genropy–url

API Documentation

November 23, 2007

Contents

1 Package gnr

1.1 Modules

2 Package gnr.app

3 Module gnr.app.gnrapp

gnrapp

3.1 Class GnrMixinObj

object
└ **gnr.app.gnrapp.GnrMixinObj**

3.1.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

3.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

3.2 Class GnrSqlAppDb

object
gnr.core.gnrlang.GnrObject
gnr.sql.gnrsql.GnrSqlDb
gnr.app.gnrapp.GnrSqlAppDb

3.2.1 Methods

checkTransactionWritable(*self*, *tblobj*)

delete(*self*, *tblobj*, *record*)
Delete a record from the table.

Overrides: gnr.sql.gnrsql.GnrSqlDb.delete extit(inherited documentation)

update(*self*, *tblobj*, *record*)
Update a record of the table.

Overrides: gnr.sql.gnrsql.GnrSqlDb.update extit(inherited documentation)

insert(*self*, *tblobj*, *record*)
Insert a record in the table.

Overrides: gnr.sql.gnrsql.GnrSqlDb.insert extit(inherited documentation)

__del__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *implementation*=`'sqlite'`, *dbname*=`'mydb'`, *host*=`None`, *user*=`None`, *password*=`None`, *port*=`None`, *main_schema*=`None`)

This is the constructor method of the GnrSqlDb class.

Parameters

`implementation:`	'sqlite' or 'postgres' or other sql implementations.
`dbname:`	the name for your db.
`host:`	the database server host (for sqlite is None)
`user:`	a database user's name (for sqlite is None)
`password:`	the user's password (for sqlite is None)
`port:`	the connection port (for sqlite is None)
`main_schema:`	the database main_schema

Overrides: gnr.core.gnrlang.GnrObject.__init__

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

analyze(*self*)

Analyze db

checkDb(*self*, *applyChanges*=`False`)

Check if there the database structure is compatible with the current model

Parameters

`applyChanges:` boolean. If True, all the changes are executed and committed

closeConnection(*self*)

Close a connection

commit(*self*)

Commit a transaction

connection(*self*)

property .connection If there's not connection open and return connection to database

createDb(*self*, *name*, *encoding*=`'unicode'`)

Create a db with given name and encoding @param name @param encoding

createSchema(*self*, *name*)

Create a db with given name and encoding @param name @param encoding

dropDb(*self*, *name*)

Drop a db with given name @param name

execute(*self*, *sql*, *sqlargs*=`None`, *cursor*=`None`, *cursorname*=`None`, *autocommit*=`False`)

Execute the sql statement using given kwargs

importModelFromDb(*self*)

Load the model.src extracting it from the database's information schema.

importXmlData(*self*, *path*)

Populates a database from an xml file

Parameters

`path:` filepath

listen(*self*, **args*, ***kwargs*)

Listen for a database event (postgres)

loadModel(*self*, *source*=`None`)

Load the model.src from a xml source

Parameters

`source:` xml model (diskfile or text or url)

mixin(*self*, *cls*, ***kwargs*)

notify(*self*, **args*, ***kwargs*)

Database Notify

package(*self*, *pkg*)

Returns a package object

Parameters

`pkw:` package name

packageMixin(*self*, *name*, *obj*)

Register a mixin for a package.

Parameters

`name:` the target package's name
`obj:` a class or an object to mixin

packageSrc(*self*, *name*)

Return a DbModelSrc corresponding to the required package

Parameters

`name:` the package name

packages(*self*)

Returns a package object

Parameters

`pkw:` package name

query(*self*, *table*, ***kwargs*)

See gnrsqltable.SqlTable.query

rollback(*self*)

Rollback a transaction

saveModel(*self*, *path*)

Save the current model as xml file at path

Parameters

`path:` the file path

startup(*self*)

Build the model.obj from the model.src

table(*self*, *tblname*, *pkg*=`None`)

returns a table object

Parameters

`table:` table name
`pkg:` package name

tableMixin(*self*, *tblpath*, *obj*)

Register an object or a class to mixin to a table.

Parameters

`name:` the target package's name
`obj:` a class or an object to mixin

vacuum(*self*)

Analyze db

3.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

3.2.3 Class Variables

Name	Description
application	**Value:** `property(_get_application, _set_application)`

3.3 Class GnrPackage

object
└ **gnr.app.gnrapp.GnrPackage**

3.3.1 Methods

__init__(*self*, *id*, *application*, *path*=`None`, *filename*=`None`, ***pkgattrs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

loadTableMixinDict(*self*, *module*, *folder*)

config_attributes(*self*)

getResource(*self*, *path*, *locale*=`None`)

configure(*self*)

Build db structure in this order:

- package config_db.xml
- custom package config_db.xml
- customized Table objects (method config_db)
- customized Table objects (method config_db_custom)
- customized Package objects (method config_db)
- customized Package objects (method config_db_custom)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

3.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

3.3.3 Class Variables

Name	Description
application	**Value:** `property(_get_application, _set_application)`

3.4 Class GnrApp

object
└
gnr.app.gnrapp.GnrApp

3.4.1 Methods

__init__(*self*, *instanceFolder*, *custom_config*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

init(*self*)

onIniting(*self*)

onInited(*self*)

getResource(*self*, *pkg*, *path*, *locale*=`None`)

guestLogin(*self*)

newUserUrl(*self*)

getAvatar(*self*, *username*, *password*=`None`, *authenticate*=`False`)

auth_xml(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

auth_py(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

auth_sql(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

checkPassword(*self*, *login_pwd*, *authenticate*=`False`, *defaultTags*=`None`, ***kwargs*)

makeAvatar(*self*, ***kwargs*)

checkResourcePermission(*self*, *pageTags*, *userTags*)

checkDb(*self*)

applyChangesToDb(*self*)

realPath(*self*, *path*)

onWebPageCreation(*self*, *page*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

3.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

3.5 Class GnrAvatar

object
└
gnr.app.gnrapp.GnrAvatar

3.5.1 Methods

__init__(*self*, *id*, *tags=''*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

3.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

3.6 Class GnrWriteInReservedTableError

exceptions.Exception
└
gnr.app.gnrapp.GnrWriteInReservedTableError

3.6.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

4 Module gnr.app.gnrtransactiond

4.1 Class GnrAppTransactionAgent

object
gnr.app.gnrapp.GnrApp
gnr.app.gnrtransactiond.GnrAppTransactionAgent

4.1.1 Methods

onInited(*self*)
Overrides: gnr.app.gnrapp.GnrApp.onInited

loop(*self*)

checkTransactions(*self*, *notify*=`None`)

expandTransaction(*self*, *transaction*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *instanceFolder*, *custom_config*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

applyChangesToDb(*self*)

auth_py(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

auth_sql(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

auth_xml(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

checkDb(*self*)

checkPassword(*self*, *login_pwd*, *authenticate*=`False`, *defaultTags*=`None`, ***kwargs*)

checkResourcePermission(*self*, *pageTags*, *userTags*)

getAvatar(*self*, *username*, *password*=`None`, *authenticate*=`False`)

getResource(*self*, *pkg*, *path*, *locale*=`None`)

guestLogin(*self*)

init(*self*)

makeAvatar(*self*, ***kwargs*)

newUserUrl(*self*)

onIniting(*self*)

onWebPageCreation(*self*, *page*)

realPath(*self*, *path*)

