Neos SEO Addon Documentation

Release dev-master

The Neos Team

May 02, 2023

Contents

1 Fallback definitions **3**
- 1.1 Open Graph Metatags . . . 3
- 1.2 TwitterCard Metatags . . . 4

2 Installation **5**

3 Page title **7**

4 Basic meta tags **9**

5 Twitter Cards **11**

6 Facebook **13**

7 Open Graph **15**

8 Define image fallback for Twitter Cards or Open Graph **17**

9 Structured data **19**
- 9.1 Google Search Console . . . 19
- 9.2 Breadcrumb . . . 19
- 9.3 Website & Searchbox . . . 20
- 9.4 Social profile . . . 20

10 XML sitemap **21**

11 Alternate Language Tag **23**

12 Dynamic robots.txt **25**

13 Disabling not needed features **27**

This documentation covering version dev-master has been rendered at: May 02, 2023

CHAPTER 1

Fallback definitions

As this package provides many SEO related fields and mechanisms it can be a lot of work for editors to fill everything the way it should be.

Therefore several default fallback chains are provided for the output. Those are described in the following sections.

Our goal is to make it easy for integrators to extend these chains. You can find them in the individual *Fusion* objects. And feel free to created issues on Github if you think that something could be improved!

1.1 Open Graph Metatags

Tag	Default	Fallbacks
og:type	Property *openGraphType*	*website*
og:title	Property *openGraphTitle*	1. Property *titleOverride* 2. Property *title*
og:description	Property *openGraphDescription*	Property *metaDescription*
og:site_name	Property *titleOverride* of homepage	Name defined in *Site Management*
og:image	Property *openGraphImage*	n/a
og:image:width	Width of processed *og:image*	n/a
og:image:height	Height of processed *og:image*	n/a
og:image:alt	Caption of image in *og:image*	Label of image
og:url	Page Url	n/a
og:locale	Locale from language dimension	n/a

1.2 TwitterCard Metatags

Tag	Default	Fallbacks
twitter:card	Property *twitterCardType*	*summary*
twitter:title	Property *twitterCardTitle*	1. Property *openGraphTitle* 2. Property *titleOverride* 3. Property *title*
twitter:description	Property *twitterCardDescription*	1. Property *openGraphDescription* 2. Property *metaDescription*
twitter:creator	Property *twitterCardCreator*	Configuration *Neos.Seo.twitterCard.siteHandle*
twitter:image	Property *twitterCardImage*	Property *openGraphImage*
twitter:url	Page Url	n/a
twitter:site	Configuration *Neos.Seo.twitterCard.siteHandle*	n/a

CHAPTER 2

Installation

Install the package through composer:

```
composer require neos/seo
```

CHAPTER 3

Page title

The default *<title>* tag rendering in the *Neos.Neos:Page* Fusion object is a "reverse breadcrumb" of the regular title field(s). This is done in *Neos.Seo:TitleTag* with a configurable separator.

A new field *titleOverride* is added to *Neos.Neos:Document* via the *Neos.Seo:TitleTagMixin*. The new field is used as the *<title>* tag content if it is filled (see *Neos.Seo:TitleTag*).

By default the sites name will be added as suffix. You can also configure this behavior in *Neos.Seo:TitleTag*.

CHAPTER 4

Basic meta tags

The fields for keywords and description are added to *Neos.Neos:Document* via the *Neos.Seo:SeoMetaTagsMixin*

If they are filled in, *<meta>* tags for their contents will be rendered (see *Neos.Seo:MetaKeywordsTag* and *Neos.Seo:MetaDescriptionTag*).

Two checkboxes allow to set the content for the *<meta name="robots">* tag to any combination of the possible values *follow*, *nofollow*, *index* and *noindex*.

CHAPTER 5

Twitter Cards

The *Neos.Seo:TwitterCardMixin* (added to *Neos.Neos:Document* by default) provides a new inspector tab to configure Twitter Cards on any document. If a Twitter Card is enabled, the related meta tags will be rendered as needed and useful.

