# Document your Nix code with Sphinx

Rémi ( minijackson)

FOSDEM 2026

# ❄️ My super Nix project ❄️

Lorem ipsum dolor sit amet, consectetur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut.

# ❄️ My super Nix project ❄️

What about these?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut.

# ❄️ My super Nix project ❄️

What about these?

**Option 1**:

Give up

*“Read the source, Luke”*

# ❄️ My super Nix project ❄️

What about these?

**Option 1:** Give up

*“Read the source, Luke”*

**Option 2:** Do it yourself

# ❄️ My super Nix project ❄️

What about these?

| **Option 1:** | **Option 2:** | **Option 3:** |
|---|---|---|
| Give up | Do it yourself | ??? |
| *“Read the source, Luke”* | | |

# Introducing `sphinxcontrib-nixdomain`

Document *NixOS* options, *Nix* packages, *Nix* libraries, using the *Sphinx* documentation ecosystem!

---

https://sphinxcontrib-nixdomain.readthedocs.io/

## Sphinx?

Sphinx is the most-used documentation generator in the Python world.

- Provides directives and roles, which are like functions callable from Markdown.
- Good at generating reference API documentation and cross-references to it.
- Made to be extensible.
- Supports languages other than Python.

## Set up `sphinxcontrib-nixdomain`

0. Make a Sphinx documentation project,
1. Import the `sphinxcontrib-nixdomain` Nix repo,
2. Set up the Sphinx extension
3. Pass your Nix objects through an environment variable in your documentation's Nix package,
4. Use the new directives and roles in your documentation.

## Passing Nix objects

```
env.NIXDOMAIN_OBJECTS = sphinxcontrib-nixdomain.lib.documentObjects {
  sources = {
    self = self.outPath;
    nixpkgs = nixpkgs.outPath;
  };
  options.options = myNixosOptions;
  packages.packages = self.packages.x86_64-linux;
  library = {
    name = "exampleLib";
    library = self.lib;
  };
};
```

## Passing Nix objects

```
env.NIXDOMAIN_OBJECTS = sphinxcontrib-nixdomain.lib.documentObjects {
  sources = {
    self = self.outPath;
    nixpkgs = nixpkgs.outPath;
  };
  options.options = myNixosOptions;
  packages.packages = self.packages.x86_64-linux;
  library = {
    name = "exampleLib";
    library = self.lib;
  };
};
```

## Passing Nix objects

```
env.NIXDOMAIN_OBJECTS = sphinxcontrib-nixdomain.lib.documentObjects {
  sources = {
    self = self.outPath;
    nixpkgs = nixpkgs.outPath;
  };
  options.options = myNixosOptions;
  packages.packages = self.packages.x86_64-linux;
  library = {
    name = "exampleLib";
    library = self.lib;
  };
};
```

## Passing Nix objects

```
env.NIXDOMAIN_OBJECTS = sphinxcontrib-nixdomain.lib.documentObjects {
  sources = {
    self = self.outPath;
    nixpkgs = nixpkgs.outPath;
  };
  options.options = myNixosOptions;
  packages.packages = self.packages.x86_64-linux;
  library = {
    name = "exampleLib";
    library = self.lib;
  };
};
```

## Passing Nix objects

```
env.NIXDOMAIN_OBJECTS = sphinxcontrib-nixdomain.lib.documentObjects {
  sources = {
    self = self.outPath;
    nixpkgs = nixpkgs.outPath;
  };
  options.options = myNixosOptions;
  packages.packages = self.packages.x86_64-linux;
  library = {
    name = "exampleLib";
    library = self.lib;
  };
};
```

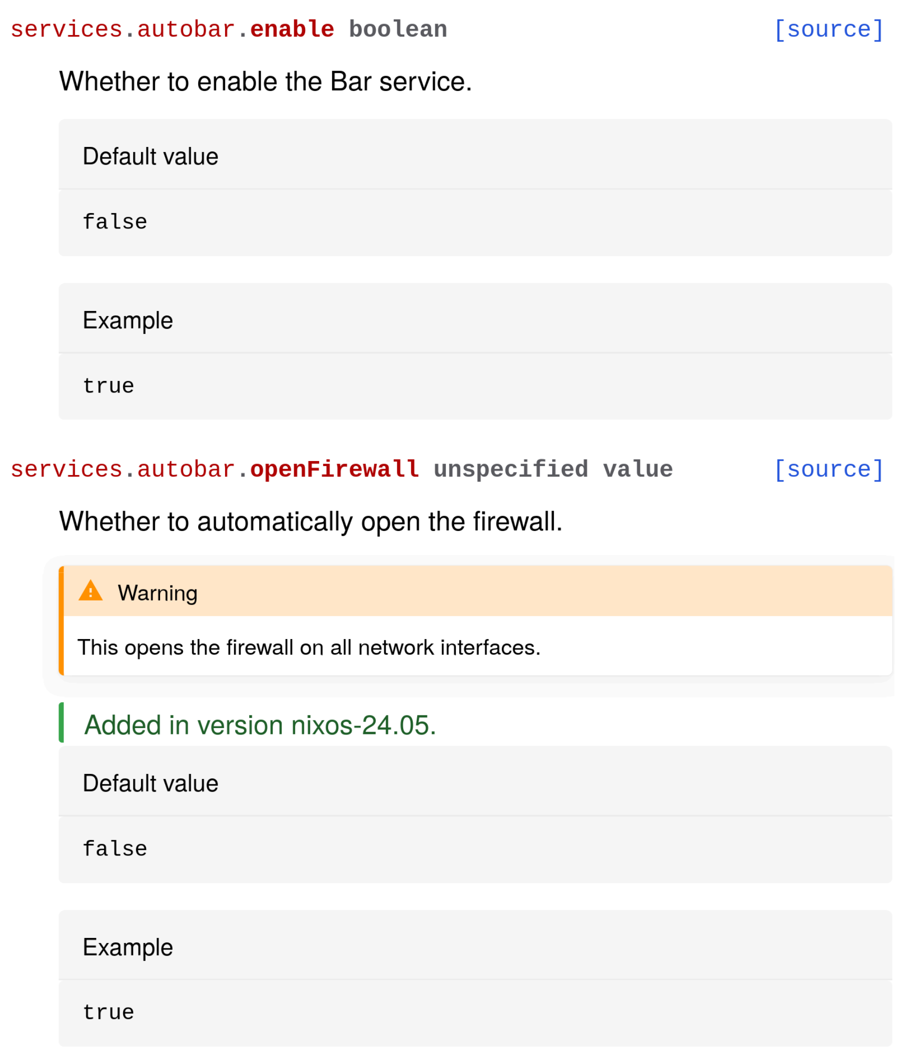
## Passing Nix objects

```
env.NIXDOMAIN_OBJECTS = sphinxcontrib-nixdomain.lib.documentObjects {
  sources = {
    self = self.outPath;
    nixpkgs = nixpkgs.outPath;
  };
  options.options = myNixosOptions;
  packages.packages = self.packages.x86_64-linux;
  library = {
    name = "exampleLib";
    library = self.lib;
  };
};
```

# Markdown Sphinx directives

~~~
```{nix:automodule} services.autobar
```
~~~

$\longrightarrow$



services.autobar.**enable** boolean [source]

Whether to enable the Bar service.

| Default value |
|---|
| `false` |

| Example |
|---|
| `true` |

services.autobar.**openFirewall** unspecified value [source]

Whether to automatically open the firewall.

