

---

# **sphinx-packaging**

*Release 0.2.0*

**A collection of Sphinx utilities related to Python  
packaging.**

**Dominic Davis-Foster**

**Sep 13, 2022**



# Contents

<b>1</b>	<b>Installation</b>	<b>1</b>
1.1	from PyPI . . . . .	1
1.2	from GitHub . . . . .	1
<b>2</b>	<b>Usage</b>	<b>3</b>
2.1	Directives . . . . .	3
2.2	Roles . . . . .	4
2.3	Configuration . . . . .	7
<b>3</b>	<b>sphinx_packaging</b>	<b>9</b>
3.1	setup . . . . .	9
<b>4</b>	<b>sphinx_packaging.peps</b>	<b>11</b>
4.1	CoreMetadata . . . . .	11
4.2	PEP . . . . .	12
4.3	PEP621Section . . . . .	12
4.4	setup . . . . .	13
<b>5</b>	<b>sphinx_packaging.tconf</b>	<b>15</b>
5.1	TConfXRefRole . . . . .	15
5.2	TOMLConf . . . . .	16
5.3	resolve_xref . . . . .	17
5.4	setup . . . . .	17
<b>6</b>	<b>sphinx_packaging.toml</b>	<b>19</b>
6.1	TOML . . . . .	19
6.2	setup . . . . .	20
<b>7</b>	<b>Downloading source code</b>	<b>21</b>
7.1	Building from source . . . . .	22
<b>8</b>	<b>License</b>	<b>23</b>
	<b>Python Module Index</b>	<b>25</b>
	<b>Index</b>	<b>27</b>



## Installation

### 1.1 from PyPI

```
$ python3 -m pip install sphinx-packaging --user
```

### 1.2 from GitHub

```
$ python3 -m pip install git+https://github.com/sphinx-toolbox/sphinx-packaging@master --user
```



## Usage

Enable `sphinx_packaging` by adding the following to the `extensions` variable in your `conf.py`:

```
extensions = [  
    ...  
    'sphinx_packaging',  
]
```

For more information see

<https://www.sphinx-doc.org/en/master/usage/extensions#third-party-extensions>.

## 2.1 Directives

**.. tconf::** name

Used to document a TOML configuration field, for example in **PEP 621** metadata.

**:type:** (string)

Indicates the field's type.

**:required:** (flag)

Indicates the whether a value is required for the field.

**:default:** (string)

Indicates the default value for the field.

**:noindex:** (flag)

Disables the index entry and cross-referencing for this field.

**Examples:**

```
.. tconf:: project.name  
    :type: :toml:`String`  
    :required: True  
  
    The name of the project.  
  
.. tconf:: project.version  
    :type: :toml:`String`  
    :required: True
```

(continues on next page)

(continued from previous page)

```
The version of the project as supported by :pep:`440`.

.. tconf:: description
   :type: :toml:`String`
   :default: ``'This is the description'``
   :required: False

A short summary description of the project.
```

**project.name**  
Type: `String`  
Required: True  
The name of the project.

**project.version**  
Type: `String`  
Required: True  
The version of the project as supported by **PEP 440**.

**description**  
Type: `String`  
Required: False  
Default: `'This is the description'`  
A short summary description of the project.

## 2.2 Roles

*sphinx\_packaging* provides the following roles:

### **:pep:**

Creates a cross-reference to a **Python Enhancement Proposal**.

The text “PEP NNN” is inserted into the document, and with supported builders is a hyperlink to the online copy of the specified PEP. This role also generates an appropriate index entry.

You can link to a specific section by writing `:pep:`number#anchor``.

A custom title can also be added to the link by writing `:pep:`title <number#anchor>``. Unlike the version of this directive which ships with Sphinx, the link is not shown in bold when there is a custom title.

### **Examples:**

```
:pep:`621`

:pep:`503#normalized-names`

.. seealso:: The :pep:`specification <427>` for wheels.
```

**PEP 621**

**PEP 503#normalized-names**



**See also:**

The [specification](#) for wheels.

**:pep621:**

Creates a cross-reference to a section in [PEP 621](#), typically the name of a field in `pyproject.toml`.

