Pico

Упрощаем работу в Интернете.

Welcome

Sub Page Index

Добро пожаловать в Pico

Поздравляем, вы успешно установили Pico 2.1.4. Pico – до глупости простая, невероятно быстрая CMS для работы с плоскими файлами.

Создание контента

Pico – это CMS с плоским файлом. Это означает, что не нужно иметь дело с серверной частью администрирования или базой данных. Вы просто создаете `.md` файлы в `content` папке, и эти файлы становятся вашими страницами. Например, этот файл называется `index.md` и отображается как главная целевая страница.

Когда вы устанавливаете Pico, он поставляется с некоторыми примерами содержимого, которые будут отображаться до тех пор, пока вы не добавите свое собственное содержимое. Просто добавьте некоторые `.md` файлы в свою `content` папку в корневом каталоге Pico. Настройка не требуется, Pico автоматически будет использовать `content` папку, как только вы создадите свою собственную `index.md`. Просто ознакомьтесь с образцами содержимого Pico для примера!

Если вы создадите папку в каталоге содержимого (например, `content/sub`) и поместите в нее `index.md`, вы сможете получить доступ к этой папке по URL `http://localhost/pico/?sub`. Если вам нужна другая страница в подпапке, просто создайте текстовый файл с соответствующим именем, и вы сможете получить к нему доступ (например, `content/sub/page.md` доступен по URL `http://localhost/pico/?sub/page`). Ниже мы показали несколько примеров местоположений и соответствующие им URL-адреса:

Физическое местоположение	URL
содержание/index.md	/

Физическое местоположение	URL
содержание/sub.md	~~?sub~~ (недоступен, см. Ниже)
содержимое/подзаголовок/index.md	?sub (то же, что и выше)
содержание/подстраница.md	?подстраница
содержание/theme.md	? тема (скрыта в меню)
содержимое/a/очень/длинное/url.md	?/очень/длинный/URL (не существует)

Если файл не может быть найден, будет показан файл `content/404.md`. Вы можете добавлять `404.md` файлы в любой каталог. Итак, например, если вы хотите использовать специальную страницу с ошибками для своего блога, вы могли бы просто создать `content/blog/404.md`.

Pico строго разделяет содержимое вашего веб-сайта (файлы Markdown в вашем `content` каталоге) и то, как это содержимое должно отображаться (шаблоны Twig в вашем `themes` каталоге). Однако не каждый файл в вашем `content` каталоге на самом деле может быть отдельной страницей. Например, некоторые темы (включая тему Pico по умолчанию) используют какой-то специальный "скрытый" файл для управления метаданными (как `_meta.md` в примере содержимого Pico). В некоторых других темах используется `_footer.md` для представления содержимого нижнего колонтитула веб-сайта. Общий момент заключается в том, `_`что все файлы и каталоги с префиксом a `_` в вашем `content` каталоге скрыты. Эти страницы недоступны из веб-браузера, вместо этого Pico отобразит страницу с ошибкой 404.

В качестве обычной практики мы рекомендуем вам разделять ваше содержимое и ресурсы (например, изображения, загрузки и т.д.). Мы даже запрещаем доступ к вашему `content` каталогу по умолчанию. Если вы хотите использовать некоторые ресурсы (например, изображение) в одном из ваших файлов контента, используйте `assets` папку Pico. Затем вы можете получить к ним доступ в своей уценке, используя `%assets_url%` заполнитель, например: `![Image Title](%assets_url%/image.png)`

Разметка текстового файла

Текстовые файлы помечаются с помощью Markdown и Markdown Extra. Они также могут содержать обычный HTML.