4.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

5 Module gnr.app.gnrtransactionmanager

5.1 Class TransactionManagerError

exceptions.Exception
└
gnr.app.gnrtransactionmanager.TransactionManagerError

5.1.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

6 Package gnr.core

6.1 Modules

- **gnrlist**: Some useful operations on lists.
  (Section 12, p. 111)
- **gnrmail** *(Section 15, p. 120)*
- **gnrsys**: sys
  (Section 18, p. 146)

6.2 Variables

Name	Description
version_info	**Value:** `(0, 0, 1)`
__version__	**Value:** `'0.0.1'`

7 Module gnr.core.gnrbag

The gnrbag module contains a single class intended to be used: Bag

A Bag is a generic container object, similar to a dictionary, with some useful properties:

- ordered
- accessible by key
- iterable
- hierarchical

A Bag can store any kind of python object with a label in an ordered list. It can also store attributes about any value stored: bag attributes are metadata, do not interfere with the stored object.

A Bag can be loaded and saved in some ways:

- saved to and loaded from pikle files or strings: the object stored in the bag must be picklable of course.
- saved to and loaded from xml files or strings with a specific syntax: the object stored in the bag must be strings, numbers or dates.
- loaded from a generic xml file preserving the whole hierarchical structure and the attributes: all values will be of type string of course.
- loaded from a generic html file: requires tidy.

Another class you could have to deal with is BagNode: they are Bag elements, basically a Bag is a list of BagNode-s.

BagNode is not intended to be instanced directly, it is used internally by Bag. However in some cases is useful to interact with BagNode instances inside a Bag.

7.1 Functions

testFormule()

testfunc(**_kwargs_)

7.2 Variables

Name	Description
logger	**Value:** `logging.getLogger('gnr.core.gnrbag')`

7.3 Class BagNodeException

exceptions.Exception
└
gnr.core.gnrbag.BagNodeException

7.3.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

7.4 Class BagException

exceptions.Exception
└
gnr.core.gnrbag.BagException

7.4.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

7.5 Class BagValidationError

exceptions.Exception
└
gnr.core.gnrbag.BagException
└
gnr.core.gnrbag.BagValidationError

7.5.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

7.6 Class BagDeprecatedCall

exceptions.Exception
└
gnr.core.gnrbag.BagException
└
gnr.core.gnrbag.BagDeprecatedCall

7.6.1 Methods

__init__(*self*, *errcode*, *message*)
Overrides: exceptions.Exception.__init__

__getitem__(...)

__str__(...)

7.7 Class BagNode

object
└ **gnr.core.gnrbag.BagNode**

```
BagNode is the element type which a Bag is composed of. That's why it's
possible to say that a Bag is a collection of BagNodes. A BagNode is an object
that gather within itself, three main things:
    -label: can be only a string.
    -value: can be anything, but a BagNode. Often value is a Bag.
    -attributes: dictionary that contains node's metadata
```

7.7.1 Methods

__init__(*self*, *parentbag*, *label*, *value*=`None`, *attr*=`None`, *resolver*=`None`, *validators*=`None`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Parameters

`parentbag:`	Bag than contains the node
`label:`	label that identifies the node
`value:`	value of the node
`attr:`	node's attributes
`resolver:`	a BagResolver
`validators:`	a dict with all validators pairs

Overrides: object.__init__

__eq__(*self*, *other*)

setValidators(*self*, *validators*)

fullpath(*self*)

getLabel(*self*)

Returns node's label

setLabel(*self*, *label*)

set node's label

getValue(*self*, *mode*=`''`)

Returns the value of the BagNode. This method is called by the property .value

Parameters

`mode:` can be one or more of:

- static: to get the resolver instance instead of the calculated value
- weak: to get a weak ref stored in the node instead of the actual object

Return Value

node's value

setValue(*self*, *value*, *trigger*=`True`, *_attributes*=`None`, *_updattr*=`None`)

Set the node's value, unless the node is locked. This method is called by the property .value

Parameters

`value:` the value to set the new bag inherits the trigger of the parentbag and calls it sending an update event

getStaticValue(*self*)

Get node's value in static mode

setStaticValue(*self*, *value*)

Set node's value in static mode

resetResolver(*self*)

getAttr(*self*, *label*=`None`, *default*=`None`)

this method returns the value of an attribute given it's label. If it doesn't exists returns a default value.

Parameters

`label:` the label of the attribute to get.

getInheritedAttributes(*self*)

hasAttr(*self*, *label*=`None`, *value*=`None`)

this method check if the node has the given pair label-value in its attributes

setAttr(*self*, *attr*=`None`, *trigger*=`True`, *_updattr*=`True`, ***kwargs*)

this method receives one or more key-value couple, passed as a dict or as named parameters, and sets them as attributes of the node @param attr the dict of attributes to set into the node.

delAttr(*self*, **attrToDelete*)

this method receives one or more attributes' labels and removes them from the node's attributes

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)
repr(x)

Overrides: object.__repr__ extit(inherited documentation)

asTuple(*self*)

addValidator(*self*, *validator*, *parameterString*)

this method set a new validator into the BagValidationList of the node. If there are no validators into the node then addValidator instantiate a new BagValidationList and append the validator to it.

Parameters

`validator:` the type of validation to set into the list of the node.
`paremeterString:` the parameters for a single validation type.

removeValidator(*self*, *validator*)

getValidatorData(*self*, *validator*, *label*=`None`, *dflt*=`None`)

subscribe(*self*, *subscriberId*, *callback*)

unsubscribe(*self*, *subscriberId*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

7.7.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.7.3 Class Variables

Name	Description
parentbag	**Value:** `property(_get_parentbag, _set_parentbag)`
value	**Value:** `property(getValue, setValue)`
staticvalue	**Value:** `property(getStaticValue, setStaticValue)`
resolver	**Value:** `property(_get_resolver, _set_resolver)`

7.8 Class Bag

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrbag.Bag

A container object like a dictionary, but ordered. Nested elements can be accessed with a path of keys joined with dots.

