Package ‘fhircrackr’

June 17, 2021

Type Package
Title Handling HL7 FHIR Resources in R
Version 1.0.1
Date 2021-06-16
Description Useful tools for conveniently downloading FHIR resources in xml format and converting them to R data frames. The package uses FHIR-search to download bundles from a FHIR server, provides functions to save and read xml-files containing such bundles and allows flattening the bundles to data.frames using XPath expressions.

BugReports https://github.com/POLAR-fhiR/fhircrackr/issues
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Imports xml2, stringr, httr, utils, dplyr, plyr, data.table, methods
Suggests knitr, rmarkdown
VignetteBuilder knitr
Depends R (>= 4.0.0)
Collate 'design.R' 'fhir_url.R' 'fhir_bundle.R' 'fhir_bundle_list.R'
 'download_resources.R' 'fhir_body.R' 'fhir_xpath_expression.R'
 'fhir_columns.R' 'fhir_style.R' 'fhir_resource_type.R'
 'fhir_table_description.R' 'fhir_design.R' 'fhir_table_list.R'
 'flatten_resources.R' 'miscellaneous.R' 'multiple_entries.R'

NeedsCompilation no
Author Thomas Peschel [aut, cre],
Julia Palm [aut] (<https://orcid.org/0000-0003-1568-5893>),
Jens Przybilla [aut],
Frank Meineke [aut] (<https://orcid.org/0000-0002-9256-7543>)
Maintainer Thomas Peschel <tpeschel@imise.uni-leipzig.de>
Repository CRAN
Date/Publication 2021-06-17 10:10:07 UTC
R topics documented:

- `example_bundles1` .............................................. 3
- `fhir_authenticate` ............................................. 7
- `fhir_body` .......................................................... 8
- `fhir_body-class` ................................................... 9
- `fhir_bundle-class` ............................................... 9
- `fhir_bundle_list-class` ........................................... 9
- `fhir_bundle_serialized-class` ................................. 10
- `fhir_bundle_xml` .................................................. 10
- `fhir_bundle_xml-class` ......................................... 10
- `fhir_canonical_design` .......................................... 11
- `fhir_capability_statement` ...................................... 11
- `fhir_columns` ...................................................... 13
- `fhir_columns-class` ................................................ 14
- `fhir_common_columns` ............................................. 14
- `fhir_crack` .......................................................... 15
- `fhir_current_request` ............................................. 18
- `fhir_design` ........................................................ 19
- `fhir_design-class` ................................................. 22
- `fhir_df_list-class` ................................................. 22
- `fhir_dt_list-class` ............................................... 23
- `fhir_load` ........................................................... 23
- `fhir_load_design` .................................................. 24
- `fhir_melt` ............................................................ 25
- `fhir_next_bundle_url` ............................................ 26
- `fhir_resource_type` ............................................... 27
- `fhir_resource_type-class` .................................. 28
- `fhir_rm_indices` .................................................. 28
- `fhir_save` ............................................................. 29
- `fhir_save_design` ................................................. 30
- `fhir_search` ......................................................... 31
- `fhir_serialize` ...................................................... 33
- `fhir_style` .......................................................... 34
- `fhir_style-class` ................................................... 35
- `fhir_table_description` ....................................... 36
- `fhir_table_description-class` ............................... 38
- `fhir_unserialize` .................................................. 39
- `fhir_url` ............................................................. 40
- `fhir_url-class` ..................................................... 41
- `fhir_xpath_expression` ........................................... 42
- `fhir_xpath_expression-class` .................................. 42
- `medication_bundles` ............................................ 43
- `paste_paths` ........................................................ 44

Index 45
example_bundles1 Toy example bundles for multiple entries

Description

These data examples are bundles that contain very few very simple Patient resources that have multiple entries and can be used for demonstration purposes. See Source for how the xml versions look.

Usage

example_bundles1
example_bundles2
example_bundles3

Format

An object of class fhir_bundle_list of length 1.
An object of class fhir_bundle_list of length 1.
An object of class fhir_bundle_list of length 1.

Details

example_bundles1 contains 1 bundle with 2 Patient resources.
example_bundles2 contains 1 bundle with 3 Patient resources.
example_bundles3 contains 1 bundle with 3 Patient resources and 1 Observation resource.

Source

example_bundles1

<Bundle>
  <Patient>
    <id value='id1'/>
    <address>
      <use value='home'/>
      <city value='Amsterdam'/>
      <type value='physical'/>
      <country value='Netherlands'/>
    </address>
    <name>
      <given value='Marie'/>
    </name>
  </Patient>
</Bundle>
<Patient>
  <id value='id3'/>
  <address>
    <use value='home'/>
    <city value='Berlin'/>
  </address>
  <address>
    <type value='postal'/>
    <country value='France'/>
  </address>
  <address>
    <use value='work'/>
    <city value='London'/>
    <type value='postal'/>
    <country value='England'/>
  </address>
  <name>
    <given value='Frank'/>
  </name>
  <name>
    <given value='Max'/>
  </name>
</Patient>
</Bundle>

displayed_bundles2

<Bundle>
  <Patient>
    <id value='id1'/>
    <address>
      <use value='home'/>
      <city value='Amsterdam'/>
      <type value='physical'/>
      <country value='Netherlands'/>
    </address>
    <given value='Marie'/>
  </name>
</Patient>

  <Patient>
    <id value='id2'/>
    <address>
      <use value='home'/>
      <city value='Rome'/>
      <type value='physical'/>
      <country value='Italy'/>
    </address>
  </Patient>
</Bundle>
<address>
<use value='work' />
<city value='Stockholm' />
<type value='postal' />
<country value='Sweden' />
</address>

<name>
<given value='Susie' />
</name>
</Patient>

<Patient>
<id value='id3' />
<address>
<use value='home' />
<city value='Berlin' />
</address>
<address>
<type value='postal' />
<country value='France' />
</address>
<address>
<use value='work' />
<city value='London' />
<type value='postal' />
<country value='England' />
</address>

<name>
<given value='Frank' />
</name>
<name>
<given value='Max' />
</name>
</Patient>

</Bundle>

document 3

<Patient>
<id value='id1' />
<address>
<use value='home' />
<city value='Amsterdam' />
<type value='physical' />
<country value='Netherlands' />
</address>
</Patient>

</Bundle>
<name>
<given value='Marie'/>
</name>  
</Patient>  

<Patient>
<id value='id2'/>
<address>
<use value='home'/>
<city value='Rome'/>
<type value='physical'/>
<country value='Italy'/>
</address>
<address>
<use value='work'/>
<city value='Stockholm'/>
<type value='postal'/>
<country value='Sweden'/>
</address>
<name>
<given value='Susie'/>
</name>  
</Patient>  

<Patient>
<id value='id3'/>
<address>
<use value='home'/>
<city value='Berlin'/>
</address>
<address>
<type value='postal'/>
<country value='France'/>
</address>
<address>
<use value='work'/>
<city value='London'/>
<type value='postal'/>
<country value='England'/>
</address>
<name>
<given value='Frank'/>
</name>  
<name>
<given value='Max'/>
</name>  
</Patient>
<Observation>
  <id value='1'/>
  <code>
    <coding>
      <system value='http://loinc.org'/>
      <code value='29463-7'/>
      <display value='Body Weight'/>
    </coding>
    <coding>
      <system value='http://snomed.info/sct'/>
      <code value='27113001'/>
      <display value='Body weight'/>
    </coding>
  </code>
</Observation>
</Bundle>"

Examples

#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles1)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(example_bundles2)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles2)


---

**fhir_authenticate**

Create token for Authentication

**Description**

This function is a wrapper to create an `http::Token` object for authentication with OAuth2/OpenID Connect. Internally, it calls `http::oauth_app()`, `http::oauth_endpoint()` and `http::oauth2.0_token()` to create a token that can then be used in `fhir_search`.

