Многоязычность в Astro

📖 Оглавление

- Минимальная рабочая конфигурация под мой случай

⚠ Проблема

```
Многоязычность в проекте на Astro
```

🎯 Цель

- Настроить и понять многоязычность в проекте на Astro

Документация

Многоязычная маршрутизация (**i18n**)
Функции интернационализации Astro

позволяют вам адаптировать ваш проект для международной аудитории. Этот API маршрутизации помогает вам создавать, использовать и проверять URL-адреса, которые производит ваш многоязычный сайт.

Маршрутизация **i18n** Astro позволяет вам представлять ваше многоязычное содержимое с поддержкой настройки языка по умолчанию, вычисления относительных URL-адресов страниц и принятия предпочтительных языков, предоставляемых браузером вашего посетителя. Вы также можете указать резервные языки для каждого языка отдельно, чтобы ваши посетители всегда могли быть направлены к существующему контенту на вашем сайте.

Логика маршрутизации

Astro использует мидлвары для реализации своей логики маршрутизации. Функция мидлвара размещается в первой позиции, где она ожидает каждого Response, поступающего от любого дополнительного мидлвара, и каждого маршрута страницы, прежде чем, наконец, выполнить свою собственную логику.

Это означает, что сначала выполняются операции (например, перенаправления) из вашего собственного мидлвара и логики вашей страницы, визуализируются ваши маршруты, а затем мидлвар **i18n** выполняет свои собственные действия, такие как проверка соответствия локализованного URL-адреса допустимому маршруту.

Вы также можете добавить свою собственную логику **i18n** в дополнение или вместо мидлвара **Astro i18n**, что даст вам ещё больший контроль над вашими маршрутами, сохраняя при этом доступ к вспомогательным функциям astro: **i18n**.

Настройка **i18n**-маршрутизации И язык по умолчанию (defaultLocale (EN)), и список всех поддерживаемых языков (locales (EN)) должны быть указаны в объекте конфигурации **i18n**. Кроме того, можно настроить более конкретную маршрутизацию и резервное поведение в соответствии с желаемыми URL-адресами.

```
astro.config.mjs
import { defineConfig }
from "astro/config"
export default
defineConfig({
```

```
  **i18n**: {
    locales: ["ru", "en",
"pt-br"],
    defaultLocale: "en",
  }
})
```

a. Создание локализованных папок

Организуйте папки с контентом, локализованным по языку. Создайте отдельные папки **/[locale]/** в любом месте **src/pages/**, и маршрутизация на основе файлов Astro создаст ваши страницы по соответствующим URL-путям.

Имена ваших папок должны точно соответствовать элементам в **locales**. Включите локализованную папку для вашей **defaultLocale**, только если вы

настроили **prefixDefaultLocale: true** для отображения локализованного URL-пути для вашего языка по умолчанию (например, **/en/about/**).

```
|- src
  |- pages
    |- about.astro
    |- index.astro
    |- ru
      |- about.astro
      |- index.astro
    |- pt-br
      |- about.astro
      |- index.astro
```

- Локализованные папки не обязательно должны находиться в корне директории **/pages/**.

b. Создание ссылок

Настроив маршрутизацию **i18n**, вы теперь можете вычислять ссылки на страницы вашего сайта, используя вспомогательные функции, такие как **getRelativeLocaleUrl()**, доступные из astro: **i18n**. Эти сгенерированные ссылки всегда будут обеспечивать правильный локализованный маршрут и могут помочь вам правильно использовать или проверять URL-адреса на вашем сайте.

Вы также можете написать ссылки вручную.

```
src/pages/ru/index.astro
---
import {
getRelativeLocaleUrl } from
'astro:i18n';

// defaultLocale = "ru"
```

```
const aboutURL =
getRelativeLocaleUrl("ru",
"about");
---

<a href="/get-
started/">Начнём!</a>
<a href=
{getRelativeLocaleUrl('ru',
'blog')}>Блог</a>
<a href={aboutURL}>О
сайте</a>
```

c. routing

Встроенная файловая маршрутизация Astro автоматически создает для вас URL-маршруты на основе вашей файловой структуры в **src/pages/**.