В верхней части текстовых файлов вы можете разместить комментарий к блоку и указать определенные мета-атрибуты страницы, используя YAML ("заголовок YAML"). Например:

```
---
Title: Welcome
```

```
Description: This description will go in the meta description tag
Author: Joe Bloggs
Date: 2001-04-25
Robots: noindex,nofollow
Template: index
---
```

Эти значения будут содержаться в `{{ meta }}` переменной в темах (см. ниже). Мета-заголовки иногда имеют особое значение: например, Pico не проходит только через `Date` мета-заголовок, а скорее оценивает его как действительно "понять", когда была создана эта страница. Это вступает в игру, когда вы хотите сортируйте свои страницы не только в алфавитном порядке, но и по дате. Другим примером является `Template` мета-заголовок: он определяет, какой шаблон Twig Pico использует для отображения эта страница (например, если вы добавляете `Template: blog`, Pico использует `blog.twig`).

В попытке разделить содержимое и стиль, мы рекомендуем вам не использовать встроенный CSS в ваших файлах Markdown. Вам лучше добавить соответствующие классы CSS в вашу тему. Например, вы можете захотеть добавить несколько классов CSS в свою тему, чтобы определить, какую часть доступного пространства должно использовать изображение (например, `img.small { width: 80%; }`). Затем вы можете использовать эти CSS-классы в своих файлах Markdown, например: `![Image Title](%assets_url%/image.png) {.small}`

Существуют также определенные переменные, которые вы можете использовать в своих текстовых файлах:

- `%site_title%` - Название вашего сайта Pico
- `%base_url%` - URL-адрес вашего сайта Pico; внутренние ссылки могут быть указаны с помощью `%base_url%?sub/page`
- `%theme_url%` - URL-адрес используемой в данный момент темы
- `%assets_url%` - URL-адрес `assets` каталога Pico
- `%themes_url%` - URL-адрес `themes` каталога Pico; не путайте это с `%theme_url%`
- `%plugins_url%` - URL-адрес `plugins` каталога Pico
- `%version%` - Строка текущей версии Pico (например, `2.0.0`)
- `%meta.*%` - Доступ к любой метапеременной текущей страницы, например, `%meta.author%` заменяется на `Joe Bloggs`
- `%config.*%` - Доступ к любой скалярной переменной конфигурации, например, `%config.theme%` заменяется на `default`

Ведение блога

Pico не является программным обеспечением для ведения блогов, но позволяет вам очень легко использовать его в качестве блога. Вы можете найти множество плагинов,

реализующих типичные функции ведения блога, такие как аутентификация, пометка, разбивка на страницы и социальные плагины. Смотрите раздел плагинов ниже для получения подробной информации.

Если вы хотите использовать Pico в качестве программного обеспечения для ведения блогов, вы, вероятно, захотите сделать что-то вроде следующего:

1. Поместите все статьи вашего блога в отдельную `blog` папку в вашем `content` каталоге. У всех этих статей должен быть `Date` мета-заголовок.
2. Создайте `blog.md` или `blog/index.md` в своем `content` каталоге. Добавьте `Template: blog-index` в заголовок YAML этой страницы. Позже будет показан список всех статей вашего блога (см. шаг 3).
3. Создайте новый шаблон Twig `blog-index.twig` (имя файла должно совпадать с `Template` заголовком meta из шага 2) в каталоге вашей темы. Этот шаблон, вероятно, не сильно отличается от вашего шаблона по умолчанию `index.twig` (т. Е. копировать `index.twig`), он создаст список всех статей вашего блога. Добавьте следующий фрагмент ветки в `blog-index.twig` рядом `{{ content }}`:

```
{% for page in pages("blog")|sort_by("time")|reverse if not page.hidden %}
    <div class="post">
        <h3><a href="{{ page.url }}">{{ page.title }}</a></h3>
        <p class="date">{{ page.date_formatted }}</p>
        <p class="excerpt">{{ page.description }}</p>
    </div>
{% endfor %}
```

Настройка

Pico легко настраивается двумя различными способами: с одной стороны, вы можете изменить внешний вид Pico с помощью тем, с другой стороны, вы можете добавить новые функциональные возможности с помощью плагинов. Выполнение первого включает в себя изменение HTML, CSS и JavaScript в Pico, последнее в основном состоит из программирования на PHP.