The title of the directive (either implicit, `:pep621:`title``, or explicit `:pep621:`title` <target>``) is inserted into the document. With supported builders is a hyperlink to the specified heading in the online copy of the PEP. This role also generates an appropriate index entry.

**Examples:**

```
The :pep621:`name` field must be provided and cannot be :pep621:`dynamic`.

:pep621:`Version <version>` may be required by some backend,
but can be determined dynamically by others.

:pep621:`authors` and :pep621:`maintainers` both point to the same section.
```

The `name` field must be provided and cannot be `dynamic`.

`Version` may be required by some backend, but can be determined dynamically by others.

`authors` and `maintainers` both point to the same section.

**:core-meta:**

Creates a cross-reference to a field in the Python [core metadata](#).

The title of the directive (either implicit, `:core-meta:`title``, or explicit `:core-meta:`title` <target>``) is inserted into the document. With supported builders is a hyperlink to the specified field in the specification on [packaging.python.org](#). This role also generates an appropriate index entry.

**Examples:**

```
:core-meta:`Supported-Platform (Multiple Use) <Supported-Platform>` specifies the
↪ OS and CPU
for which the binary distribution was compiled.

The project's :core-meta:`description` may have multiple lines.

:pep621:`requires-python` in ``pyproject.toml`` maps to :core-meta:`Requires-Python`
```

`Supported-Platform (Multiple Use)` specifies the OS and CPU for which the binary distribution was compiled.

The project's `description` may have multiple lines.

`requires-python` in `pyproject.toml` maps to `Requires-Python`

**:toml:**

Creates a cross-reference to a section in the [TOML specification](#).

The title of the directive (either implicit, `:toml:`title``, or explicit `:toml:`title` <target>``) is inserted into the document. With supported builders is a hyperlink to section in the web version of the specification. This role also generates an appropriate index entry.

**Examples:**

```
TOML's :toml:string type accepts either single or double quotes.
```

```
:toml:Inline Tables <Inline Table> must be on a single line.
```

There are four date/time types in TOML:

```
* :toml:Offset Date-Time
* :toml:Local Date-Time
* :toml:Local Date
* :toml:!Local Time
```

TOML's `string` type accepts either single or double quotes.

`Inline Tables` must be on a single line.

There are four date/time types in TOML:

- `Offset Date-Time`
- `Local Date-Time`
- `Local Date`
- `Local Time`

The last xref will not appear in the index because the target is prefixed with a `!`. This also works when there is an explicit title:

```
The following xrefs are not indexed: :toml:!Float, :toml:array <!Array>.
```

The following xrefs are not indexed: `Float`, `array`.

### **:tconf:**

Role which provides a cross-reference to a `tconf` directive.

#### **Examples:**

```
:tconf:project.name and :tconf:~project.version are required.
Some backends may be able to determine a value for :tconf:version dynamically.
```

```
:tconf:description will be displayed this towards the top of the project page on PyPI.
```

```
Links can also be written with a shorter name: :tconf:~.name.
```

`project.name` and `version` are required. Some backends may be able to determine a value for `version` dynamically.

`description` will be displayed this towards the top of the project page on PyPI.

Links can also be written with a shorter name: `name`.

## 2.3 Configuration

**toml\_spec\_version**

Type: `string`

**Required:** False

**Default:** 1.0.0

The version of the [TOML specification](#) to link to.

For example, this documentation links to `v0.5.0` with the following setting:

```
# conf.py
toml_spec_version = "0.5.0"
```

**tconf\_show\_full\_name**

Type: `bool`

**Required:** False

**Default:** True

Whether to show the full name for field.

For example, with `tconf_show_full_name = True`:

**project.description**

Type: `String`

And with `tconf_show_full_name = False`:

**description**

Type: `String`



## **sphinx\_packaging**

A collection of Sphinx utilities related to Python packaging.

### **Functions:**

---

<code>setup(app)</code>	Setup <i>sphinx_packaging</i> .
-------------------------	---------------------------------

---

#### **setup**(*app*)

Setup *sphinx\_packaging*.