⚠ Warning

This opens the firewall on all network interfaces.

Added in version nixos-24.05.

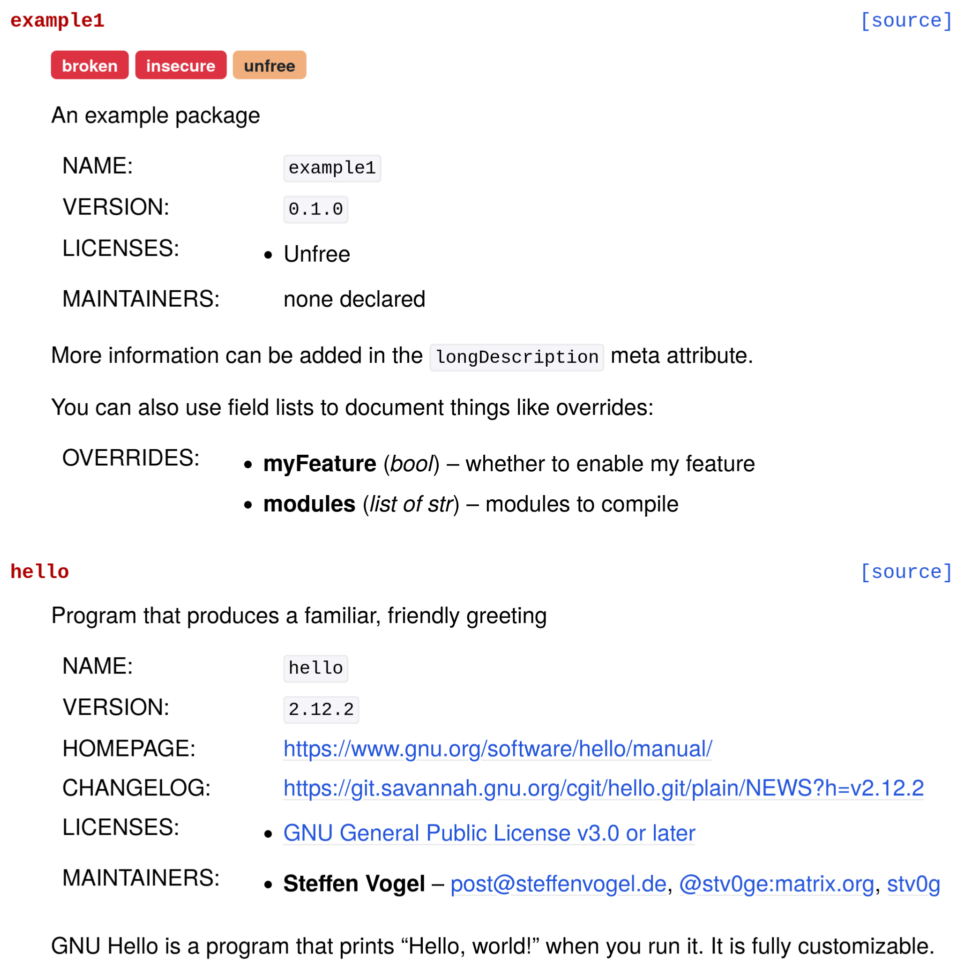
| Default value |
|---|
| `false` |

| Example |
|---|
| `true` |

# Markdown Sphinx directives

~~~
```{nix:autopackages}
```
~~~

$\longrightarrow$



**example1** [source]

broken insecure unfree

An example package

| | |
|---|---|
| NAME: | `example1` |
| VERSION: | `0.1.0` |
| LICENSES: | • Unfree |
| MAINTAINERS: | none declared |

More information can be added in the `longDescription` meta attribute.

You can also use field lists to document things like overrides:

| | |
|---|---|
| OVERRIDES: | • **myFeature** (*bool*) – whether to enable my feature<br>• **modules** (*list of str*) – modules to compile |

**hello** [source]

Program that produces a familiar, friendly greeting

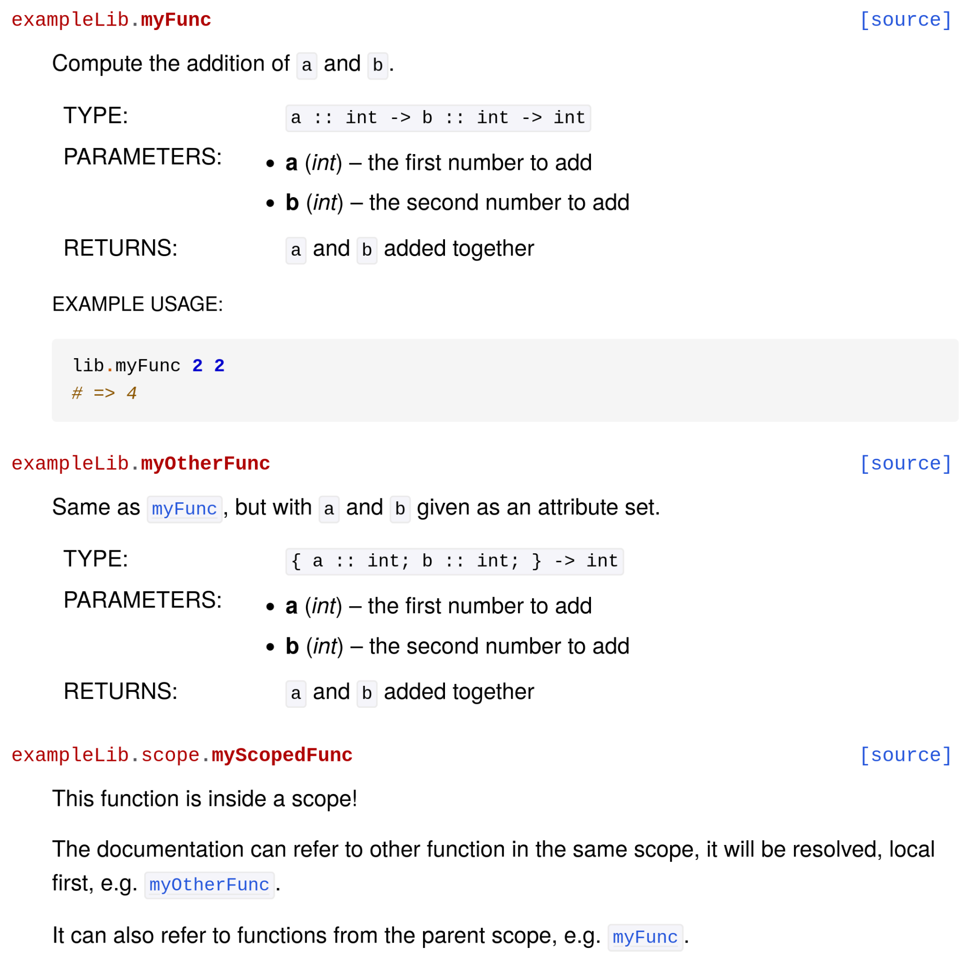
| | |
|---|---|
| NAME: | `hello` |
| VERSION: | `2.12.2` |
| HOMEPAGE: | https://www.gnu.org/software/hello/manual/ |
| CHANGELOG: | https://git.savannah.gnu.org/cgit/hello.git/plain/NEWS?h=v2.12.2 |
| LICENSES: | • GNU General Public License v3.0 or later |
| MAINTAINERS: | • **Steffen Vogel** – post@steffenvogel.de, @stv0ge:matrix.org, stv0g |