7.8.1 Methods

__init__(*self*, *source*=`None`)

A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrlang.GnrObject.__init__

fullpath(*self*)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

sum(*self*, *what*=`'#v'`)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

__iter__(*self*)

__len__(*self*)

__call__(*self*, *what*=`None`)

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value
a formatted representation of the bag contents (unicode)

Overrides: object.__str__

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters
`encoding:` default is 'UTF-8'

Return Value
a formatted representation of the bag contents (ascii)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value
a list containing all the keys of the Bag.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value
a list containing all the values of the Bag.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value
a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

popNode(*self*, *path*)

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

clear(*self*)

This method clears the Bag.

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

__eq__(*self*, *other*)

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

copy(*self*)

This method returns a copy of the Bag.

Return Value

a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value

a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

delAttr(*self*, *path*=`None`, *attr*=`None`)

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value

a Dictionary equivalent to the given Bag.

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
- `_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

- `kwargs:` a dict of symbol to define for a formula.

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters

- `kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

- `formula:` a string that represents the expression with symbolic vars
- `kwargs:` links between symbols and paths associated to their values

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

- `path:` path of the node.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

- `path:` path of the node.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

`path:` path of the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

makePicklable(*self*)

This method make a Bag picklable.

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

backref(*self*)

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:`	path of the node.
`validator:`	the type of validation.
`parameterString:`	string which contains the params for validation.

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

`subscriberId:`	an ID can be assigned for a subscription
`update:`	the eventhandler function linked to update event.
`insert:`	the eventhandler function linked to insert event.
`delete:`	the eventhandler function linked to delete event.
`any:`	the eventhandler function linked to do whenever something happens.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:`	an ID can be assigned for a subscription
`update:`	the eventhandler function to remove
`insert:`	the eventhandler function to remove
`delete:`	the eventhandler function to remove
`any:`	the eventhandler function to remove

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback:`	the function which is called.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

mixin(*self*, *cls*, ***kwargs*)

7.8.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.8.3 Class Variables

Name	Description
parent	**Value:** `property(_get_parent, _set_parent)`
node	**Value:** `property(_get_node, _set_node)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`
modified	**Value:** `property(_get_modified, _set_modified)`

7.9 Class BagValidationList

object
└
gnr.core.gnrbag.BagValidationList

This class provides the validation system for a BagNode. This is a list of validators related to a BagNode. This class is used only from a the Bag's and BagNode's accessor methods addValidator and removeValidator. All the methods of this class must be considered private.

7.9.1 Methods

__init__(*self*, *parentNode*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

getdata(*self*, *validator*, *label*=`None`, *dflt*=`None`)

This method get the validatorsdata of a validator.

add(*self*, *validator*, *parameterString*)

This method add a new validator to the BagValidationList.

Parameters

`validator:`	type of validator
`parameterString:`	the string that contains the parameters for the validators.

remove(*self*, *validator*)

This method remove a validator

__call__(*self*, *value*, *oldvalue*)

This method apply the validation to a BagNode value.

validate_case(*self*, *value*, *oldvalue*, *parameterString*)

This method is set a validation for the case of a string value.

Parameters

`parameterString:`	this can be 'upper' 'lower' 'capitalize'

validate_inList(*self*, *value*, *oldvalue*, *parameterString*)

validate_hostaddr(*self*, *value*, *oldvalue*)

This method provides a validaton for Host address value

validate_length(*self*, *value*, *oldvalue*, *parameterString*)

This method provides a validaton for the length of a string value

coerceFromText(*self*, *value*)

validate_db(*self*, *value*, *oldvalue*, *parameterString*)

defaultExt(*self*, *value*, *oldvalue*, *parameterString*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

7.9.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.9.3 Class Variables

Name	Description
node	**Value:** `property(_get_node, _set_node)`

7.10 Class BagResolver

object
└
gnr.core.gnrbag.BagResolver

BagResolver is an abstract class, that defines the interface for a new kind of dynamic objects. By "Dynamic" we mean, properties that are calculated in real-time but looks like static ones.

7.10.1 Methods

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__eq__(*self*, *other*)

instanceKwargs(*self*)

reset(*self*)

expired(*self*)

__call__(*self*, ***kwargs*)

load(*self*)

must be reimplemented

init(*self*)

resolverSerialize(*self*)

__getitem__(*self*, *k*)

keys(*self*)

items(*self*)

values(*self*)

digest(*self*, *k*=`None`)

sum(*self*, *k*=`None`)

iterkeys(*self*)

iteritems(*self*)

itervalues(*self*)

__iter__(*self*)

__contains__(*self*)

__len__(*self*)

getAttributes(*self*)

setAttributes(*self*, *attributes*)

resolverDescription(*self*)

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

7.10.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.10.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True}`
classArgs	**Value:** `[]`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
attributes	**Value:** `property(getAttributes, setAttributes)`

7.11 Class BagCbResolver

object

gnr.core.gnrbag.BagResolver

gnr.core.gnrbag.BagCbResolver

This is a standard resolver. It calls a callback method, passing its kwargs parameters

7.11.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.11.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.11.3 Class Variables

Name	Description
classArgs	**Value:** `['method']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True}`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

7.12 Class UrlResolver

object
gnr.core.gnrbag.BagResolver
gnr.core.gnrbag.UrlResolver

7.12.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.12.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.12.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 300, 'readOnly': True}`
classArgs	**Value:** `['url']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

7.13 Class DirectoryResolver

object

gnr.core.gnrbag.BagResolver

gnr.core.gnrbag.DirectoryResolver

7.13.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

makeLabel(*self*, *name*, *ext*)

processor_directory(*self*, *path*)

processor_xml(*self*, *path*)

processor_html(*self*, *path*)

processor_txt(*self*, *path*)

processor_default(*self*, *path*)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.13.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.13.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 500, 'readOnly': True, 'invisible': False, ...`
classArgs	**Value:** `['path', 'relocate']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

7.14 Class TxtDocResolver

object

gnr.core.gnrbag.BagResolver

gnr.core.gnrbag.TxtDocResolver

7.14.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.14.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.14.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 500, 'readOnly': True}`
classArgs	**Value:** `['path']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

7.15 Class XmlDocResolver

object

gnr.core.gnrbag.BagResolver

gnr.core.gnrbag.XmlDocResolver

7.15.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.15.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.15.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 500, 'readOnly': True}`
classArgs	**Value:** `['path']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