The *twitter:site* handle can be configured with the setting *Neos.Seo.twitterCard.siteHandle* by providing a valid Twitter handle:

```
Neos:
  Seo:
    twitterCard:
      siteHandle: '@neoscms'
```

Check the documentation on https://dev.twitter.com/cards/overview for more on Twitter Cards.

Also check out the defined **fallback-definitions_** for the individual tags.

CHAPTER 6

Facebook

The optional *fb:profile_id*, *fb:admins* and *fb:pages* tags can be configured with the settings in *Neos.Seo.facebook* and by providing valid Facebook user/channel ids:

```
Neos:
  Seo:
    facebook:
      profileId: 'myUserId'
      admins:
        - 'myAdmin1'
        - 'myAdmin2'
      pages:
        - 'myChannelId1'
        - 'myChannelId2'
```

Check the documentation on https://developers.facebook.com/docs/reference/opengraph/object-type/article for more on facebook specific tags.

You can add further tags if needed by extending the *Neos.Seo:FacebookMetaTags* prototype via Fusion in your own package.

CHAPTER 7

Open Graph

The *Neos.Seo:OpenGraphMixin* (added to *Neos.Neos:Document* by default) provides a new inspector tab to configure Open Graph on any document. The Open Graph protocol enables any web page to become a rich object in a social graph. The essential ones are:

- *og:type*
- *og:site_name*
- *og:title*
- *og:locale*
- *og:description*
- *og:image*
- *og:url*

Also check out the defined **fallback-definitions_** for the individual tags as most fields will be prefilled with values from other fields.

For more information about Open Graph please have a look at http://ogp.me/.

CHAPTER 8

Define image fallback for Twitter Cards or Open Graph

The fusion object of this package defines Case-Objects that you can use to add your fallback. Here are some examples how to do so:

Example: Add a document property *headerImage* as fallback, if no Twitter Card image is present:

```
prototype(Neos.Seo:TwitterCard) {
    image {
        asset {
            headerImage {
                condition = ${Type.instance(q(node).property('headerImage'),
↪'Neos\Media\Domain\Model\ImageInterface')}
                renderer = ${q(node).property('headerImage')}
            }
        }
    }
}
```

Example: Add the image of the first content node found that has an image property as fallback, when no Twitter Card image is present:

```
prototype(Neos.Seo:OpenGraphMetaTags) {
    image {
        contentImage {
            firstContentWithImage = ${q(node).children('[instanceof Neos.
↪Neos:ContentCollection]').find('[instanceof Neos.Neos:Content][image instanceof
↪"Neos\Media\Domain\Model\ImageInterface"]')}
            condition = ${this.firstContentWithImage.count()}
            renderer = ${this.firstContentWithImage.property('image')}
        }
    }
}
```

CHAPTER 9

Structured data

This package provides basic prototypes for structured data and some already implemented elements. You can activate and configure them according to your needs.

The prototypes are also meant to be a very basic standard for your own structured elements. So feel free to adjust and reuse them.

9.1 Google Search Console

When you activate structured data elements and have them on your live site you should use the Google Search Console to verify that they work as expected: https://search.google.com/search-console.

You can verify site ownership for the Google Search Console with the following setting:

```
Neos:
    Seo:
        google: siteVerification: 'yourVerificationProperty'
```

Or if you prefer having a property to edit in the Neos backend (or on multi site setups), add the following nodeType to the nodeType definition of your site node:

```
'Vendor.Package:Document.Home':
    superTypes: 'Neos.Seo:GoogleSiteVerificationMixin': true
```

Both options will add a meta tag required for Google to verify the ownership.

9.2 Breadcrumb

Search engines can output breadcrumbs by passing a structured data representation of a documents rootline. More information is available here: https://developers.google.com/search/docs/data-types/breadcrumb.

These are enabled by default.

