Package 'geosapi'

September 15, 2021

Type Package

Title GeoServer REST API R Interface

Version 0.5-1

Date 2021-09-14

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 <https://docs.geoserver.org/stable/en/user/rest/>.

Depends R (>= 3.1.0)

Imports R6, openssl, httr, XML, keyring

Suggests testthat, roxygen2, covr

License MIT + file LICENSE

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

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

LazyLoad yes

RoxygenNote 7.1.0

NeedsCompilation no

Author Emmanuel Blondel [aut, cre] (<https://orcid.org/0000-0002-5870-5762>)

Repository CRAN

Date/Publication 2021-09-15 17:30:02 UTC

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 <https://docs.geoserver.org/stable/en/user/rest/> .

Details

Package:	geosapi
Type:	Package
Version:	0.5-1
Date:	2021-09-14
License:	MIT
LazyLoad:	yes

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

GSAbstractDBDataStore *Geoserver REST API AbstractDBDataStore*

Description

Geoserver REST API AbstractDBDataStore

Format

R6Class object.

Value

Object of R6Class for modelling a GeoServer abstract DB dataStore

Methods

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

`setDatabaseType(dbtype)` Sets the database type

`setNamespace(namespace)` Sets the datastore namespace

`setHost(host)` Sets the database host

`setPort(port)` Set the database port

`setDatabase(database)` Set the database name

`setSchema(schema)` Set the database schema

`setUser(user)` Set the database username

`setPassword(password)` Set the database password

`setJndiReferenceName(jndiReferenceName)` Set a JNDI reference name `setExposePrimaryKeys(exposePrimaryKeys)` Set TRUE if primary keys have to be exposed to datastore, FALSE otherwise.

`setMaxConnections(maxConnections)` Set the maximum number of connections. Default is set to 10.

`setMinConnections(minConnections)` Set the minimum number of connections. Default is set to 1.

`setFetchSize(fetchSize)` Set the fetch size. Default is set to 10.

`setConnectionTimeout(seconds)` Set the connection timeout. Default is set to 20s.

`setValidateConnections(validateConnections)` Set TRUE if connections have to be validated, FALSE otherwise.

`setPrimaryKeyMetadataTable(primaryKeyMetadataTable)` Set the name of the primaryKey metadata table

`setLooseBBox(looseBBox)` Set losse bbox parameter.

`setPreparedStatements(preparedStatements)` Set prepared statements

`setMaxOpenPreparedStatements(maxOpenPreparedStatements)` Set maximum open prepared statements

`setEstimatedExtends(estimatedExtends)` Set estimatedExtend parameter

`setDefautConnectionParameters()` Set default connection parameters

Note

Internal abstract class used for setting DB stores

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

`GSDataStore` *Geoserver REST API DataStore*

Description

Geoserver REST API DataStore

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer dataStore

Fields

`full` completeness of datastore description

`name` datastore name

`description` datastore description

`type` datastore type

`connectionParmaeters` list of datastore connection parameters

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

Format

`R6Class` object.

Value

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

Constructor

`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

`DataStore` methods

`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.

`FeatureType` methods

`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 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.

`Layer` methods

- `getLayers()` Get the list of layers. Returns an object of class `list` giving items of class `GSLayer`
- `getLayerNames()` Get the list of layer names.
- `getLayer(lyr)` Get an object of class `GSLayer` if existing
- `createLayer(layer)` Creates a new layer given an object of class `GSLayer`
- `updateLayer(layer)` Creates a layer given an object of class `GSLayer`
- `deleteLayer(layer)` Deletes a layer given an object of class `GSLayer`

`LayerGroup` methods

- `getLayerGroups()` Get the list of layers. Returns an object of class `list` giving items of class `GSLayer`
- `getLayerGroupNames()` Get the list of layer names.
- `getLayerGroup(lyr, ws)` Get an object of class `GSLayerGroup` if existing. Can be restrained to a workspace.
- `createLayerGroup(layerGroup, ws)` Creates a new layer given an object of class `GSLayerGroup`. Can be restrained to a particular workspace.
- `updateLayerGroup(layerGroup, ws)` Creates a layer given an object of class `GSLayerGroup`. Can be restrained to a particular workspace.
- `deleteLayerGroup(layerGroup, ws)` Deletes a layer given an object of class `GSLayerGroup`. Can be restrained to a particular workspace.