Когда вы настраиваете **i18n**-маршрутизацию, информация об этой

структуре файлов (и соответствующие сгенерированные пути URL) доступна вспомогательным функциям **i18n**, чтобы они могли генерировать, использовать и проверять маршруты в вашем проекте. Многие из этих опций можно использовать вместе для ещё большей настройки и гибкости для каждого языка.

Вы даже можете реализовать собственную логику маршрутизации вручную для ещё большего контроля.

d. prefixDefaultLocale

- *Добавлено в: astro@3.5.0*

Этот параметр маршрутизации определяет, должны ли URL-адреса вашего языка по умолчанию использовать языковой префикс (например, **/en/about/**).

Все языки, поддерживаемые не по умолчанию, будут использовать локализованный префикс (например, **/ru/** или **/russian/**), а файлы содержимого должны находиться в соответствующих папках. Этот параметр конфигурации позволяет указать, должен ли язык по умолчанию также следовать локализованной структуре URL.

Этот параметр также определяет, где должны находиться файлы страниц для вашего языка по умолчанию (например, **src/pages/about/** или **src/pages/en/about**), так как структура файлов и структура URL должны совпадать для всех языков.

"prefixDefaultLocale: false" (по умолчанию): URL-адреса на вашем языке по умолчанию не будут иметь префикса **/[locale]/**. Все остальные локали будут.

"prefixDefaultLocale: true": Все URL-адреса, включая язык по умолчанию, будут иметь префикс **/[locale]/**.

```
prefixDefaultLocale: false
astro.config.mjs
```

```
import { defineConfig }
from "astro/config"
export default
defineConfig({
  **i18n**: {
    defaultLocale: "en",
    locales: ["ru", "en",
"fr"],
    routing: {

prefixDefaultLocale: false
    }
```

```
    }
  })
```

Это значение по умолчанию. Установите эту опцию, если URL-адреса на вашем языке по умолчанию не будут иметь префикс **/[locale]/**, а файлы на вашем языке по умолчанию существуют в корне **src/pages/**:

```
|-src
  |-pages
    |-about.astro
    |-index.astro
    |-ru
      |-about.astro
      |-index.astro
    |-fr
      |-about.astro
      |-index.astro
```

```
src/pages/about.astro    --
-> создаст маршрут --->
example.com/about/
src/pages/fr/about.astro --
-> создаст маршрут --->
example.com/fr/about/
```

```
prefixDefaultLocale: true
astro.config.mjs
```

```
import { defineConfig }
from "astro/config"
export default
defineConfig({
  **i18n**: {
    defaultLocale: "en",
    locales: ["ru", "en",
"fr"],
```

```
    routing: {

prefixDefaultLocale: true
    }
  }
})
```

Установите эту опцию, когда все маршруты будут иметь префикс **/locale/** в своем URL и когда все файлы содержимого страницы, включая файлы для вашей **defaultLocale**, существуют в локализованной папке:

```
|-src
  |-pages
    |-index.astro //
Примечание: этот файл
всегда необходим
    |-en
      |-index.astro
```

```
        |-about.astro
      |-ru
        |-about.astro
        |-index.astro
      |-pt-br
        |-about.astro
        |-index.astro
```

URL-адреса без префикса локали, (например, **example.com/about/**) вернут код состояния **404** (страница не найдена), если вы не укажете резервную локаль.

e. redirectToDefaultLocale

- *Добавлено в: astro@4.2.0*

Определяет, будет ли URL главной страницы (**/**), сгенерированный **src/pages/index.astro**, перенаправляться на **/<defaultLocale>**.

Установка **prefixDefaultLocale: true** также автоматически установит **redirectToDefaultLocale: true** в вашем объекте конфигурации **routing**. По умолчанию требуемый файл **src/pages/index.astro** автоматически перенаправляется на индексную страницу вашего языка по умолчанию.

Вы можете отказаться от такого поведения, установив **redirectToDefaultLocale: false (EN)**. Это позволяет вам иметь домашнюю страницу сайта, которая существует за пределами настроенной вами структуры папок локали.

f. manual

- *Добавлено в: astro@4.6.0*

Когда эта опция включена, Astro отключит свой мидлвар **i18n**, чтобы вы могли реализовать собственную пользовательскую логику. Никакие другие параметры **routing** (например, **prefixDefaultLocale**) не могут быть настроены с помощью **routing: "manual"**.