**Usage**

```r
fhir_authenticate(
  secret,
  key,
  base_url,
  access,
  authorize,
  query_authorize_extra = list()
)
```
Arguments

| secret | The consumer/client secret, belonging to key. |
| key    | Consumer key, also called client ID. For Keycloak this would for instance be the Keycloak client, e.g. "postman". |
| base_url | The URL the user will be redirected to after authorization is complete. This will usually be the base url of you FHIR server. |
| access | The url used to exchange unauthenticated for authenticated token. This can be identical to authorize. |
| authorize | The url to send the client for authorization. |
| query_authorize_extra | A named list holding query parameters to append to initial auth page query. Could hold info about user identity and scope for keycloak like this: |

```r
list(scope = "openid",
     grant_type = "password",
     username = "fhir-user",
     password = "fhirtest")
```

---

**fhir_body**

Create *fhir_body* object

Description

Create *fhir_body* object

Usage

```r
fhir_body(content, type)
```

## S4 method for signature 'list,missing'
```r
default(fhir_body)
```

## S4 method for signature 'list,character'
```r
default(fhir_body)
```

## S4 method for signature 'character,character'
```r
default(fhir_body)
```

Arguments

| content | A character vector of length one representing the body for the post in the format specified in type. If you provide a named list here, it will be taken as key value pairs of FHIR search parameters and will be concatenated appropriately. In this case the type will automatically be set to "application/x-www-form-urlencoded". See examples. |
| type | A string defining the type of the body e.g. "application/x-www-form-urlencoded" or "xml". |
**Value**

An object of type `fhir_body`.

**Examples**

```r
#body that could be used in a FHIR search request POSTed to an URL like baseurl/Patient/_search
fhir_body(content = "gender=female&_summary=count", type="application/x-www-form-urlencoded")
fhir_body(content = list("gender" = "female", "_summary" = "count"))
```

**Description**

Objects of this class should always be created with a call to the function `fhir_body()`.

**Slots**

- `content` A vector of length one representing the body for the post.
- `type` A vector of length one defining the type of the body e.g. "application/x-www-form-urlencoded" or "xml".

---

**fhir_bundle-class**

An S4 class to represent FHIR bundles

**Description**

An S4 class to represent FHIR bundles

---

**fhir_bundle_list-class**

S4 class to represent a list of FHIR bundles

**Description**

A `fhir_bundle_list` is a list of `fhir_bundle_xml` or `fhir_bundle Serialized` objects. It should not be created by the user but returned by a call to `fhir_search()`.
**fhir_bundle_serialized-class**

An S4 class to represent a FHIR bundle in serialized form

**Description**

A `fhir_bundle_serialized` is a `fhir_bundle_xml` that has been serialized using `fhir_serialize()`. In this form, the bundle cannot be used in any meaningful way, but it can be saved and loaded as an `.RData` or `.rds` object without breaking the external pointers in the xml. See `?fhir_serialize` and `?fhir_unserialize`.

**fhir_bundle_xml**

Create `fhir_bundle_xml` object

**Description**

This should only be used if you want to create small examples. Usually, a `fhir_bundle_xml` will be returned by `fhir_search()`.

**Usage**

`fhir_bundle_xml(bundle)`

**Arguments**

- `bundle` A xml-object representing a FHIR bundle

**Examples**

`fhir_bundle_xml(bundle = xml2::xml_unserialize(patient_bundles[[1]]))`

**fhir_bundle_xml-class**

An S4 class to represent a FHIR bundle in xml form

**Description**

A `fhir_bundle_xml` is an xml representation of a FHIR bundle (https://www.hl7.org/fhir/bundle.html). It is usually found inside a `fhir_bundle_list` which is returned by a call to `fhir_search()`.

**Slots**

- `next_link` A `fhir_url` pointing to the next bundle on the server.
- `self_link` A `fhir_url` pointing to this bundle on the server.
fhir_canonical_design  Retrieve design of last call to fhir_crack

Description

Returns the fhir_design of the last call to fhir_crack().

Usage

fhir_canonical_design()

See Also

fhir_design(), fhir_table_description()

Examples

#load example bundles
bundles <- fhir_unserialize(bundles = patient_bundles)

#define design
patients <- fhir_table_description(resource = "Patient")
design <- fhir_design(patients)

result <- fhir_crack(bundles = bundles, design = design)
fhir_canonical_design()

fhir_capability_statement  Get capability statement

Description

Get the capability statement of a FHIR server.

This function downloads a capability statement and creates three data.frames from it:

- Meta contains general information on the server
- Rest contains information on the Rest operations the server supports
- Resources contains information on the supported resource types

When there is more than one piece of information regarding a variable in these data.frames, they are divided by the string specified in sep. If brackets is not NULL, those entries will also be assigned indices so you can melt them using fhir_melt().
Usage

fhir_capability_statement(
  url = "https://hapi.fhir.org/baseR4",
  username = NULL,
  password = NULL,
  token = NULL,
  brackets = NULL,
  sep = " || ",
  log_errors = NULL,
  verbose = 2
)

Arguments

url The base URL of the FHIR server.
username A character vector of length one containing the username for basic authentication. Defaults to NULL, meaning no authentication.
password A character vector of length one containing the password for basic authentication. Defaults to NULL, meaning no authentication.
token A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this.
brackets A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c("<",">"). Defaults to NULL. If NULL, no indices will be added to multiple entries.
sep A character vector of length one to separate pasted multiple entries
log_errors Either NULL or a character vector of length one indicating the name of a file in which to save the http errors. NULL means no error logging. When a file name is provided, the errors are saved in the specified file. Defaults to NULL.
verbose An integer Scalar. If 0, nothing is printed, if 1, only finishing message is printed, if > 1, extraction progress will be printed. Defaults to 2.

Value

A list of data frames containing the information from the statement

Examples

#without indices
cap <- fhir_capability_statement("https://server.fire.ly")

#with indices
cap <- fhir_capability_statement("https://server.fire.ly", brackets = c("[","]"))

#melt searchInclude variable
resources <- fhir_melt(cap$Resources,
columns = "searchInclude",
brackets = c("[", "]"), sep = " || ",
all_columns = TRUE)

#remove indices
resources <- fhir_rm_indices(resources, brackets = c("[", "]"))

head(resources)

---

**Create fhir_columns object**

**Description**

An object of class fhir_columns is part of a fhir_table_description in a fhir_design and holds information on the elements that should be extracted from the FHIR resources, as well as the column names of the resulting dataframe. The elements to be extracted are indicated by XPath xpaths. If no column names are provided, they are generated automatically and reflect the elements position in the resource.