7.16 Class BagFormula

object
gnr.core.gnrbag.BagResolver
gnr.core.gnrbag.BagFormula

This resolver calculates the value of an algebric espression

7.16.1 Methods

__init__(*self*)

Parameters

`root:`
`expr:` expression with symbolic terms
`symbols:`

Overrides: gnr.core.gnrbag.BagResolver.init

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.16.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.16.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'formula': '', 'parameters': None, 'read...`
classArgs	**Value:** `['formula', 'parameters']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

7.17 Class BagResolverNew

object
└ **gnr.core.gnrbag.BagResolverNew**

docstring for BagResolver

7.17.1 Methods

__init__(*self*, *cacheTime*=`0`, *readOnly*=`True`, *serializerStore*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__eq__(*self*, *other*)

reset(*self*)

expired(*self*)

__call__(*self*, ***kwargs*)

load(*self*)

must be reimplemented

init(*self*)

resolverSerialize(*self*)

__getitem__(*self*, *k*)

keys(*self*)

items(*self*)

values(*self*)

digest(*self*, *k*=`None`)

sum(*self*, *k*=`None`)

iterkeys(*self*)

iteritems(*self*)

itervalues(*self*)

__iter__(*self*)

__contains__(*self*)

__len__(*self*)

getAttributes(*self*)

setAttributes(*self*, *attributes*)

resolverDescription(*self*)

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

7.17.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.17.3 Class Variables

Name	Description
serializerStore	**Value:** `property(_get_serializerStore, _set_serializerStore)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
attributes	**Value:** `property(getAttributes, setAttributes)`

7.18 Class TraceBackResolver

object
gnr.core.gnrbag.BagResolver
gnr.core.gnrbag.TraceBackResolver

7.18.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

7.18.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

7.18.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'limit': None}`
classArgs	**Value:** `[]`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

8 Module gnr.core.gnrbagxml

8.1 Variables

Name	Description
REGEX_XML_ILLEGAL	**Value:** `re.compile(r'<|>|&')`

8.2 Class BagFromXml

object
└ **gnr.core.gnrbagxml.BagFromXml**

8.2.1 Methods

build(*self*, *source*, *fromFile*, *catalog*=`None`, *bagcls*=`Bag`, *empty*=`None`)

do_build(*self*, *source*, *fromFile*, *catalog*=`None`, *bagcls*=`Bag`, *empty*=`None`, *testmode*=`False`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

8.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

8.3 Class BagToXml

object
└
gnr.core.gnrbagxml.BagToXml

8.3.1 Methods

nodeToXmlBlock(*self*, *node*)

This method handles all the different node types, calls the method build tag and returns its result.

Return Value

the XML tag that represent self BagNode.

bagToXmlBlock(*self*, *bag*)

This method returns an XML block version of the Bag. The XML block version of the Bag uses XML attributes for an efficient representation of types: If the element-leaf is a simple string, there are no type attributes in the corresponding XML nodes otherwise a 'T' attribute is set to the node and the value of 'T' changes in function of the type (value of 'T' is 'B' for boolean, 'L' for integer, 'R' for float, 'D' for date, 'H' for time).

Return Value

the Bag represented as XML block in a List.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag['aa.cc']='test'
>>> mybag.toXmlBlock()
['<aa>', u'<cc>test</cc>', u'<bb T="L">4567</bb>', '</aa>']
```

build(*self*, *bag*, *filename*=`None`, *encoding*=`'UTF-8'`, *catalog*=`None`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> ; the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'

`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?><GenRoBag><aa><bb T="L">4567</bb></aa></GenRoBag>'
```

buildTag(*self*, *tagName*, *value*, *attributes*=`None`, *cls*=`''`, *xmlMode*=`False`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

8.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

9 Module gnr.core.gnrclasses

9.1 Class GnrClassCatalog

object
└ **gnr.core.gnrclasses.GnrClassCatalog**

9.1.1 Methods

convert(*cls*)

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

addClass(*self*, *cls*, *key*, *aliases*=`None`, *altcls*=`None`, *align*=`'L'`, *empty*=`None`)

Add a python class to the list of objects known by the Catalog. cls: the class itself by reference key: a string, is a short name of the class, as found in textual values to parse or write altcls: other classes to write in the same way. All values will be parsed with the main class. aliases: other keys to parse using this class empty: the class or value to be used for empty parsed values, defualt None, example " for strings

getEmpty(*self*, *key*)

getAlign(*self*, *key*)

addSerializer(*self*, *mode*, *cls*, *funct*)

```
Given a mode and a class to convert, specifies the function to use for the actual conversion:
funct: is a function by reference or lambda,
       will receive an instance and return an appropriate value for the conversion mode
```

addParser(*self*, *cls*, *funct*)

```
Given a class to convert, specifies the function to use for the actual conversion from text:
funct: is a function by reference or lambda,
       will receive a text and return an instance
```

getClassKey(*self*, *o*)

getClass(*self*, *name*)

asText(*self*, *o*, *quoted*=`False`)

quoted(*self*, *s*)

fromText(*self*, *txt*, *clsname*)

fromTypedText(*self*, *txt*)

asTypedText(*self*, *o*, *quoted*=`False`)

asTextAndType(*self*, *o*)

getType(*self*, *o*)

standardClasses(*self*)

parse_float(*self*, *txt*)

parse_date(*self*, *txt*)

parse_time(*self*, *txt*)

toJson(*self*, *data*)

fromJson(*self*, *data*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

9.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

10 Module gnr.core.gnrdict

10.1 Class GnrDict

object
dict
gnr.core.gnrdict.GnrDict

An ordered dictionary

10.1.1 Methods

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Return Value

```
new empty dictionary
```

Overrides: dict.__init__ extit(inherited documentation)

__setitem__(*self*, *key*, *value*)
x[i]=y

Overrides: dict.__setitem__ extit(inherited documentation)

__iter__(*self*)
iter(x)

Overrides: dict.__iter__ extit(inherited documentation)

__delitem__(*self*, *key*)
del x[y]

Overrides: dict.__delitem__ extit(inherited documentation)

get(*self*, *label*, *default*=`None`)
d defaults to None.

Return Value

```
D[k] if k in D, else d
```

Overrides: dict.get extit(inherited documentation)

__getitem__(*self*, *label*)
x[y]

Overrides: dict.__getitem__ extit(inherited documentation)

items(*self*)
Return Value

```
list of D's (key, value) pairs, as 2-tuples
```

Overrides: dict.items extit(inherited documentation)

keys(*self*)
Return Value
`list of D's keys`

Overrides: dict.keys extit(inherited documentation)

index(*self*, *value*)

values(*self*)
Return Value
`list of D's values`

Overrides: dict.values extit(inherited documentation)

pop(*self*, *key*, *dflt*=`None`)
If key is not found, d is returned if given, otherwise KeyError is raised

Return Value
`v, remove specified key and return the corresponding value`

Overrides: dict.pop extit(inherited documentation)

__str__(*self*)
repr(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)
repr(x)

Overrides: dict.__repr__ extit(inherited documentation)

clear(*self*)
Remove all items from D.

