

Package ‘eurlex’

September 11, 2020

Type Package

Title Retrieve Data on European Union Law

Version 0.3.1

Description Access to data on European Union laws and court decisions made easy with pre-defined 'SPARQL' queries and 'GET' requests.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports magrittr, dplyr, xml2, tidyr, httr, rvest, rlang, stringr,
readr, pdfutils, antiword

RoxygenNote 7.1.0

Suggests knitr, rmarkdown, tidytext, wordcloud, purrr, ggplot2, glue

URL <https://michalovadek.github.io/eurlex/>

VignetteBuilder knitr

NeedsCompilation no

Author Michal Ovadek [aut, cre, cph] (<<https://orcid.org/0000-0002-2552-2580>>)

Maintainer Michal Ovadek <michal.ovadek@gmail.com>

Repository CRAN

Date/Publication 2020-09-11 05:30:03 UTC

R topics documented:

elx_council_votes	2
elx_curia_list	2
elx_fetch_data	3
elx_label_eurovoc	3
elx_make_query	4
elx_parse_xml	5
elx_run_query	6

Index	7
--------------	----------

elx_council_votes	<i>Retrieve Council votes on EU acts</i>
-------------------	--

Description

Executes a SPARQL query to the Council's endpoint.

Usage

```
elx_council_votes()
```

Examples

```
votes <- elx_council_votes()
```

elx_curia_list	<i>Scrape list of court cases from Curia</i>
----------------	--

Description

Harvests data from lists of EU court cases from curia.europa.eu. CELEX identifiers are extracted from hyperlinks where available.

Usage

```
elx_curia_list(data = c("all", "ecj_old", "ecj_new", "gc_all", "cst_all"))
```

Arguments

data	Data to be scraped from four separate lists of cases maintained by Curia, defaults to "all" which contains cases from Court of Justice, General Court and Civil Service Tribunal.
------	---

Examples

```
elx_curia_list(data = "cst_all")
```

elx_fetch_data	<i>Retrieve additional data on EU documents</i>
----------------	---

Description

Wraps `http::GET` with pre-specified headers to retrieve data.

Usage

```
elx_fetch_data(
  url,
  type = c("title", "text", "ids"),
  language_1 = "en",
  language_2 = "fr",
  language_3 = "de",
  include_breaks = TRUE
)
```

Arguments

<code>url</code>	A valid url based on a resource identifier such as CELEX or Cellar URI.
<code>type</code>	The type of data to be retrieved. When <code>type = "text"</code> , the returned list contains named elements reflecting the source of each text.
<code>language_1</code>	The priority language in which the data will be attempted to be retrieved, in ISO 639 2-char code
<code>language_2</code>	If data not available in <code>'language_1'</code> , try <code>'language_2'</code>
<code>language_3</code>	If data not available in <code>'language_2'</code> , try <code>'language_3'</code>
<code>include_breaks</code>	If <code>TRUE</code> , text includes tags showing where pages (" <code>—pagebreak—</code> ", for pdfs) and documents (" <code>—documentbreak—</code> ") were concatenated

Examples

```
elx_fetch_data(url = "http://publications.europa.eu/resource/celex/32014R0001", type = "title")
```

elx_label_eurovoc	<i>Label EuroVoc concepts</i>
-------------------	-------------------------------

Description

Create a look-up table with labels for EuroVoc concept URIs. Only unique identifiers are returned.

Usage

```
elx_label_eurovoc(uri_eurovoc = "", alt_labels = FALSE, language = "en")
```

Arguments

uri_eurovoc	Character vector with valid EuroVoc URIs
alt_labels	If 'TRUE', results include comma-separated alternative labels in addition to the preferred label
language	Language in which to return the labels, in ISO 639 2-char code

Examples

```
elx_label_eurovoc(uri_eurovoc = "http://eurovoc.europa.eu/5760", alt_labels = TRUE, language = "fr")
elx_label_eurovoc(uri_eurovoc = c("http://eurovoc.europa.eu/5760", "http://eurovoc.europa.eu/576"))
```

elx_make_query

Create SPARQL queries

Description

Generates pre-defined or manual SPARQL queries to retrieve document ids from Cellar. List of available resource types: <http://publications.europa.eu/resource/authority/resource-type> . Note that not all resource types are compatible with the pre-defined query.

Usage

```
elx_make_query(
  resource_type = c("directive", "regulation", "decision", "recommendation", "intagr",
    "caselaw", "manual", "proposal", "national_impl"),
  manual_type = "",
  include_corrigenda = FALSE,
  include_celex = TRUE,
  include_lbs = FALSE,
  include_date = FALSE,
  include_date_force = FALSE,
  include_date_endvalid = FALSE,
  include_date_transpos = FALSE,
  include_force = FALSE,
  include_eurovoc = FALSE,
  order = FALSE,
  limit = NULL
)
```

Arguments

resource_type	Type of resource to be retrieved via SPARQL query
manual_type	Define manually the type of resource to be retrieved
include_corrigenda	If 'TRUE', results include corrigenda
include_celex	If 'TRUE', results include CELEX identifier for each resource URI
include_lbs	If 'TRUE', results include legal bases of legislation
include_date	If 'TRUE', results include document date
include_date_force	If 'TRUE', results include date of entry into force
include_date_endvalid	If 'TRUE', results include date of end of validity
include_date_transpos	If 'TRUE', results include date of transposition deadline for directives
include_force	If 'TRUE', results include whether legislation is in force
include_eurovoc	If 'TRUE', results include EuroVoc descriptors of subject matter
order	Order results by ids
limit	Limit the number of results, for testing purposes mainly

Examples

```

elx_make_query(resource_type = "directive", include_date = TRUE, include_force = TRUE)
elx_make_query(resource_type = "regulation", include_corrigenda = TRUE, order = TRUE)
elx_make_query(resource_type = "caselaw")
elx_make_query(resource_type = "manual", manual_type = "SWD")

```

elx_parse_xml

Parse RDF/XML triplets to data frame

Description

An internal function to parse RDF/XML output from SPARQL queries.

Usage

```
elx_parse_xml(sparql_response = "")
```

Arguments

sparql_response	A valid response from the SPARQL endpoint
-----------------	---

elx_run_query	<i>Execute SPARQL queries</i>
---------------	-------------------------------

Description

Executes cURL request to a pre-defined endpoint of the EU Publications Office. Relies on `elx_make_query` to generate valid SPARQL queries

Usage

```
elx_run_query(  
  query = "",  
  endpoint = "http://publications.europa.eu/webapi/rdf/sparql"  
)
```

Arguments

query	A valid SPARQL query specified by 'elx_make_query' or manually
endpoint	SPARQL endpoint

Examples

```
elx_run_query(elx_make_query("directive", include_force = TRUE))
```

Index

`elx_council_votes`, [2](#)
`elx_curia_list`, [2](#)
`elx_fetch_data`, [3](#)
`elx_label_eurovoc`, [3](#)
`elx_make_query`, [4](#)
`elx_parse_xml`, [5](#)
`elx_run_query`, [6](#)