9.3 Website & Searchbox

To output general information including url and name of your site you can enable the *Website* object:

```
prototype(Neos.Seo:StructuredData.Container).website.@if.enabled = true
```

Additionally if you have a local search on your site you can enable the *Sitelinks Searchbox* object. More information is available here: https://developers.google.com/search/docs/data-types/sitelinks-searchbox

You can enable this feature with the following code and by adjusting *targetNode*:

```
prototype(Neos.Seo:StructuredData.Website) {
    potentialAction.targetNode = ${<reference to your search page>}
}
```

9.4 Social profile

Search engines can output information about the social profile by passing a structured data representation of a documents rootline. Depending whether the site is run by a person or organization some parts need to be configured.

More information is available here: * https://developers.google.com/search/docs/data-types/social-profile * https://developers.google.com/search/docs/data-types/logo

You can enable this feature with the following code:

```
prototype(Neos.Seo:StructuredData.Container).socialProfile.@if.enabled = true
```

To adjust the profile configure this according to your requirements in your *Settings.yaml*:

```
Neos:
  Seo:
    socialProfile:
      type: 'set either to "Person" or "Organization"'
      logo: 'resource://Vendor.Site/Public/Images/MyLogo.png'
      profiles:
        twitter: 'your twitter name'
        facebook: 'your facebook name'
        instagram: 'your instagram name'
        linkedIn: 'your linkedin name'
        youTube: 'your YouTube channel identifier'
```

CHAPTER 10

XML sitemap

The generation of an XML sitemap to submit to search engines can be used as follows:

The change frequency and priority for each sitemap entry are used as specified in the respective fields added to the SEO tab in the inspector of *Neos.Neos:Document* nodes via the *Neos.Seo:XmlSitemapMixin*. For priority the default value is 0.5 (neutral) and the change frequency is omitted unless specified.

The necessary route to make the sitemap available is automatically included via *Settings.yaml* and will provide the sitemap via *your.domain/sitemap.xml*. See Settings on how to disable or change the route.

To include contained images of pages in the xml sitemap use the following fusion code:

```
prototype(Neos.Seo:XmlSitemap) {
    body.includeImageUrls = true
}
```

Be aware that the sitemap will output all images referenced in a page and it's content. If you reference images that should not render in the frontend you might need to adjust the sitemap according to your needs.

By default all shortcuts are ignored in the sitemap. They inherit from the prototype *Neos.Seo:NoindexMixin*. If you have other document types that should not appear in the sitemap you can also let them inherit from that prototype.

To include alternate language links of pages in the xml sitemap use the following fusion code:

```
prototype(Neos.Seo:XmlSitemap) {
    body.includeAlternateLanguageLinks = true
}
```

Be aware of possible performance issues. Rendering the sitemap with all optional features might be slow for larger installations and needs an optimized *XmlSitemapImplementation* which could use ElasticSearch for example. Alternatively you can change the caching behavior and have a cron job that recreates the sitemap for example once per day.

CHAPTER 11

Alternate Language Tag

The *Alternate Language Tag* provides information that the site is also available in other languages. Given the Neos Demo Site Package as an example the rendered tags for the homepage would be.

```
<link rel="alternate" hreflang="en_US" href="http://neos.dev/"/>
<link rel="alternate" hreflang="en_UK" href="http://neos.dev/uk"/>
<link rel="alternate" hreflang="de" href="http://neos.dev/de/"/>
```

If you only want to render a subset of the available language dimensions (e.g., if the content is not yet ready) you can configure this in the *Settings.yaml*:

```
Neos:
  Seo:
    alternateLanguageLinks:
      # Show all but exclude German
      excludedDimensionsPresets: ['de']
```

CHAPTER 12

Dynamic robots.txt

To activate the automatic *robots.txt* you have to delete the *robots.txt* inside the */Web* folder. You also have to edit the *.htaccess*:: Change the line *RewriteRule ^(_Resources/Packages/|robots.txt|favicon.ico) - [L]* to *RewriteRule ^(_Resources/Packages/|favicon.ico) - [L]*.