Для тебя это все по-гречески? Не волнуйтесь, вам не придется тратить время на эти технические лекции - очень просто использовать одну из замечательных тем или плагинов, разработанных другими и выпущенных для широкой публики. Пожалуйста, обратитесь к следующим разделам для получения подробной информации.

Темы

Вы можете создавать темы для вашей установки Pico в `themes` папке. Pico использует Twig для рендеринга шаблона. Вы можете выбрать свою тему, задав `theme` параметр в

`config/config.yml` имени папки вашей темы.

Тема по умолчанию Pico на самом деле не предназначена для использования на продуктивном веб-сайте, это скорее отправная точка для создания вашей собственной темы. Если вам недостаточно темы по умолчанию и вы не хотите создавать свою собственную тему, вы можете использовать одну из замечательных тем, созданных сторонними разработчиками и дизайнерами в прошлом. Как и в случае с плагинами, вы можете найти темы в нашей Wiki и на нашем веб-сайте.

Все темы должны содержать `index.twig` файл для определения HTML-структуры темы и `pico-theme.yml` для установки необходимых параметров конфигурации. Просто обратитесь к теме Pico по умолчанию в качестве примера. Вы можете использовать разные шаблоны для разных файлов содержимого, указав `Template` мета-заголовок. Просто добавьте, например, `Template: blog` в заголовок YAML файла содержимого, и Pico будет использовать `blog.twig` шаблон в вашей папке темы для отображения страницы.

Ниже приведены переменные Twig, которые доступны для использования в темах. Пожалуйста, обратите внимание, что URL-адреса (например, `{{ base_url }}`) никогда не содержат косую черту в конце.

- `{{ site_title }}` - Сокращение к названию сайта (см. `config/config.yml`)
- `{{ config }}` - Содержит значения, которые вы задаете в `config/config.yml` (например, `{{ config.theme }}` становится `default`)
- `{{ base_url }}` - URL-адрес вашего сайта Pico; используйте `link` фильтр Twig для указания внутренних ссылок (например, `{{ "sub/page"|link }}`), это гарантирует, что ваша ссылка работает независимо от того, включена перезапись URL или нет
- `{{ theme_url }}` - URL-адрес текущей активной темы
- `{{ assets_url }}` - URL-адрес `assets` каталога Pico
- `{{ themes_url }}` - URL-адрес `themes` каталога Pico; не путайте это с `{{ theme_url }}`
- `{{ plugins_url }}` - URL-адрес `plugins` каталога Pico
- `{{ version }}` - Строка текущей версии Pico (например, `2.1.4`)
- `{{ meta }}` - Содержит мета-значения текущей страницы
  - `{{ meta.title }}` - `Title` Заголовок YAML
  - `{{ meta.description }}` - `Description` Заголовок YAML
  - `{{ meta.author }}` - `Author` Заголовок YAML
  - `{{ meta.date }}` - `Date` Заголовок YAML
  - `{{ meta.date_formatted }}` - Дата форматирования страницы, указанная `date_format` параметром в вашем `config/config.yml`
  - `{{ meta.time }}` - Временная метка Unix, полученная из `Date` заголовка YAML

    - `{{ meta.robots }}` - `Robots` Заголовок YAML
    - ...
- `{{ content }}` - Содержимое текущей страницы после ее обработки с помощью Markdown
- `{{ previous_page }}` - Данные предыдущей страницы, относящиеся к `current_page`
- `{{ current_page }}` - Данные текущей страницы; обратитесь к разделу "Страницы" ниже для получения подробной информации
- `{{ next_page }}` - Данные следующей страницы, относящиеся к `current_page`

Чтобы вызвать ресурсы из вашей темы, используйте `{{ theme_url }}`. Например, чтобы включить файл CSS `themes/my_theme/example.css`, добавьте `<link rel="stylesheet" href="{{ theme_url }}/example.css" type="text/css" />` в свой `index.twig`. Это работает для произвольных файлов в папке вашей темы, включая изображения и файлы JavaScript.

