Package 'geosapi'

February 22, 2017

Type Package

Title GeoServer REST API R Interface

Version 0.1-0

Date 2017-02-21

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

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

Description Provides an R interface to the GeoServer REST API, allowing to upload and publish data in a GeoServer web-application and expose data to OGC Web-Services. The package currently supports all CRUD (Create,Read,Update,Delete) operations on GeoServer workspaces, namespaces, datastores (stores of vector data), featuretypes, layers, styles, as well as vector data upload operations. For more information about the GeoServer REST API, see <http://docs.geoserver.org/stable/en/user/rest/>.

Depends R (>= 3.1.0)

Imports R6, openssl, httr, XML

Suggests testthat, roxygen2

License MIT + file LICENSE

URL https://github.com/eblondel/geosapi/wiki, http://geoserver.org/

BugReports https://github.com/eblondel/geosapi/issues

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2017-02-22 11:40:02

R topics documented:

`geosapi` *GeoServer REST API R Interface*

Description

Provides an R interface to the GeoServer REST API, allowing to upload and publish data in a GeoServer web-application and expose data to OGC Web-Services. The package currently supports all CRUD (Create,Read,Update,Delete) operations on GeoServer workspaces, namespaces, datastores (stores of vector data), featuretypes, layers, styles, as well as vector data upload operations. For more information about the GeoServer REST API, see <http://docs.geoserver.org/stable/en/user/rest/> .

Details

Package:	geosapi
Type:	Package
Version:	0.1-0
Date:	2017-02-21
License:	MIT
LazyLoad:	yes

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSDataStore *Geoserver REST API DataStore*

Description

Geoserver REST API DataStore

Usage

```
GSDataStore
```

Format

R6Class object.

Value

Object of R6Class for modelling a GeoServer dataStore

Fields

`name`

`workspace`

Methods

`new(xml, dataStore, description, type, enabled, connectionParameters)` This method is used to instantiate a GSDataStore

`decode(xml)` This method is used to decode a GSDataStore from XML

`encode()` This method is used to encode a GSNamespace to XML. Inherited from the generic `GSRESTResource` encoder

`setEnabled(enabled)` Sets the datastore as enabled if TRUE, disabled if FALSE

`setDescription(description)` Sets the datastore description

`setType(type)` Sets the datastore type

`setConnectionParameters(parameters)` Sets the datastore connection parameters. The argument should be an object of class `GSRESTEntrySet` giving a list of key/value parameter entries.

`addConnectionParameter(key, value)` Adds a datastore connection parameter. Convenience wrapper of `GSRESTEntrySet` `addEntry` method.

`setConnectionParameter(key, value)` Sets a datastore connection parameter. Convenience wrapper of `GSRESTEntrySet` `setEntry` method.

`delConnectionParameter(key)` Deletes a datastore connection parameter. Convenience wrapper of `GSRESTEntrySet` `delEntry` method.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSDataStoreManager *Geoserver REST API DataStore Manager*

Description

Geoserver REST API DataStore Manager

Usage

```
GSDataStoreManager
```

Format

R6Class object.

Value

Object of R6Class with methods for managing GeoServer DataStores (i.e. stores of vector data)

Methods

`new(url, user, pwd, logger)` This method is used to instantiate a GSManager with the `url` of the GeoServer and credentials to authenticate (`user`/`pwd`). By default, the `logger` argument will be set to `NULL` (no logger). This argument accepts two possible values: `INFO`: to print only geosapi logs, `DEBUG`: to print geosapi and CURL logs

`getDataStores(ws)` Get the list of available dataStores. Returns an object of class `list` giving items of class GSDataStore

`getDataStoreNames(ws)` Get the list of available dataStore names. Returns an vector of class `character`

`getDataStore(ws, ds)` Get an object of class GSDataStore given a workspace and datastore names.

`createDataStore(ws, dataStore)` Creates a new datastore given a workspace and an object of class GSDataStore

`updateDataStore(ws, dataStore)` Updates an existing dataStore given a workspace and an object of class GSDataStore

`deleteDataStore(ws, ds, recurse)` Deletes a datastore given a workspace and an object of class GSDataStore. By defaut, the option `recurse` is set to FALSE, ie datastore layers are not removed. To remove all datastore layers, set this option to TRUE.