**Usage**

fhir_columns(xpaths, colnames)

## S4 method for signature 'missing,missing'
fhir_columns()

## S4 method for signature 'NULL,missing'
fhir_columns(xpaths)

## S4 method for signature 'character,character'
fhir_columns(xpaths, colnames)

## S4 method for signature 'character,missing'
fhir_columns(xpaths)

## S4 method for signature 'list,missing'
fhir_columns(xpaths)

**Arguments**

- **xpaths** A (named) character vector or (named) list containing xpath xpaths, or a fhir_xpath_expression object.
- **colnames** The names of the columns to create. If no colnames are provided and the list or vector in xpaths has names, those names are taken as the colnames. If no colnames are provided and xpaths is unnamed too, the colnames are generated automatically from the xpath xpaths. See examples.
Examples

# provide colnames explicitly
fhir_columns(xpaths = c("name/given", "code/coding/code"),
  colnames = c("name", "code"))

# colnames are taken from xpaths argument
fhir_columns(xpaths = c(name = "name/given", code = "code/coding/code"))

# colnames are taken from xpaths argument
fhir_columns(xpaths = list(name = "name/given", code = "code/coding/code"))

# colnames are generated automatically
fhir_columns(xpaths = c("name/given", "code/coding/code"))

---

fhir_columns-class  
A S4 class to represent columns in a fhir_table_description

Description

An object of class fhir_columns is part of a fhir_table_description in a fhir_design and holds information on the elements that should be extracted from the FHIR resources, as well as the column names of the resulting data.frame. The elements to be extracted are indicated by XPath xpaths.

Slots

names  The column names

---

fhir_common_columns  
Find common columns

Description

This is a convenience function to find all column names in a data frame starting with the same string that can then be used for fhir_melt().

Usage

fhir_common_columns(data_frame, column_names_prefix)

Arguments

data_frame  A data.frame/data.table with automatically named columns as produced by fhir_crack().
column_names_prefix  A string containing the common prefix of the desired columns.
**Details**

It is intended for use on data frames with column names that have been automatically produced by `fhir_design()`/`fhir_crack()` and follow the form `level1.level2.level3` such as `name.given` or `code.coding.system`. Note that this function will only work on column names following exactly this scheme.

The resulting character vector can be used for melting all columns belonging to the same attribute in an indexed data frame, see `?fhir_melt`.

**Value**

A character vector with the names of all columns matching `column_names_prefix`.

**See Also**

`fhir_melt()`, `fhir_rm_indices()`

**Examples**

```r
# unserialize example bundles
bundles <- fhir_unserialize(bundles = medication_bundles)

# crack Patient Resources
pats <- fhir_table_description(resource = "Patient")

df <- fhir_crack(bundles = bundles, design = pats)

# look at automatically generated names
names(df)

# extract all column names beginning with the string "name"

fhir_common_columns(data_frame = df, column_names_prefix = "name")
```

---

**Description**

Converts a `fhir_bundle_list` (the result of `fhir_search()`) to a list of data.frames/data.tables, i.e. a `fhir_df_list`/`fhir_dt_list` if a `fhir_design` is given in the argument `design`. Creates a single data.frame/data.table, if only a `fhir_table_description` is given in the argument `design`.

**Usage**

```r
fhir_crack(
  bundles, 
  design, 
  sep = NULL, 
  remove_empty_columns = NULL, 
)```
brackets = NULL,
        verbose = 2,
        data.table = FALSE
    )

## S4 method for signature 'ANY,fhir_design'
fhir_crack(
    bundles,
    design,
    sep = NULL,
    remove_empty_columns = NULL,
    brackets = NULL,
    verbose = 2,
    data.table = FALSE
)

## S4 method for signature 'ANY,fhir_table_description'
fhir_crack(
    bundles,
    design,
    sep = NULL,
    remove_empty_columns = NULL,
    brackets = NULL,
    verbose = 2,
    data.table = FALSE
)

## S4 method for signature 'ANY,list'
fhir_crack(
    bundles,
    design,
    sep = NULL,
    remove_empty_columns = NULL,
    brackets = NULL,
    verbose = 2,
    data.table = FALSE
)

### Arguments

#### bundles
A FHIR search result as returned by `fhir_search()`.

#### design
A `fhir_design` or `fhir_table_description` object. See `fhir_design()`/`fhir_table_description()` and the corresponding vignette (`vignette("flattenResources", package = "fhircrackr")`) for a more detailed explanation and comprehensive examples of both.

#### sep
Optional. A character vector of length ones to separate pasted multiple entries which will overwrite the `sep` defined in `design`. If `sep = NULL`, it is looked up in `design`, where the default is " ".

Optional. Remove empty columns? Logical scalar which will overwrite the `rm_empty_cols` defined in design. If `remove_empty_columns = NULL`, it is looked up in design, where the default is FALSE.

Optional. A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. `c("<",">")`, which will overwrite the brackets defined in design. If `brackets = NULL`, it is looked up in design, where the default is `character(0)`, i.e. no indices are added to multiple entries. Empty strings ("") are not allowed.

An integer vector of length one. If 0, nothing is printed, if 1, only finishing message is printed, if > 1, extraction progress will be printed. Defaults to 2.

A logical vector of length one. If it is set to TRUE the `fhir_crack`-function returns a data.table, otherwise a data.frame. Defaults to FALSE.

If a `fhir_design` was used, the result is a list of data.frames, i.e. a `fhir_df_list` object, or a list of data.tables, i.e. a `fhir_dt_list` object. If a `fhir_table_description` was used, the result is a single data.frame/data.table.

- Downloading bundles from a FHIR server: `fhir_search()`
- Creating designs/table_descriptions: `fhir_table_description()` and `fhir_design()`
- Dealing with multiple entries: `fhir_melt()`, `fhir_rm_indices()`

```r
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

###Example 1###
#Extract just one resource type

#define attributes to extract
medications <- fhir_table_description(
  resource = "MedicationStatement",
  cols = c(
    MS.ID = "id",
    STATUS.TEXT = "text/status",
    STATUS = "status",
    MEDICATION.SYSTEM = "medicationCodeableConcept/coding/system",
    MEDICATION.CODE = "medicationCodeableConcept/coding/code",
    MEDICATION.DISPLAY = "medicationCodeableConcept/coding/display",
    DOSAGE = "dosage/text",
    PATIENT = "subject/reference",
    LAST.UPDATE = "meta/lastUpdated"
  ),
  style = fhir_style(
  
  
  )
)
```
```r
sep = " ",
  brackets = c("[", "]"),
  rm_empty_cols= FALSE
)

med_df <- fhir_crack(bundles = bundles, design = medications)
head(med_df) #data.frame

###Example 2###
#extract more resource types
patients <- fhir_table_description(
  resource = "Patient"
)

design <- fhir_design(medications, patients)
df_list <- fhir_crack(bundles = bundles, design = design)
#list of data.frames/fhir_df_list
head(df_list$medications)
head(df_list$patients)

#The design that was used can be extracted from a fhir_df_list
fhir_design(df_list)
```

---

**fhir_current_request**  
Return FHIR search request used in last call to `fhir_search()` or `fhir_url()`

### Description ###

Return FHIR search request used in last call to `fhir_search()` or `fhir_url()`

### Usage ###

```r
fhir_current_request()
```

### Value ###

An object of class `fhir_url()`
**fhir_design**

**Examples**

```r
text <- fhir_url(url = "https://server.fire.ly", resource = "Patient")
text

fhir_search("https://server.fire.ly/Medication", max_bundles = 1)
text
```

---

**Description**

A `fhir_design` is a named list of `fhir_table_description` objects (See `fhir_table_description()`) and should be created using the function described here. The design is used in `fhir_crack()` to tell the function how to flatten each resource type.

**Usage**