GNU Hello is a program that prints "Hello, world!" when you run it. It is fully customizable.

# Markdown Sphinx directives

~~~
```{nix:autolibrary}
```
~~~

$\longrightarrow$



exampleLib.**myFunc** [source]

Compute the addition of `a` and `b`.

TYPE: `a :: int -> b :: int -> int`

PARAMETERS:
- **a** (*int*) – the first number to add
- **b** (*int*) – the second number to add

RETURNS: `a` and `b` added together

EXAMPLE USAGE:

```
lib.myFunc 2 2
# => 4
```

exampleLib.**myOtherFunc** [source]

Same as `myFunc`, but with `a` and `b` given as an attribute set.

TYPE: `{ a :: int; b :: int; } -> int`

PARAMETERS:
- **a** (*int*) – the first number to add
- **b** (*int*) – the second number to add

RETURNS: `a` and `b` added together

exampleLib.scope.**myScopedFunc** [source]

This function is inside a scope!

The documentation can refer to other function in the same scope, it will be resolved, local first, e.g. `myOtherFunc`.

It can also refer to functions from the parent scope, e.g. `myFunc`.

## The fun part

Sphinx understands that these are “objects” from Nix.

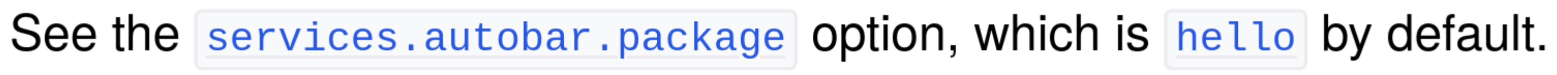
## Cross-referencing (Sphinx roles)

```
See the {nix:option}`services.autobar.package` option,
which is {nix:pkg}`hello` by default.
```

## Cross-referencing (Sphinx roles)

```
See the {nix:option}`services.autobar.package` option,
which is {nix:pkg}`hello` by default.
```

↓



See the `services.autobar.package` option, which is `hello` by default.

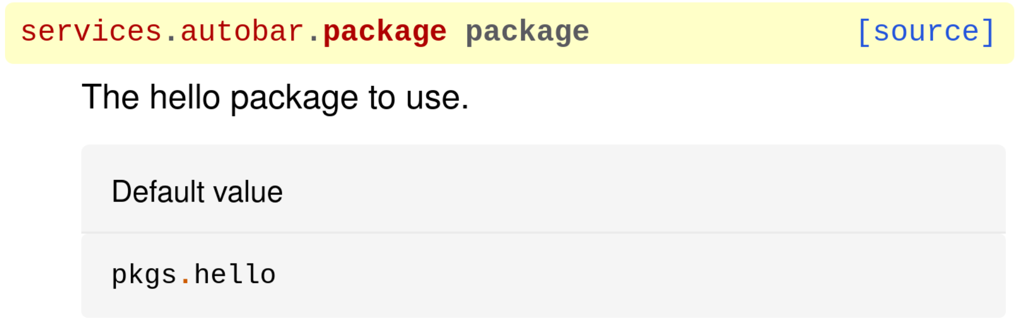
## Cross-referencing (Sphinx roles)

```
See the {nix:option}`services.autobar.package` option,
which is {nix:pkg}`hello` by default.
```

↓

See the `services.autobar.package` option, which is `hello` by default.

↙ ↘

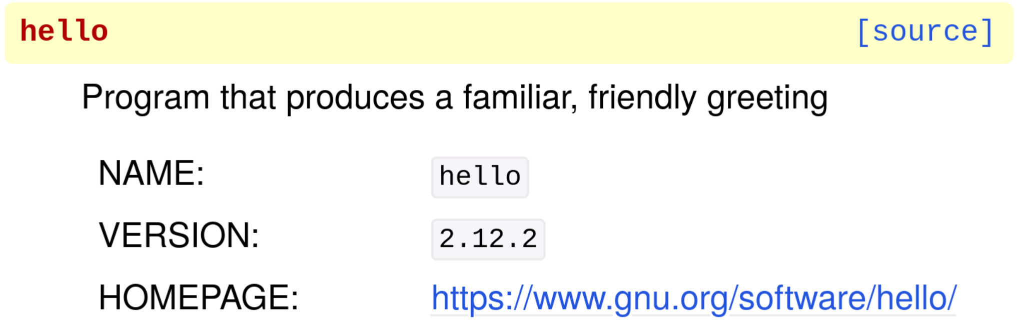


**services.autobar.package** package [source]

The hello package to use.