`getFeatureTypes(ws, ds)` Get the list of available feature types for given workspace and datastore. Returns an object of class `list` giving items of class GSFeatureType

`getFeatureTypeNames(ws, ds)` Get the list of available feature type names for given workspace and datastore. Returns an vector of class`character`

`getFeatureType(ws, ds, ft)` Get an object of class GSFeatureType given a workspace, datastore and feature type names.

`createFeatureType(ws, ds, featureType)` Creates a new featureType given a workspace, datastore names and an object of class `GSFeatureType`

`updateFeatureType(ws, ds, FeatureType)` Updates a new featureType given a workspace, datastore names and an object of class `GSFeatureType`

`deleteFeatureType(ws, ds, featureType, recurse)` Deletes a featureType given a workspace, datastore names, and an object of class `GSFeatureType`. By defaut, the option `recurse` is set to FALSE, ie datastore layers are not removed.

`uploadData(ws, ds, endpoint, extension, configure, update, filename, charset, contentType)` Uploads data to a target dataStore

`uploadShapefile(ws, ds, endpoint, configure, update, filename, charset)` Uploads a zipped ESRIshapefile to a target dataStore

`uploadProperties(ws, ds, endpoint, configure, update, filename, charset)` Uploads a properties file to a target dataStore

`uploadH2(ws, ds, endpoint, configure, update, filename, charset)` Uploads a H2 database to a target dataStore

`uploadSpatialite(ws, ds, endpoint, configure, update, filename, charset)` Uploads a Spatialite database to a target dataStore

`uploadAppschema(ws, ds, endpoint, configure, update, filename, charset)` Uploads a appschema file to a target dataStore

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
   GSDataStoreManager$new("http://localhost:8080/geoserver", "admin", "geoserver")

## End(Not run)
```

`GSDimension` *A GeoServer dimension*

Description

This class models a GeoServer resource dimension.

This class models a GeoServer feature dimension.

Usage

```
GSDimension

GSFeatureDimension
```

Format

R6Class object.

Details

Geoserver REST API Dimension

Geoserver REST API FeatureDimension

Value

Object of R6Class for modelling a GeoServer dimension

Object of R6Class for modelling a GeoServer feature dimension

Fields

enabled

presentation

resolution

units

unitSymbol

attribute

endAttribute

Methods

new(xml) This method is used to instantiate a GSResource

decode(xml) This method is used to decode a GSResource from XML

encode() This method is used to encode a GSFeatureType to XML. Inherited from the generic GSRESTResource encoder

Methods

new(xml) This method is used to instantiate a GSResource

decode(xml) This method is used to decode a GSResource from XML

encode() This method is used to encode a GSFeatureType to XML. Inherited from the generic GSRESTResource encoder

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
dim <- GSDimension$new()
dim <- GSFeatureDimension$new()
```

GSFeatureType *A GeoServer feature type*

Description

This class models a GeoServer feature type. This class is to be used for manipulating representations of vector data with GeoServer.

Usage

```
GSFeatureType
```

Format

R6Class object.

Details

Geoserver REST API Resource

Value

Object of R6Class for modelling a GeoServer feature type

Methods

`new(rootName, xml)` This method is used to instantiate a GSResource

`decode(xml)` This method is used to decode a GSResource from XML

`encode()` This method is used to encode a GSFeatureType to XML. Inherited from the generic `GSRESTResource` encoder

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
ft <- GSFeatureType$new()
```

GSLayer *A GeoServer layer resource*

Description

This class models a GeoServer layer. This class is to be used for published resource (feature type or coverage).

This class models a GeoServer style.

Usage