Пожалуйста, обратите внимание, что Twig экранирует HTML во всех строках перед их выводом. Так, например, если вы добавите `headline: My <strong>favorite</strong> color` в заголовок страницы YAML и выведете его с помощью `{{ meta.headline }}`, вы в конечном итоге увидите `My <strong>favorite</strong> color` - да, включая разметку! Чтобы на самом деле проанализировать его, вы должны использовать `{{ meta.headline|raw }}` (что приводит к ожидаемому `My **favorite** color` результату). Заметными исключениями из этого правила являются `content` переменная Pico (например, `{{ content }}`), `content` фильтр Pico (например, `{{ "sub/page"|content }}`) и `markdown` фильтр Pico, все они помечены как безопасные для HTML.

Работа со страницами

Существует несколько способов получить доступ к списку страниц Pico. Вы можете получить доступ к данным текущей страницы с помощью `current_page` переменной или использовать `prev_page` и / или `next_page` переменные для доступа к соответствующей предыдущей / следующей странице в списке страниц Pico. Но что более важно, есть `pages()` функция. Независимо от того, как вы получаете доступ к странице, она всегда будет состоять из следующих данных:

- `{{ id }}` - Относительный путь к файлу содержимого (уникальный идентификатор)
- `{{ url }}` - URL-адрес страницы
- `{{ title }}` - Заголовок страницы (`Title` заголовок YAML)
- `{{ description }}` - Описание страницы (`Description` заголовок YAML)
- `{{ author }}` - Автор страницы (`Author` заголовок YAML)
- `{{ date }}` - Дата страницы (`Date` заголовок YAML)
- `{{ date_formatted }}` - Дата форматирования страницы, указанная `date_format` параметром в вашем `config/config.yml`
- `{{ time }}` - Временная метка Unix, полученная из даты страницы

- `{{ raw_content }}` - Необработанное, еще не проанализированное содержимое страницы; используйте фильтр, чтобы получить проанализированное содержимое страницы, передав ее уникальный идентификатор (например, `{{ "sub/page"|content }}`)
- `{{ meta }}` - Мета-значения страницы (см. Глобальные `{{ meta }}` выше)
- `{{ prev_page }}` - Данные соответствующей предыдущей страницы
- `{{ next_page }}` - Данные соответствующей следующей страницы
- `{{ tree_node }}` - Узел страницы в дереве страниц Pico; подробности см. в документации по дереву страниц Pico

`pages()` Функция Pico - это лучший способ получить доступ ко всем страницам вашего сайта. Он использует дерево страниц Pico, чтобы легко перемещаться по подмножеству списка страниц Pico. Это позволяет фильтровать страницы и создавать рекурсивные меню (например, выпадающие списки). По умолчанию `pages()` возвращает список всех основных страниц (например, `content/page.md` и `content/sub/index.md`, но не `content/sub/page.md` или `content/index.md`). Если вы хотите вернуть все страницы, расположенные ниже определенной папки (например, `content/blog/`), передайте название папки в качестве первого параметра функции (например, `pages("blog")`). Естественно, вы также можете передавать функции переменные. Например, чтобы вернуть список всех дочерних страниц текущей страницы, используйте `pages(current_page.id)`. Ознакомьтесь со следующим фрагментом кода:

```
<section class="articles">
    {% for page in pages(current_page.id) if not page.hidden %}
        <article>
            <h2><a href="{{ page.url }}">{{ page.title }}</a></h2>
            {{ page.id|content }}
        </article>
    {% endfor %}
</section>
```

`pages()` Функция очень мощная, а также позволяет возвращать не только дочерние страницы страницы путем передачи параметров `depth` и `depthOffset`. Для например, если вы пройдете `pages(depthOffset=-1)`, в список также будут включены имена Пико главная страница индекса (т.е. `content/index.md`). Этот обычно используется для создания основная навигация по теме. Если вы хотите узнать больше, зайдите в Pico заполните `pages()` документацию по функциям.

