="name"
id="name" value="[[!+fi.name]]">

  <div class="invalid-feedback">[[!+fi.error.name]]</div>

 </div>

 <div class="mb-3">

  <label for="email" class="form-label">Email</label>

  <input type="email" class="form-control[[!+fi.error.email:notempty=` is-invalid`]]" name="email"
id="email" value="[[!+fi.email]]">

  <div class="invalid-feedback">[[!+fi.error.email]]</div>

 </div>

 <div class="mb-3">

  <label for="phone" class="form-label">Phone</label>

  <input type="tel" class="form-control[[!+fi.error.phone:notempty=` is-invalid`]]" name="phone"
id="phone"

    value="[[!+fi.phone]]">

  <div class="invalid-feedback">[[!+fi.error.phone]]</div>
```

```
 </div>
 <div class="mb-3">
  <label for="message" class="form-label">Message</label>
  <textarea class="form-control[[!+fi.error.message:notempty=` is-invalid`]]" name="message" id="message"
      rows="5">[[!+fi.message]]</textarea>
  <div class="invalid-feedback">[[!+fi.error.message]]</div>
 </div>
 <div class="mb-3">
  <input type="submit" class="btn btn-primary" name="feedback-form" value="Отправить">
 </div>
</form>
```

Оформление формы выполнено на Bootstrap 5.

Скрытое поле nospam используется для защиты от спама. Обычно спам боты пытаются заполнить все поля, а это поле должно быть пустым. Если поле nospam не пустое, то форма не пройдёт проверку и её дальнейшая обработка выполняться не будет.

Сниппет FormIt будет заполнять в форме заданные плейсхолдеры. Например, для поля email:

fi.email – значением поля email;

fi.error.email – сообщением об ошибке при валидации.

Выделение полей, которые не прошли валидацию, выполняется посредством добавления к ним класса is-invalid. Например, для поля email это будет выглядеть так:

```
[[!+fi.error.email:notempty=` is-invalid`]]
```

Т.е. когда плейсхолдер fi.error.email не пустой, у элемента <input> появится класс is-invalid.

Для вывода сообщения об успешной обработки формы используется следующая конструкция:

```
[[!+fi.successMessage:notempty=`<div class="alert alert-success" role="alert">[[!+fi.successMessage]]</div>`]]
```

Она работает очень просто: если плейсхолдер fi.successMessage что-то содержит, то выведи HTML блок с содержимым [[!+fi.successMessage]]:

```
<div class="alert alert-success" role="alert">[[!+fi.successMessage]]</div>
```

Этот HTML блок является компонентом Alert фреймворка Bootstrap.

В теге <form> для атрибута action укажем URL на эту же страницу:

```
<form action="[[~[[*id]]]]" method="post">
```

Содержимое чанка myEmailTpl:

```
<p style="font-weight: bold;">Сообщение с формы обратной связи.</p>
```

<mark>[[+name]]</mark> (email: <mark>[[+email]]</mark>, phone: <mark>[[+phone:default=`-`]]</mark>) оставил следующее сообщение:

[[+message]]

Отправка формы через AJAX

По умолчанию FormIt не умеет работать с формой через AJAX. Но в репозитории modstore.pro имеется дополнение AjaxForm, которое позволяет отправлять формы через AJAX. По умолчанию оно работает с FormIt. Но при необходимости вместо FormIt можно использовать собственный сниппет.

При использовании AjaxForm работа с формой будет осуществляться следующим образом:

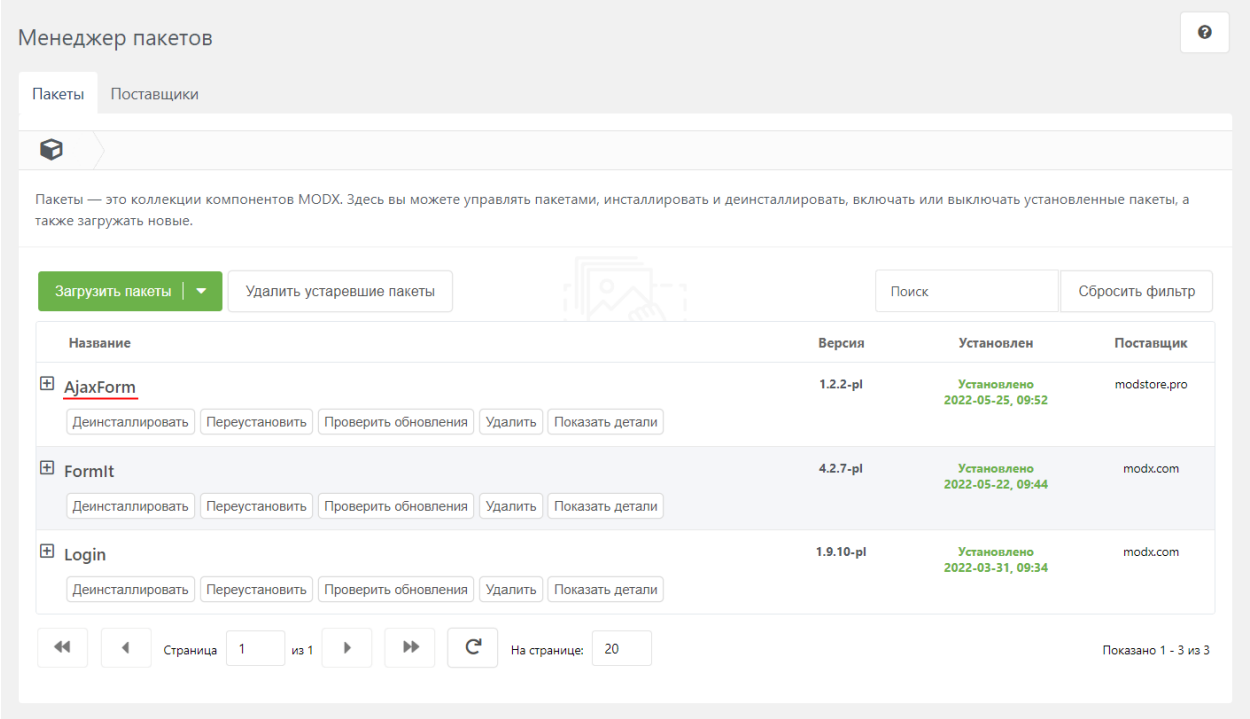
вывод HTML формы будет осуществляться из чанка, который передан AjaxForm в свойстве form;

отправка форма выполняется через AJAX, для этого сниппет AjaxForm регистрирует на фронтенде нужные скрипты;

на сервере запускает сниппет FormIt и передаёт ему данные формы;

после получения ответа от FormIt передаёт его браузеру, который выводится пользователю на страницу с использованием JavaScript.

Перед тем как начать переделывать форму проверим установлено ли у нас расширение AjaxForm.



Ход работы:

1. Создадим чанк myFormTpl и вставим в него HTML код формы.

AjaxForm вывод сообщений об ошибках, например, для поля email, осуществляет в элемент с классом error_mail. Кроме этого, к полям, которые не прошли валидацию добавляет класс error.

Но нас это не устраивает, т.к. сообщения об ошибке нужно помещать в элемент с классом invalid-feedback, а к полям, которые не прошли проверку необходимо добавлять класс is-invalid.

Это мы будем выполнять с помощью JavaScript при получении ответа от сервера:

```
$(document).on('af_complete', function (event, response) {
  const elForm = $(response.form);
  elForm.find('.is-invalid').each(function (index, el) {
    $(el).removeClass('is-invalid');
  });
  for (let key in response.data) {
    const elInput = elForm.find(`[name=${key}]`);
    elInput.addClass('is-invalid');
    elInput.next('.invalid-feedback').text($(response.data[key]).text());
  }
});
```

После этого из формы необходимо удалить плейсхолдеры [[!+fi.name]], [[!+fi.error.name]] и др. Они используются в качестве заполнителей для сниппета FormIt и позволяют нам сохранять значения, введённые пользователем и отображать ошибки валидации. Но так как сейчас работа ведётся через AJAX, то они нам не нужны.

В итоге HTML форма будет иметь следующий код:

```
<form action="#" method="post" class="feedback-form">
  <input type="hidden" name="nospam" id="nospam">
  <div class="mb-3">
    <label for="name" class="form-label">Имя</label>
    <input type="text" class="form-control" name="name" id="name">
    <div class="invalid-feedback"></div>
  </div>
  <div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input type="email" class="form-control" name="email" id="email">
    <div class="invalid-feedback"></div>
  </div>
  <div class="mb-3">
    <label for="message" class="form-label">Сообщение</label>
    <textarea class="form-control" name="message" id="message" rows="3"></textarea>
    <div class="invalid-feedback"></div>
```

