Package 'thredds'

May 8, 2026

Title Crawler for Navigating THREDDS Catalogs

Description Provides a crawler for programmatically navigating THREDDS Data Server (<https://www.unidata.ucar.edu/software/tds/>) catalogs, and access dataset metadata and resources.

Version 0.1-4

Date 2023-09-01

Depends R (>= 3.0)

Imports R6, rlang, magrittr, httr, xml2

Suggests ncdf4, testthat

Maintainer Emmanuel Blondel <emmanuel.blondel1@gmail.com>

URL https://github.com/BigelowLab/thredds, https://www.unidata.ucar.edu/software/tds/

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Ben Tupper [aut], Emmanuel Blondel [aut, cre] (ORCID: <https://orcid.org/0000-0002-5870-5762>), Bigelow Laboratory for Ocean Sciences [cph]

Repository CRAN

Date/Publication 2023-09-01 09:40:01 UTC

Contents

`build_xpath` *Build and xpath string, possibly using the user specified namespace prefix.*

Description

Build and xpath string, possibly using the user specified namespace prefix.

Usage

```
build_xpath(x, prefix = "d1", select = ".//")
```

Arguments

`x`	character one or more path segments
`prefix`	character by default "d1" prepended to each of the segements in `x`. If NA or length is 0 then ignore.
`select`	charcater, by default search anywhere in the current node with ".//"

Value

xpath descriptor

`CatalogNode` *A class for Catalogs (which may contain catalogs references or datasets)*

Description

A catalog representation that sublcasses from ThreddsNode

Super class

`thredds::ThreddsNode` -> `CatalogNode`

Methods

Public methods:

- CatalogNode$list_services()
- CatalogNode$list_catalogs()
- CatalogNode$list_datasets()
- CatalogNode$get_catalogs()
- CatalogNode$get_datasets()
- CatalogNode$get_dataset_names()
- CatalogNode$get_catalog_names()
- CatalogNode$parse_catalog_node()
- CatalogNode$parse_dataset_node()
- CatalogNode$print()
- CatalogNode$clone()

Method `list_services()`**:** list available services

Usage:

```
CatalogNode$list_services(
  xpath = build_xpath("service", prefix = self$prefix),
  form = "list"
)
```

Arguments:

`xpath` character, the xpath specifications

`form` character, either "list" or "table"

Returns: list of zero or more character vectors

Method `list_catalogs()`**:** list available catalogRefs

Usage:

```
CatalogNode$list_catalogs(
  xpath = build_xpath(c("dataset", "catalogRef"), prefix = self$prefix),
  form = "list"
)
```

Arguments:

`xpath` character, the xpath descriptor

`form` character, either "list" or "table"

Returns: a list with zero or more character vectors

Method `list_datasets()`**:** list available datasets

Usage:

```
CatalogNode$list_datasets(
  xpath = build_xpath(c("dataset", "dataset"), prefix = self$prefix),
  form = "list"
)
```

Arguments:

`xpath` character, the xpath descriptor

`form` character, either "list" or "table"

Returns: a list with zero or more character vectors

Method `get_catalogs()`**:** Retrieve a list one or more of child catalogs

Usage:

```
CatalogNode$get_catalogs(
  index,
  xpath = build_xpath(c("dataset", "catalogRef"), prefix = self$prefix)
)
```

Arguments:

`index` integer index (1,...,nChild), indices or name(s)

`xpath` character xpath representation

Returns: a list of Catalog class objects, possibly NULL

Method `get_datasets()`**:** Retrieve list one or more dataset children

Usage:

```
CatalogNode$get_datasets(
  index,
  xpath = build_xpath(c("dataset", "dataset"), prefix = self$prefix)
)
```

Arguments:

`index` the integer index (1,...,nChild), indices or name(s)

`xpath` character xpath representation

Returns: a list of Dataset objects or NULL

Method `get_dataset_names()`**:** Retrieve list zero or more dataset child names. If unnnamed, then we substitute "title", "ID", "urlPath", or "href" in that order of availability.

Usage:

```
CatalogNode$get_dataset_names(
  xpath = build_xpath(c("dataset", "dataset"), prefix = self$prefix)
)
```

Arguments:

`xpath` character xpath representation

`index` the integer index (1,...,nChild), indices or name(s)

Returns: character vector of zero or more names

Method `get_catalog_names()`**:** Retrieve list zero or more catalog child names. If unnnamed, then we substitute "title", "ID", "urlPath" or href" in that order of availability.