Если вы хотите получить доступ к данным определенной страницы, используйте `pages` переменную Pico. Просто возьмите `content/_meta.md` пример содержимого Pico для примера: `content/_meta.md` содержит некоторые метаданные, которые вы, возможно, захотите использовать в своей теме. Если вы хотите вывести `tagline` мета-значение страницы,

используйте `{{ pages["_meta"].meta.logo }}`. Никогда не пытайтесь использовать `pages` переменную Pico в качестве замены `pages()` функции Pico. Его использование выглядит очень похоже, оно вроде как работает, и вы даже можете увидеть, как оно используется в старых темах, но имейте в виду: это замедляет работу Pico. Всегда используйте `pages()` функцию Pico при повторении списка страниц Pico (например, `{% for page in pages() %}…{% endfor %}`).

Фильтры и функции Twig

В дополнение к обширному списку фильтров, функций и тегов Twig, Pico также предоставляет несколько полезных дополнительных фильтров и функций, которые еще больше упрощают тематизацию.

- Передайте уникальный идентификатор страницы `link` фильтру, чтобы вернуть URL страницы (например, `{{ "sub/page"|link }}` возвращает `http://localhost/pico/?sub/page`).
- Вы можете заменить заполнители URL-адресов (например, `%base_url%`) в произвольных строках, используя `url` фильтр. Это полезно вместе с метапеременными, например, если вы добавите `image: %assets_url%/stock.jpg` в заголовок страницы YAML, `{{ meta.image|url }}` вернутся `http://localhost/pico/assets/stock.jpg`.
- Чтобы получить проанализированное содержимое страницы, передайте ее уникальный идентификатор `content` фильтру (например `{{ "sub/page"|content }}`).
- Вы можете проанализировать любую строку Markdown с помощью `markdown` фильтра. Например, вы могли бы использовать Markdown в `description` мета-переменной, а затем проанализировать ее в своей теме с помощью `{{ meta.description|markdown }}`. Вы также можете передать метаданные в качестве параметра для замены `%meta.*%` заполнителей (например, `{{ "Written by **"|markdown(meta) }}` выдает "Написано *Джоном Доу*"). Однако, пожалуйста, обратите внимание, что все содержимое будет заключено в элементы абзаца HTML (т.е. `<p>…</p>`). Если вы хотите проанализировать только одну строку разметки Markdown, передайте `singleLine` параметр в `markdown` фильтр (например `{{ "This really is a *single* line"|markdown(singleLine=true) }}`).
- Массивы могут быть отсортированы по одному из его ключей с помощью `sort_by` фильтра (например, `{% for page in pages|sort_by([ 'meta', 'nav' ]) %}...{% endfor %}` выполняется итерация по всем страницам, упорядоченным по `nav` мета-заголовку; пожалуйста, обратите внимание на `[ 'meta', 'nav' ]` часть примера, она предписывает Pico сортировать по `page.meta.nav`). Элементы, которые не удалось отсортировать, перемещаются в нижнюю часть массива; вы можете указать `bottom` (переместить элементы в нижнюю часть; по умолчанию), `top` (переместить элементы в начало), `keep`

(сохранить первоначальный порядок) или `remove` (удалить элементы) в качестве второго параметра для изменения этого поведения.

- Вы можете вернуть все значения данного ключа массива, используя `map` фильтр (например, `{{ pages|map("title") }}` возвращает все заголовки страниц).
- Используйте функции `url_param` и `form_param` Twig для доступа к параметрам HTTP GET (т. Е. к строке запроса URL, подобной `?some-variable=my-value`) и HTTP POST (т.Е. к данным отправленной формы). Это позволяет реализовать такие вещи, как разбивка на страницы, теги и категории, динамические страницы и даже больше – с помощью pure Twig! Просто перейдите на нашу вводную страницу для получения доступа к параметрам HTTP для получения подробной информации.