```
GSLayer

GSStyle
```

Format

R6Class object.

Details

Geoserver REST API Resource

Geoserver REST API Style

Value

Object of R6Class for modelling a GeoServer layer

Object of R6Class for modelling a GeoServer style

Methods

`new(rootName, xml)` This method is used to instantiate a GSLayer

`decode(xml)` This method is used to decode a GSLayer from XML

`encode()` This method is used to encode a GSLayer to XML. Inherited from the generic `GSRESTResource` encoder

`setName(name)` Sets the layer name.

`setPath(path)` Sets the layer path.

`setDefaultStyle(style)` Sets the default style.

`setStyles(styles)` Sets a list of optional styles

`addStyle(style)` Sets an available style. Returns TRUE if set, FALSE otherwise

`delStyle(name)` Deletes an available. Returns TRUE if deleted, FALSE otherwise

`setEnabled(enabled)` Sets if the layer is enabled (TRUE) or not (FALSE)

`setQueryable(queryable)` Sets if the layer is queryable (TRUE) or not (FALSE)

`setAdvertised(advertised)` Sets if the layer is advertised (TRUE) or not (FALSE)

Methods

`new(xml)` This method is used to instantiate a GS Style

`decode(xml)` This method is used to decode a GSStyle from XML

`encode()` This method is used to encode a GSStyle to XML. Inherited from the generic `GSRESTResource` encoder

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
lyr <- GSLayer$new()
lyr <- GSStyle$new()
```

`GSManager` *Geoserver REST API Manager*

Description

Geoserver REST API Manager

Usage

```
GSManager
```

Format

`R6Class` object.

Value

Object of `R6Class` with methods for communication with the REST API of a GeoServer instance.

Fields

`loggerType` the type of logger

`verbose.info` if geosapi logs have to be printed

`verbose.debug` if curl logs have to be printed

`url` the Base url of GeoServer

`version` the version of Geoserver. Handled as `GSVersion` object

Methods

`new(url, user, pwd, logger)` This method is used to instantiate a GSManager with the `url` of the GeoServer and credentials to authenticate (`user`/`pwd`). By default, the `logger` argument will be set to `NULL` (no logger). This argument accepts two possible values: `INFO`: to print only geosapi logs, `DEBUG`: to print geosapi and CURL logs

`logger(type, text)` Basic logger to report geosapi logs. Used internally

`INFO(text)` Logger to report information. Used internally

`WARN(text)` Logger to report warnings. Used internally

`ERROR(text)` Logger to report errors. Used internally

`getUrl()` Get the authentication URL

`connect()` This methods attempts a connection to GeoServer REST API. User internally during initialization of `GSManager`.

`reload()` Reloads the GeoServer catalog.

`getClassName()` Retrieves the name of the class instance

`getWorkspaceManager()` Retrieves an instance of workspace manager

`getNamespaceManager()` Retrieves an instance of namespace manager

`getDataStoreManager()` Retrieves an instance of datastore manager

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
   GSManager$new("http://localhost:8080/geoserver", "admin", "geoserver")

## End(Not run)
```

GSMetadataLink *A GeoServer resource metadataLink*

Description

This class models a GeoServer resource metadataLink made of a type (free text e.g. text/xml, text/html), a metadataType (Possible values are ISO19115:2003, FGDC, TC211, 19139, other), and a content: an URL that gives the metadataLink

Usage

```
GSMetadataLink
```

Format

`R6Class` object.

Details

Geoserver REST API Metadatalink

Value

Object of R6Class for modelling a GeoServer resource metadataLink

Methods

`new(xml, type, metadataType, content)` This method is used to instantiate a GSMetadataLink

`decode(xml)` This method is used to decode a GSMetadataLink from XML

`encode()` This method is used to encode a GSMetadataLink to XML. Inherited from the generic `GSRESTResource` encoder

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSNamespace *Geoserver REST API Namespace*

Description

Geoserver REST API Namespace

Usage

```
GSNamespace
```

Format

R6Class object.

Value

Object of R6Class for modelling a GeoServer namespace

Fields

`name`

`prefix`

`uri`

`full`

Methods

`new(xml, prefix, uri)` This method is used to instantiate a GSNamespace

`decode(xml)` This method is used to decode a GSNamespace from XML

`encode()` This method is used to encode a GSNamespace to XML. Inherited from the generic `GSRESTResource` encoder

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSNamespace$new(prefix = "prefix", uri = "http://prefix")
```

GSNamespaceManager *Geoserver REST API Namespace Manager*

Description

Geoserver REST API Namespace Manager

Usage

```
GSNamespaceManager
```

Format

`R6Class` object.

