

# Package ‘rols’

March 25, 2021

**Type** Package

**Title** An R interface to the Ontology Lookup Service

**Version** 2.18.2

**Description** The rols package is an interface to the Ontology Lookup Service (OLS) to access and query hundred of ontologies directly from R.

**Depends** methods

**Imports** httr, progress, jsonlite, utils, Biobase, BiocGenerics (>= 0.23.1)

**Suggests** GO.db, knitr (>= 1.1.0), BiocStyle (>= 2.5.19), testthat, lubridate, DT, rmarkdown,

**biocViews** ImmunoOncology, Software, Annotation, MassSpectrometry, GO

**VignetteBuilder** knitr

**License** GPL-2

**Encoding** UTF-8

**URL** <http://lgatto.github.com/rols/>

**BugReports** <https://github.com/lgatto/rols/issues>

**Collate** AllClasses.R AllGenerics.R utils.R cvparam.R  
methods-OlsSearch.R methods-Ontologies.R methods-Terms.R  
methods-Properties.R zzz.R

**RoxygenNote** 6.1.0

**git\_url** <https://git.bioconductor.org/packages/rols>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** 9fa9e4a

**git\_last\_commit\_date** 2021-02-17

**Date/Publication** 2021-03-24

**Author** Laurent Gatto [aut, cre],  
Tiage Chedraoui Silva [ctb]

**Maintainer** Laurent Gatto <laurent.gatto@uclouvain.be>

## R topics documented:

.properties . . . . .	2
.term . . . . .	2
.terms . . . . .	3
CVParam-class . . . . .	3
makeOntologies . . . . .	5
makeOntology . . . . .	6
makeProperty . . . . .	6
makeTerm . . . . .	7
OlsSearch-class . . . . .	7
Ontology-class . . . . .	8
Properties-class . . . . .	11
Term-class . . . . .	12
<b>Index</b>	<b>15</b>

---

.properties	<i>Constructs the query for all properties from a given ontology</i>
-------------	--

---

### Description

Constructs the query for all properties from a given ontology

### Usage

```
.properties(oid, pagesize = 200)
```

### Arguments

oid	A character with an ontology or an ontology
pagesize	How many results per page to return

### Value

An object of class Terms

---

.term	<i>Constructs the query for a single term from a given ontology</i>
-------	---

---

### Description

Constructs the query for a single term from a given ontology

### Usage

```
.term(oid, termid)
```

**Arguments**

oid	A character with an ontology or an ontology
termid	A character with a term id

**Value**

An object of class Term

---

.terms	<i>Constructs the query for all term from a given ontology</i>
--------	--

---

**Description**

Constructs the query for all term from a given ontology

**Usage**

```
.terms(oid, pagesize = 1000)
```

**Arguments**

oid	A character with an ontology or an ontology
pagesize	How many results per page to return

**Value**

An object of class Terms

---

CVParam-class	<i>Class "CVParam"</i>
---------------	------------------------

---

**Description**

CVParam objects instantiate controlled vocabulary entries.

**Usage**

```
CVParam(label, name, accession, value, exact = TRUE)
```

**Arguments**

label	A character with the ontology label. If missing, a user-defined parameter is created.
name	A character with the name of the CVParam to be constructed. This argument can be omitted if accession is used and label is not missing.
accession	A character with the accession of the CVParam to be constructed. This argument can be omitted if name is used. Ignored for user-defined instances.
value	A character with the value of the CVParam to be constructed. This argument is optional.
exact	A logical defining whether the query to retrieve the accession (when name is used) should be an exact match.

## Objects from the Class

Objects can be created with the CVParam constructor.

## Slots

**label:** Object of class "character" that defines the label of the instance, i.e the ontology abbreviation/prefix. See [Ontologies](#) to generate a list of available ontologies and [olsPrefix](#) for existing labels.

**accession:** Object of class "character" with the parameter's valid label ontology accession number. See below for validity constrains.

**name:** Object of class "character" with the instance's valid name, i.e matching with the accession. name and accession must follow `term(accession, label) == name` for the instance to be valid.

**value:** Object of class "character" with the CVParams value, if applicable, of empty string ("") otherwise.

**user:** Object of class "logical" defining if the instance is a user-defined parameter (also called User params).

**.\_\_classVersion\_\_:** Object of class "[Versions](#)" describing the instance's class definition version. For development use.