##### **Parameters**

**app** (*Sphinx*) – The Sphinx application.

##### **Return type**

*Dict[str, Any]*



## sphinx\_packaging.peps

Sphinx extension which modifies the `pep` role to use normal (i.e. not bold) text for custom titles.

Also adds the `pep621` role for referencing sections within **PEP 621**, and the `core-meta` role for referencing sections in Python's core metadata`.

### Classes:

<code>CoreMetadata()</code>	Sphinx role for referencing a <code>core metadata</code> field.
<code>PEP()</code>	Sphinx role for referencing a PEP or a section thereof.
<code>PEP621Section()</code>	Sphinx role for referencing a section within <b>PEP 621</b> .

### Functions:

<code>setup(app)</code>	Setup <code>sphinx_packaging.peps</code> .
-------------------------	--

### class CoreMetadata

Bases: `ReferenceRole`

Sphinx role for referencing a `core metadata` field.

#### Methods:

<code>build_uri()</code>	Construct the target URI for the reference node.
<code>run()</code>	Process the role.

#### `build_uri()`

Construct the target URI for the reference node.

#### Return type

`str`

#### `has_explicit_title`

Type: `bool`

#### `run()`

Process the role.

#### Return type

`Tuple[List[Node], List[system_message]]`

**target**

Type: `str`

**title**

Type: `str`

**class PEP**

Bases: `ReferenceRole`

Sphinx role for referencing a PEP or a section thereof.

**Methods:**

<code>build_uri()</code>	Constrict the target URI for the reference node.
<code>run()</code>	Process the role.

**build\_uri()**

Constrict the target URI for the reference node.

**Return type**

`str`

**has\_explicit\_title**

Type: `bool`

**run()**

Process the role.

**Return type**

`Tuple[List[Node], List[system_message]]`

**target**

Type: `str`

**title**

Type: `str`

**class PEP621Section**

Bases: `PEP`

Sphinx role for referencing a section within **PEP 621**.

**Methods:**

<code>__call__(name, rawtext, text, lineno, inliner)</code>	Call self as a function.
---	--------------------------

`__call__(name, rawtext, text, lineno, inliner, options={}, content=[])`

Call self as a function.

**Return type**



`Tuple[List[Node], List[system_message]]`

**has\_explicit\_title**

**Type:** `bool`

**target**

**Type:** `str`

**title**

**Type:** `str`

**setup**(*app*)

Setup *sphinx\_packaging.peps*.

**Parameters**

**app** (*Sphinx*) – The Sphinx application.

**Return type**

`Dict[str, Any]`



## sphinx\_packaging.tconf

The *tconf* directive and role for configuration fields in `pyproject.toml` etc.

### Classes:

<i>TConfXRefRole</i> ( <i>fix_parens</i> , <i>lowercase</i> , ...)	Customised XRef role for <i>tconf</i> roles.
<i>TOMLConf</i> ( <i>name</i> , <i>arguments</i> , <i>options</i> , <i>content</i> , ...)	The <i>tconf</i> directive.

### Functions:

<i>resolve_xref</i> ( <i>app</i> , <i>env</i> , <i>node</i> , <i>contnode</i> )	Resolve as-yet-unresolved XRefs for <i>tconf</i> roles.
<i>setup</i> ( <i>app</i> )	Setup <i>sphinx_packaging.tconf</i> .

```
class TConfXRefRole(fix_parens=False, lowercase=False, nodeclass=None, innernodeclass=None,
                    warn_dangling=False)
```

Bases: `XRefRole`

Customised XRef role for *tconf* roles.

### Methods:

<i>process_link</i> ( <i>env</i> , <i>refnode</i> , ...)	Construct a link from the parsed content of the role.
--	---

```
process_link(env, refnode, has_explicit_title, title, target)
```

Construct a link from the parsed content of the role.

### Parameters

- **env** (`BuildEnvironment`) – The Sphinx build environment.
- **refnode** (`Element`) – The reference node.
- **has\_explicit\_title** (`bool`) – Whether the role has an explicit title.
- **title** (`str`) – The title of the XRef role.
- **target** (`str`) – The target of the XRef role. (`:tconf:`title <target>``)

### Return type

`Tuple[str, str]`

### Returns

A tuple of (`title`, `target`).