Main `Layer` user publication methods

- `publishLayer(ws, ds, featureType, layer)` Publish a web-layer (including the featureType and 'layer' resources), given a workspace, a datastore, providing an object of class `GSFeatureType`, and `GSLayer`
- `unpublishLayer(ws, ds, lyr)` Unpublish a web-layer (including the featureType and 'layer' resources), given a workspace, a datastore, and a layer name

Data upload methods

- `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 app-schema file to a target dataStore
- `uploadGeopackage(ws, ds, endpoint, configure, update, filename, charset)` Uploads a GeoPackage 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.

Format

`R6Class` object.

`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` true/false

`presentation` dimension presentation

`resolution` dimension resolution

`units` dimension units

`unitSymbol` dimension unitsSymbol

`attribute` attribute

`endAttribute` end attribute

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

`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.

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

`setCqlFilter(filter)` Sets a CQL filter for the feature type.

`setVirtualTable(vt)` Sets a virtual table for the feature type.

`delVirtualTable()` Deletes the virtual table for the feature type

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

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

`GSGeoPackageDataStore` *Geoserver REST API GeoPackageDataStore*

Description

Geoserver REST API GeoPackageDataStore

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer GeoPackage dataStore

Methods inherited from `GSAbstractDBDataStore`

`setDatabaseType(dbtype)` Sets the database type, here "geopkg

`setNamespace(namespace)` Sets the datastore namespace

`setHost(host)` Sets the database host

`setPort(port)` Set the database port

`setDatabase(database)` Set the database name

`setSchema(schema)` Set the database schema

`setUser(user)` Set the database username

`setPassword(password)` Set the database password `setJndiReferenceName(jndiReferenceName)` Set a JNDI reference name

`setExposePrimaryKeys(exposePrimaryKeys)` Set TRUE if primary keys have to be exposed to datastore, FALSE otherwise.

`setMaxConnections(maxConnections)` Set the maximum number of connections. Default is set to 10.

`setMinConnections(minConnections)` Set the minimum number of connections. Default is set to 1.

`setFetchSize(fetchSize)` Set the fetch size. Default is set to 10.

`setConnectionTimeout(seconds)` Set the connection timeout. Default is set to 20s.

`setValidateConnections(validateConnections)` Set TRUE if connections have to be validated, FALSE otherwise.

`setPrimaryKeyMetadataTable(primaryKeyMetadataTable)` Set the name of the primaryKey metadata table

`setLooseBBox(looseBBox)` Set losse bbox parameter.

`setPreparedStatements(preparedStatements)` Set prepared statements

`setMaxOpenPreparedStatements(maxOpenPreparedStatements)` Set maximum open prepared statements

`setEstimatedExtends(estimatedExtends)` Set estimatedExtend parameter

`setDefautConnectionParameters()` Set default connection parameters

Methods

`new(xml, dataStore, description, enabled, database)` Instantiates a GSGeoPackageDataStore object

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
ds <- GSGeoPackageDataStore$new(
 dataStore="ds", description = "des",
 enabled = TRUE, database = NULL
)
```

`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 layer. This class is to be used internally by **geosapi** for configuring layers or layer groups within an object of class `GSLayerGroup`

This class models a GeoServer style.

Format

`R6Class` object.

`R6Class` object.

`R6Class` object.

Details

Geoserver REST API Resource

Geoserver REST API Publishable

Geoserver REST API Style

Value

Object of `R6Class` for modelling a GeoServer layer

Object of `R6Class` for modelling a GeoServer layer group publishable

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)

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

`decode(xml)` This method is used to decode a GSPublishable

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

`setName(name)` Sets the publishable name.

`setType(type)` Sets the publishable type.

`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>

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
lyr <- GSLayer$new()

publishable <- GSPublishable$new(name = "name", type = "layer")

lyr <- GSStyle$new()
```

`GSLayerGroup` *A GeoServer layergroup resource*

Description

This class models a GeoServer layer group. This class is to be used for clustering layers into a group.

Format

`R6Class` object.

Details

Geoserver REST API LayerGroup

Value

Object of `R6Class` for modelling a GeoServer layergroup

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 name.

`setTitle(title)` Sets the title.

`setAbstract(abstract)` Sets the abstract.

`setMode(mode)` Sets the mode.

`setWorkspace(ws)` Sets the worksapce

`addLayer(layer)` Adds a layer

`delLayer(layer)` Deletes a layer

`addLayerGroup(layerGroup)` Adds a layer group

`delLayerGroup(layerGroup)` Deletes a layer group

`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

`setMetadataLinks(metadataLinks)` Sets a list of `GSMetadataLinks`

`addMetadataLink(metadataLink)` Adds a metadataLink

`delMetadataLink(metadataLink)` Deletes a metadataLink

`setBounds(minx, miny, maxx, maxy, bbox, crs)` Sets the layer group bounds. Either from coordinates or from a bbox object (matrix).