Usage:

```
CatalogNode$get_catalog_names(
  xpath = build_xpath(c("dataset", "catalogRef"), prefix = self$prefix)
)
```

Arguments:

`xpath` character xpath representation

`index` the integer index (1,...,nChild), indices or name(s)

Returns: character vector of zero or more names

Method `parse_catalog_node()`**:** Parse a catalog node

Usage:

```
CatalogNode$parse_catalog_node(x)
```

Arguments:

`x` xml_node

Returns: Catalog class object

Method `parse_dataset_node()`**:** Parse a dataset node

Usage:

```
CatalogNode$parse_dataset_node(x)
```

Arguments:

`x` xml_node

Returns: Dataset class object

Method `print()`**:** print method

Usage:

```
CatalogNode$print(prefix = "")
```

Arguments:

`prefix` character, to be printed before each line of output (like spaces)

`...` other arguments for superclass

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CatalogNode$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(thredds)
top_uri <- 'https://oceandata.sci.gsfc.nasa.gov/opendap/catalog.xml'
Top <- thredds::CatalogNode$new(top_uri)

#to browse catalogue
#Top$browse()

#go down in 'MODISA' catalog
L3 <- Top$get_catalogs("MODISA")[["MODISA"]]$get_catalogs()[[1]]
```

```
#see what's available for 2009
catalog2009 <- L3$get_catalogs("2009")[[1]]

#get catalog for 2009-01-20
doy <- format(as.Date("2009-01-20"), "%m%d")
catalog20 <- catalog2009$get_catalogs(doy)[[doy]]

#get dataset node
chl <- catalog20$get_datasets("AQUA_MODIS.20090120.L3m.DAY.CHL.chlor_a.4km.nc")[[1]]

#retrieve the relative URL, and add it to the base URL for the service.
#Somewhat awkwardly, the relative URL comes prepended with a path separator, so we
#use straight up `paste0` to append to the base_uri.
#if(require("ncdf4")){
#  base_uri <- "https://oceandata.sci.gsfc.nasa.gov:443/opendap"
#  uri <- paste0(base_uri, chl[["AQUA_MODIS.20090120.L3m.DAY.CHL.chlor_a.4km.nc"]]$url)
#  NC <- ncdf4::nc_open(uri)
#}
```

`DatasetNode` *A class for a single datatset reference*

Description

A direct Dataset representation that subclasses from ThreddsNode

Super class

`thredds::ThreddsNode` -> `DatasetNode`

Public fields

`name` character, often the filename

`dataSize` numeric, size in bytes

`date` character, modification date

Methods

Public methods:

- `DatasetNode$new()`
- `DatasetNode$GET()`
- `DatasetNode$get_url()`
- `DatasetNode$list_access()`
- `DatasetNode$print()`
- `DatasetNode$clone()`

Method `new()`: initialize an instance of ServiceNode

Usage:

```
DatasetNode$new(x, ...)
```

Arguments:

`x` url or xml2::xml_node

`...` arguments for superclass initialization

Method `GET()`: Overrides the GET method of the superclass. GET is not permitted

Usage:

```
DatasetNode$GET()
```

Returns: NULL

Method `get_url()`: Retrieve the relative URL for a dataset.

Usage:

```
DatasetNode$get_url(
  service = c("dap", "opendap", "wms")[1],
  sep = c("/", "")[2],
  ...
)
```

Arguments:

`service` character, the service to use. (default 'dap' equivalent to 'opendap') Ignored if 'urlPath' or 'href' is in the nodes' attributes.

`sep` character, typically "/" or "" (default), used for joined base_url to relative url

`...` other arguments for `DatasetNode$list_access`

Returns: character

Method `list_access()`: list access methods

Usage:

```
DatasetNode$list_access(xpath = build_xpath("access", prefix = self$prefix))
```

Arguments:

`xpath` charcater, xpath descriptor

Returns: named list of character vectors or NULL

Method `print()`: print method

Usage:

```
DatasetNode$print(prefix = "")
```

Arguments:

`prefix` character, to be printed before each line of output (like spaces)

`...` other arguments for superclass

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
DatasetNode$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

For examples see CatalogNode

---

get_catalog *Retrieve a catalog*

Description

Retrieve a catalog

Usage

```
get_catalog(uri, ...)
```

Arguments

uri	the URI of the catalog
...	further arguments for parse_node

Value

ThreddsNodeRefClass or subclass or NULL

---

get_xml_ns *Retrieve the namespaces for a resource*

Description

Retrieve the namespaces for a resource

Usage

```
get_xml_ns(uri)
```

Arguments

uri	the URI of the catalog

Value

the output of xml_ns

grepl_it *Determine if a vector of names match the greplargs*

Description

Determine if a vector of names match the greplargs

Usage

```
grepl_it(x, greplargs = NULL)
```

Arguments

x	a vector of names
greplargs	NULL, vector or list

Value

logical vector

is_xmlNode *Test if an object inherits from xml2::xml_node*

Description

Test if an object inherits from xml2::xml_node

Usage