```r
fhir_design("")
```

**Arguments**

... One or more `fhir_table_description` objects or a named list containing `fhir_table_description` objects, or an object of class `fhir_df_list/fhir_dt_list`. See `fhir_table_description()`.

**Details**

A `fhir_design` looks for example like this:

A `fhir_design` with 2 table descriptions:

```
Name: Patients
Resource type: Patient
```
Columns:
column name | xpath expression
------------------------
name | name/family
gender | gender
id | id

Style:
sep: ||
brackets: '[]'
rm_empty_cols: FALSE

Name: MedicationAdministrations

Resource type: MedicationAdministration

Columns:
An empty fhir_columns object

Style:
sep: ','
brackets: character(0)
rm_empty_cols: TRUE

The names of the table_descriptions are taken from the names of the arguments. If the table_descriptions are created within the call to fhir_design and therefore have no names, the names will be created from the respective resource type. See examples.

For backwards compatibility it is for the moment also possible to build it from an old-style design as used in fhircrackr (< 1.0.0). See examples.

If this function is given an object of class fhir_df_list or fhir_dt_list, it will extract the design that was used to create the respective list.

See Also

fhir_table_description(), fhir_crack()

Examples

####Example 1####

```r
# create fhir_table_descriptions first
# see ?fhir_table_description for explanation
pat <- fhir_table_description(resource = "Patient",
cols = c(name = "name/family",
        gender = "gender",
        id = "id"),
```
```r
style = fhir_style(sep = "||", 
    brackets = c("[", "]"), 
    rm_empty_cols = FALSE 
)

meds <- fhir_table_description(resource = "MedicationAdministration")

### create design

# First option: Explicitly define names

design1 <- fhir_design(Pats = pat, Medics = meds)
print(design1)

# Second option: Names are taken from the object names

design2 <- fhir_design(pat, meds)
print(design2)

# Third option: Create table_description within fhir_design

design3 <- fhir_design(fhir_table_description(resource = "MedicationAdministration"))
print(design3)

# Fourth option: Names are taken from named list

design3 <- fhir_design(list(Patients = pat, Medications = meds))
print(design3)

#### Example 2 ####

# This option will be deprecated at some point

# Old style design

old_design <- list(
    Patients = list(
        resource = "//Patient",
        cols = list(
            name = "name/family",
            gender = "gender",
            id = "id"),
        style = list(
            sep = "||",
            brackets = c("[", "]"),
            rm_empty_cols = FALSE 
        )
    ),
    Medications = list(
        resource = "//Medication"
    )
)
new_design <- fhir_design(old_design)
print(new_design)

###Example 3###
###Extract design from fhir_df_list/fhir_dt_list

#unserialize and crack example bundles
med_bundles <- fhir_unserialize(bundles = medication_bundles)
dfs <- fhir_crack(bundles = med_bundles, design = design)

#extract design
fhir_design(dfs)

fhir_design-class  
A S4 class containing a design for fhir_crack()

Description
A fhir_design is a named list of fhir_table_description objects. Each table_description contains information on how to flatten one resource type which will result in one data.frame. The fhir_design is passed to the function fhir_crack() along with a list of bundles containing FHIR resources.

Slots
.Data  The list of fhir_table_description objects.
.names  The names of the table_descriptions. Those will also be the names of the resulting data.frames.

See Also
fhir_table_description(), fhir_crack()

fhir_df_list-class  
List of data.frames as returned by fhir_crack()

Description
Objects of this class are returned by fhir_crack() when data.table=FALSE (the default). They behave like an ordinary named list of data.frames but have some additional information in the slot design.

Slots
.names  Character vector containing the names of the data.frames.
design  An object of class fhir_design that was used to create the df_list.
**fhir_dt_list-class**  
*List of data.tables as returned by fhir_crack()*

**Description**

Objects of this class are returned by `fhir_crack()` when `data.table=TRUE`. They behave like an ordinary named list of data.tables but have some additional information in the slot `design`.

**Slots**

- `names` A character vector containing the names of the data.tables.
- `design` An object of class `fhir_design` that was used to create the `dt_list`.

**fhir_load**  
*Load bundles from xml-files*

**Description**

Reads all bundles stored as xml files from a directory.

**Usage**

`fhir_load(directory)`

**Arguments**

- `directory` A character vector of length one containing the path to the folder were the files are stored.

**Value**

A `fhir_bundle_list`.

**Examples**

```r
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

#save to temporary directory
fhir_save(bundles, directory = tempdir())

#load from temporary directory
loaded_bundles <- fhir_load(tempdir())
```
fhir_load_design  

Load design from xml

Description
Loads a fhir_design for use with fhir_crack() from an xml file into R.

Usage
fhir_load_design(file)

Arguments
file  
A string specifying the file from which to read.

Value
A fhir_design object. See ?fhir_design.

See Also
fhir_design(), fhir_table_description(), fhir_save_design()

Examples

# create and save design
df_desc1 <- fhir_table_description(resource = "Patient",
cols = c(name = "name/family",
gender = "gender",
id = "id"),
style = fhir_style(sep = "||",
brackets = c("[", "]"),
rm_empty_cols = FALSE
)

df_desc2 <- fhir_table_description(resource = "Observation",
cols = c("code/coding/system", "code/coding/code")
)

# create design
design <- fhir_design(Patients = df_desc1, Observations = df_desc2)
temp <- tempfile()
fhir_save_design(design = design, file = temp)
design <- fhir_load_design(file = temp)
fhir_melt

Melt multiple entries

Description

This function divides multiple entries in an indexed data frame as produced by `fhir_crack()` into separate rows.

Usage

```r
fhir_melt(
  indexed_data_frame,
  columns,
  brackets = c("<", ">"),
  sep = " ",
  id_name = "resource_identifier",
  all_columns = FALSE
)
```

Arguments

- `indexed_data_frame`: A data.frame/data.table with indexed multiple entries.
- `columns`: A character vector specifying the names of all columns that should be molten simultaneously. It is advisable to only melt columns simultaneously that belong to the same (repeating) attribute!
- `brackets`: A character vector of length two, defining the brackets used for the indices.
- `sep`: A character vector of length one defining the separator that was used when pasting together multiple entries in `fhir_crack()`.
- `id_name`: A character vector of length one, the name of the column that will hold the identification of the origin of the new rows.
- `all_columns`: Return all columns? Defaults to `FALSE`, meaning only those specified in `columns` are returned.

Details

Every row containing values that consist of multiple entries on the variables specified by the argument `columns` will be turned into multiple rows, one for each entry. Values on other variables will be repeated in all the new rows.

The new data.frame will contain only the molten variables (if `all_columns = FALSE`) or all variables (if `all_columns = TRUE`) as well as an additional variable `resource_identifier` that maps which rows came from the same origin. The name of this column can be changed in the argument `id_name`.

For a more detailed description on how to use this function please see the corresponding package vignette.
Value

A data.frame/data.table where each entry from the variables in columns appears in a separate row.