## Extends

Class "[Versioned](#)", directly.

## Methods

**charIsCVParam(x)** Checks if x, a character of the form "[ONTO, ACCESSION, NAME, VALUE]", is a valid (possibly user-defined) CVParam. "ONTO" is the ontology label (prefi), "ACCESSION" is the term accession number, "NAME" is the term's name and "VALUE" is the value. Note that only syntax validity is verified, not semantics. See example below.

## Methods

**coerce** signature(from = "CVParam", to = "character"): Coerces CVParam from to a character of the following form: [label, accession, name, value]. `as.character` is also defined.

**coerce** signature(from = "character", to = "CVParam"): Coerces character from to a CVParam. `as.CVParam` is also defined. If a label is absent, the character is converted to a User param, else, the label and accession are used to query the Ontology Lookup Service (see [OlsSearch](#)). If a name is provided and does not match the retrieved name, a warning is thrown.

This function is vectorised; if the from character is of length greater than 1, then a list of CVParam is returned. The queries to the OLS are processed one-by-one, though.

**show** signature(object = "CVParam"): Prints the CVParam instance as text.

**rep** signature(x = "CVParam", times = "numeric"): Replicates the CVParam x times times.

## Author(s)

Laurent Gatto <lg390@cam.ac.uk>

**Examples**

```
## a user param
CVParam(name = "A user param", value = "the value")
## a CVParam from PSI's Mass Spectrometry ontology
term("MS", "MS:1000073")
CVParam(label = "MS", accession = "MS:1000073")
CVParam(label = "MS", name = "electrospray ionization")
CVParam(label = "MS", name = "ESI") ## using a synonym

## From a CVParam object to a character
cv <- as(CVParam(label = "MS", accession = "MS:1000073"), "character")
cv

## From a character object to a CVParam
as(cv, "CVParam")
as("[MS, MS:1000073, , ]", "CVParam") ## no name
as("[MS, MS:1000073, ESI, ]", "CVParam") ## name does not match
as(c(cv, cv), "CVParam") ## more than 1 character

x <- c("[MS, MS:1000073, , ]", ## valid CV param
      "[, , Hello, world]", ## valid User param
      "[this, one is, not, valid]", ## not valid
      "[ , , , ]") ## not valid

stopifnot(charIsCVParam(x) == c(TRUE, TRUE, FALSE, FALSE))

## A list of expected valid and non-valid entries
rols::validCVchars
rols::notvalidCVchars
```

---

makeOntologies	<i>Makes an Ontologies instance based on the response from api/ontologies @return</i>
----------------	---

---

**Description**

Makes an Ontologies instance based on the response from api/ontologies @return

**Usage**

```
makeOntologies(pagesize = 150)
```

**Arguments**

pagesize	A numeric indicating the number of elements per page (default in method is 150).
----------	--

**Value**

An object of class Ontologies

---

makeOntology	<i>Makes an Ontology instance based on the response from /api/ontologies/ontology_id</i>
--------------	--

---

**Description**

Makes an Ontology instance based on the response from /api/ontologies/ontology\_id

**Usage**

makeOntology(x)

**Arguments**

x                    A valid ontology prefix

**Value**

An object of class Ontology

---

makeProperty	<i>Makes a Property instance based on the response from /api/ontologies/ontology/terms/iri</i>
--------------	--

---

**Description**

Makes a Property instance based on the response from /api/ontologies/ontology/terms/iri

**Usage**

makeProperty(x)

**Arguments**

x                    The content from the response

**Value**

An object of class Property

---

makeTerm	<i>Makes a Term instance based on the response from /api/ontologies/ontology/terms/iri</i>
----------	--

---

**Description**

Makes a Term instance based on the response from /api/ontologies/ontology/terms/iri

**Usage**

```
makeTerm(x)
```

**Arguments**

x	The content from the response
---	-------------------------------

**Value**

An object of class Term

---

OlSearch-class	<i>Class "OlSearch"</i>
----------------	-------------------------

---

**Description**

Searching the OLS is done using the OlSearch data structure.

**Objects from the Class**

Objects can be created with the constructor function OlSearch.