Value

Object of `R6Class` with methods for managing the namespaces of a GeoServer instance.

Methods

`new(url, user, pwd, logger)` This method is used to instantiate a GSManager with the `url` of the GeoServer and credentials to authenticate (`user`/`pwd`). By default, the `logger` argument will be set to `NULL` (no logger). This argument accepts two possible values: `INFO`: to print only geosapi logs, `DEBUG`: to print geosapi and CURL logs

`getNamespaces()` Get the list of available namespace. Returns an object of class `list` containing items of class `GSNamespace`

`getNamespaceNames()` Get the list of available namespace names. Returns an vector of class `character`

`getNamespace(ns)` Get a `GSNamespace` object given a namespace name.

`createNamespace(prefix, uri)` Creates a GeoServer namespace given a prefix, and an optional URI. Returns `TRUE` if the namespace has been successfully created, `FALSE` otherwise

`updateNamespace(ns, uri)` Updates a GeoServer namespace given a name, and an optional URI. Returns `TRUE` if the namespace has been successfully updated, `FALSE` otherwise

`deleteNamespace(ns)` Deletes a GeoServer namespace given a name. Returns `TRUE` if the namespace has been successfully deleted, `FALSE` otherwise

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
   GSNamespaceManager$new("http://localhost:8080/geoserver", "admin", "geoserver")

## End(Not run)
```

`GSResource` *A GeoServer abstract resource*

Description

This class models an abstract GeoServer resource. This class is used internally for modelling instances of class `GSFeatureType` or `GSCoverage`

Usage

```
GSResource
```

Format

`R6Class` object.

Details

Geoserver REST API Resource

Value

Object of `R6Class` for modelling a GeoServer resource

Fields

`name`

`nativeName`

`title`

`description`

`abstract`

keywords

metadataLinks

projectionPolicy

srs

nativeCRS

latLonBoundingBox

nativeBoundingBox

Methods

new(rootName, xml) This method is used to instantiate a GSResource

decode(xml) This method is used to decode a GSResource from XML

encode() This method is used to encode a GSResource to XML. Inherited from the generic GSRESTResource encoder

setEnabled(enabled) Sets if the resource is enabled or not in GeoServer

setName(name) Sets the resource name

setNativeName(nativeName) Sets the resource native name

setTitle(title) Sets the resource title

setDescription(description) Sets the resource description

setAbstract(abstract) Sets the resource abstract

setKeywords(keywords) Sets a list of keywords

addKeyword(keyword) Sets a keyword. Returns TRUE if set, FALSE otherwise

delKeyword(keyword) Deletes a keyword. Returns TRUE if deleted, FALSE otherwise

setMetadataLinks(metadataLinks) Sets a list of GSMetadataLinks

addMetadataLink(metadataLink) Adds a metadataLink

delMetadataLink(metadataLink) Deletes a metadataLink

setNativeCRS(nativeCRS) Sets the resource nativeCRS

setSrs(srs) Sets the resource srs

setNativeBoundingBox(minx, miny, maxx, maxy, bbox, crs) Sets the resource nativeBoundingBox. Either from coordinates or from a bbox object (matrix).

setLatLonBoundingBox(minx, miny, maxx, maxy, bbox, crs) Sets the resource latLonBoundingBox. Either from coordinates or from a bbox object (matrix).

setProjectionPolicy(policy) Sets the resource projection policy

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
res <- GSResource$new(rootName = "featureType")
```

GSRESTEntrySet *Geoserver REST API XML entry set*

Description

Geoserver REST API XML entry set

Usage

```
GSRESTEntrySet
```

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a entry set

Fields

`entryset`

Methods

`new(xml)` This method is used to instantiate a GSDataStore

`decode(xml)` This method is used to decode a GSRESTEntrySet from XML

`encode()` This method is used to encode a GSRESTEntrySet as XML

`setEntryset(entryset)` Sets an entryset (list)

`addEntry(key, value)` Adds an entry (key/value pair). Returns TRUE if added, FALSE otherwise

`setEntry(key, value)` Sets an entry (key/value pair).