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
lyr <- GSLayerGroup$new()
```

`GSManager` *Geoserver REST API Manager*

Description

Geoserver REST API Manager

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, keyring_backend)` 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.

The `keyring_backend` can be set to use a different backend for storing the Geoserver user password with **keyring** (Default value is 'env').

`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

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

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer namespace

Fields

`name` namespace name

`prefix` namespace prefix

`uri` namespace URI

`full` completeness of the namespace description

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

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)
```

`GSOracleNGDataStore` *Geoserver REST API OracleNGDataStore*

Description

Geoserver REST API OracleNGDataStore

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer OracleNG dataStore

Methods inherited from `GSAbstractDBDataStore`

`setDatabaseType(dbtype)` Sets the database type, here "OracleNG"

`setNamespace(namespace)` Sets the datastore namespace

`setHost(host)` Sets the database host

`setPort(port)` Set the database port

`setDatabase(database)` Set the database name

`setSchema(schema)` Set the database schema

`setUser(user)` Set the database username

`setPassword(password)` Set the database password

`setJndiReferenceName(jndiReferenceName)` Set a JNDI reference name

`setExposePrimaryKeys(exposePrimaryKeys)` Set TRUE if primary keys have to be exposed to datastore, FALSE otherwise.

`setMaxConnections(maxConnections)` Set the maximum number of connections. Default is set to 10.

`setMinConnections(minConnections)` Set the minimum number of connections. Default is set to 1.

`setFetchSize(fetchSize)` Set the fetch size. Default is set to 10.

`setConnectionTimeout(seconds)` Set the connection timeout. Default is set to 20s.

`setValidateConnections(validateConnections)` Set TRUE if connections have to be validated, FALSE otherwise.

`setPrimaryKeyMetadataTable(primaryKeyMetadataTable)` Set the name of the primaryKey metadata table

`setLooseBBox(looseBBox)` Set losse bbox parameter.

`setPreparedStatements(preparedStatements)` Set prepared statements

`setMaxOpenPreparedStatements(maxOpenPreparedStatements)` Set maximum open prepared statements

`setEstimatedExtends(estimatedExtends)` Set estimatedExtend parameter

`setDefautConnectionParameters()` Set default connection parameters

Methods

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

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSOracleNGDataStore$new(dataStore="ds", description = "des", enabled = TRUE)
```

`GSPostGISDataStore` *Geoserver REST API PostGISDataStore*

Description

Geoserver REST API PostGISDataStore

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer PostGIS dataStore

Methods inherited from `GSAbstractDBDataStore`

`setDatabaseType(dbtype)` Sets the database type, here "postgis"

`setNamespace(namespace)` Sets the datastore namespace

`setHost(host)` Sets the database host

`setPort(port)` Set the database port

`setDatabase(database)` Set the database name

`setSchema(schema)` Set the database schema

`setUser(user)` Set the database username

`setPassword(password)` Set the database password

`setJndiReferenceName(jndiReferenceName)` Set a JNDI reference name

`setExposePrimaryKeys(exposePrimaryKeys)` Set TRUE if primary keys have to be exposed to datastore, FALSE otherwise.

`setMaxConnections(maxConnections)` Set the maximum number of connections. Default is set to 10.

`setMinConnections(minConnections)` Set the minimum number of connections. Default is set to 1.

`setFetchSize(fetchSize)` Set the fetch size. Default is set to 10.

`setConnectionTimeout(seconds)` Set the connection timeout. Default is set to 20s.

`setValidateConnections(validateConnections)` Set TRUE if connections have to be validated, FALSE otherwise.

`setPrimaryKeyMetadataTable(primaryKeyMetadataTable)` Set the name of the primaryKey metadata table

`setLooseBBox(looseBBox)` Set losse bbox parameter.

`setPreparedStatements(preparedStatements)` Set prepared statements

`setMaxOpenPreparedStatements(maxOpenPreparedStatements)` Set maximum open prepared statements

`setEstimatedExtends(estimatedExtends)` Set estimatedExtend parameter

`setDefautConnectionParameters()` Set default connection parameters

Methods

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

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSPostGISDataStore$new(dataStore="ds", description = "des", enabled = TRUE)
```

`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`

Format

`R6Class` object.