Default value

`pkgs.hello`



**hello** [source]

Program that produces a familiar, friendly greeting

| | |
|---|---|
| NAME: | `hello` |
| VERSION: | `2.12.2` |
| HOMEPAGE: | https://www.gnu.org/software/hello/ |

## External cross-referencing (Intersphinx)

*In your Sphinx* `conf.py`*:*

```python
intersphinx_mapping = {
  "nixdomain": ("https://sphinxcontrib-nixdomain.readthedocs.io/en/latest/", None),
}
```

## External cross-referencing (Intersphinx)

*In your Sphinx `conf.py`:*

```
intersphinx_mapping = {
  "nixdomain": ("https://sphinxcontrib-nixdomain.readthedocs.io/en/latest/", None),
}
```

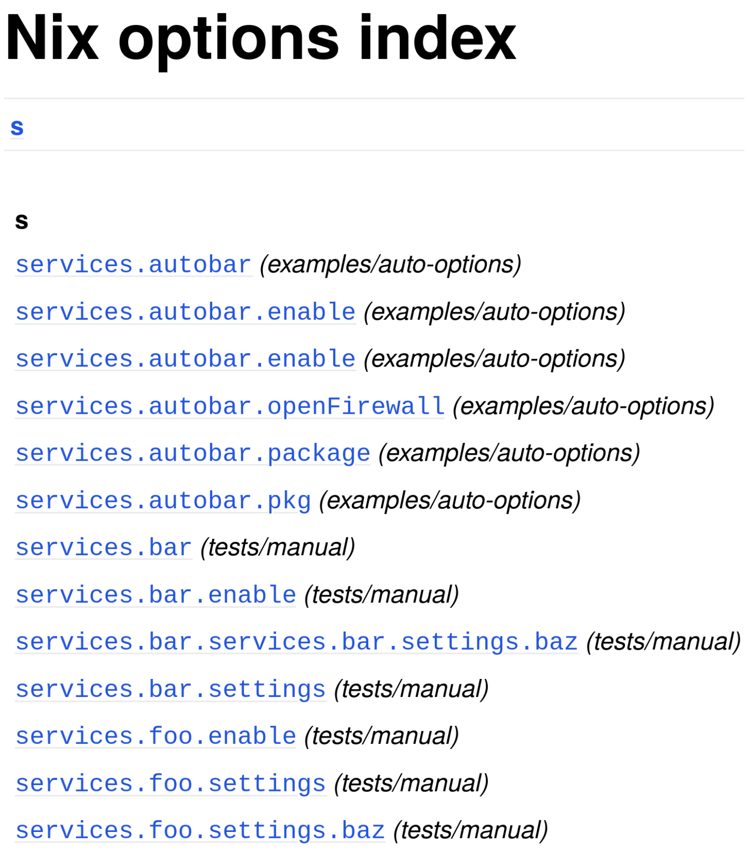
---

*In your documentation:*

```
See the {nix:func}`nixdomainLib.documentObjects` function.
---or---
See the {external+nixdomain:nix:func}`nixdomainLib.documentObjects` function.
```

*will resolve to the external `sphinxcontrib-nixdomain` documentation website.*

## Nix-specific index generation



### Nix options index

s

s

services.autobar *(examples/auto-options)*

services.autobar.enable *(examples/auto-options)*

services.autobar.enable *(examples/auto-options)*

services.autobar.openFirewall *(examples/auto-options)*

services.autobar.package *(examples/auto-options)*

services.autobar.pkg *(examples/auto-options)*

services.bar *(tests/manual)*

services.bar.enable *(tests/manual)*

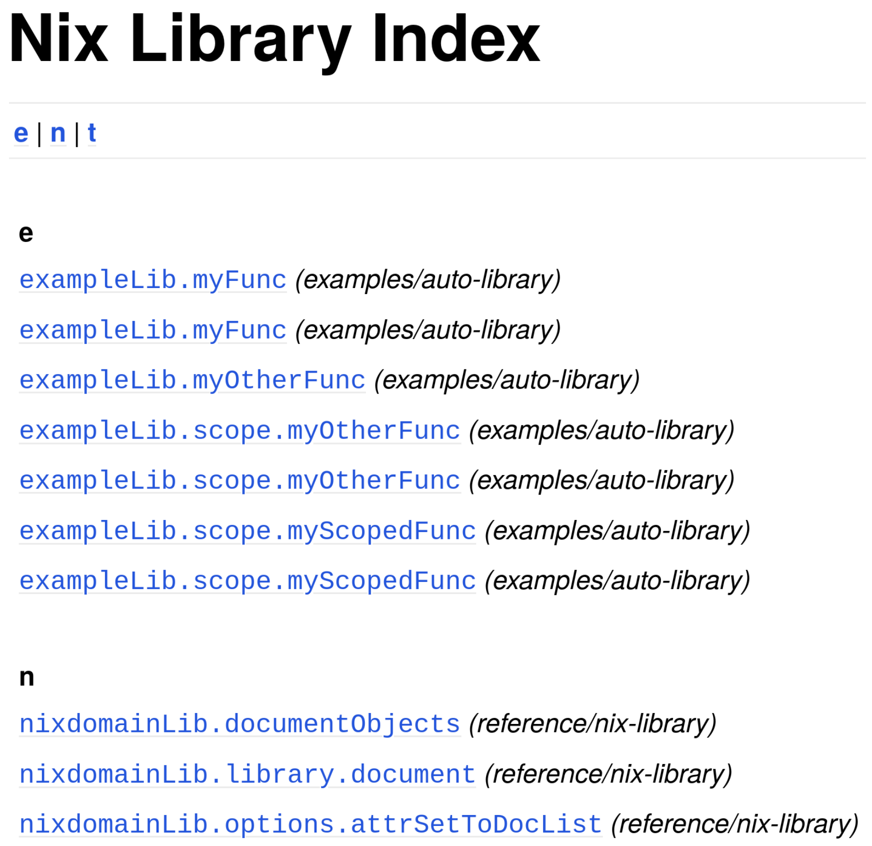
services.bar.services.bar.settings.baz *(tests/manual)*

services.bar.settings *(tests/manual)*

services.foo.enable *(tests/manual)*

services.foo.settings *(tests/manual)*

services.foo.settings.baz *(tests/manual)*



### Nix Library Index

e | n | t

e

exampleLib.myFunc *(examples/auto-library)*

exampleLib.myFunc *(examples/auto-library)*

exampleLib.myOtherFunc *(examples/auto-library)*

exampleLib.scope.myOtherFunc *(examples/auto-library)*

exampleLib.scope.myOtherFunc *(examples/auto-library)*

exampleLib.scope.myScopedFunc *(examples/auto-library)*

exampleLib.scope.myScopedFunc *(examples/auto-library)*

n

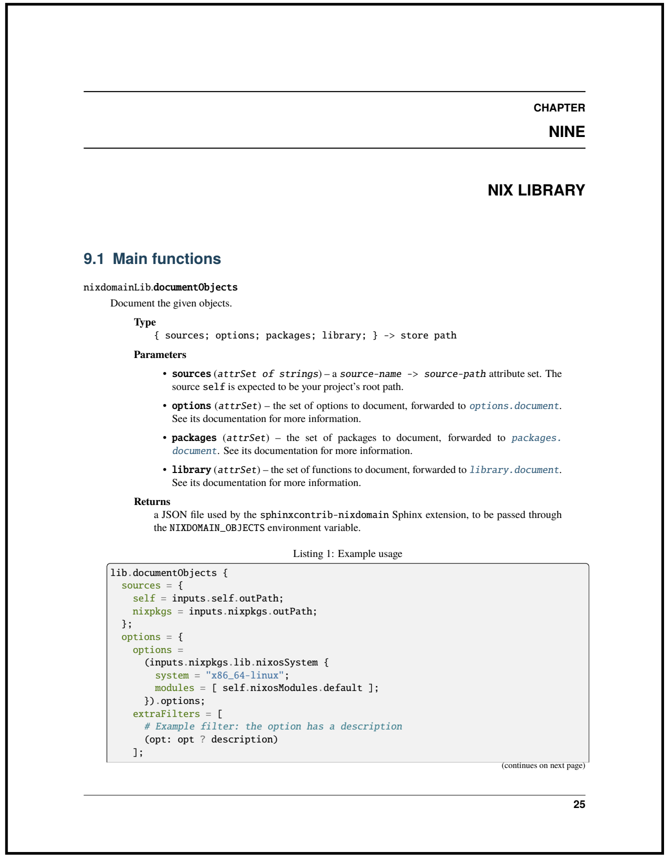
nixdomainLib.documentObjects *(reference/nix-library)*