`delEntry(key)` Deletes an entry by key. Returns TRUE if removed, FALSE otherwise

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSRESTResource *Geoserver REST API REST Resource interface*

Description

Geoserver REST API REST Resource interface

Usage

```
GSRESTResource
```

Format

R6Class object.

Value

Object of R6Class for modelling a GeoServer REST resource interface

Abstract Methods

`new()` This method is used to instantiate a GSRESTResource

`decode(xml)` Decodes a GS* R6 object from XML representation

`encode()` Encodes a GS* R6 object to XML representation

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSShapefileDataStore *Geoserver REST API ShapeFileDataStore*

Description

Geoserver REST API ShapeFileDataStore

Usage

```
GSShapefileDataStore
```

Format

R6Class object.

Value

Object of R6Class for modelling a GeoServer Shapefile dataStore

Methods

`new(xml, dataStore, description, enabled, url)` Instantiates a GSShapefileDataStore object

`setUrl(url)` Set the spatial files data URL

`setCharset(charset)` Set the charset used for DBF file. Default value is 'ISO-8859-1'

`setCreateSpatialIndex(create)` Set the 'Create Spatial Index' option. Default is TRUE

`setMemoryMappedBuffer(buffer)` Set the 'Memory Mapped Buffer' option. Default is TRUE

`CacheReuseMemoryMaps(maps)` Set the 'Cache & Reuse Memory Maps' option. Default is TRUE

`setDefautConnectionParameters()` Set the defaut connection paramaters

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSShapefileDataStore$new(dataStore="ds", description = "des",
                         enabled = TRUE, url = "file://data/shape.shp")
```

GSStyleManager *Geoserver REST API Style Manager*

Description

Geoserver REST API Style Manager

Usage

```
GSStyleManager
```

Format

R6Class object.

Value

Object of R6Class with methods for managing the styles of a GeoServer instance.

Methods

`new(url, user, pwd, logger)` This method is used to instantiate a GSManager with the `url` of the GeoServer and credentials to authenticate (`user`/`pwd`). By default, the `logger` argument will be set to `NULL` (no logger). This argument accepts two possible values: `INFO`: to print only geosapi logs, `DEBUG`: to print geosapi and CURL logs

`getStyles()` Get the list of available styles. Returns an object of class `list` containing items of class `GSStyle`

`getStyleNames()` Get the list of available style names. Returns an vector of class `character`

`getStyle(style)` Get a `GSStyle` object given a style name.

`createStyle(file, sldBody, name, raw, ws)` Creates a GeoServer style given a name. Returns `TRUE` if the style has been successfully created, `FALSE` otherwise

`updateStyle(file, sldBody, name, raw, ws)` Updates a GeoServer style. Returns `TRUE` if the style has been successfully updated, `FALSE` otherwise

`deleteStyle(style, recurse, purge, ws)` Deletes a GeoServer style given a name. Returns `TRUE` if the style has been successfully deleted, `FALSE` otherwise

`getSLDVersion(sldBody)` Get the SLD version from the XML object (of class `XMLInternalDocument`)

`getSLDBody(style, ws = NULL)` Get the SLD Body given a style name. This method is only supported for Geoserver >= 2.2.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
   GSStyleManager$new("http://localhost:8080/geoserver", "admin", "geoserver")

## End(Not run)
```

GSUtils *Geoserver REST API Manager Utils*

Description

Geoserver REST API Manager Utils

Usage

```
GSUtils
```

Format

`R6Class` object.

Value

Object of R6Class with static util methods for communication with the REST API of a GeoServer instance.

Static methods

`getUserAgent()` This method is used to get the user agent for performing GeoServer API requests. Here the user agent will be compound by geosapi package name and version.

`getUserToken(user, pwd)` This method is used to get the user authentication token for performing GeoServer API requests. Token is given a Base64 encoded string.