**Slots**

```
q: Object of class "character" ~~
ontology: Object of class "character" ~~
type: Object of class "character" ~~
slim: Object of class "character" ~~
fieldList: Object of class "character" ~~
queryFields: Object of class "character" ~~
exact: Object of class "logical" ~~
groupField: Object of class "logical" ~~
obsoletes: Object of class "logical" ~~
local: Object of class "character" ~~
childrenOf: Object of class "character" ~~
rows: Object of class "integer" ~~
start: Object of class "integer" ~~
url: Object of class "character" ~~
numFound: Object of class "integer" ~~
response: Object of class "data.frame" ~~
```

**Methods and functions**

**coerce** signature(from = "OlsSearch", to = "data.frame"): ...

**coerce** signature(from = "OlsSearch", to = "Terms"): ...

**show** signature(object = "OlsSearch"): ...

**olsRows** signature(object = "OlsSearch"): ... The value can be updated with the `olsRows` replacement method. To request all responses, use `allRows`.

**Author(s)**

Laurent Gatto <lg390@cam.ac.uk>

**Examples**

```
OlsSearch(q = "trans-golgi")
OlsSearch(q = "cell")
OlsSearch(q = "cell", exact = TRUE)
OlsSearch(q = "cell", exact = TRUE, ontology = "go")
OlsSearch(q = "cell", exact = TRUE, ontology = "GO")

OlsSearch(q = "electrospray", ontology = "MS")
OlsSearch(q = "ionization", ontology = "MS")
OlsSearch(q = "electrospray ionization", ontology = "MS")
OlsSearch(q = "electrospray ionization", ontology = "MS", exact=TRUE)

## Request 5 results instead of 20 (default)
OlsSearch(q = "plasma,membrane", ontology = "go", rows = 5)

## or, once the object was created
(res <- OlsSearch(q = "plasma,membrane", ontology = "go"))
olsRows(res) <- 5
res
## all results
res <- allRows(res)
res

res <- OlsSearch(q = "trans-golgi", ontology = "go", rows = 5)
res
res <- olsSearch(res)
res
as(res, "data.frame")
res <- as(res, "Terms")
res
termPrefix(res)
termId(res)
```

---

Ontology-class

*Class "Ontology"*

---

**Description**

Ontologies are stored as `Ontology` and `Ontologies` instances, and contain various information as provided by the Ontology Lookup Service.



## Details

Ontologies are referred to by their namespace, which is lower case: the Gene Ontology is "go", the Mass spectrometry ontology is "ms", etc. The ontologies also have prefixes, which are upper case: the Gene Ontology prefix "GO", the Mass spectrometry ontology prefix "MS". The only exception to this rule is the Drosophila Phenotype Ontology, whose namespace and prefix are "dpo" and "FBcv" respectively. This is particularly confusing as the FlyBase Controlled Vocabulary has "fbcv" and "FBcv" as namespace and prefix respectively.

When using a character to initialise an ontology or query a term, "fbcv" (this is case insensitive) will refer to the the FlyBase Controlled Vocabulary. The the Drosophila Phenotype Ontology will have to be referred as "dpo" (also case insensitive).

## Objects from the Class

Objects can be created in multiple ways. The `Ontologies` function will initialise all available ontologies as an `Ontologies` object, while a call to `Ontology` with an ontology namespace or prefix (but see Details section) as argument will initialise the ontology of interest as an `Ontology` instance.

`Ontologies` instances can be subset with `[]` and `[[` (using their namespace, see Details) and iterated over with `lapply`. `Ontologies` can be converted into a simple `data.frame` containing the ontology prefixes, namespaces and titles using `as(., "data.frame")`. An `Ontologies` can also be coerced to lists of `Ontology` objects with `as(., "list")`.

## Slots

**loaded:** Object of class `NULL` or character containing the date the ontology was loaded on the backend side. Accessed with the `olsLoaded` method.

**updated:** Object of class `NULL` or character containing the date the ontology was last updated on the backend side. Accessed with the `olsUpdated` method.

**status:** Object of class `NULL` or character documenting the status of the ontology on the backend side. For example "LOADED", "FAILED" or "NOTLOADED". Accessed with the `olsStatus` method.

**message:** Object of class `NULL` or character documentating the status of the ontology on the backend side.

**version:** Object of class `NULL` or character documenting the version of the ontology. Note that there is also a `version` field in the `config` slot below. Use `olsVersion` to access the appropriate date.