nixdomainLib.library.document *(reference/nix-library)*

nixdomainLib.options.attrSetToDocList *(reference/nix-library)*

# General index

# Other formats: PDFs, man pages, etc.



CHAPTER

NINE

NIX LIBRARY

### 9.1 Main functions

`nixdomainLib.`**`documentObjects`**

Document the given objects.

**Type**

`{ sources; options; packages; library; } -> store path`

**Parameters**

- **`sources`** (*`attrSet of strings`*) – a *`source-name -> source-path`* attribute set. The source `self` is expected to be your project's root path.
- **`options`** (*`attrSet`*) – the set of options to document, forwarded to *`options.document`*. See its documentation for more information.
- **`packages`** (*`attrSet`*) – the set of packages to document, forwarded to *`packages.document`*. See its documentation for more information.
- **`library`** (*`attrSet`*) – the set of functions to document, forwarded to *`library.document`*. See its documentation for more information.

**Returns**

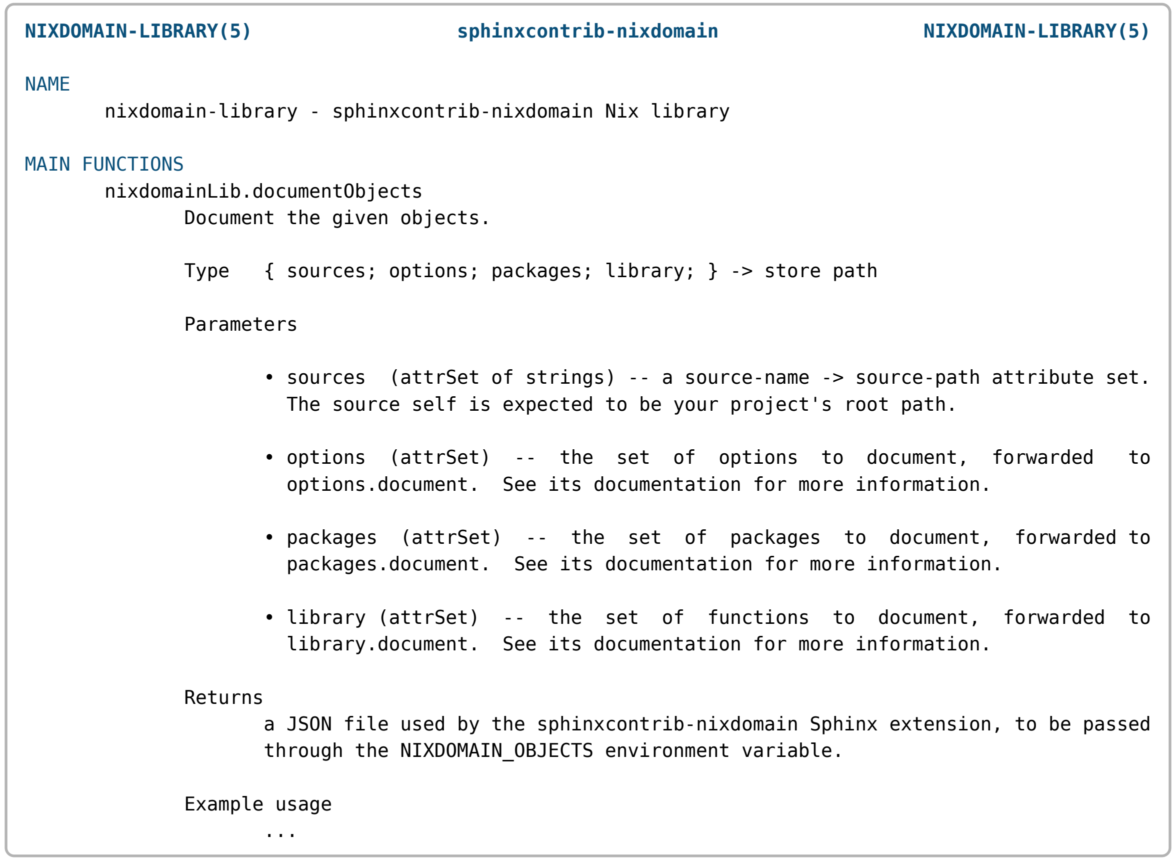
a JSON file used by the `sphinxcontrib-nixdomain` Sphinx extension, to be passed through the `NIXDOMAIN_OBJECTS` environment variable.

Listing 1: Example usage

```nix
lib.documentObjects {
  sources = {
    self = inputs.self.outPath;
    nixpkgs = inputs.nixpkgs.outPath;
  };
  options = {
    options =
      (inputs.nixpkgs.lib.nixosSystem {
        system = "x86_64-linux";
        modules = [ self.nixosModules.default ];
      }).options;
    extraFilters = [
      # Example filter: the option has a description
      (opt: opt ? description)
    ];
```

(continues on next page)