**class** `TOMLConf`(*name, arguments, options, content, lineno, content\_offset, block\_text, state, state\_machine*)

Bases: `GenericObject`

The `tconf` directive.

**Methods:**

<code>format_default</code> (default)	Formats the <code>:default:</code> option.
<code>format_required</code> (required)	Formats the <code>:required:</code> option.
<code>format_type</code> (the_type)	Formats the <code>:type:</code> option.
<code>handle_signature</code> (sig, signode)	Parse the signature of the <code>tconf</code> directive.
<code>run</code> ()	Process the content of the directive.

**Attributes:**

<code>indextemplate</code>	The template string for index entries.
----------------------------	--

**static** `format_default`(*default*)

Formats the `:default:` option.

**Parameters**

**default** (*str*)

**Return type**

*str*

**static** `format_required`(*required*)

Formats the `:required:` option.

**Parameters**

**required** (*str*)

**Return type**

*bool*

**static** `format_type`(*the\_type*)

Formats the `:type:` option.

**Parameters**

**the\_type** (*str*)

**Return type**

*str*

**handle\_signature**(*sig, signode*)

Parse the signature of the `tconf` directive.

**Parameters**

- **sig** (*str*) – The name of the field.
- **signode** (*desc\_signature*) – The signature node created by Sphinx.

**Return type**

*str*

**Returns**

The final component of the field path (e.g. `foo.bar` -> `bar`).

**indextemplate** = 'pair: %s; TOML configuration field'

**Type:** `str`

The template string for index entries.

**option\_spec** = {'default': <function unchanged\_required>, 'noindex': <function \_flag>, 'required': <function unchanged\_required>, 'type': <function unchanged\_required>}

**Type:** `Mapping[str, Callable[[str], Any]]`

**run()**

Process the content of the directive.

**Return type**

`List[Node]`

**resolve\_xref**(*app, env, node, contnode*)

Resolve as-yet-unresolved XRefs for *tconf* roles.

**Parameters**

- **app** (`Sphinx`) – The Sphinx application.
- **env** (`BuildEnvironment`) – The Sphinx build environment.
- **node** (`Node`) – The cross reference node which has not yet been.
- **contnode** (`Node`) – The child node of the reference node, which provides the formatted text.

**Return type**

`Optional[reference]`

**setup**(*app*)

Setup *sphinx\_packaging.tconf*.

**Parameters**

**app** (`Sphinx`) – The Sphinx application.

**Return type**

`Dict[str, Any]`



## **sphinx\_packaging.toml**

Sphinx extension which adds the *toml* role for referencing sections of the TOML specification.

### **Classes:**

<i>TOML()</i>	Sphinx role for referencing a section of the TOML specification.
---------------	--

### **Functions:**

<i>setup</i> (app)	Setup <i>sphinx_packaging.toml</i> .
--------------------	--------------------------------------

### **class TOML**

Bases: *ReferenceRole*

Sphinx role for referencing a section of the TOML specification.

#### **Methods:**

<i>build_uri()</i>	Constrict the target URI for the reference node.
<i>run()</i>	Process the role.

#### **build\_uri()**

Constrict the target URI for the reference node.

#### **Return type**

*str*

#### **has\_explicit\_title**

**Type:** *bool*

#### **run()**

Process the role.

#### **Return type**

*Tuple*[*List*[*Node*], *List*[*system\_message*]]

#### **target**

**Type:** *str*

#### **title**

**Type:** *str*

**setup**(*app*)

Setup *sphinx\_packaging.toml*.