Вы будете нести ответственность за написание своей собственной логики маршрутизации или выполнение мидлвара **Astro i18n** вручную наряду со своим собственным.

astro.config.mjs

```
import { defineConfig }
from "astro/config"
export default
```

```
defineConfig({
  **i18n**: {
    defaultLocale: "en",
    locales: ["ru", "en",
"fr"],
    routing: "manual"
  }
})
```

Astro предоставляет вспомогательные функции для вашего мидлвара, чтобы вы могли самостоятельно управлять маршрутизацией по умолчанию, исключениями, резервным поведением, отслеживанием ошибок и т. д.:

```
redirectToDefaultLocale(),
NotFound(), и
redirectToFallback():
```

src/middleware.js

```
import { defineMiddleware }
from "astro:middleware";
import {
redirectToDefaultLocale }
from "astro:i18n";
//  ^ функция доступна с
ручной маршрутизацией ^
export const onRequest =
defineMiddleware(async
(ctx, next) => {
  if
(ctx.url.startsWith("/about
")) {
    return next();
  } else {
    return
redirectToDefaultLocale(302
);
```

```
  }
})
```

g. Функция middleware

Функция middleware вручную создаёт мидлвар **Astro i18n**. Это позволяет расширить маршрутизацию **Astro i18n** вместо её полной замены.

Вы можете запустить middleware с параметрами маршрутизации в сочетании с вашим собственным мидлваром, используя утилиту **sequence (EN)** для определения порядка:

```
src/middleware.js
```

```
```

```
import {defineMiddleware,
sequence} from
"astro:middleware";
import { middleware } from
"astro:i18n"; //
Собственная конфигурация
маршрутизации **Astro
i18n**

export const userMiddleware
= defineMiddleware(async
(ctx, next) => {
  // этот ответ может
исходить от мидлвара
**Astro i18n**, и он может
возвращать 404
  const response = await
next();
  // страница /about
является исключением, и мы
хотим отобразить её
  if
(ctx.url.startsWith("/about
```

```
")) {
    return new
Response("About page", {
      status: 200
    });
  } else {
    return response;
  }
});


export const onRequest =
sequence(
  userMiddleware,
  middleware({

redirectToDefaultLocale:
false,
    prefixDefaultLocale:
true
  })
)
```

h. domains

- *Добавлено в: astro@4.9.0*

Эта опция маршрутизации позволяет вам настраивать ваши домены для каждого языка для проектов, отображаемых на server, с использованием **@astrojs/node (EN)** или адаптера **@astrojs/ vercel (EN)** с настроенным параметром **site**.

Добавьте **i18n.domains**, чтобы сопоставить любые поддерживаемые вами **locales** с пользовательскими URL-адресами:

```
astro.config.mjs
```

```
import { defineConfig }
from "astro/config"
export default
```

```
defineConfig({
  site:
"https://example.com",
  output: "server", //
обязательно, без
предварительно отрисованных
страниц
  adapter: node({
    mode: 'standalone',
  }),
  i18n: {
    locales: ["ru", "en",
"fr", "ja"],
    defaultLocale: "en",
    routing: {
      prefixDefaultLocale:
false
    },
    domains: {
        fr:
"https://fr.example.com",
        ru:
"https://example.ru"
    }
```

```
  }
})
```

Все несопоставленные **locales** будут соответствовать вашей конфигурации **prefixDefaultLocales**. Однако, даже если это значение равно **false**, файлы для вашей **defaultLocale** также должны существовать в локализованной папке. Для приведённой выше конфигурации требуется папка **/en/**.

С вышеуказанной конфигурацией:

```
Файл /fr/about.astro
создаст URL-адрес
https://fr.example.com/abou
t.
Файл /ru/about.astro
создаст URL-адрес
https://example.ru/about.
```

```
Файл /ja/about.astro
создаст URL-адрес
https://example.com/ja/abou
t.
Файл /en/about.astro
создаст URL-адрес
https://example.com/about.
```

Вышеуказанные URL-адреса также будут возвращены функциями

```
getAbsoluteLocaleUrl() и
getAbsoluteLocaleUrlList().
```

i. Резервная локаль

Когда страница на одном языке не существует (например, страница, которая ещё не переведена), вместо отображения страницы **404** вы можете выбрать

отображение резервного контента из другой **locale** на основе языка. Это полезно, когда у вас ещё нет страницы для каждого маршрута, но вы хотите предоставить некоторый контент вашим посетителям.