Плагины

Плагины для пользователей

Официально протестированные плагины можно найти по адресу http://picocms.org/plugins /, но существует множество потрясающих сторонних плагинов! Хорошей отправной точкой для открытий является наша Wiki.

Pico позволяет вам очень легко добавлять новые функции на ваш сайт с помощью плагинов. Так же, как и Pico, вы можете устанавливать плагины либо с помощью Composer (например, `composer require phrozenbyte/pico-file-prefixes`), либо вручную, загрузив файл плагина (только для небольших плагинов, состоящих из одного файла, например, `PicoFilePrefixes.php`) или каталог (например, `PicoFilePrefixes`) в свой `plugins` каталог. Мы всегда рекомендуем вам использовать Composer, когда это возможно, потому что это упрощает обновление как Pico, так и ваших плагинов. В любом случае, в зависимости от плагина, который вы хотите установить, вам, возможно, придется выполнить еще несколько шагов (например, указать переменные конфигурации), чтобы плагин заработал. Таким образом, вы всегда должны проверять документы плагина или `README.md` файл, чтобы узнать необходимые шаги.

Плагины, которые были написаны для работы с Pico 1.0 и более поздними версиями, можно включать и отключать с помощью вашего `config/config.yml`. Если вы хотите, например, отключить `PicoDeprecated` плагин, добавьте следующую строку в свой `config/config.yml`: `PicoDeprecated.enabled: false`. Чтобы принудительно включить плагин, замените его `false` на `true`.

Плагины для разработчиков

Вы разработчик плагинов? Мы любим вас, ребята! Вы можете найти массу информации о том, как разрабатывать плагины на http://picocms.org/development /. Если вы ранее

разрабатывали плагин и хотите обновить его до Pico 2.0, обратитесь к разделу "Обновление" документации.

Конфигурация

Настройка Pico действительно до глупости проста: просто создайте `config/config.yml`, чтобы переопределить настройки Pico по умолчанию (и добавить свои собственные пользовательские настройки). Взгляните на `config/config.yml.template`, чтобы получить краткий обзор доступных настроек и их значений по умолчанию. Чтобы переопределить параметр, просто скопируйте строку из `config/config.yml.template` в `config/config.yml` и задайте свое пользовательское значение.

Но мы на этом не остановились. Вместо того, чтобы использовать только один конфигурационный файл, вы можете использовать произвольное количество конфигурационных файлов. Просто создайте `.yml` файл в `config` каталоге Pico, и все готово. Это позволяет вам добавить некоторую структуру в вашу конфигурацию, например, отдельный конфигурационный файл для вашей темы (`config/my_theme.yml`).

Пожалуйста, обратите внимание, что Pico загружает конфигурационные файлы особым способом, о котором вам следует знать. Прежде всего, он загружает основной конфигурационный файл `config/config.yml`, а затем любой другой `*.yml` файл в `config` каталоге Pico в алфавитном порядке. Порядок файлов имеет решающее значение: значения конфигурации, которые уже были установлены, не могут быть перезаписаны последующим файлом. Например, если вы укажете `site_title: Pico` в `config/a.yml` и `site_title: My awesome site!` в `config/b.yml`, название вашего сайта будет "Pico".

Поскольку файлы YAML представляют собой обычные текстовые файлы, пользователи могут ознакомиться с вашей конфигурацией Pico, перейдя по `http://localhost/pico/config/config.yml`. Изначально это не проблема, но может возникнуть проблема, если вы используете плагины, которые требуют, чтобы вы хранили в конфигурации данные, относящиеся к безопасности (например, учетные данные). Таким образом, вы должны *всегда* быть уверены, что настроили свой веб-сервер на запрет доступа к `config` каталогу Pico. Просто обратитесь к разделу "Переписывание URL" ниже. Следуя инструкциям, вы не только разрешите перезапись URL-адреса, но и запретите доступ к `config` каталогу Pico.