25

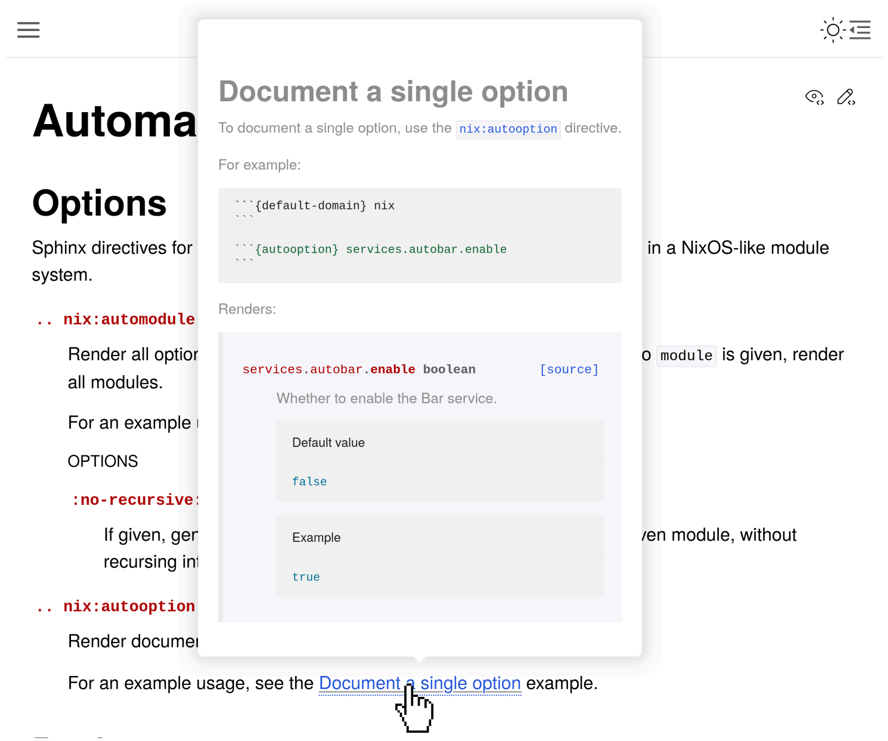


```
NIXDOMAIN-LIBRARY(5)          sphinxcontrib-nixdomain          NIXDOMAIN-LIBRARY(5)

NAME
       nixdomain-library - sphinxcontrib-nixdomain Nix library

MAIN FUNCTIONS
       nixdomainLib.documentObjects
              Document the given objects.

              Type   { sources; options; packages; library; } -> store path

              Parameters

                     • sources  (attrSet of strings) -- a source-name -> source-path attribute set.
                       The source self is expected to be your project's root path.

                     • options  (attrSet)  --  the  set  of  options  to  document,  forwarded   to
                       options.document.  See its documentation for more information.

                     • packages  (attrSet)  --  the  set  of  packages  to  document,  forwarded to
                       packages.document.  See its documentation for more information.

                     • library (attrSet)  --  the  set  of  functions  to  document,  forwarded  to
                       library.document.  See its documentation for more information.

              Returns
                     a JSON file used by the sphinxcontrib-nixdomain Sphinx extension, to be passed
                     through the NIXDOMAIN_OBJECTS environment variable.

              Example usage
                     ...
```

## *Read the Docs*

A platform for hosting your Sphinx documentation.

# Link previews



Document a single option

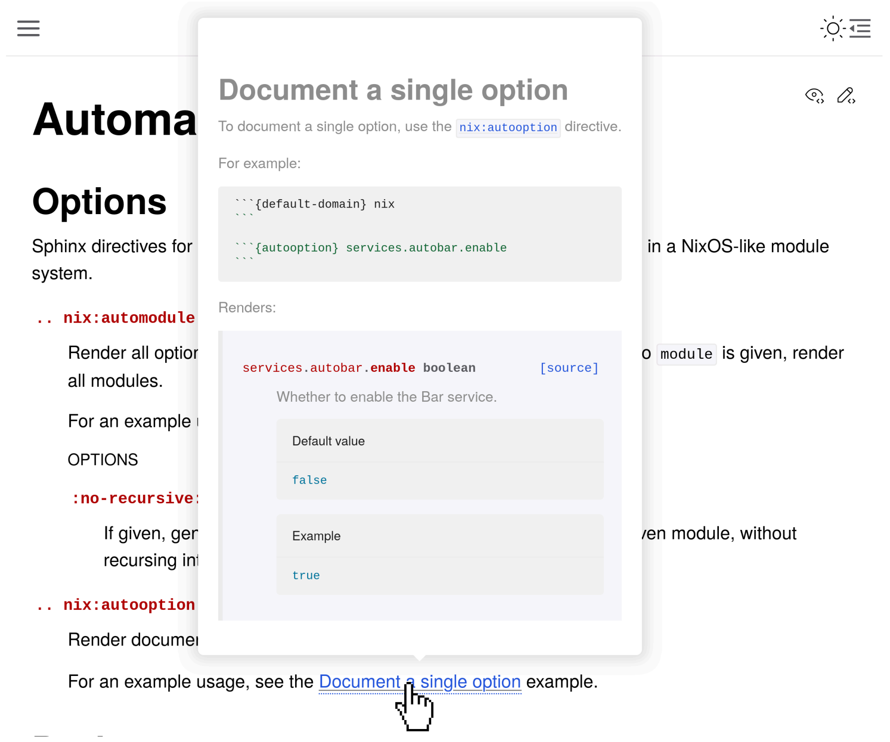
To document a single option, use the `nix:autooption` directive.

For example:

~~~
```{default-domain} nix
~~~

```{autooption} services.autobar.enable
```
```

Renders:

`services.autobar.enable boolean` [source]

Whether to enable the Bar service.

Default value

`false`

Example

`true`

Automa

Options

Sphinx directives for … in a NixOS-like module system.

.. nix:automodule

Render all optior … o `module` is given, render all modules.

For an example

OPTIONS

:no-recursive:

If given, ger … ven module, without recursing int

.. nix:autooption

Render documer

For an example usage, see the Document a single option example.