See Also

fhir_common_columns(), fhir_rm_indices()

Examples

#unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

#crack fhir resources
table_desc <- fhir_table_description(resource = "Patient",
                               style = fhir_style(brackets = c("[","]"),
                                              sep = " "))

df <- fhir_crack(bundles = bundles, design = table_desc)

#find all column names associated with attribute address
col_names <- fhir_common_columns(df, "address")

#original data frame
df

#only keep address columns
fhir_melt(indexed_data_frame = df, columns = col_names,
          brackets = c("[","]"), sep = " ")

#keep all columns
fhir_melt(indexed_data_frame = df, columns = col_names,
          brackets = c("[","]"), sep = " ", all_columns = TRUE)

---

fhir_next_bundle_url  Next Bundle’s URL

Description

fhir_next_bundle_url() gives the link to the next available bundle, either of the bundle you provided in the argument bundle or of the last call to fhir_search(), if bundle=NULL (the default).

This function is useful when you don’t have a lot of memory available or when a download of bundles was interrupted for some reason. In case of small memory, you can use fhir_next_bundle_url together with the max_bundle argument from fhir_search() to download bundles in smaller batches in a loop. See details in the example.

Usage

fhir_next_bundle_url(bundle = NULL)
**Arguments**

**bundle**
The bundle from which you wish to extract the next link. If this is NULL (the default), the function will extract the next link from the last bundle that was downloaded in the most recent call to `fhir_search()`.

**Value**

A `fhir_url` object referencing next bundle available on the FHIR server. Empty `fhir_url` / character vector, if no further bundle is available.

**Examples**

```r
# workflow for small memory environments, downloading small batches of bundles
# for really small memory environments consider also using the `\_count` option in
# your FHIR search request.
# You can iteratively download, crack and save the bundles until all bundles are processed or the
# desired number of bundles is reached.
url <- fhir_url("https://server.fire.ly/Patient")
count <- 0
obs <- fhir_table_description(resource = "Patient")
design <- fhir_design(obs)
while(length(url)>0 & count < 5){
bundles <- fhir_search(url, max_bundles = 2)
tables <- fhir_crack(bundles, design)
save(tables, file = paste0(tempdir(),"/table_", count, ".RData"))
count <- count + 1
url <- fhir_next_bundle_url()
}
#you can see the saved tables here:
dir(tempdir())
```
Arguments

string  A length one character vector containing the resource type. Will usually be one of the official FHIR resource types listed at https://hl7.org/FHIR/resourcelist.html

fix_capitalization  Correct wrong capitalization for known resource types? E.g. patients -> Patients or medicationstatement -> MedicationStatement. Defaults to TRUE.

Value

An fhir_resource_type object

Examples

fhir_resource_type(string = "Patient")
fhir_resource_type(string = "medicationadministration")

fhir_resource_type-class

A representation of a FHIR resource type

Description

An object of class fhir_resource_type is a string containing a FHIR resource type. It is part of a fhir_table_description which in turn is part of a fhir_design and used in fhir_crack().

fhir_rm_indices  Remove indices from data.frame/data.table

Description

Removes the indices in front of multiple entries as produced by fhir_crack() when brackets are provided in the design.

Usage

fhir_rm_indices(
  indexed_data_frame,  
  brackets = c("<", ">"),  
  columns = names(indexed_data_frame)
)
**Arguments**

- **indexed_data_frame**  
  A data frame with indices for multiple entries as produced by `fhir_crack()`.

- **brackets**  
  A character vector of length two defining the brackets that were used in `fhir_crack()`.

- **columns**  
  A character vector of column names, indicating from which columns indices should be removed. Defaults to all columns.

**Value**

A data frame without indices.

**See Also**

- `fhir_melt()`

**Examples**

```r
# unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)
patients <- fhir_table_description(resource = "Patient")

df <- fhir_crack(bundles = bundles,  
                 design = patients,  
                 brackets = c("[", "]"))

df_indices_removed <- fhir_rm_indices(indexed_data_frame = df, brackets = c("[", "]"))
```

---

**fhir_save**

`Save FHIR bundles as xml-files`

**Description**

Writes a list of FHIR bundles as numbered xml files into a directory.

**Usage**

`fhir_save(bundles, directory = "result")`

**Arguments**

- **bundles**  
  A list of xml objects representing the FHIR bundles.

- **directory**  
  A character vector of length one containing the path to the folder to store the data in.
Examples

#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

#save to temporary directory
fhir_save(bundles, directory = tempdir())

Description

Writes a `fhir_design` for use with `fhir_crack()` to an xml file.

Usage

fhir_save_design(design, file = "design.xml")

Arguments

design A `fhir_design` object. See `fhir_design()`.
file A string specifying the file to write to, defaults to writing "design.xml" into the current working directory.

See Also

`fhir_design()`, `fhir_table_description()`, `fhir_load_design()`

Examples

#create design
df_desc1 <- fhir_table_description(resource = "Patient",
cols = c(name = "name/family",
gender = "gender",
id = "id"),
style = fhir_style(sep = "||",
brackets = c("[", "]"),
rm_empty_cols = FALSE)
)

df_desc2 <- fhir_table_description(resource = "Observation",
cols = c("code/coding/system", "code/coding/code")
}
design <- fhir_design(Patients = df_desc1, Observations = df_desc2)
fhir_save_design(design = design, file = tempfile())
Description

Downloads all FHIR bundles of a FHIR search request from a FHIR server by iterating through the bundles. Search via GET and POST is possible, see Details.

Usage

```r
fhir_search(
    request = fhir_current_request(),
    body = NULL,
    username = NULL,
    password = NULL,
    token = NULL,
    max_bundles = Inf,
    verbose = 1,
    max_attempts = 5,
    delay_between_attempts = 10,
    log_errors = NULL,
    save_to_disc = NULL,
    delay_between_bundles = 0,
    directory = paste0("FHIR_bundles_", gsub("-| |:|", "", Sys.time())
)
```

Arguments

- **request**: An object of class `fhir_url` or a character vector of length one containing the full FHIR search request. It is recommended to explicitly create the request via `fhir_url()` as this will do some validity checks and format the url properly. Defaults to `fhir_current_request()`.

- **body**: A character vector of length one or object of class `fhir_body` with type "application/x-www-form-urlencoded". A body should be provided when the FHIR search request is too long and might exceed the maximal allowed length of the URL when send to the server. In this case a search via POST (see https://www.hl7.org/fhir/search.html#Introduction) can be used. The body should contain all the parameters that follow after the ? in the FHIR search request. When a body is provided, the required _search is automatically added to the url in request. See examples and ?fhir_body.

- **username**: A character vector of length one containing the username for basic authentication.

- **password**: A character vector of length one containing the password for basic authentication.

- **token**: A character vector of length one or object of class `httr::Token`, for bearer token authentication (e.g. OAuth2). See `fhir_authenticate()` for how to create this.
max_bundles  Maximal number of bundles to get. Defaults to Inf meaning all available bundles are downloaded.
verbose  An integer vector of length one. If 0, nothings is printed, if 1, only finishing message is printed, if > 1, downloading progress will be printed. Defaults to 2.
max_attempts  A numeric vector of length one. The maximal number of attempts to send a request, defaults to 5.
delay_between_attempts  A numeric vector of length one specifying the delay in seconds between two attempts. Defaults to 10.
log_errors  Either NULL or a character vector of length one indicating the name of a file in which to save the http errors. NULL means no error logging. When a file name is provided, the errors are saved in the specified file. Defaults to NULL.
save_to_disc  Either NULL or a character vector of length one indicating the name of a directory in which to save the bundles. If a directory name is provided, the bundles are saved as numerated xml-files into the directory specified and not returned as a bundle list in the R session. This is useful when a lot of bundles are to be downloaded and keeping them all in one R session might overburden working memory. When the download is complete, the bundles can be loaded into R using `fhir_load()`. Defaults to NULL, i.e. bundles are returned as a list within the R session.
delay_between_bundles  A numeric scalar specifying a time in seconds to wait between pages of the search result, i.e. between downloading the current bundle and the next bundle. This can be used to avoid choking a weak server with too many requests to quickly. Defaults to zero.
directory  Deprecated. Please specify the directory directly in the `save_to_disc` argument.