**numberOfTerms:** Object of class "integer" documenting the number of terms available in the ontology.

**numberOfProperties:** Object of class "integer" documenting the number of properties available in the ontology.

**numberOfIndividuals:** Object of class "integer" documenting the number of individuals available in the ontology.

**config:** Object of class "list" containing further ontology configuration and metadata.

## Methods and functions

**Ontologies** signature(object = "numeric"):

**Ontology** signature(object = "character"):

**olsDesc** signature(object = "Ontology"): returns the description of an ontology. Also works for `Ontologies` objects and character describing an ontology namespace or prefix (see Details).

- olsPrefix** signature(object = "Ontology"): retruns the prefix of an ontology. Also works for Ontologies objects describing an ontology namespace or prefix (see Details).
- olsRoot** signature(object = "Ontology"): returns the root of the ontology as a [Terms](#) instance. object could also be a character with an ontology namespace or prefix (see Details). If object is of class Ontologies, it returns a list of [Terms](#).
- olsVersion** signature(object = "Ontology"): returns the version of the ontology. Also works with an ontology namespace or prefix (see Details) as a character or an object of class Ontologies, in which case it returns a list of versions.
- olsLoaded** signature(object = "Ontology"): returns the loading date of the ontology. Also works with a character containing the ontology namespace or prefix (see Details) or an object of class Ontologies.
- olsUpdated** signature(object = "Ontology"): returns the update date of the ontology. Also works with a character containing the ontology namespace or prefix (see Details) or an object of class Ontologies.
- olsStatus** signature(object = "Ontology"): returns the status of the ontology. Also works with a character containing the ontology namespace or prefix (see Details) or an object of class Ontologies.
- olsStatus** signature(object = "Ontology"): returns the namespace of the ontology. Also works with a character containing the ontology namespace or prefix (see Details) or an object of class Ontologies.
- olsTitle** signature(object = "Ontology"): returns the title of an ontology. Also works with a character containing the ontology namespace or prefix (see Details) or an object of class Ontologies.
- show** signature(object = "Ontology"): prints a short summary of Ontology and Ontologies objects.
- length** signature(object = "Ontologies"): returns the number of ontolgies described by the Ontologies object.
- all.equal** signature(target = "Ontologies", current = "Ontologies"): ...

### Author(s)

Laurent Gatto <lg390@cam.ac.uk>

### Examples

```
## Get all ontolgies
ol <- Ontologies()
ol

head(as(ol, "data.frame"))
length(ol)

## Individual ontologies
(go <- ol[["go"]])
(efo <- ol[["efo"]])

## some basic information
olsVersion(go)
olsDesc(go)
olsTitle(go)
olsPrefix(go)
```

```

olsNamespace(go)

olsRoot(go)

## works with Ontology objects or their namespace
identical(olsRoot("go"), olsRoot(go))
identical(olsVersion("go"), olsVersion(go))

## Directly initialise a single ontology
go1 <- Ontology("go") ## using the namespace (preferred)
go2 <- Ontology("G0") ## using the prefix (see Details)
all.equal(go, go1)
all.equal(go, go2)

```

---

Properties-class	Class "Properties"
------------------	--------------------

---

## Description

Properties (relationships) between terms can be queried for complete [Ontology](#) objects and [Term/Terms](#) instances, and the results are stored as objects of class Property or Properties.

## Objects from the Class

Objects can be created by calls to `properties`, as described below.

## Slots

See the [Term](#) and [Terms](#) classes.

## Extends

Class "[Terms](#)", directly.

## Methods and functions

**properties** signature(object = "Ontology", pagesize = 200): ... Also works with a character with the ontology namespace. See [Ontology](#) for details.

**properties** signature(object = "Term"): retrieves the properties of term object and returns a Properties object. Returns NULL when no properties are available.

**proteties** signature(object = "Terms", ...): retrieves the properties of each term of object and returns a list of Properties (or NULL) items.

**show** signature(object = "Properties"): shows a textual summary of the object.

**length** signature(object = "Properties"): returns the number of properties in object.

## Author(s)

Laurent Gatto <lg390@cam.ac.uk>

**Examples**

```
trm <- term("uberon", "UBERON:0002107")
trm
properties(trm)

trm2 <- term("GO", "GO:0005326")
trm2
properties(trm2)
```

---

Term-class	<i>Class "Term"</i>
------------	---------------------

---

**Description**

The Term class describes an ontology term. A set of terms are instantiated as a Terms class.