Ваша стратегия резервирования состоит из двух частей: выбор языков, которые должны стать резервными (**i18n.fallback** (EN)), и включение перенаправления или переписывание для отображения резервного контента (**i18n.routing.fallbackType** (EN) — это было добавлено в **Astro v4.15.0**).

Например, когда вы настраиваете **i18n.fallback: { fr: "ru" }**, **Astro** обеспечит создание страницы в **src/pages/fr/** для каждой страницы, которая существует в **src/pages/ru/**.

Если какая-либо страница ещё не существует, то страница будет создана в зависимости от вашего **fallbackType**:

С перенаправлением на соответствующий маршрут **ru** (поведение по умолчанию).

С содержимым страницы **/ru/** (**i18n.routing.fallbackType: "rewrite"**).

Например, приведённая ниже конфигурация устанавливает **ru** в качестве резервной локали для любых отсутствующих маршрутов **fr**. Это означает, что пользователь, посещающий *example.com/fr/my-page/*, будет перенаправлен на *example.com/ru/my-page/* вместо перехода на страницу **404**, если **src/pages/fr/my-page.астро** не существует.

astro.config.mjs

```
import { defineConfig }
from "astro/config"
export default
defineConfig({
  **i18n**: {
    locales: ["ru", "en",
"fr"],
    defaultLocale: "en",
    fallback: {
      fr: "ru"
    },
    routing: {
      fallbackType:
"rewrite"
    }
  }
})
```

j. Пользовательские пути локалей

В дополнение к определению поддерживаемых вашим сайтом языковых стандартов в виде строк (например, **en**, **pt-br**), **Astro** также позволяет сопоставить произвольное количество языковых кодов, распознаваемых браузером с пользовательским URL-адресом **path**. Хотя локали могут быть строками любого формата, если они соответствуют структуре папок вашего проекта, codes должны соответствовать принятому в браузере синтаксису.

Передайте объект в массив **locales** с ключом **path** для определения пользовательского префикса URL-адреса и **codes** для указания языков,

сопоставленных с этим URL-адресом. В этом случае имя вашей папки **/[locale]/** должно точно соответствовать значению **path**, и ваши URL-адреса будут созданы с использованием значения **path**.

Это полезно, если вы поддерживаете несколько вариантов языка (например, **"fr"**, **"fr-BR"** и **"fr-CA"**) и хотите, чтобы все эти варианты отображались под одним и тем же URL-адресом **/fr/**, или даже полностью настроить его (например, **/french/**):

astro.config.mjs

```
import { defineConfig }
from "astro/config"
export default
```

```
defineConfig({
  i18n: {
    defaultLocale: "en",
    locales: ["ru", "en",
"fr"],
    locales: ["ru", "en", {
        path: "french", //
косые черты не включены
        codes: ["fr", "fr-
BR", "fr-CA"]
    }],
    routing: {

prefixDefaultLocale: true
    }
  }
})
```

При использовании функций из виртуального модуля astro: **i18n** для вычисления допустимых URL-путей на основе вашей конфигурации (например,

getRelativeLocaleUrl()), используйте path как значение **locale**.

k. Ограничения

Эта функция имеет некоторые ограничения:

```
Опция site является
обязательной.
Опция output должна иметь
значение "server".
```

Не может быть отдельных пререндеренных страниц. Для поддержки этой функции **Astro** использует следующие заголовки:

X-Forwarded-Host и **Host**. **Астро** будет использовать первый вариант, а если его

нет, то попробует второй. **X-Forwarded-Proto** и **URL#protocol** запроса сервера.