*Read the Docs* link preview…
```

# Link previews



Automa

Options

Sphinx directives for … in a NixOS-like module system.

.. nix:automodule

Render all optio… o `module` is given, render all modules.

For an example …

OPTIONS

:no-recursive:

If given, ge… …en module, without recursing in…

.. nix:autooption

Render documen…

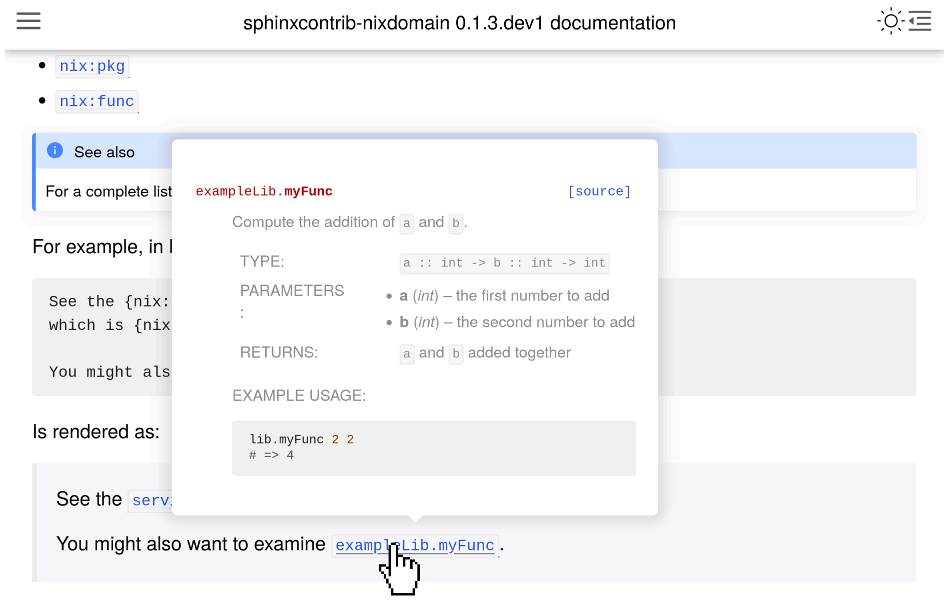
For an example usage, see the Document a single option example.

Document a single option

To document a single option, use the `nix:autooption` directive.

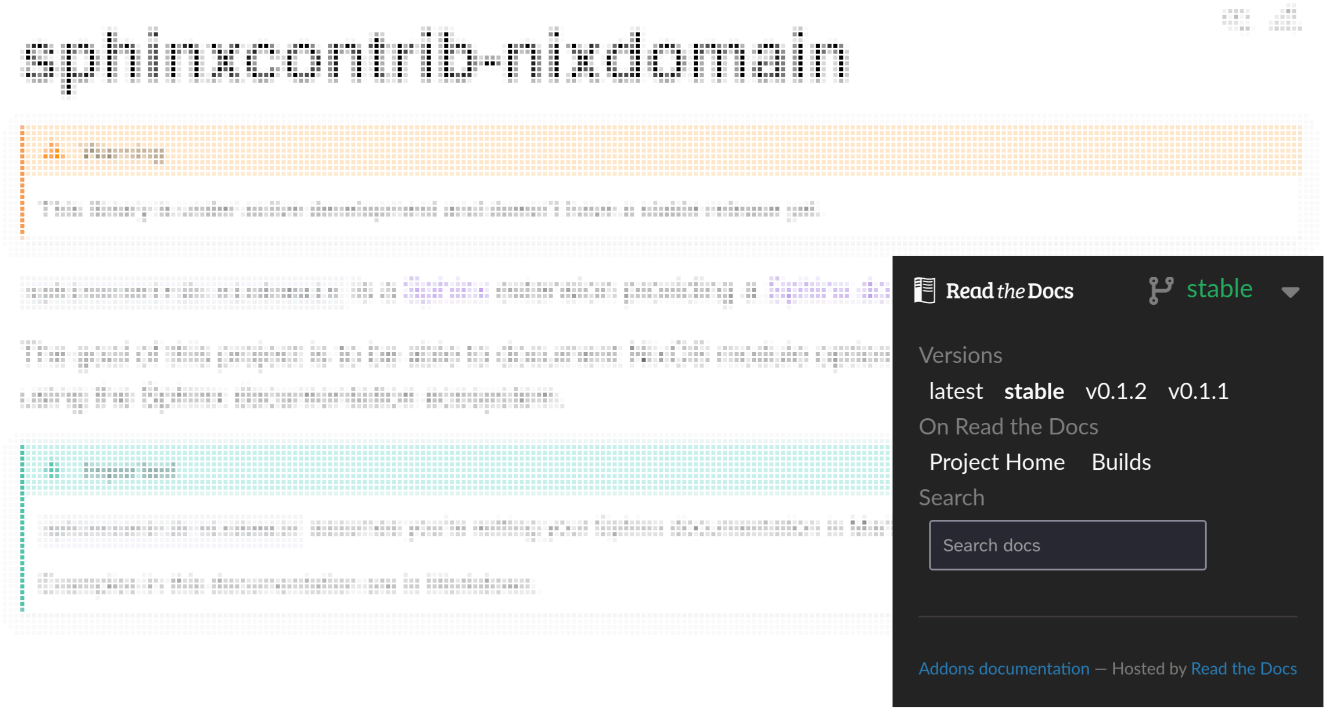
For example:

~~~
```{default-domain} nix
~~~

```{autooption} services.autobar.enable
```
```

Renders:

services.autobar.enable boolean [source]

Whether to enable the Bar service.

Default value

`false`

Example

`true`

*Read the Docs* link preview…



sphinxcontrib-nixdomain 0.1.3.dev1 documentation

- `nix:pkg`
- `nix:func`

See also

For a complete list …

For example, in …

```
See the {nix:
which is {nix

You might als
```

Is rendered as:

See the `serv`…

You might also want to examine `exampleLib.myFunc`.

exampleLib.myFunc [source]

Compute the addition of `a` and `b`.

TYPE: `a :: int -> b :: int -> int`

PARAMETERS:
- **a** (*int*) – the first number to add
- **b** (*int*) – the second number to add

RETURNS: `a` and `b` added together

EXAMPLE USAGE:

```
lib.myFunc 2 2
# => 4
```

…working with my plugin :)
```

## Multi-version support



sphinxcontrib-nixdomain

Read the Docs stable

Versions

latest **stable** v0.1.2 v0.1.1

On Read the Docs

Project Home Builds

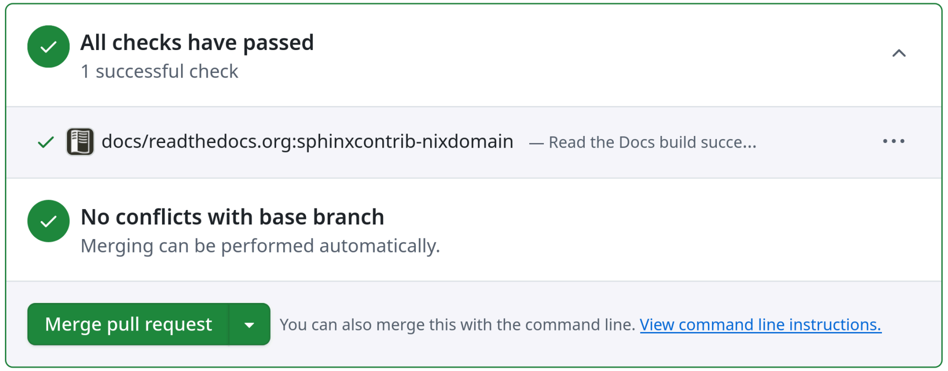
Search

Search docs

Addons documentation — Hosted by Read the Docs

*Read the Docs'* flyout menu

## PR building

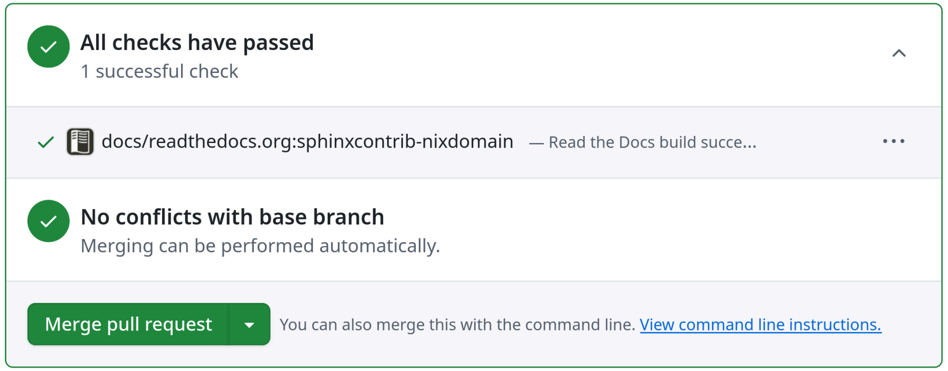

All checks have passed
1 successful check

✓ docs/readthedocs.org:sphinxcontrib-nixdomain — Read the Docs build succe...