**Objects from the Class**

Objects can be created using the `term` and `terms` functions. The latter is used with an object of class `Ontology` or a character describing a valid ontology prefix to download and instantiate all terms of an ontology of interest. The former takes an `Ontology` object (or an ontology prefix) and a term identifier to instantiate that specific term. See also the 'Methods and functions' sections.

For any given Term object, the children, parents, ancestors, descendants, `partOf` and `derivesFrom` terms can be generated and returned as Terms objects.

Terms instances can be subset with `[]` and `[[` and iterated over with `lapply`.

**Slots**

```
iri: Object of class "character" ~~
label: Object of class "character" ~~
description: Object of class "NullOrList" ~~
annotation: Object of class "list" ~~
synonym: Object of class "NullOrList" ~~
ontology_name: Object of class "character" ~~
ontology_prefix: Object of class "character" ~~
ontology_iri: Object of class "character" ~~
is_obsolete: Object of class "logical" ~~
is_defining_ontology: Object of class "logical" ~~
has_children: Object of class "logical" ~~
is_root: Object of class "logical" ~~
short_form: Object of class "character" ~~
obo_id: Object of class "NullOrChar" ~~
links: Object of class "list" ~~
```

**Methods and functions**

- term** signature(object = "Ontology", id = "character"): ...
- terms** signature(x = "Ontology", pagesize = "numeric"): ...
- termDesc** signature(object = "Term"): ...
- termLabel** signature(object = "Term"): ...
- termPrefix** signature(object = "Term"): ...
- termSynonym** signature(object = "Term"): ...
- termNamespace** signature(object = "Term"): ...
- termOntology** signature(object = "Term"): ...
- isRoot** signature(object = "Term"): ...
- isObsolete** signature(object = "Term"): ...
- termId** signature(object = "Term"): ...
- children** signature(object = "Term"): Returns a new Terms instance with the object's children. NULL if there are not children.
- parents** signature(object = "Term"): Returns a new Terms instance with the object's parents. NULL if there are not parents.
- ancestors** signature(object = "Term"): Returns a new Terms instance with the object's ancestors. NULL if there are not ancestors.
- descendants** signature(object = "Term"): Returns a new Terms instance with the object's descendants. NULL if there are not descendants.
- partOf** signature(object = "Term"): Returns a new Terms instance with terms the object's is a part of. NULL if none.
- derivesFrom** signature(object = "Term"): Returns a new Terms instance with terms the object's is derived from. NULL if none.
- show** signature(object = "Term"): ...
- show** signature(object = "Terms"): ...
- all.equal** signature(target = "Term", current = "Term"): ...
- all.equal** signature(target = "Terms", current = "Terms"): ...
- length** signature(object = "Terms"): returns the number of ontologies described by the Terms object.
- unique** signature(x = "Terms"): returns a new Terms object where all duplicated Term instances, i.e. those with the same term id (even when stemming from different ontologies), have been removed (only the first occurrence is retained).
- as(x, "data.frame") Coerces a single Term or Terms into a data.frame of length 1 (for the former) or length length(x) for the latter. The result will contain the following columns: id, label, description of the term(s), their ontology, whether they are obsolete, have children or are root node, the first synonym only, their iri and whether they are defining the ontology. Any missing value will be reported as NA.

**Author(s)**

Laurent Gatto <lg390@cam.ac.uk>

**Examples**

```

## (all) terms
(gotrms <- terms("go", pagesize = 10000))

## Not run:
## or, using on ontology object
go <- Ontology("go")
gotrms <- terms(go, pagesize = 10000)

## End(Not run)

as(gotrms[1:10], "data.frame") ## data,frame with 10 rows

## (one) term
(trm <- gotrms[[1]])
termPrefix(trm)
gotrms[1:3]
gotrms[["GO:0005230"]]

as(trm, "data.frame") ## data,frame with 1 row

## using an Ontology object
go <- Ontology("GO")
term(go, "GO:0032801")
## using an ontology prefix
term("GO", "GO:0032801")

isObsolete(gotrms[["GO:0005230"]])
isObsolete(gotrms[["GO:0005232"]])

isRoot(gotrms[["GO:0005230"]])
isRoot(gotrms[["GO:0005232"]])

i <- isRoot(gotrms) & !isObsolete(gotrms)
gotrms[i]
for (ii in which(i))
  show(gotrms[[ii]])

## or, directly querying the ontology
olsRoot(go)

parents(trm)
ancestors(trm)
children(trm)
descendants(trm)

partOf(gotrms[["GO:0044429"]])
partOf(term("BTO", "BTO:0000142"))

derivesFrom(term("BTO", "BTO:0002600"))
derivesFrom(term("BTO", "BTO:0001023"))