**Parameters**

**app** (*Sphinx*) – The Sphinx application.

**Return type**

*Dict[str, Any]*



## Downloading source code

The `sphinx-packaging` source code is available on GitHub, and can be accessed from the following URL: <https://github.com/sphinx-toolbox/sphinx-packaging>

If you have git installed, you can clone the repository with the following command:

```
$ git clone https://github.com/sphinx-toolbox/sphinx-packaging
```

```
Cloning into 'sphinx-packaging'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a ‘zip’ file by clicking:

*Clone or download → Download Zip*

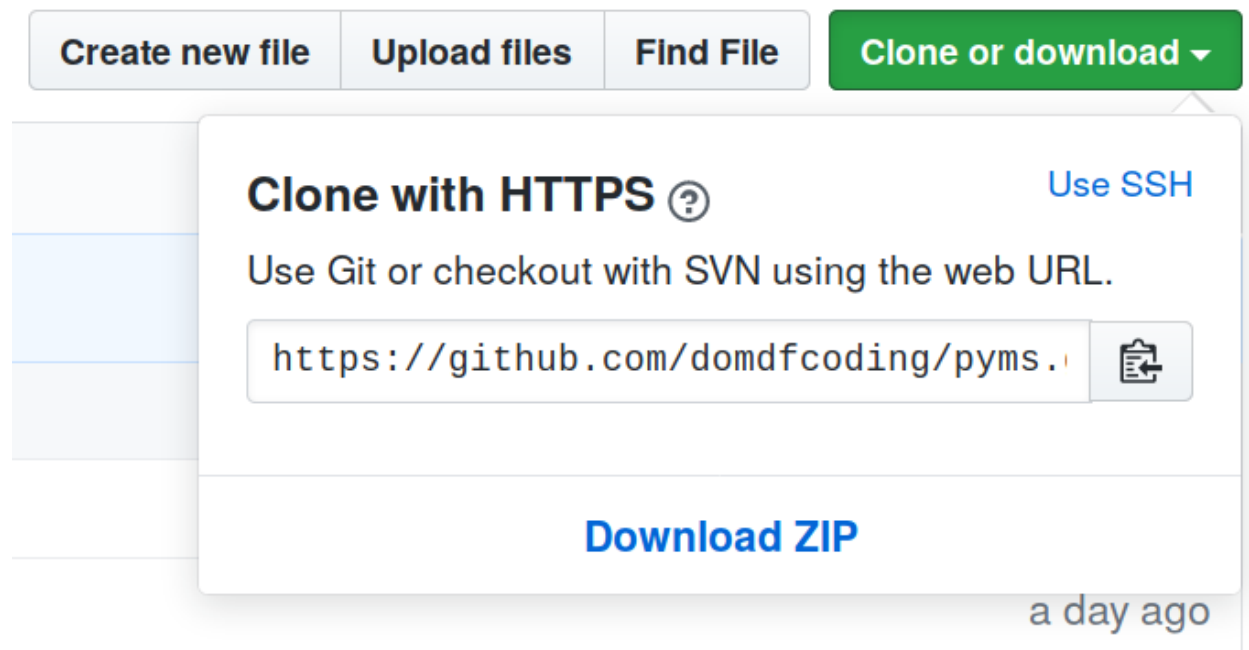


Fig. 1: Downloading a ‘zip’ file of the source code

## 7.1 Building from source

The recommended way to build `sphinx-packaging` is to use `tox`:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use `pep517.build` or another **PEP 517**-compatible build tool.

## License

sphinx-packaging is licensed under the [BSD 3-Clause “New” or “Revised” License](#)

---

A permissive license similar to the [BSD 2-Clause License](#), but with a 3rd clause that prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent.

### Permissions

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Private use – The licensed material may be used and modified in private.

### Conditions

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.

### Limitations

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

[See more information on choosealicense.com](#) ⇒

---

Copyright (c) 2021, Dominic Davis-Foster

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER

(continues on next page)

(continued from previous page)

OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Python Module Index

### S

- [sphinx\\_packaging](#), 9
- [sphinx\\_packaging.peps](#), 11
- [sphinx\\_packaging.tconf](#), 15
- [sphinx\\_packaging.toml](#), 19



## Symbols

`:default:` (directive option)  
     `tconf` (directive), 3  
`:noindex:` (directive option)  
     `tconf` (directive), 3  
`:required:` (directive option)  
     `tconf` (directive), 3  
`:type:` (directive option)  
     `tconf` (directive), 3  
`__call__()` (PEP621Section method), 12

## B

BSD 3-Clause "New" or "Revised" License, 23  
`build_uri()` (CoreMetadata method), 11  
`build_uri()` (PEP method), 12  
`build_uri()` (TOML method), 19

## C

Core Metadata Field description, 5  
 Core Metadata Field Requires-Python, 5  
 Core Metadata Field Supported-Platform, 5  
`core-meta` (role), 5  
`CoreMetadata` (class in `sphinx_packaging.peps`), 11

## D

description  
     TOML configuration field, 4

## F

`format_default()` (TOMLConf static method), 16  
`format_required()` (TOMLConf static method), 16  