Return Value
`None`

Overrides: dict.clear extit(inherited documentation)

update(*self*, *o*, *removeNone*=`False`)
Update D from E and F: for k in E: D[k] = E[k] (if E has keys else: for (k, v) in E: D[k] = v) then: for k in F: D[k] = F[k]

Return Value
`None`

Overrides: dict.update extit(inherited documentation)

copy(*self*)
Return Value
`a shallow copy of D`

Overrides: dict.copy extit(inherited documentation)

setdefault(*key*, *d*=`None`)
Return Value
`D.get(k,d), also set D[k]=d if k not in D`

Overrides: dict.setdefault extit(inherited documentation)

popitem(*self*)
2-tuple; but raise KeyError if D is empty

Return Value
`(k, v), remove and return some (key, value) pair as a`

Overrides: dict.popitem extit(inherited documentation)

iteritems(*self*)
Return Value
`an iterator over the (key, value) items of D`

Overrides: dict.iteritems extit(inherited documentation)

iterkeys(*self*)
Return Value
`an iterator over the keys of D`

Overrides: dict.iterkeys extit(inherited documentation)

itervalues(*self*)
Return Value
`an iterator over the values of D`

Overrides: dict.itervalues extit(inherited documentation)

__add__(*self*, *o*)

__sub__(*self*, *o*)

__getslice__(*self*, *start*=`None`, *end*=`None`)

__setslice__(*self*, *start*=`None`, *end*=`None`, *val*=`None`)

reverse(*self*)

sort(*self*, *cmpfunc*=`None`)

__cmp__(*x*, *y*)

cmp(x,y)

__contains__(*D*, *k*)
Return Value
`True if D has a key k, else False`

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*x*, *y*)

x==y

__ge__(*x*, *y*)

x>=y

__getattribute__(...)

x.__getattribute__('name') <==> x.name

Overrides: object.__getattribute__

__gt__(*x*, *y*)

x>y

__hash__(*x*)

hash(x)

Overrides: object.__hash__

__le__(*x*, *y*)

x<=y

__len__(*x*)

len(x)

__lt__(*x*, *y*)

x<y

__ne__(*x*, *y*)

x!=y

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

Overrides: object.__new__

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

fromkeys(*dict*, *S*, *v*=...)

v defaults to None.

Return Value

`New dict with keys from S and values equal to v`

has_key(*D*, *k*)

Return Value

`True if D has a key k, else False`

10.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

10.2 Class GnrNumericDict

object
dict
gnr.core.gnrdict.GnrDict
gnr.core.gnrdict.GnrNumericDict

10.2.1 Methods

__getitem__(*self*, *label*)

x[y]

Overrides: gnr.core.gnrdict.GnrDict.__getitem__

__iter__(*self*)

iter(x)

Overrides: gnr.core.gnrdict.GnrDict.__iter__

__add__(*self*, *o*)

__cmp__(*x*, *y*)

cmp(x,y)

__contains__(*D*, *k*)
Return Value

```
True if D has a key k, else False
```

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *key*)
del x[y]

Overrides: dict.__delitem__ extit(inherited documentation)

__eq__(*x*, *y*)

x==y

__ge__(*x*, *y*)

x>=y

__getattribute__(...)

x.__getattribute__('name') <==> x.name

Overrides: object.__getattribute__

__getslice__(*self*, *start*=`None`, *end*=`None`)

__gt__(*x*, *y*)

x>y

__hash__(*x*)

hash(x)

Overrides: object.__hash__

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Return Value

```
new empty dictionary
```

Overrides: dict.__init__ extit(inherited documentation)

__le__(*x*, *y*)

x<=y

__len__(*x*)

len(x)

__lt__(*x*, *y*)

x<y

__ne__(*x*, *y*)

x!=y

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

Overrides: object.__new__

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*self*)

repr(x)

Overrides: dict.__repr__ extit(inherited documentation)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *key*, *value*)

x[i]=y

Overrides: dict.__setitem__ extit(inherited documentation)

__setslice__(*self*, *start*=`None`, *end*=`None`, *val*=`None`)

__str__(*self*)

repr(x)

Overrides: object.__str__ extit(inherited documentation)

__sub__(*self*, *o*)

clear(*self*)
Remove all items from D.

Return Value
`None`

Overrides: dict.clear extit(inherited documentation)

copy(*self*)
Return Value
`a shallow copy of D`

Overrides: dict.copy extit(inherited documentation)

fromkeys(*dict*, *S*, *v*=`...`)

v defaults to None.

Return Value
`New dict with keys from S and values equal to v`

get(*self*, *label*, *default*=`None`)
d defaults to None.

Return Value
`D[k] if k in D, else d`

Overrides: dict.get extit(inherited documentation)

has_key(*D*, *k*)
Return Value
`True if D has a key k, else False`

index(*self*, *value*)

items(*self*)
Return Value
`list of D's (key, value) pairs, as 2-tuples`

Overrides: dict.items extit(inherited documentation)

iteritems(*self*)
Return Value
`an iterator over the (key, value) items of D`

Overrides: dict.iteritems extit(inherited documentation)

iterkeys(*self*)
Return Value
`an iterator over the keys of D`

Overrides: dict.iterkeys extit(inherited documentation)

itervalues(*self*)
Return Value
`an iterator over the values of D`

Overrides: dict.itervalues extit(inherited documentation)

keys(*self*)
Return Value
`list of D's keys`

Overrides: dict.keys extit(inherited documentation)

pop(*self*, *key*, *dflt*=`None`)
If key is not found, d is returned if given, otherwise KeyError is raised

Return Value
`v, remove specified key and return the corresponding value`

Overrides: dict.pop extit(inherited documentation)

popitem(*self*)
2-tuple; but raise KeyError if D is empty

Return Value
`(k, v), remove and return some (key, value) pair as a`

Overrides: dict.popitem extit(inherited documentation)

reverse(*self*)

setdefault(*key*, *d*=`None`)
Return Value
`D.get(k,d), also set D[k]=d if k not in D`

Overrides: dict.setdefault extit(inherited documentation)

sort(*self*, *cmpfunc*=`None`)

update(*self*, *o*, *removeNone*=`False`)
Update D from E and F: for k in E: D[k] = E[k] (if E has keys else: for (k, v) in E: D[k] = v) then: for k in F: D[k] = F[k]