```

# Index

## \* classes

- CVParam-class, 3
- OlsSearch-class, 7
- Ontology-class, 8
- Properties-class, 11
- Term-class, 12
- .properties, 2
- .term, 2
- .terms, 3
- [, Ontologies-method (Ontology-class), 8
- [, Terms-method (Term-class), 12
- [[, Ontologies-method (Ontology-class), 8
- [[, Terms-method (Term-class), 12
- all.equal, Ontologies, Ontologies-method (Ontology-class), 8
- all.equal, Ontology, Ontology-method (Ontology-class), 8
- all.equal, Term, Term-method (Term-class), 12
- all.equal, Terms, Terms-method (Term-class), 12
- allRows (OlsSearch-class), 7
- ancestors (Term-class), 12
- as.character.CVParam (CVParam-class), 3
- as.CVParam.character (CVParam-class), 3
- as.Term.data.frame (Term-class), 12
- as.Terms.data.frame (Term-class), 12
- charIsCVParam (CVParam-class), 3
- children (Term-class), 12
- class:OlsSearch (OlsSearch-class), 7
- class:Ontologies (Ontology-class), 8
- class:Ontology (Ontology-class), 8
- class:Properties (Properties-class), 11
- class:Property (Properties-class), 11
- class:Term (Term-class), 12
- class:Terms (Term-class), 12
- coerce, character, CVParam-method (CVParam-class), 3
- coerce, CVParam, character-method (CVParam-class), 3
- coerce, OlsSearch, data.frame-method (OlsSearch-class), 7
- coerce, OlsSearch, Terms-method (OlsSearch-class), 7
- coerce, Ontologies, data.frame-method (Ontology-class), 8
- coerce, Ontologies, list-method (Ontology-class), 8
- coerce, Term, data.frame-method (Term-class), 12
- coerce, Terms, data.frame-method (Term-class), 12
- CVParam (CVParam-class), 3
- CVParam-class, 3
- derivesFrom (Term-class), 12
- descendants (Term-class), 12
- isObsolete (Term-class), 12
- isObsolete, Term-method (Term-class), 12
- isObsolete, Terms-method (Term-class), 12
- isRoot (Term-class), 12
- isRoot, Term-method (Term-class), 12
- isRoot, Terms-method (Term-class), 12
- lapply, Ontologies-method (Ontology-class), 8
- lapply, Terms-method (Term-class), 12
- length, Ontologies-method (Ontology-class), 8
- length, Properties-method (Properties-class), 11
- length, Terms-method (Term-class), 12
- makeOntologies, 5
- makeOntology, 6
- makeProperty, 6
- makeTerm, 7
- olsDesc (Ontology-class), 8
- olsDesc, character-method (Ontology-class), 8
- olsDesc, Ontologies-method (Ontology-class), 8
- olsDesc, Ontology-method (Ontology-class), 8
- olsLoaded (Ontology-class), 8