Details

**Request type:**

`fhir_search` allows for two types of search request:

1. FHIR search via GET: This is the more common approach. All information on which resources to download is contained in the URL that is send to the server (`request` argument). This encompasses the base url of the server, the resource type and possible search parameters to further qualify the search (see `fhir_url()`). The search via GET is the default and performed whenever the argument body is `NULL`.

2. FHIR search via POST: This option should only be used when the parameters make the search URL so long the server might deny it because it exceeds the allowed length. In this case the search parameters (everything that would usually follow the resource type after the `?`) can be transferred to a body of type "application/x-www-form-urlencoded" and send via POST. If you provide a body in `fhir_search()`, the url in request should only contain the base URL and the resource type. The function will automatically amend it with `_search` and perform a POST.

**Authentication:**
There are several ways of authentication implemented in `fhir_search()`. If you don’t need any authentication, just leave the arguments described in the following at their default values of `NULL`.

1. Basic Authentication: Provide the username and the password for basic authentication in the respective arguments.
2. Token Authentication: Provide a token in the argument token, either as a character vector of length one or as an object of class `httr::Token`. You can use the function `fhir_authenticate()`

Value

A `fhir_bundle_list` when `save_to_disc = NULL` (the default), else `NULL`.

See Also

- Creating a FHIR search request: `fhir_url()` and `fhir_body()` (for POST based search)
- OAuth2 Authentication: `fhir_authenticate()`
- Saving/reading bundles from disc: `fhir_save()` and `fhir_load()`
- Flattening the bundles: `fhir_crack()`

Examples

```r
# Search with GET
# create fhir search url
request <- fhir_url(url = "https://server.fire.ly",
                   resource = "Patient",
                   parameters = c(gender="female"))

# download bundles
bundles <- fhir_search(request, max_bundles = 5)

# Search with POST (should actually be used for longer requests)
request <- fhir_url(url = "https://server.fire.ly",
                    resource = "Patient")

body <- fhir_body(content = list(gender = "female"))

bundles <- fhir_search(request = request,
                        body = body,
                        max_bundles = 5)
```

---

### `fhir_serialize`

Serialize a `fhir_bundle` or `fhir_bundle_list`

**Description**

Serializes FHIR bundles to allow for saving in `.rda` or `.RData` format without losing integrity of pointers i.e. it turns a `fhir_bundle_xml` object into an `fhir_bundle_serialized` object.
Usage

```r
fhir_serialize(bundles)
```

## S4 method for signature 'fhir_bundle_xml'

```r
fhir_serialize(bundles)
```

## S4 method for signature 'fhir_bundle_serialized'

```r
fhir_serialize(bundles)
```

## S4 method for signature 'fhir_bundle_list'

```r
fhir_serialize(bundles)
```

Arguments

- `bundles`: A `fhir_bundle` or `fhir_bundle_list` object.

Value

A `fhir_bundle_xml` or `fhir_bundle_list` object.

Examples

```r
#example bundles are serialized, unserialize like this:
bundles <- fhir_unserialize(medication_bundles)

#Serialize like this:
bundles_for_saving <- fhir_serialize(bundles)

#works also on single bundles
fhir_serialize(bundles[[1]])
```

---

### fhir_style

Create `fhir_style` object

Description

This function creates an object of class `fhir_style`. It contains the three elements `sep`, `brackets` and `rm_empty_cols`. See Details.

Usage

```r
fhir_style(sep = " ", brackets = character(), rm_empty_cols = FALSE)
```
Arguments

sep: A character vector of length one to separate pasted multiple entries. Defaults to " ".

brackets: A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c( "<", ">" ). If this is empty (i.e. character of length zero, the default) or 'NULL', no indices will be added to multiple entries. If it is a character vector of length one, it will be recycled to length two, i.e. "|" will become c("|", "|" ). Empty strings (""") are not allowed.

rm_empty_cols: A logical vector of length one. Remove empty columns? Defaults to FALSE.

Details

A fhir_style object is part of a fhir_table_description which in turn is part of a fhir_design and ultimately used in fhir_crack(). A fhir_style object contains three elements:

• sep: A string defining the separator used to separate multiple entries for the same element in a FHIR resource, e.g. multiple address/city elements in a Patient resource.

• brackets: A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c( "<", ">" ). If this is empty (i.e. character of length 0, the default), no indices will be added to multiple entries. Empty strings (""") are not allowed.

• rm_empty_cols: A logical scalar defining whether or not to remove empty columns after cracking. Empty columns arise when you try to extract an element that doesn’t appear in any of the resources. A fhir_style object looks for example like this:

  sep: ' '
  brackets: '[ ' ' ]'
  rm_empty_cols: FALSE

Value

A fhir_style object

Examples

  fhir_style(sep = " ",
             brackets = c("[", "]"),
             rm_empty_cols = TRUE)
Slots

sep  A string to separate pasted multiple entries. Defaults to " ".

brackets  A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c("<",">"). If this is empty (i.e. character of length 0, the default), no indices will be added to multiple entries. Empty strings ("") are not allowed.

rm_empty_cols  Logical scalar. Remove empty columns? Defaults to FALSE.

fhir_table_description

Create fhir_table_description object

Description

A fhir_table_description is part of a fhir_design and holds the information fhir_crack() needs to flatten (aka crack) FHIR resources from a FHIR bundle. There should be one fhir_table_description per resource type as fhir_crack() will create one data.frame/data.table per resource type. See Details.

Usage

fhir_table_description(resource, cols = fhir_columns(), style = fhir_style())

Arguments

resource  A character vector of length one or fhir_resource_type object indicating which resource type should be extracted.

cols  Optional. A fhir_columns object or something that can be coerced to one, like a (named) character vector, a (named) list containing xpath expressions, or a fhir_xpath_expression object. See fhir_columns() and the examples. If this argument is omitted, an empty fhir_columns object will be supplied. This means that in the call to fhir_crack(), all available elements are extracted in put in automatically named columns.

style  Optional. A fhir_style object, as created by fhir_style(). If this argument is omitted, default values will be assumed, see fhir_style().

Details

A fhir_table_description consists of the following elements:

- The resource element: Defines the resource type (e.g. Patient or Observation). See ?fhir_resource.
- The cols element: Contains the column names and XPath expressions defining the columns to extract. If this element is empty, fhir_crack() will extract all available elements of the resource and name the columns automatically. See ?fhir_columns.
- The style element: Defines how to deal with multiple entries to the same element and whether empty columns are removed. See ?fhir_style
A full `fhir_table_description` looks for example like this:

```r
fhir_resource_type: Patient

fhir_columns:
  column name | xpath expression
  ------------------------
  name | name/family
  gender | gender
  id | id

fhir_style:
  sep: ||
  brackets: ']['
  rm_empty_cols: FALSE
```

Value

An object of class `fhir_table_description`.