```
  </div>
  <div class="mb-3">
    <input type="submit" class="btn btn-primary" name="feedback-form" value="Отправить">
  </div>
</form>
```

2. Добавим в нужный ресурс или шаблон вызов сниппета AjaxForm и скрипт:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
[[AjaxForm?
  &form=`myFormTpl`
  &formSelector=`feedback-form`
  &emailTpl=`myEmailTpl`
  &emailTo=`alexander@gmail.com`
  &hooks=`email`
  &successMessage=`Форма успешно отправлена!`
  &validationErrorMessage=`В форме содержатся ошибки!`
  &submitVar=`feedback-form`
  &validate=`nospam:blank,
    name:required,
    email:email:required,
    message:required:minLength=^20^`
]]
<script>
  $(document).on('af_complete', function(event, response) {
    const elForm = $(response.form);
    elForm.find('.is-invalid').each(function(index, el) {
      $(el).removeClass('is-invalid');
    });
    for (let key in response.data) {
      const elInput = elForm.find(`[name=${key}]`);
      elInput.addClass('is-invalid');
      elInput.next('.invalid-feedback').text($(response.data[key]).text());
    }
```

```
});
</script>
```

Свойство form в вызове сниппета AjaxForm указывает чанк, в котором содержится HTML форма. Остальные свойства данный сниппет просто передаёт FormIt.

Скрипт необходим для установки не валидным полям класса is-invalid и вставки сведений об ошибках в элементы .invalid-feedback. Кроме этого JavaScript файлы, которые регистрирует AjaxForm во фронтенде, написаны с использованием библиотеки jQuery. Поэтому её нужно подключить.

Дополнительно

1. Редирект на другую страницу после успешной обработки формы.

Перенаправить пользователя на другую страницу после успешной отправки формы можно с помощью хука redirect:

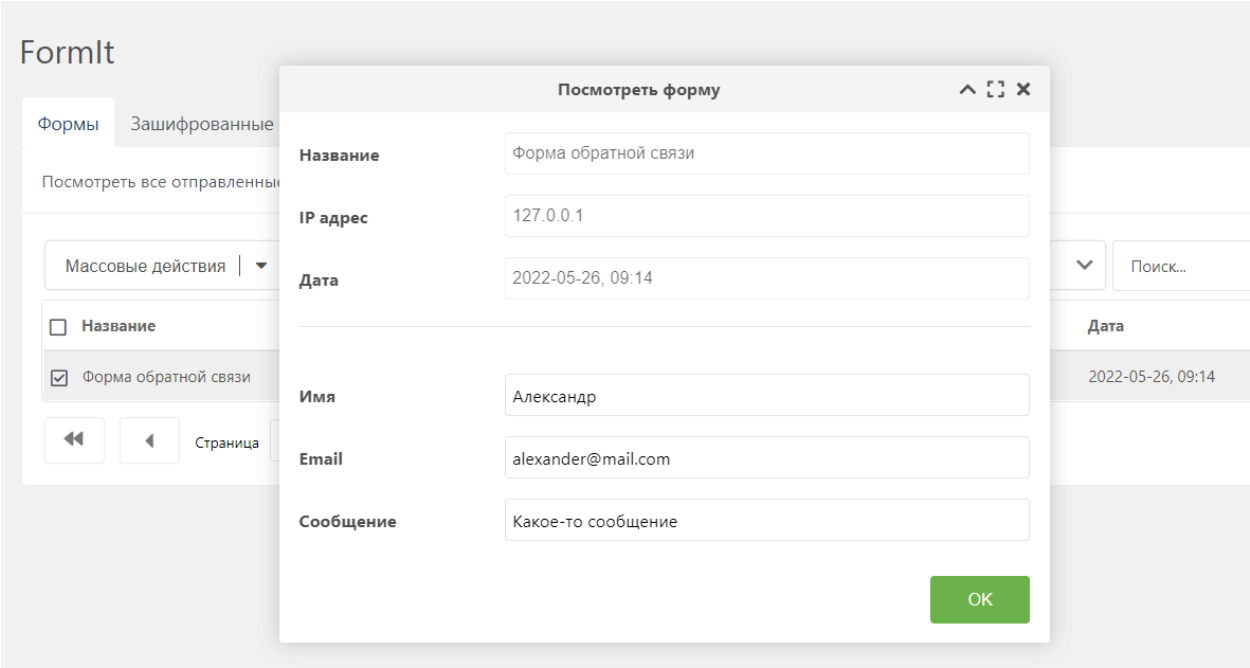
```
&hooks=`email,redirect`
&redirectTo=`7`
```

Указание ресурса, на который необходимо перенаправить пользователя осуществляется с помощью свойства redirectTo. В данном примере пользователь будет перенаправлен на ресурс, который имеет в качестве id значение 7.