`format_type()` (TOMLConf static method), 16

## H

`handle_signature()` (TOMLConf method), 16  
`has_explicit_title` (CoreMetadata attribute), 11  
`has_explicit_title` (PEP attribute), 12  
`has_explicit_title` (PEP621Section attribute), 13  
`has_explicit_title` (TOML attribute), 19

## I

`indextemplate` (TOMLConf attribute), 17

## M

module  
     `sphinx_packaging`, 9  
     `sphinx_packaging.peps`, 11  
     `sphinx_packaging.tconf`, 15  
     `sphinx_packaging.toml`, 19

## O

`option_spec` (TOMLConf attribute), 17

## P

PEP (class in `sphinx_packaging.peps`), 12  
`pep` (role), 4  
`pep621` (role), 5  
`PEP621Section` (class in `sphinx_packaging.peps`), 12  
`process_link()` (TConfXRefRole method), 15  
`project.name`  
     TOML configuration field, 4  
`project.version`  
     TOML configuration field, 4  
 Python Enhancement Proposals  
     PEP 427, 5  
     PEP 440, 4  
     PEP 503#normalized-names, 4  
     PEP 517, 22  
     PEP 621, 3–5, 11, 12  
     PEP 621#authors-maintainers, 5  
     PEP 621#dynamic, 5  
     PEP 621#name, 5  
     PEP 621#requires-python, 5  
     PEP 621#version, 5

## R

`resolve_xref()` (in module `sphinx_packaging.tconf`), 17  
`run()` (CoreMetadata method), 11  
`run()` (PEP method), 12  
`run()` (TOML method), 19  
`run()` (TOMLConf method), 17

## S

`setup()` (in module `sphinx_packaging`), 9  
`setup()` (in module `sphinx_packaging.peps`), 13

`setup()` (in module *sphinx\_packaging.tconf*), 17  
`setup()` (in module *sphinx\_packaging.toml*), 20  
`sphinx_packaging`  
    module, 9  
`sphinx_packaging.peps`  
    module, 11  
`sphinx_packaging.tconf`  
    module, 15  
`sphinx_packaging.toml`  
    module, 19

## T

`target` (*CoreMetadata* attribute), 12  
`target` (*PEP* attribute), 12  
`target` (*PEP621Section* attribute), 13  
`target` (*TOML* attribute), 19  
`tconf` (directive), 3  
    :default: (directive option), 3  
    :noindex: (directive option), 3  
    :required: (directive option), 3  
    :type: (directive option), 3  
`tconf` (role), 6  
`tconf_show_full_name` (configuration value), 7  
*TConfXRefRole* (class in *sphinx\_packaging.tconf*), 15  
`title` (*CoreMetadata* attribute), 12  
`title` (*PEP* attribute), 12  
`title` (*PEP621Section* attribute), 13  
`title` (*TOML* attribute), 19  
*TOML* (class in *sphinx\_packaging.toml*), 19  
`toml` (role), 5  
*TOML* configuration field  
    description, 4  
    project.name, 4  
    project.version, 4  
*TOML*: Inline Table, 6  
*TOML*: Local Date, 6  
*TOML*: Local Date-Time, 6  
*TOML*: Offset Date-Time, 6  
*TOML*: String, 4, 7  
*TOML*: string, 6  
`toml_spec_version` (configuration value), 7  
*TOMLConf* (class in *sphinx\_packaging.tconf*), 16