- olsLoaded, character-method  
(Ontology-class), 8
- olsLoaded, Ontologies-method  
(Ontology-class), 8
- olsLoaded, Ontology-method  
(Ontology-class), 8
- olsNamespace (Ontology-class), 8
- olsNamespace, character-method  
(Ontology-class), 8
- olsNamespace, Ontologies-method  
(Ontology-class), 8
- olsNamespace, Ontology-method  
(Ontology-class), 8
- olsPrefix, 4
- olsPrefix (Ontology-class), 8
- olsPrefix, character-method  
(Ontology-class), 8
- olsPrefix, Ontologies-method  
(Ontology-class), 8
- olsPrefix, Ontology-method  
(Ontology-class), 8
- olsRoot (Ontology-class), 8
- olsRoot, character-method  
(Ontology-class), 8
- olsRoot, Ontologies-method  
(Ontology-class), 8
- olsRoot, Ontology-method  
(Ontology-class), 8
- olsRows (OlsSearch-class), 7
- olsRows<- (OlsSearch-class), 7
- OlsSearch, 4
- OlsSearch (OlsSearch-class), 7
- olsSearch (OlsSearch-class), 7
- OlsSearch-class, 7
- olsStatus (Ontology-class), 8
- olsStatus, character-method  
(Ontology-class), 8
- olsStatus, Ontologies-method  
(Ontology-class), 8
- olsStatus, Ontology-method  
(Ontology-class), 8
- olsTitle (Ontology-class), 8
- olsTitle, character-method  
(Ontology-class), 8
- olsTitle, Ontologies-method  
(Ontology-class), 8
- olsTitle, Ontology-method  
(Ontology-class), 8
- olsUpdated (Ontology-class), 8
- olsUpdated, character-method  
(Ontology-class), 8
- olsUpdated, Ontologies-method  
(Ontology-class), 8
- olsUpdated, Ontology-method  
(Ontology-class), 8
- olsVersion (Ontology-class), 8
- olsVersion, character-method  
(Ontology-class), 8
- olsVersion, Ontologies-method  
(Ontology-class), 8
- olsVersion, Ontology-method  
(Ontology-class), 8
- Ontologies, 4
- Ontologies (Ontology-class), 8
- Ontologies, missing-method  
(Ontology-class), 8
- Ontologies, numeric-method  
(Ontology-class), 8
- Ontologies-class (Ontology-class), 8
- Ontology, 11, 12
- Ontology (Ontology-class), 8
- Ontology, character-method  
(Ontology-class), 8
- Ontology, Ontology-method  
(Ontology-class), 8
- Ontology-class, 8
- parents (Term-class), 12
- partOf (Term-class), 12
- Properties (Properties-class), 11
- properties (Properties-class), 11
- properties, character-method  
(Properties-class), 11
- properties, Ontology-method  
(Properties-class), 11
- properties, Term-method  
(Properties-class), 11
- properties, Terms-method  
(Properties-class), 11
- Properties-class, 11
- Property (Properties-class), 11
- Property-class (Properties-class), 11
- rep, CVParam-method (CVParam-class), 3
- show, CVParam-method (CVParam-class), 3
- show, OlsSearch-method  
(OlsSearch-class), 7
- show, Ontologies-method  
(Ontology-class), 8
- show, Ontology-method (Ontology-class), 8
- show, Properties-method  
(Properties-class), 11
- show, Property-method  
(Properties-class), 11



show,Term-method (Term-class), [12](#)  
show,Terms-method (Term-class), [12](#)

Term, [11](#)  
Term (Term-class), [12](#)  
term (Term-class), [12](#)  
term,character,character-method  
    (Term-class), [12](#)  
term,Ontology,character-method  
    (Term-class), [12](#)  
Term-class, [12](#)  
termDesc (Term-class), [12](#)  
termDesc,Term-method (Term-class), [12](#)  
termDesc,Terms-method (Term-class), [12](#)  
termId (Term-class), [12](#)  
termId,Term-method (Term-class), [12](#)  
termId,Terms-method (Term-class), [12](#)  
termLabel (Term-class), [12](#)  
termLabel,Term-method (Term-class), [12](#)  
termLabel,Terms-method (Term-class), [12](#)  
termNamespace (Term-class), [12](#)  
termNamespace,Term-method (Term-class),  
    [12](#)  
termNamespace,Terms-method  
    (Term-class), [12](#)  
termOntology (Term-class), [12](#)  
termOntology,Term-method (Term-class),  
    [12](#)  
termOntology,Terms-method (Term-class),  
    [12](#)  
termPrefix (Term-class), [12](#)  
termPrefix,Term-method (Term-class), [12](#)  
termPrefix,Terms-method (Term-class), [12](#)  
Terms, [10](#), [11](#)  
Terms (Term-class), [12](#)  
terms (Term-class), [12](#)  
terms,character-method (Term-class), [12](#)  
terms,Ontology-method (Term-class), [12](#)  
Terms-class (Term-class), [12](#)  
termSynonym (Term-class), [12](#)  
termSynonym,Term-method (Term-class), [12](#)  
termSynonym,Terms-method (Term-class),  
    [12](#)

unique,Terms-method (Term-class), [12](#)

Versioned, [4](#)  
Versions, [4](#)