No conflicts with base branch
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. View command line instructions.

## PR building



All checks have passed

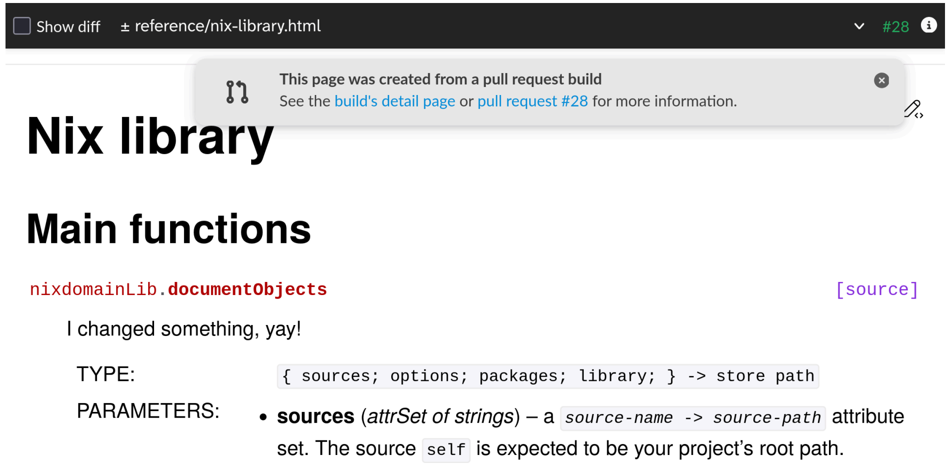
1 successful check

✓ docs/readthedocs.org:sphinxcontrib-nixdomain — Read the Docs build succe...

No conflicts with base branch

Merging can be performed automatically.

You can also merge this with the command line. View command line instructions.



This page was created from a pull request build

See the build's detail page or pull request #28 for more information.

Nix library

Main functions

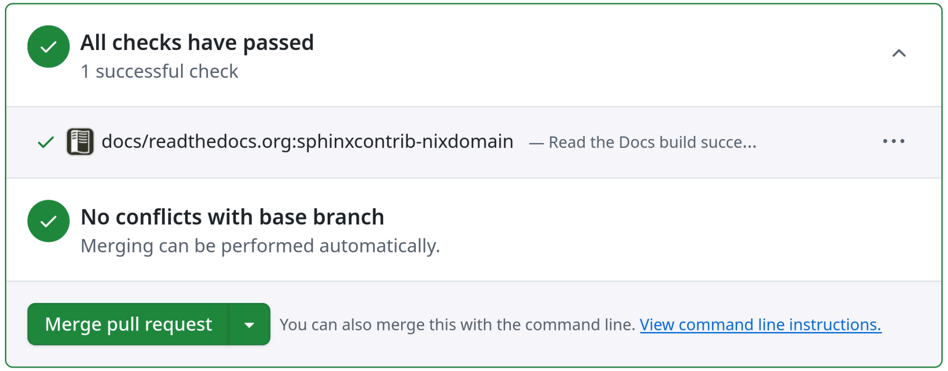
nixdomainLib.documentObjects [source]

I changed something, yay!

TYPE: `{ sources; options; packages; library; } -> store path`

PARAMETERS:
- **sources** (*attrSet of strings*) – a `source-name -> source-path` attribute set. The source `self` is expected to be your project's root path.

# PR building



All checks have passed

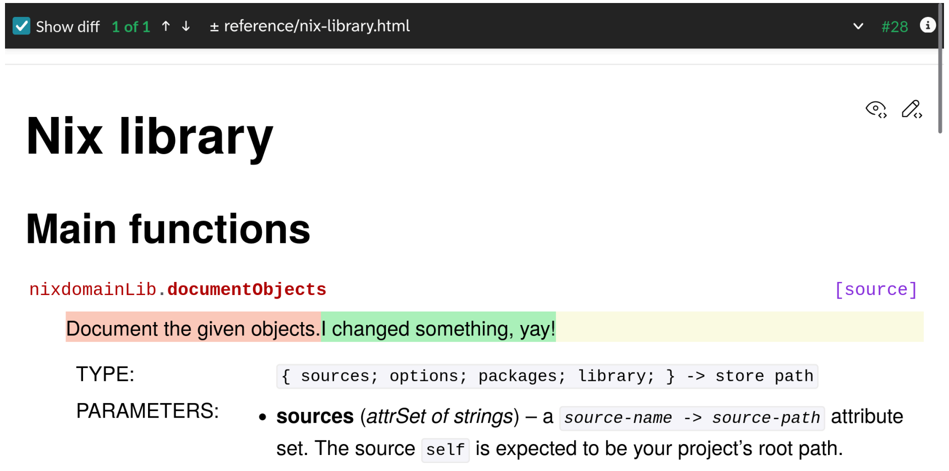
1 successful check

docs/readthedocs.org:sphinxcontrib-nixdomain — Read the Docs build succe...

No conflicts with base branch

Merging can be performed automatically.

You can also merge this with the command line. View command line instructions.



Nix library

Main functions

`nixdomainLib.documentObjects` [source]

~~Document the given objects.~~I changed something, yay!

TYPE: `{ sources; options; packages; library; } -> store path`

PARAMETERS:
- **sources** (*attrSet of strings*) – a `source-name -> source-path` attribute set. The source `self` is expected to be your project's root path.

## Future work

- Upstreaming to Nixpkgs.
- Cross-reference options, packages, and functions defined in Nixpkgs.
- Support the `meta.doc` NixOS option.
- Create pages automatically.
- Warn if some Nix objects aren’t referenced in the documentation.
- Weird Sphinx MyST bug when using headers in object documentation.

## Resources

**sphinxcontrib-nixdomain**:

- **documentation**: https://sphinxcontrib-nixdomain.readthedocs.io/
- **source**: https://github.com/minijackson/sphinxcontrib-nixdomain/
- **project using it**: https://epics-extensions.github.io/EPNix/

---

**Diátaxis**: https://diataxis.fr/

**Write the Docs**: https://www.writethedocs.org/guide/