Return Value
`None`

Overrides: dict.update extit(inherited documentation)

values(*self*)
Return Value
`list of D's values`

Overrides: dict.values extit(inherited documentation)

10.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11 Module gnr.core.gnrlang

11.1 Functions

getUuid()

safe_dict(*d*)

moduleDict(*module*, *proplist*)

gnrImport(*source*, *importAs*=`None`)

addCallable(*obj*, *method*)

addBoundCallable(*obj*, *method*, *importAs*=`None`)

setMethodFromText(*obj*, *src*, *importAs*)

getObjCallables(*obj*)

getObjAttributes(*obj*)

callables(*obj*)

testbound(*self*, *n*)

compareInstances(*a*, *b*, *__visited*=`None`)

args(**args*, ***kwargs*)

setCallable(*obj*, *name*, *argstring*=`None`, *func*=`'pass'`)

instanceMixin(*obj*, *source*, *methods*=`None`, ***kwargs*)

Add to the instance obj methods from 'source'. Source can be an instance or a class If not 'methods' all methods are added.

safeStr(*self*, *o*)

instanceOf(*obj*, **args*, ***kwargs*)

errorLog(*proc_name*, *host*=`None`, *from_address*=`''`, *to_address*=`None`, *user*=`None`, *password*=`''`)

11.2 Class GnrException

exceptions.Exception
└
gnr.core.gnrlang.GnrException

11.2.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

11.3 Class GnrObject

object
└
gnr.core.gnrlang.GnrObject

11.3.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

mixin(*self*, *cls*, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

11.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.4 Class GnrImportedModule

object
└
gnr.core.gnrlang.GnrImportedModule

11.4.1 Methods

__init__(*self*, *source*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

getPath(*self*)

getModule(*self*)

getName(*self*)

getDoc(*self*, *memberName*=`None`)

getMember(*self*, *memberName*)

getImportedMember(*self*, *memberName*)

load(*self*)

update(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

11.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.5 Class GnrAddOn

object
└ **gnr.core.gnrlang.GnrAddOn**

A class to be subclassed to inherit some introspection methods

11.5.1 Methods

className(*self*)

recorderReset(*self*)

recorderWrite(*self*)

recorderGet(*self*)

recorderDo(*self*, *recorder*=`None`)

superdo(*self*, **args*, ***kwargs*)

like calling super() with the right arguments **** verificare se funziona a piu livelli

dosuper(*self*, **args*, ***kwargs*)

like calling super() with the right arguments **** verificare se funziona a piu livelli

setCallable(*self*, *src*, *importAs*=`None`, *bound*=`True`)

Parameters

`src:`	is a string of a python function or an imported function
`importAs:`	a name for identify the function in error messages
`bound:`	if true the function will be bounded to this instance

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

11.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.6 Class GnrRemeberableAddOn

object

gnr.core.gnrlang.GnrAddOn

gnr.core.gnrlang.GnrRemeberableAddOn

11.6.1 Methods

__del__(*self*)

rememberMe(*self*, *name*=`None`)

rememberedMembers(*cls*)

rememberedNamedMembers(*cls*)

rememberedGet(*cls*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

className(*self*)

dosuper(*self*, **args*, ***kwargs*)

like calling super() with the right arguments **** verificare se funziona a piu livelli

recorderDo(*self*, *recorder*=`None`)

recorderGet(*self*)

recorderReset(*self*)

recorderWrite(*self*)

setCallable(*self*, *src*, *importAs*=`None`, *bound*=`True`)

Parameters

`src:` is a string of a python function or an imported function
`importAs:` a name for identify the function in error messages
`bound:` if true the function will be bounded to this instance

superdo(*self*, **args*, ***kwargs*)

like calling super() with the right arguments **** verificare se funziona a piu livelli

11.6.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.6.3 Class Variables

Name	Description
_gnr_members__	**Value:** {}
_gnr_namedmembers__	**Value:** {}
_gnr_remembered_as__	**Value:** `None`

11.7 Class GnrMetaString

object
└
gnr.core.gnrlang.GnrMetaString

11.7.1 Methods

glossary(*cls*)

__init__(*self*, *value*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__repr__(*self*)
repr(x)

Overrides: object.__repr__ extit(inherited documentation)

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

__eq__(*self*, *value*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

11.7.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.8 Class SuperdoTest

object
└ **gnr.core.gnrlang.SuperdoTest**

11.8.1 Methods

__init__(*self*, *first*, *second*, *alfa*=`'alfadef'`, *beta*=`'betadef'`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

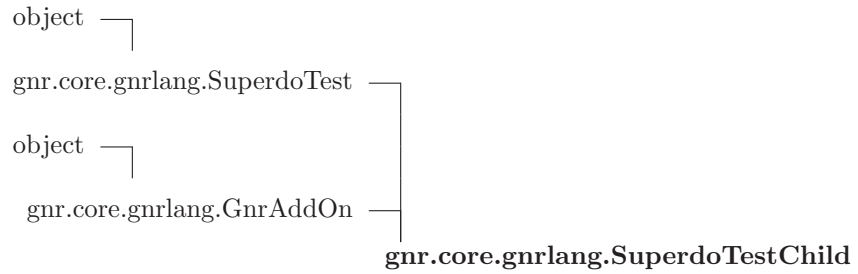
__str__(*x*)

str(x)

11.8.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.9 Class SuperdoTestChild



11.9.1 Methods

__init__(*self*, *a*, *b*, *alfa*=`'alfachildef'`, *beta*=`'betachildefd'`, *gamma*=78, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.SuperdoTest.__init__

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

className(*self*)

dosuper(*self*, **args*, ***kwargs*)

like calling super() with the right arguments **** verificare se funziona a piu livelli

recorderDo(*self*, *recorder*=`None`)

recorderGet(*self*)

recorderReset(*self*)

recorderWrite(*self*)

setCallable(*self*, *src*, *importAs*=`None`, *bound*=`True`)

Parameters

`src:`	is a string of a python function or an imported function
`importAs:`	a name for identify the function in error messages
`bound:`	if true the function will be bounded to this instance

superdo(*self*, **args*, ***kwargs*)

like calling super() with the right arguments **** verificare se funziona a piu livelli

11.9.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.10 Class SuperdoTestChildX

object

gnr.core.gnrlang.SuperdoTest

gnr.core.gnrlang.SuperdoTestChildX

11.10.1 Methods

__init__(*self*, *a*, *b*, *alfa*=`'alfachildef'`, *beta*=`'betachildefd'`, *gamma*=`78`, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.SuperdoTest.__init__

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

11.10.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.11 Class waz

object

gnr.core.gnrlang.waz

11.11.1 Methods

somma(*self*, *a*, *b*)

differenza(*self*, *a*, *b*)

stampa(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

11.11.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

11.12 Class GnrExpandible

object
└
gnr.core.gnrlang.GnrExpandible

11.12.1 Methods

__onmixin__(*self*)

addExpander(*self*, *expander*)

delExpander(*self*, *expander*)

__getattr__(*self*, *attr*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

11.12.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

12 Module gnr.core.gnrlist

Some useful operations on lists.