2. Сохранение данных формы в базу данных.

Сохранить отправленные формы в базу данных можно с помощью хука FormItSaveForm:

```
&hooks=`email,FormItSaveForm`
&formName=`Форма обратной связи`
&formFields=`name,email,message`
&fieldNames=`name==Имя,email==Email,message==Сообщение`
```



Свойства:

formName – название формы. По умолчанию: "form-{resourceid}";

formFields – список полей, которые следует сохранить. По умолчанию: все поля, включая кнопку submit;

fieldNames – позволяет задать отображение полей на форме в менеджере.

3. Добавление в форму заголовка и URL страницы

Добавить в форму дополнительные данные, например заголовок страницы и её URL, можно с помощью скрытых полей:

<input type="hidden" name="title" value="[[*pagetitle]]">

<input type="hidden" name="url" value="[[~[[*id]]? &scheme=`full`]]">

Для отправки этой информации на email добавьте в шаблон письма:

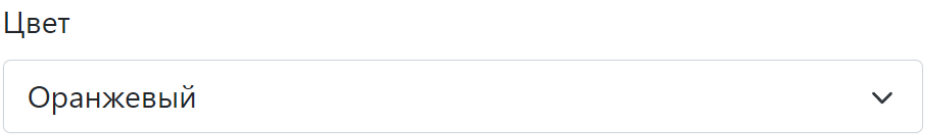
Страница, с которой отправлена форма: [[+title]]

4. Обработка полей select

Для обработки <select> FormIt предоставляет фильтр вывода FormItIsSelected. Он устанавливает атрибут selected, если значение <option> соответствует выбранной:

```
<div class="mb-3">
  <label for="color" class="form-label">Цвет</label>
  <select class="form-select[[!+fi.error.color:notempty=` is-invalid`]]" name="color" id="color" value="[[!+fi.color]]">
    <option value="" selected disabled>Выберите цвет</option>
    <option value="Красный" [[!+fi.color:FormItIsSelected=`Красный`]]>Красный</option>
    <option value="Оранжевый" [[!+fi.color:FormItIsSelected=`Оранжевый`]]>Оранжевый</option>
```

```
  <option value="Желтый" [[!+fi.color:FormItIsSelected=`Желтый`]]>Желтый</option>
 </select>
 <div class="invalid-feedback">[[!+fi.error.color]]</div>
</div>
```



5. Обработка чекбоксов и радиокнопок

FormIt для обработки type="checkbox" и type="radio" предоставляет фильтр FormItIsChecked аналогичный FormItIsSelected.

Это пример с чекбоксами:

```
<div class="mb-3">
 <div class="form-check">
  <input type="checkbox" class="form-check-input [[!+fi.error.colors:notempty=` is-invalid`]]" name="colors[]" id="colors-red" value="Красный" [[!+fi.colors:FormItIsChecked=`Красный`]]>
  <label class="form-check-label" for="colors-red">Красный</label>
 </div>
 <div class="form-check">
  <input type="checkbox" class="form-check-input [[!+fi.error.colors:notempty=` is-invalid`]]" name="colors[]" id="colors-orange" value="Оранжевый" [[!+fi.colors:FormItIsChecked=`Оранжевый`]]>
  <label class="form-check-label" for="colors-orange">Оранжевый</label>
 </div>
 <div class="form-check">
  <input type="checkbox" class="form-check-input [[!+fi.error.colors:notempty=` is-invalid`]]" name="colors[]" id="colors-yellow" value="Желтый" [[!+fi.colors:FormItIsChecked=`Желтый`]]>
  <label class="form-check-label" for="colors-yellow">Желтый</label>
 </div>
 <input type="hidden" class="[[!+fi.error.colors:notempty=` is-invalid`]]" name="colors[]" value="">
 <div class="invalid-feedback">[[!+fi.error.colors]]</div>
</div>
```

Если colors должно быть обязательным, то нужно добавить «скрытое» поле, как это показано в примере выше. После этого в свойство validate добавить colors:required:

```
[[!FormIt?
```

```
  ...
  &validate=`...
    colors:required`
]]
```

- [ ] Красный
- [ ] Оранжевый
- [ ] Желтый

Это поле обязательно для заполнения.