Examples

```r
#named character for cols
fhir_table_description(resource = "Patient",
    cols = c(name = "name/family",
              gender = "gender",
              id = "id"),
    style = fhir_style(sep = "||",
                       brackets = c("[", "]"),
                       rm_empty_cols = FALSE))

#If style is ommitted, default values are assumed
fhir_table_description(resource = "Patient",
    cols = c(name = "name/family",
              gender = "gender",
              id = "id"))

#named list for cols
fhir_table_description(resource = "Patient",
    cols = list(name = "name/family",
                 gender = "gender",
                 id = "id"))

#unnamed character for cols, colnames are generated automatically
fhir_table_description(resource = "Patient",
    cols = c("name/family",
             "gender",
             "id"))
```
A S4 class describing the form of data.frame produced by \texttt{fhir\_crack()}

Description

A \texttt{fhir\_table\_description} is part of a \texttt{fhir\_design} and holds the information \texttt{fhir\_crack()} needs to flatten (aka crack) FHIR resources from a FHIR bundle and is created with \texttt{fhir\_table\_description()}. There should be one \texttt{fhir\_table\_description} per resource type as \texttt{fhir\_crack()} will create one data.frame/data.table per resource type. See Details.

Details

A \texttt{fhir\_table\_description} consists of the following elements:

- The resource element: Defines the resource type (e.g. Patient or Observation). See \texttt{fhir\_resource\_type()}
- The cols element: Contains the column names and XPath expressions defining the columns to extract. If this element is empty, \texttt{fhir\_crack()} will extract all available elements of the resource and name the columns automatically. See \texttt{fhir\_columns()}.
- The style element: Defines how to deal with multiple entries to the same element and whether empty columns are removed. See \texttt{fhir\_style()}.

A full \texttt{fhir\_table\_description} looks for example like this:

\begin{verbatim}
  fhir_resource_type: Patient

  fhir_columns:
  column name  | xpath expression
                  ---------------
  name          | name/family
  gender        | gender
  id            | id

  fhir_style:
  sep: ||
  brackets: [' ' ']
  rm_empty_cols: FALSE
\end{verbatim}

Slots

- \texttt{resource} An object of class \texttt{fhir\_resource\_type} defining the resource type that should be extracted.
- \texttt{cols} An object of class \texttt{fhir\_columns} describing which columns should be created and how. If this is an empty \texttt{fhir\_columns} object, the call to \texttt{fhir\_crack()} will extract all available elements and put them in automatically named columns.
- \texttt{style} An object of class \texttt{fhir\_style} describing how to deal with multiple entries and empty columns.
**fhir_unserialize**

Unserialize a fhir_bundle or fhir_bundle_list

---

**Description**

Unserializes FHIR bundles that have been serialized to allow for saving in .rda or .RData format, i.e. it turns a fhir_bundle_serialized object into an fhir_bundle_xml object.

**Usage**

```r
fhir_unserialize(bundles)
```

## S4 method for signature 'fhir_bundle_xml'

```r
def(fhir_unserialize(bundles))
```

## S4 method for signature 'fhir_bundle_serialized'

```r
def(fhir_unserialize(bundles))
```

## S4 method for signature 'fhir_bundle_list'

```r
def(fhir_unserialize(bundles))
```

**Arguments**

- **bundles**: A fhir_bundle or fhir_bundle_list object.

**Value**

A fhir_bundle_serialized or fhir_bundle_list object.

**Examples**

```r
#unserialize bundle list
fhir_unserialize(patient_bundles)

#unserialize single bundle
fhir_unserialize(patient_bundles[[1]])
```
fhir_url

Create FHIR URL

Description
This function creates an object of class `fhir_url` which mostly represents a URL-encoded URL for a FHIR search request. A valid Search URL contains a base URL and a resource type and may contain additional search parameters. For more info on FHIR search see https://www.hl7.org/fhir/search.html.

Usage

```r
fhir_url(url, resource, parameters, url_enc = TRUE)
```

## S4 method for signature 'character,missing,missing'
```
fhir_url(url, url_enc = TRUE)
```

## S4 method for signature 'character,character,missing'
```
fhir_url(url, resource, url_enc = TRUE)
```

## S4 method for signature 'character,character,character'
```
fhir_url(url, resource, parameters, url_enc = TRUE)
```

## S4 method for signature 'character,character,list'
```
fhir_url(url, resource, parameters, url_enc = TRUE)
```

Arguments

- `url`: A character of length one specifying either the full search request, e.g. "http://hapi.fhir.org/baseR4/Patient?gender=male&_summary=count", or the base URL to the FHIR server, e.g. "http://hapi.fhir.org/baseR4".
- `resource`: A character of length one of `fhir_resource_type` object with the resource type to be searched, e.g. "Patient".
- `parameters`: Optional. Either a length 1 character containing properly formatted FHIR search parameters, e.g. "gender=male&_summary=count" or a named list or named character vector e.g. `list(gender="male","_summary"="count")` or `c(gender="male","_summary"="count")`. Note that parameter names beginning with _ have to be put in quotation marks!
- `url_enc`: Should the url be URL-encoded? Defaults to TRUE.

Details

You can use this function in two ways. If you provide just one string in the argument `url` with the full FHIR search request, this string will be taken as a full FHIR search request. If you also provide the arguments `resource` and/or `parameters`, the string in `url` will be taken as the base url of your FHIR server and the arguments will be concatenated appropriately to form the full request. See examples.

Note that only the latter approach does a validity check on the resource type!

You can disable URL-encoding by setting `url_enc=FALSE`. 
Value

An object of class `fhir_url`

Examples

```r
# provide full FHIR search request
fhir_url(url = "http://hapi.fhir.org/baseR4/Patient?gender=male&_summary=count")

# provide base url and resource type
fhir_url(
  url = "http://hapi.fhir.org/baseR4",
  resource = "Patient"
)

# parameters in one string
fhir_url(
  url = "http://hapi.fhir.org/baseR4",
  resource = "Patient",
  parameters = "gender=male&_summary=count"
)

# parameters as a named character
fhir_url(
  url = "http://hapi.fhir.org/baseR4",
  resource = "Patient",
  parameters = c("gender" = "male", "_summary" = "count")
)

# parameters as a named list
fhir_url(
  url = "http://hapi.fhir.org/baseR4",
  resource = "Patient",
  parameters = list("gender" = "male", "_summary" = "count")
)
```

---

**fhir_url-class**

An S4 object to represent a URL for a FHIR server

**Description**

Objects of this class are basically strings (character vectors of length one) representing a URL. They are usually url encoded. See `fhir_url()` for how to build them.
**fhir_xpath_expression-class**

An S4 class for xpath_expressions Objects of this class are essentially character vectors, but can only be valid XPath (1.0) expressions. They are mostly used in the fhir_columns class.

**Description**

An S4 class for xpath_expressions Objects of this class are essentially character vectors, but can only be valid XPath (1.0) expressions. They are mostly used in the fhir_columns class.

---

**fhir_xpath_expression**  
*Create fhir_xpath_expression*

---

**Description**

This function takes a character vector, checks whether it contains valid XPath (1.0) expressions and returns it as an fhir_xpath_expression object. These objects are used in fhir_parameters objects.