12.1 Functions

findByAttr(*l*, ***kwargs*)

Find elements in list "l" having attributes with names and values as kwargs items

sortByItem(*l*, **args*, ***kwargs*)

Sort the list "l", filled of objects with dict interface by items with key in *args.

Parameters

`*args:` a list of keys to sort for. Each key can be reverse sorted by adding ':rev' to the key.

`hkeys:` if True and a key contains '.' it is interpreted as a hyerarchical path and sub dict are looked for

sortByAttr(*l*, **args*)

12.2 Class GnrNamedList

object
list
gnr.core.gnrlist.GnrNamedList

A row object that allow by-colun-name access to data, the capacity to add columns and alter data.

12.2.1 Methods

__init__(*self*, *index*, *values*=`None`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Return Value

`new list`

Overrides: list.__init__ extit(inherited documentation)

__getitem__(*self*, *x*)

x[y]

Overrides: list.__getitem__ extit(inherited documentation)

__setitem__(*self*, *x*, *v*)

x[i]=y

Overrides: list.__setitem__ extit(inherited documentation)

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

__repr__(*self*)
repr(x)

Overrides: list.__repr__ extit(inherited documentation)

items(*self*)

keys(*self*)

values(*self*)

has_key(*self*, *x*)

get(*self*, *x*, *default*=`None`)

update(*self*, *d*)

iteritems(*self*)

extractItems(*self*, *columns*)

extractValues(*self*, *columns*)

__add__(*x*, *y*)

x+y

__contains__(*x*, *y*)

y in x

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*x*, *y*)

del x[y]

__delslice__(*x*, *i*, *j*)

del x[i:j]
Use of negative indices is not supported.

__eq__(*x*, *y*)

x==y

__ge__(*x*, *y*)

x>=y

__getattribute__(...)

x.__getattribute__('name') <==> x.name

Overrides: object.__getattribute__

__getslice__(*x*, *i*, *j*)

x[i:j]
Use of negative indices is not supported.

__gt__(*x*, *y*)

x>y

__hash__(*x*)

hash(x)

Overrides: object.__hash__

__iadd__(*x*, *y*)

x+=y

__imul__(*x*, *y*)

x*=y

__iter__(*x*)

iter(x)

__le__(*x*, *y*)

x<=y

__len__(*x*)

len(x)

__lt__(*x*, *y*)

x<y

__mul__(*x*, *n*)

x*n

__ne__(*x*, *y*)

x!=y

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

Overrides: object.__new__

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__reversed__(*L*)

return a reverse iterator over the list

__rmul__(*x*, *n*)

n*x

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setslice__(*x*, *i*, *j*, *y*)

x[i:j]=y
Use of negative indices is not supported.

append(*L*, *object*)

append object to end

count(*L*, *value*)

return number of occurrences of value

Return Value

```
integer
```

extend(*L*, *iterable*)

extend list by appending elements from the iterable

index(...)

L.index(value, [start, [stop]]) -> integer – return first index of value

insert(*L*, *index*, *object*)

insert object before index

pop(*L*, *index*=`...`)

remove and return item at index (default last)

Return Value

`item`

remove(*L*, *value*)

remove first occurrence of value

reverse(*L*)

reverse *IN PLACE*

sort(*L*, *cmp*=`None`, *key*=`None`, *reverse*=`False`)

stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1

12.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

13 Module gnr.core.gnrlocale

13.1 Functions

localize(*obj*, *format*=`None`, *currency*=`None`, *locale*=`None`)

localize_number(*obj*, *locale*, *format*=`None`, *currency*=`None`)

localize_date(*obj*, *locale*, *format*=`None`, ***kwargs*)

localize_datetime(*obj*, *locale*, *format*=`None`, ***kwargs*)

localize_time(*obj*, *locale*, *format*=`None`, ***kwargs*)

parselocal_number(*txt*, *locale*)

parselocal_float(*txt*, *locale*)

parselocal_decimal(*txt*, *locale*)

parselocal_date(*txt*, *locale*)

parselocal_datetime(*txt*, *locale*)

parselocal_time(*txt*, *locale*)

parselocal(*txt*, *cls*, *locale*=`None`)

return an object of class cls

13.2 Variables

Name	Description
DEFAULT_LOCALE	**Value:** `'en_US'`
TYPES_LOCALIZERS_DICT	**Value:** {`int: localize_number, float: localize_number, Decimal: l...`
TYPES_LOCALPARSERS_DICT	**Value:** {`int: parselocal_number, float: parselocal_float, Decimal...`

14 Module gnr.core.gnrlog

14.1 Variables

Name	Description
gnrlogging	**Value:** `GnrLog()`

14.2 Class GnrLogger

object
└ **gnr.core.gnrlog.GnrLogger**

14.2.1 Methods

__init__(*self*, *path*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

info(*self*, *msg*, **args*, ***kwargs*)

debug(*self*, *msg*, **args*, ***kwargs*)

warning(*self*, *msg*, **args*, ***kwargs*)

error(*self*, *msg*, **args*, ***kwargs*)

critical(*self*, *msg*, **args*, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

14.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

14.3 Class GnrLog

object
└
gnr.core.gnrlog.GnrLog

14.3.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

getLogger(*self*, *path*, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, *...*)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

14.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

15 Module gnr.core.gnrmail

15.1 Functions

sendmail(*host*, *from_address*, *to_address*, *subject*, *body*, *user*=''', *password*=''')

16 Module gnr.core.gnrstring

16.1 Functions

getUntil(*myString*, *chunk*)

Returns a string until a given chunk. @param myString

Parameters

`chunk:` substring that bounds the result string >>> getUntil('teststring', 'st') 'te' >>> getUntil('teststring', 'te') '' >>> getUntil('teststring', 'co') ''

getUntilLast(*myString*, *chunk*)

returns a string until last occurence of a given chunk @param myString

Parameters

`chunk:` substring that bounds the result string >>> getUntilLast('teststring', 'st') 'test' >>> getUntilLast('teststring', 'te') '' >>> getUntilLast('teststring', 'co') ''

getFrom(*myString*, *chunk*)

returns a string from a given chunk @param myString

Parameters

`chunk:` substring that bounds the result string >>> getFrom('teststring', 'st') 'string' >>> getFrom('teststring', 'te') 'ststring' >>> getFrom('teststring', 'co') ''

getFromLast(*myString*, *chunk*)

returns a string from last occurence of a given chunk. @param myString

Parameters

`chunk:` substring that bounds the result string >>> getFromLast('teststring', 'st') 'ring' >>> getFromLast('teststring', 'ng') '' >>> getFromLast('teststring', 'co') ''

wordSplit(*text*)