Details

Geoserver REST API Resource

Value

Object of `R6Class` for modelling a GeoServer resource

Fields

`name` resource name

`nativeName` resource native name

`title` resource title

`description` resource description

`abstract` resource abstract

`keywordsr` resource keywords

`metadataLinks` resource metadata links

`projectionPolicy` resource projection policy

`srs` resource srs

`nativeCRS` resource native CRS

`latLonBoundingBox` resource lat/lon bounding box

`nativeBoundingBox` resource lat/lon native bounding box

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

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

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>

`GSServiceManager` *Geoserver REST API Service Manager*

Description

Geoserver REST API Service Manager

Format

`R6Class` object.

Value

Object of `R6Class` with methods for managing GeoServer services

Constructor

`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

`getServiceSettings(service, ws)` Get the service settings. To get the service settings for a specific workspace, specify the workspace name as `ws` parameter, otherwise global settings are retrieved.

`getWmsSettings(ws)` Get WMS settings. To get the WMS settings for a specific workspace, specify the workspace name as `ws` parameter, otherwise global settings are retrieved.

`getWfsSettings(ws)` Get WFS settings. To get the WFS settings for a specific workspace, specify the workspace name as `ws` parameter, otherwise global settings are retrieved.

`getWcsSettings(ws)` Get WCS settings. To get the WCS settings for a specific workspace, specify the workspace name as `ws` parameter, otherwise global settings are retrieved.

`updateServiceSettings(serviceSettings, service, ws)` Updates the service settings with an object of class `GSServiceSetting`. An optional workspace name `ws` can be specified to update service settings applying to a workspace.

`deleteServiceSettings(service, ws)` Deletes the service settings. This method is used internally by **geosapi** for disabling a service setting at workspace level.

`updateWmsSettings(serviceSettings, ws)` Updates the WMS settings with an object of class `GSServiceSetting`. An optional workspace name `ws` can be specified to update WMS settings applying to a workspace.

`updateWfsSettings(serviceSettings, ws)` Updates the WFS settings with an object of class `GSServiceSetting`. An optional workspace name `ws` can be specified to update WFS settings applying to a workspace.

`updateWcsSettings(serviceSettings, ws)` Updates the WCS settings with an object of class `GSServiceSettings`. An optional workspace name `ws` can be specified to update WCS settings applying to a workspace.

`enableWMS(ws)` Enables the WMS, either globally, or for a given workspace (optional)

`enableWFS(ws)` Enables the WFS, either globally, or for a given workspace (optional)

`enableWCS(ws)` Enables the WCS, either globally, or for a given workspace (optional)

`disableServiceSettings(service, ws)` Disables a service, either globally, or for a given workspace (optional). For a global service setting, an UPDATE operation will be applied, while for a workspace service setting, a DELETE operation is applied.

`disableWMS(ws)` Disables the WMS, either globally, or for a given workspace (optional)

`disableWFS(ws)` Disables the WFS, either globally, or for a given workspace (optional)

`disableWCS(ws)` Disables the WCS, either globally, or for a given workspace (optional)

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

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

## End(Not run)
```

`GSServiceSettings` *A GeoServer service settings resource*

Description

This class models a GeoServer OWS service settings.

Format

`R6Class` object.

Details

Geoserver REST API Service Setting

Value

Object of `R6Class` for modelling a GeoServer OWS service setting

Fields

`enabled` is service enabled or not?

`citeCompliant` is service cite compliant?

`name` service name

`title` service title

`maintainer` service maintainer

`abstract` service abastract

`accessConstraints` service access constraints

`fees` service fees

`keywords` services keywords

`onlineResource` service online resource

`schemaBaseURL` service schema base URL

`verbose` service verbose or not?

Methods

- `new(rootName, xml)` This method is used to instantiate a `GSServiceSettings`. This settings object is required to model/manipulate an OGC service configuration, using the method `GSManager$updateServiceSetti` or derivates.
- `decode(xml)` This method is used to decode a `GSServiceSettings` from XML
- `encode()` This method is used to encode a `GSServiceSettings` to XML. Inherited from the generic `GSRESTResource` encoder
- `setEnabled(enabled)` Sets if the service is enabled (TRUE) or not (FALSE)
- `setCiteCompliant(citeCompliant)` Sets if the service is compliant with CITE (TRUE) or not (FALSE)
- `setName(name)` Sets the service name
- `setTitle(title)` Sets the service title
- `setAbstract(abstract)` Sets the service abstract
- `setMaintainer(maintainer)` Sets the service maintainer
- `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
- `setOnlineResource(onlineResource)` Sets the online resource
- `setSchemaBaseURL(schemaBaseURL)` Sets the schema base URL. Default is http://schemas.opengis.net
- `setVerbose(verbose)` Sets verbose

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
settings <- GSServiceSettings$new(service = "WMS")
settings$setEnabled(TRUE)
```