```
is_xmlNode(x, classname = "xml_node")
```

Arguments

x	object to test
classname	character, the class name to test against, by default 'xml_node'

Value

logical

parse_node *Convert a node to an object inheriting from ThreddsNode*

Description

Convert a node to an object inheriting from ThreddsNode

Usage

```
parse_node(node, url = NULL, verbose = FALSE, encoding = "UTF-8", ...)
```

Arguments

node	xml2::xml_node or an httr::response object
url	character, optional url if a catalog or direct dataset
verbose	logical, by default FALSE
encoding	character, by default UTF-8
...	further arguments for instantiation of classes (such as ns = "foo")

Value

ThreddsNode class object or subclass

ServiceNode *A simple class for parsing and holdoing service info*

Description

A Service representation that subclasses from ThreddsNode

Super class

thredds::ThreddsNode -> ServiceNode

Public fields

name character

serviceType character

base character base url

Methods

Public methods:

- `ServiceNode$new()`
- `ServiceNode$print()`
- `ServiceNode$clone()`

Method `new()`**:** initialize an instance of ServiceNode

Usage:

```
ServiceNode$new(x, ...)
```

Arguments:

`x` url or xml2::xml_node

`...` arguments for superclass initialization

Method `print()`**:** print method

Usage:

```
ServiceNode$print(prefix = "")
```

Arguments:

`prefix` character, to be printed before each line of output (like spaces)

`...` other arguments for superclass

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ServiceNode$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

For examples see CatalogNode

`thredds` *thredds: Crawler for Navigating THREDDS Catalogs*

Description

A limited crawler for programmatically navigating THREDDS catalogs.

ThreddsNode *A base representation that other nodes subclass from*

Description

R6 base class for all other to inherit from

Public fields

`url` character - possibly wrong but usually right!

`node` xml2::xml_node

`verbose` logical

`prefix` xpath namespace prefix, NA or NULL or charcater() to ignore

`tries` numeric number of requests attempts before failing

`encoding` character, by default 'UTF-8'

`base_url` character, the base URL for the service

Methods

Public methods:

- `ThreddsNode$new()`
- `ThreddsNode$print()`
- `ThreddsNode$GET()`
- `ThreddsNode$browse()`
- `ThreddsNode$children_names()`
- `ThreddsNode$clone()`

Method `new()`**:** initialize an instance of ThreddsNode

Usage:

```
ThreddsNode$new(
  x,
  verbose = FALSE,
  n_tries = 3,
  prefix = NULL,
  ns_strip = FALSE,
  encoding = "UTF-8",
  base_url = ""
)
```

Arguments:

`x` url or xml2::xml_node

`verbose` logical, TRUE to be noisy (default FALSE)

`n_tries` numeric, defaults to 3

`prefix` character, the namespace to examine (default NULL, inherited when initialized)

`ns_strip` logical, if TRUE then strip namespace (default FALSE)

`encoding` character, by default 'UTF-8'

`base_url` character, the base URL for the service

Method `print()`**:** print method

Usage:

```
ThreddsNode$print(prefix = "", ...)
```

Arguments:

`prefix` character, to be printed before each line of output (like spaces)

`...` other argum,ents (ignored for now)

Method `GET()`**:** Retrieve a node of the contents at this nodes URL

Usage:

```
ThreddsNode$GET()
```

Returns: ThreddsNode or subclass or NULL

Method `browse()`**:** Browse the URL if possible

Usage:

```
ThreddsNode$browse()
```

Method `children_names()`**:** Retrieve a vector of unique child names

Usage:

```
ThreddsNode$children_names(...)
```

Arguments:

`...` further arguments for `xml_children_names`

Returns: a vector of zero or more child names

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ThreddsNode$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

Abstract class. For examples see CatalogNode

xmlString *Convert xm2::xml_node to character*

Description

Convert xm2::xml_node to character

Usage

```
xmlString(x)
```

Arguments

x	xmlNode

Value

character

xml_children_names *Get the names of children*

Description

Get the names of children

Usage

```
xml_children_names(x, unique_only = TRUE)
```

Arguments

x	xml2::xml_node
unique_only	logical if TRUE remove duplicates

Value

zero or more child names.

`xml_id` *Retrieve an ID value for a node from it's attributes.*

Description

Retrieve an ID value for a node from it's attributes.

Usage

```
xml_id(x, atts = c("name", "title", "ID", "urlPath", "href"))
```

Arguments

`x`	xml node or a named character vector as per `xml_attrs`
`atts`	character, ordered vector of attribute names to use as an ID value As the list is stepped through if an attribute is missing or empty character then advance to the next, otherwise return that value

Value

character identifier, possibly an empty character (`character()`)

Index