! Убедитесь, что ваш прокси-сервер/хостинговая платформа способны предоставить эту информацию. Невозможность получить эти заголовки приведет к появлению страницы **404** (код состояния).

l. Определение языка браузера

Маршрутизация **Astro i18n** позволяет вам получить доступ к двум свойствам для определения языка браузера на страницах, отображаемых по запросу: **Astro.preferredLocale** и **Astro.preferredLocaleList**. Все страницы, включая статические страницы с

предварительным отображением, имеют доступ к **Astro.currentLocale**.

Они объединяют заголовок **Accept-Language** браузера и ваши **locales** (строки или codes) для автоматического учёта предпочитаемых языков вашего посетителя.

-Astro.preferredLocale: **Astro** может вычислить предпочтительный языковой стандарт для вашего посетителя, если предпочтительный языковой стандарт его браузера включен в ваш массив **locales**. Это значение не определено, если такого соответствия не существует.

-Astro.preferredLocaleList: Массив всех локалей, которые запрашиваются браузером и поддерживаются вашим веб-сайтом. В результате выводится список всех языков, совместимых между вашим

сайтом и посетителем. Значением является [], если ни один из запрошенных браузером языков не найден в массиве **locales**. Если в браузере не указаны предпочтительные языки, то это значение будет **i18n.locales**.

-Astro.currentLocale: Локаль, рассчитанная на основе текущего URL-адреса с использованием синтаксиса, указанного в опции locales. Если URL-адрес не содержит префикс **/[locale]/**, то значением по умолчанию будет **i18n.defaultLocale** (EN).

Чтобы успешно соответствовать предпочтениям ваших посетителей, предоставляйте свои codes, используя тот же шаблон, используемый браузером.

Что такое i18n-маршрутизация в Astro “по-человечески”

Когда ты подключаешь **i18n** в astro.config.mjs, ты говоришь Астро:

“У меня сайт на нескольких языках. Вот список языков, какой из них основной, и как должны выглядеть URL”.

```
export default
defineConfig({
  i18n: {
    locales: ["cs", "en",
"ru", "uk"], // чешский,
английский, русский,
украинский
```

```
    defaultLocale: "cs",
  }
})
```

После этого Astro ожидает, что у тебя будут папки с контентом:

```
src/
  |-pages/
    |-cs/
      |-index.astro
    |-en/
      |-index.astro
    |-ru/
      |-index.astro
    |-uk/
      |-index.astro
```

🚫 Почему дублирование кода — не обязательно Astro очень гибкий. Ты не

обязан создавать отдельные файлы для каждой локали, если сам хочешь управлять контентом и переводами.

а. 3 нормальных способа избежать *копипасты*

Вариант 1: Один шаблон, разные переводы (лучший для статичных сайтов)

Ты создаёшь один общий файл, например:

```
src/pages/[locale]/index.as
tro
Внутри [locale] — это
параметр маршрута.
Astro сам подставит cs, en,
ru, uk в зависимости от
URL.
```

Пример:

```
---
//
src/pages/[locale]/index.as
tro
import translations from
"../../translations.js";

// Получаем текущий язык из
URL
const { locale } =
Astro.params;

// Если нет перевода,
используем чешский по
умолчанию
const t =
translations[locale] ||
translations["cs"];
---

<html lang={locale}>
```

```
  <head><title>{t.title}
</title></head>
  <body>
    <h1>{t.heading}</h1>
    <p>{t.description}</p>
  </body>
</html>
```

И отдельный файл с переводами:

```
// src/translations.js
export default {
  cs: {
    title: "Vítejte na
našem webu",
    heading: "Ahoj světe!",
    description: "Toto je
česká verze stránky.",
  },
  en: {
    title: "Welcome to our
```

```
website",
    heading: "Hello
world!",
    description: "This is
the English version.",
  },
  ru: {
    title: "Добро
пожаловать на наш сайт",
    heading: "Привет,
мир!",
    description: "Это
русская версия сайта.",
  },
  uk: {
    title: "Ласкаво просимо
на наш сайт",
    heading: "Привіт,
світе!",
    description: "Це
українська версія
сторінки.",
  },
};
```

✅ Плюсы:

один файл .astro для всех языков;

структура не дублируется;

легко добавлять новые языки (только в переводах);

всё работает с **i18n**-маршрутизацией.