Переписывание URL

URL-адреса Pico по умолчанию (например http://localhost/pico/?sub/page) уже очень удобны для пользователя. Кроме того, Pico предлагает вам функцию перезаписи URL, чтобы сделать URL-адреса еще более удобными для пользователя (например

http://localhost/pico/sub/page). Ниже вы найдете некоторую базовую информацию о том, как настроить ваш веб-сервер, вероятно, для включения перезаписи URL.

Apache

Если вы используете веб-сервер Apache, перезапись URL, вероятно, уже включена – попробуйте сами, нажмите на второй URL. Если перезапись URL не работает (вы получаете `404 Not Found` сообщения об ошибках от Apache), пожалуйста, обязательно включите `mod_rewrite` модуль и включите `.htaccess` переопределения. Возможно, вам придется установить для `AllowOverride` директивы значение `AllowOverride All` в вашем файле конфигурации виртуального хоста или глобальном `httpd.conf`/`apache.conf`. Предполагая, что перезаписанные URL-адреса работают, но Pico по-прежнему не показывает перезаписанные URL-адреса, принудительно измените URL-адрес, установив `rewrite_url: true` в вашем `config/config.yml`. Если вы предпочитаете получать `500 Internal Server Error` независимо от того, что вы делаете, попробуйте удалить `Options` директиву из `.htaccess` файла Pico (это последняя строка).

Nginx

Если вы используете Nginx, вы можете использовать следующую конфигурацию, чтобы включить перезапись URL (строки `5` на `8`) и запретить доступ к внутренним файлам Pico (строки `1` на `3`). Вам нужно будет настроить путь (`/pico` в строках `1`, `2`, `5` и `7`) в соответствии с вашим каталогом установки. Кроме того, вам нужно будет включить перезапись URL-адреса, настроив `rewrite_url: true` в вашем `config/config.yml`. Конфигурация Nginx должна обеспечивать *абсолютный минимум*, необходимый для Pico. Nginx – это очень обширная тема. Если у вас возникнут какие-либо проблемы, пожалуйста, ознакомьтесь с нашей документацией по конфигурации Nginx.

```
location ~ ^/pico/((config|content|vendor|composer\.(json|lock|phar))(/|$)|(.+/)?\.(
    try_files /pico/index.php$is_args$args =404;
}

location /pico/ {
    index index.php;
    try_files $uri $uri/ /pico/index.php$is_args$args;
}
```

Полиция света

Pico работает без сбоев в Lighttpd. Вы можете использовать следующую конфигурацию, чтобы включить перезапись URL (строки `6` на `9`) и запретить доступ к внутренним файлам Pico (строки `1` на `4`). Обязательно настройте путь (`/pico` в строках `2`, `3` и `7`) в соответствии с вашим каталогом установки и сообщите Pico о возможности перезаписи URL-

адреса, установив `rewrite_url: true` в вашем `config/config.yml`. Приведенная ниже конфигурация должна обеспечить *абсолютный минимум*, необходимый для Pico.

```
url.rewrite-once = (
    "^/pico/(config|content|vendor|composer\.(json|lock|phar))(/|$)" => "/pico/inde
    "^/pico/(.+/)?\.(?!well-known(/|$))" => "/pico/index.php"
)

url.rewrite-if-not-file = (
    "^/pico(/|$)" => "/pico/index.php"
)
```

Документация

Для получения дополнительной помощи ознакомьтесь с документацией Pico по адресу http://picocms.org/docs.

Pico был создан Жильбером Пеллегромом и поддерживается Сообществом Pico. Выпущен по лицензии MIT.