`GSShapefileDataStore` *Geoserver REST API ShapeFileDataStore*

Description

Geoserver REST API ShapeFileDataStore

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")
```

`GSShapefileDirectoryDataStore` *Geoserver REST API ShapeFileDirectoryDataStore*

Description

Geoserver REST API ShapeFileDirectoryDataStore

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer Shapefile directory dataStore

Methods

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

```
GSShapefileDirectoryDataStore$new(dataStore="ds", description = "des",
                          enabled = TRUE, url = "file://data")
```

`GSStyleManager` *Geoserver REST API Style Manager*

Description

Geoserver REST API Style Manager

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

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.

`setBbox(minx, miny, maxx, maxy, bbox, crs)` Creates an list object representing a bbox. Either from coordinates or from a bbox object (matrix).

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.

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)
```

GSVirtualTable *Geoserver REST API GSVirtualTable*

Description

Geoserver REST API GSVirtualTable

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer virtual table

Fields

`name` name

`sql` SQL statement

`escapeSql` escape SQL?

`keyColumn` key column

`geometry` geometry

`parameters` list of virtual parameters

Methods

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

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

`encode()` This method is used to encode a GSVirtualTable to XML

`setName(name)` Sets the name of the virtual table

`setSql(sql)` Sets the sql of the virtual table `setEscapeSql(escapeSql)` Sets the escapeSql. Default is FALSE

`setKeyColumn(keyColumn)` Sets the keyColumn. Name of the column to be the primary key

`setGeometry(vtg)` Sets the virtual table geometry

`addParameter(vtp)` Adds a virtual table parameter

`delParameter(param)` Removes a virtual table parameter.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSVirtualTable$new()
```

`GSVirtualTableGeometry` *Geoserver REST API GSVirtualTableGeometry*

Description

Geoserver REST API GSVirtualTableGeometry

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer virtual table geometry

Fields

`name` geometry name

`type` geometry type

`srid` geometry SRID

Methods

`new(xml, name, type, srid)` This method is used to instantiate a GSVirtualTableGeometry

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

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

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSVirtualTableGeometry$new(name = "work", type = "MultiPolygon", srid = 4326)
```

GSVirtualTableParameter

Geoserver REST API GSVirtualTableParameter

Description

Geoserver REST API GSVirtualTableParameter

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer virtual table parameter

Fields

`name` parameter name

`defaultValue` parameter default value

`regexpValidator` parameter regexp validator

Methods

`new(xml, name, defaultValue, regexpValidator` This method is used to instantiate a GSVirtualTableParameter

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

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

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
GSVirtualTableParameter$new(name = "fieldname", defaultValue = "default_value",
                            regexpValidator = "someregexp")
```

`GSWorkspace` *Geoserver REST API Workspace*

Description

Geoserver REST API Workspace

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

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

`getWorkspaceSettings(ws)` Get the workspace settings (if existing) as object of class `GSWorkspaceSettings`

`createWorkspaceSettings(ws, workspaceSettings)` Creates a workspace settings for the workspace `ws`

`updateWorkspaceSettings(ws, workspaceSettings)` Updates a workspace settings for the workspace `ws`

`deleteWorkspaceSettings(ws)` Deletes a workspace settings for the workspace `ws`

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

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

## End(Not run)
```

`GSWorkspaceSettings` *Geoserver REST API Workspace Setting*

Description

Geoserver REST API Workspace Setting

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a GeoServer workspace settings

Methods

`new(xml)` This method is used to instantiate a `GSWorkspaceSettings`. This settings object is required to activate a workspace configuration, using the method `GSManager$createWorkspaceSettings`. Supported from GeoServer 2.12

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

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

`setCharset(charset)` Set charset

`setNumDecimals(numDecimals)` Set number of decimals

`setOnlineResource(onlineResource)` Set the online resource

`setVerbose(verbose)` Set verbose

`setVerboseExceptions(verboseExceptions)` Set verbose exceptions

`setLocalWorkspaceIncludesPrefix(includesPrefix)` Set if the Local workspace includes prefix

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
settings <- GSWorkspaceSettings$new()
settings$setCharset("UTF-8")
settings$setNumDecimals(5)
```

Index