Вариант 2: Использовать Markdown или Content Collections

Если у тебя контентный сайт (например, блог), то переводы можно хранить в **/src/content**, например:

```
src/
  |-content/
    |-posts/
```

```
        |-cs/
          |-hello.md
        |-en/
          |-hello.md
        |-ru/
          |-hello.md
```

А шаблон страницы (**src/pages/[locale]/blog/[slug].astro**) будет один. Он просто подгружает нужный файл контента на нужном языке.

Вариант 3: Ручная маршрутизация (routing: "manual")

Если тебе нужно максимум контроля — можно полностью отключить встроенную логику **Astro** и самому решать, какой язык показывать.

Тогда ты сам обрабатываешь **/**, **/en/**, **/ru/** и т.п. Но это уже “профи-режим” — удобнее

начать с встроенной **i18n**.

Итого

Задача ---> Что делать

Не хочу копировать файлы для каждого языка --->

```
Используй
**src/pages/[locale]/page.a
stro**
```

Хочу разные переводы --->

```
Подключай JS-объект или
**JSON** с переводами
```

Хочу контролировать всё сам --->

```
Включай routing: "manual" и
используй middleware
```

Хочу автоматический редирект по языку браузера --->

```
**Astro** сделает сам, если
включено **i18n**
```

Минимальная рабочая конфигурация под мой случай

```
```

```
🧱 Структура проекта
|-astro.config.mjs
|-05_src/
  |-translations.js
  |- pages/
      |-index.astro //
требует корневой
index.astro в /src/pages/,
даже если ты используешь
i18n
      |- [locale]/
            |- index.astro
```

```
// src/pages/index.astro
---
import {
redirectToDefaultLocale }
from "astro:i18n";

// перенаправляем на локаль
по умолчанию (cs)
```

```
export const prerender =
false;
export async function GET()
{
  return
redirectToDefaultLocale();
}
---
```

```
//⚙ astro.config.mjs
import { defineConfig }
from "astro/config";

export default
defineConfig({
  projectRoot: "./05_src",
  srcDir: "./05_src",
  outDir: '02_dist', // вот
сюда будет билд
  i18n: {
    locales: ["cs", "en",
"ru", "uk"], // чешский,
```

```
английский, русский,
украинский
    defaultLocale: "cs",
    routing: {
      prefixDefaultLocale:
true,
    },
  },
});
```

```
//🌍 src/translations.js
export default {
  cs: {
    title: "Vítejte na
našem webu",
    heading: "Ahoj světe!",
    description: "Toto je
česká verze stránky.",
  },
  en: {
    title: "Welcome to our
website",
```

```
    heading: "Hello
world!",
    description: "This is
the English version.",
  },
  ru: {
    title: "Добро
пожаловать на наш сайт",
    heading: "Привет,
мир!",
    description: "Это
русская версия страницы.",
  },
  uk: {
    title: "Ласкаво просимо
на наш сайт",
    heading: "Привіт,
світе!",
    description: "Це
українська версія
сторінки.",
  },
};
```

```
//🧩
src/pages/[locale]/index.as
tro
---
import translations from
"../../translations.js";

export async function
getStaticPaths() {
  // Astro требует объявить
доступные маршруты для
[locale]
  return [
    { params: { locale:
"cs" } },
    { params: { locale:
"en" } },
    { params: { locale:
"ru" } },
    { params: { locale:
"uk" } },
  ];
}
```

```
const { locale } =
Astro.params;
const t =
translations[locale] ||
translations["cs"];
---

<html lang={locale}>
  <head>
    <meta charset="utf-8"
/>
    <title>{t.title}
</title>
  </head>
  <body>
    <h1>{t.heading}</h1>
    <p>{t.description}</p>
    <nav>
      <a
href="/cs/">Čeština</a> |
      <a
href="/en/">English</a> |
      <a
href="/ru/">Русский</a> |
```

```
      <a
href="/uk/">Українська</a>
    </nav>
  </body>
</html>
```

✅ Теперь запускаешь:

```
npm run dev
```

и переходишь:

```
http://localhost:4321/ --->
редирект на /cz/

http://localhost:4321/cs

http://localhost:4321/en
```

```
http://localhost:4321/ru

http://localhost:4321/uk
```

- Всё работает: никаких ошибок *getStaticPathsRequired*, один .astro файл, 4 языка