**Usage**

```r
fhir_xpath_expression(expression)
```

**Arguments**

- `expression`  
  A character vector of the XPath expressions

**Value**

A XPath expression object

**Examples**

```r
fhir_xpath_expression(c("//Patient", "name/given"))
```
medication_bundles  Exemplary FHIR bundles

Description
These data examples can be used to explore some of the functions from the fhircrackr package when direct access to a FHIR server is not possible.

All example data sets are fhir_bundle_lists containing fhir_bundle_serialized objects representing FHIR bundles as returned by fhir_search(). They have to be unserialized (once per R session), before you can work with them!

Usage

medication_bundles

patient_bundles

Format
An object of class fhir_bundle_list of length 3.
An object of class fhir_bundle_list of length 2.

Details

medication_bundles contains 3 bundles with MedicationStatement resources representing Medications with Snomed CT code 429374003 and the respective Patient resources that are linked to these MedicationStatements.

patient_bundles contains 2 bundles with Patient resources.

Source
The data sets are generated by the following code:

medication_bundles (Downloaded 10-05-21)

                  
"_include" = "MedicationStatement:subject"))

bundles <- fhir_search(request = search_request, max_bundles = 3)

medication_bundles <- fhir_serialize(bundles = bundles)

patient_bundles (Downloaded 10-05-21)
bundles <- fhir_search(request="https://hapi.fhir.org/baseR4/Patient",
  max_bundles=2,
  verbose = 0)

patient_bundles <- fhir_serialize(bundles = bundles)

Examples

#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = medication_bundles)

#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = patient_bundles)

---

paste_paths

Concatenate paths

Description

Concatenates two strings to a path string correctly.

Usage

paste_paths(path1 = "w", path2 = "d", os = "LiNuX")

Arguments

path1 A a character vector of length one specifying the left hand part of the resulting path.
path2 A a character vector of length one specifying the right hand part of the resulting path.
os A a character vector of length one specifying the operating system you’re operating on: windows or linux.

Value

A a character vector of length one containing the concatenated path.

Examples

paste_paths(path1 = "data", path2 = "patients")
paste_paths(path1 = "/data", path2 = "patients")
paste_paths(path1 = "/data/", path2 = "patients")
paste_paths(path1 = "/data", path2 = "/patients")
paste_paths(path1 = "/data/", path2 = "/patients/"
paste_paths(path1 = "data", path2 = "patients", os = "windows")
Index

* datasets
  example_bundles1, 3
  medication_bundles, 43
  example_bundles1, 3
  example_bundles2 (example_bundles1), 3
  example_bundles3 (example_bundles1), 3
  fhir_authenticate, 7
  fhir_authenticate(), 12, 31, 33
  fhir_body, 8, 8, 9
  fhir_body(), 9, 33
  fhir_body, character, character-method
    (fhir_body), 8
  fhir_body, character, character-methods
    (fhir_body), 8
  fhir_body, list, character-method
    (fhir_body), 8
  fhir_body, list, character-methods
    (fhir_body), 8
  fhir_body, list, missing-method
    (fhir_body), 8
  fhir_body, list, missing-methods
    (fhir_body), 8
  fhir_body-class, 9
  fhir_bundle, 33, 34, 39
  fhir_bundle-class, 9
  fhir_bundle_list, 15, 23, 33, 34, 39, 43
  fhir_bundle_list-class, 9
  fhir_bundle_serialized, 33, 39, 43
  fhir_bundle_serialized-class, 10
  fhir_bundle_xml, 10, 10, 33, 34, 39
  fhir_bundle_xml-class, 10
  fhir_canonical_design, 11
  fhir_capability_statement, 11
  fhir_columns, 13, 13, 36, 38
  fhir_columns(), 36, 38, 39
  fhir_columns, character, character-method
    (fhir_columns), 13
  fhir_columns, character, missing-method
    (fhir_columns), 13
  fhir_columns, list, missing-method
    (fhir_columns), 13
  fhir_columns, missing, missing-method
    (fhir_columns), 13
  fhir_columns, NULL, missing-method
    (fhir_columns), 13
  fhir_columns-class, 14
  fhir_common_columns, 14
  fhir_common_columns(), 26
  fhir_crack, 15
  fhir_crack(), 11, 14, 15, 19, 20, 22–25,
    28–30, 33, 35, 36, 38, 39
  fhir_crack, ANY, fhir_design-method
    (fhir_crack), 15
  fhir_crack, ANY, fhir_table_description-method
    (fhir_crack), 15
  fhir_crack, ANY, list-method
    (fhir_crack), 15
  fhir_crack, fhir_design-method
    (fhir_crack), 15
  fhir_crack, fhir_table_description-method
    (fhir_crack), 15
  fhir_crack, list-method
    (fhir_crack), 15
  fhir_current_request, 18
  fhir_current_request(), 31
  fhir_design, 11, 14–17, 19, 19, 22–24, 30, 35
  fhir_design(), 11, 15–17, 24, 30, 39
  fhir_design, fhir_table_description-method
    (fhir_design), 19
  fhir_design, fhir_table_list-method
    (fhir_design), 19
  fhir_design, list-method
    (fhir_design), 19
  fhir_design-class, 22
  fhir_df_list, 15, 17, 19, 20
  fhir_df_list-class, 22
  fhir_dt_list, 15, 17, 19, 20
fhir_dt_list-class, 23
fhir_load, 23
fhir_load(), 32, 33
fhir_load_design, 24
fhir_load_design(), 30
fhir_melt, 25
fhir_melt(), 11, 14, 15, 17, 29
fhir_next_bundle_url, 26
fhir_resource_type, 27, 27, 28, 36, 38, 40
fhir_resource_type(), 38, 39
fhir_resource_type-class, 28
fhir_rm_indices, 28
fhir_rm_indices(), 15, 17, 26
fhir_save, 29
fhir_save(), 33
fhir_save_design, 30
fhir_save_design(), 24
fhir_search, 7, 31
fhir_search(), 9, 10, 15–18, 26, 27, 43
fhir_serialize, 33
fhir_serialize(), 10
fhir_serialize,fhir_bundle_list-method (fhir_serialize), 33
fhir_serialize,fhir_bundle_serialized-method (fhir_serialize), 33
fhir_serialize,fhir_bundle_xml-method (fhir_serialize), 33
fhir_style, 34, 36, 38
fhir_style(), 36, 38, 39
fhir_style-class, 35
fhir_table_description, 14–17, 22, 35, 36, 36, 37
fhir_table_description(), 11, 16, 17, 19, 20, 22, 24, 30, 38
fhir_table_description-class, 38
fhir_unserialize, 39
fhir_unserialize,fhir_bundle_list-method (fhir_unserialize), 39
fhir_unserialize,fhir_bundle_serialized-method (fhir_unserialize), 39
fhir_unserialize,fhir_bundle_xml-method (fhir_unserialize), 39
fhir_url, 10, 27, 31, 40, 40, 41
fhir_url(), 18, 31–33, 41
fhir_url,character,character,character-method (fhir_url), 40
fhir_url,character,character,list-method (fhir_url), 40
fhir_url,character,character,missing-method (fhir_url), 40
fhir_url,character,missing,missing-method (fhir_url), 40
fhir_url-class, 41
fhir_xpath_expression, 13, 36, 42
fhir_xpath_expression-class, 42
httr::oauth2.0_token(), 7
httr::oauth_app(), 7
httr::oauth_endpoint(), 7
httr::Token, 7, 12, 31, 33
medication_bundles, 43
paste_paths, 44
patient_bundles (medication_bundles), 43