If you don't want to delete 'robots.txt' after every update, you should add following lines to your '.htaccess'::

Use Neos robots.txt RewriteCond %{REQUEST_URI} ^/robots.txt RewriteRule (.*) index.php [L]

If you use nginx you should disable the following entry if you have it::

location = /robots.txt { allow all; log_not_found off; access_log off;

}

If you only want to render a subset of the available language dimensions (e.g., if the content is not yet ready) you can configure this in the *Settings.yaml*:

```
Neos:
  Seo:
    robotsTxt:
      # Activate only English and German
      dimensionsPresets: ['en','de']
      # Or show all but exclude French
      excludedDimensionsPresets: ['fr']
```

You can also add your own definitions to the *robots.txt*. They can be passed by adding them to definitions array. For example to block the GoogleBot from a directory use this fusion code:

```
prototype(Neos.Seo:RobotsTxt) {
    data {
        userAgentGoogleBot = 'User-agent: Googlebot'
        disallowAll = 'Disallow: /private'
        disallowAll.@position = 'after userAgentGoogleBot'
    }
}
```

You should work with the position argument to ensure that everything is where you want it. By default a definition is preconfigured that blocks */neos* for all user agents.

CHAPTER 13

Disabling not needed features

The package provides a number of mixins to help rendering SEO metadata. By default, they are enabled in the *Configuration/NodeTypes.yaml* file, along with an inspector tab:

```
'Neos.Neos:Document':
  superTypes:
    'Neos.Seo:TitleTagMixin': true
    'Neos.Seo:SeoMetaTagsMixin': true
    'Neos.Seo:TwitterCardMixin': true
    'Neos.Seo:CanonicalLinkMixin': true
    'Neos.Seo:OpenGraphMixin': true
    'Neos.Seo:XmlSitemapMixin': true
  ui:
    inspector:
      tabs:
        seo:
          label: 'Neos.Seo:NodeTypes.Document:tabs.seo'
          position: 30
          icon: 'icon-bullseye'

'Neos.Neos:Shortcut':
  superTypes:
    'Neos.Seo:TitleTagMixin': false
    'Neos.Seo:SeoMetaTagsMixin': false
    'Neos.Seo:NoindexMixin': true
    'Neos.Seo:TwitterCardMixin': false
    'Neos.Seo:CanonicalLinkMixin': false
    'Neos.Seo:OpenGraphMixin': false
    'Neos.Seo:XmlSitemapMixin': false
```

Then to enable rendering of all SEO meta tags, the following code is used:

```
prototype(Neos.Neos:Page) {
  htmlTag.attributes.lang = Neos.Seo:LangAttribute
  head {
    titleTag >
    titleTag = Neos.Seo:TitleTag
    metaDescriptionTag = Neos.Seo:MetaDescriptionTag
    metaKeywordsTag = Neos.Seo:MetaKeywordsTag
    metaRobotsTag = Neos.Seo:MetaRobotsTag
```

(continues on next page)

(continued from previous page)

```
    canonicalLink = Neos.Seo:CanonicalLink
    alternateLanguageLinks = Neos.Seo:AlternateLanguageLinks
    twitterCard = Neos.Seo:TwitterCard
    openGraphMetaTags = Neos.Seo:OpenGraphMetaTags
    facebookMetaTags = Neos.Seo:FacebookMetaTags
    structuredData = Neos.Seo:StructuredData.Container
  }
}
```

If not all of the features are needed in a project, they can be disabled as needed. This example removes OpenGraph support.

Packages/Sites/Acme.AcmeCom/Configuration/NodeTypes.yaml:

```
'Neos.Neos:Document':
  superTypes:
    'Neos.Seo:OpenGraphMixin': false
```

Packages/Sites/Acme.AcmeCom/Resources/Private/Fusion/Root.fusion:

```
prototype(Neos.Neos:Page).head.openGraphMetaTags >
```