`GET(url, user, pwd, path, verbose)` This method performs a GET request for a given `path` to GeoServer REST API

`PUT(url, user, pwd, path, filename, contentType, verbose)` This method performs a PUT request for a given `path` to GeoServer REST API, to upload a file of name `filename` with given `contentType`

`POST(url, user, pwd, path, content, contentType, verbose)` This method performs a POST request for a given `path` to GeoServer REST API, to post content of given `contentType`

`DELETE(url, user, pwd, path, verbose)` This method performs a DELETE request for a given GeoServer resource identified by a `path` in GeoServer REST API

`parseResponseXML(req)` Convenience method to parse XML response from GeoServer REST API. Although package **httr** suggests the use of **xml2** package for handling XML, **geosapi** still relies on the package **XML**. Response from **httr** is retrieved as text, and then parsed as XML using `xmlParse` function.

`getPayloadXML(obj)` Convenience method to create payload XML to send to GeoServer.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSVersion *A GeoServer version*

Description

This class allows to grab the GeoServer version. By default, a tentative is made to fetch version from web admin default page, since Geoserver REST API did not support GET operation for the Geoserver version in past releases of Geoserver.

Usage

```
GSVersion
```

Format

R6Class object.

Details

Geoserver REST API - Geoserver Version

Value

Object of R6Class for modelling a GeoServer version

Methods

`new(url, user, pwd)` This method is used to instantiate a GSVersion object.

`lowerThan(version)` Compares to a version and returns TRUE if it is lower, FALSE otherwise

`greaterThan(version)` Compares to a version and returns TRUE if it is greater, FALSE otherwise

`equalTo(version)` Compares to a version and returns TRUE if it is equal, FALSE otherwise

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
version <- GSVersion$new(
             url = "http://localhost:8080/geoserver",
             user = "admin", pwd = "geoserver"
           )

## End(Not run)
```

GSWorkspace *Geoserver REST API Workspace*

Description

Geoserver REST API Workspace

Usage

`GSWorkspace`

Format

R6Class object.

Value

Object of R6Class for modelling a GeoServer workspace

Fields

`name`

Methods

`new(xml, name)` This method is used to instantiate a GSWorkspace

`decode(xml)` This method is used to decode a GSWorkspace from XML

`encode()` This method is used to encode a GSWorkspace to XML. Inherited from the generic `GSRESTResource` encoder

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSWorkspace$new(name = "work")
```

GSWorkspaceManager *Geoserver REST API Workspace Manager*

Description

Geoserver REST API Workspace Manager

Usage

```
GSWorkspaceManager
```

Format

`R6Class` object.

Value

Object of `R6Class` with methods for managing the workspaces of a GeoServer instance.

Methods

`new(url, user, pwd, logger)` This method is used to instantiate a GSManager with the `url` of the GeoServer and credentials to authenticate (`user`/`pwd`). By default, the `logger` argument will be set to `NULL` (no logger). This argument accepts two possible values: `INFO`: to print only geosapi logs, `DEBUG`: to print geosapi and CURL logs

`getWorkspaces()` Get the list of available workspace. Returns an object of class `list` containing items of class `GSWorkspace`

`getWorkspaceNames()` Get the list of available workspace names. Returns an vector of class `character`

`getWorkspace(ws)` Get a `GSWorkspace` object given a workspace name.

`createWorkspace(name, uri)` Creates a GeoServer workspace given a name, and an optional URI. If the URI is not specified, GeoServer will automatically create an associated Namespace with the URI being "http://workspaceName. If the URI is specified, the method invokes the method `createNamespace(ns, uri)` of the `GSNamespaceManager`. Returns `TRUE` if the workspace has been successfully created, `FALSE` otherwise

`updateWorkspace(name, uri)` Updates a GeoServer workspace given a name, and an optional URI. If the URI is not specified, GeoServer will automatically update the associated Namespace with the URI being "http://workspaceName. If the URI is specified, the method invokes the method `updateNamespace(ns, uri)` of the `GSNamespaceManager`. Returns `TRUE` if the workspace has been successfully updated, `FALSE` otherwise

`deleteWorkspace(ws)` Deletes a GeoServer workspace given a name. Returns `TRUE` if the workspace has been successfully deleted, `FALSE` otherwise

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
   GSWorkspaceManager$new("http://localhost:8080/geoserver", "admin", "geoserver")

## End(Not run)
```

Index