Returns a list that contains the words of the given text

Parameters

`text:` text to split

```
>>> wordSplit('hello, my dear friend')
['hello', 'my', 'dear', 'friend']
```

splitLast(*myString*, *chunk*)

returns a tuple of two strings, splitting the string at the last occurence of a given chunk. >>> splitLast('hello my dear friend', 'e') ('hello my dear fri', 'nd')

getBetween(*myString*, *startChunk*, *endChunk*)

returns a string between two given chunks. @param myString

Parameters

`startChunk:` substring that bounds the result string from left
`startChunk:` substring that bounds the result string from right

```
>>> getBetween('teststring', 'st','in')
'str'
>>> getBetween('teststring', 'st','te')
''
>>> getBetween('teststring', 'te','te')
''
```

like(*s1*, *s2*, *wildcard*=`'%'`)

Parameters

`s1:` first string
`s2:` second string

ilike(*s1*, *s2*, *wildcard*=`'%'`)

Returns the result of like() function ignoring upper-lowercase differencies

filter(*item*, *include*=`None`, *exclude*=`None`, *wildcard*=`'%'`)

regexDelete(*myString*, *pattern*)

Returns a string obtained deleting from the given string any occurrency of the given pattern. @param myString

Parameters

`pattern:` pattern of chars that must be deleted from myString

```
>>> regexDelete('When he turns the steering  wheel', 'he')
'Wn  turns t steering  wel'
```

templateReplace(*myString*, *symbolDict*=`None`, *safeMode*=`False`)

Parameters

`myString:` template string
`symbolDict:` dictionary that links symbol and values
`safeMode:` flag that implies the use of substitute() or safe_substitute()

```
>>> templateReplace('$foo loves $bar but she loves $aux and not $foo', {'foo':'John','bar':
'John loves Sandra but she loves Steve and not John'
```

asDict(*myString*, *itemSep*=`','`, *argSep*=`'='`, *symbols*=`None`)

Parameters

`myString:` a string that represent a list of key-value pairs
`itemSep:` the separator char between each key-value pair
`argSep:` the separator key and value
`symbols:` a dictionary that eventually contains value for templates in myString

```
>>> asDict('height=22, weight=73')
{'weight': '73', 'height': '22'}
>>> asDict('height=$myheight, weight=73', symbols={'myheight':55})
{'weight': '73', 'height': '55'}
```

stringDict(*myDict*, *itemSep*=`','`, *argSep*=`'='`)

returns a string that represents a list of list of key-value pairs taken from a dictionary.

Parameters

`myDict:` dictionay to transform in string
`itemSep:` the separator char between each key-value pair
`argSep:` the separator key and value

```
>>> stringDict({'height':22,'width':33})
 'width=33,height=22'
```

updateString(*source*, *s*, *sep*=`','`)

This method appends to a string that represents a set of elements separated by a separation cha a new element.

Parameters

`source:` first string
`s:` string that must be added @param separator string

```
>>> updateString('I drink cola', 'beer')
'I drink cola,beer'
>>> gs.updateString('I drink cola', 'beer', ' and ')
'I drink cola and beer'
```

updateStringList(*s1*, *s2*, *sep*=`','`)

makeSet(**args*, ***kwargs*)

splitAndStrip(*myString*, *sep*=`','`, *n*=`-1`, *fixed*=`0`)

```
This methods splits a string in a list of n+1 items, striping white spaces.
@param myString
@param sep: separation character
@param n: how many split operations must be done on myString
@param fixed: use fixed if the resulting list must be always
              with a fixed length.

>>> splitAndStrip('cola, beer, milk')
['cola', 'beer', 'milk']
>>> splitAndStrip('cola, beer, milk', n=2)
['cola', 'beer, milk']
```

countOf(*myString*, *srcString*)

split(*path*, *sep*=`'.'`)

Returns a list splittng a path string at any occurrency of separation character. This methods checks brackets >>> split('first.second.third') ['first', 'second', 'third']

smartjoin(*mylist*, *on*)

Joins the given list with the separator substring on and escape each occurrency of on within mylist @param mylist

Parameters

`on:` separator substring

```
>>> smartjoin(['Hello, dog', 'you', 'are', 'yellow'], ',')
'Hello\, dog,you,are,yellow'
```

smartsplit(*path*, *on*)

Splits the string "path" with the separator substring "on" ignoring the escaped separator chars @param path

Parameters

`on:` separator substring

dotEsc(*txt*)

returns a text with all dot char escaped

encode(*number*, *base*=`'/16'`, *nChars*=`None`)

Returns a string that contains the given number in the specified base

Parameters

`number:` number to encode
`base:` base of encoding
`nChar:` number of characters of the result return: encoded number as string

fromIsoDate(*datestring*)

fromText(*mystring*, *obj*, *locale*=`None`)

toText(*obj*, *locale*=`None`, *format*=`None`, *mask*=`None`, *encoding*=`None`)

Return a unicode string representing an object of any class. If there are locale or format parameters Babel is used to format the value according to the given localization or format.

boolean(*obj*)

pickleObject(*obj*, *zipfilename*=`None`)

Return the Pickle string for the given object

unpickleObject(*objstr*, *zipfilename*=`None`)

Load an object from a pikle string

zipString(*mystring*, *filename*)

Return a zip compressed version of mystring

unzipString(*mystring*, *filename*)

Extract a zip compressed string

toJson(*obj*)

fromJson(*obj*)

16.2 Variables

Name	Description
logger	**Value:** `logging.getLogger('gnr.core.gnrstring')`
REGEX_WRDSPLIT	**Value:** `re.compile(r'\W+')`
BASE_ENCODE	**Value:** `{'/2': '01', '/8': '012345678', '/16': '0123456789ABCDEF'...`

16.3 Class JsonEncoder

simplejson.JSONEncoder

gnr.core.gnrstring.JsonEncoder

16.3.1 Methods

default(*self*, *obj*)

17 Module gnr.core.gnrstructures

17.1 Class GnrStructData

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.core.gnrstructures.GnrStructData

17.1.1 Methods

makeRoot(*cls*, *source*=`None`, *protocls*=`None`)

This method builds the root instance for the given class.

Parameters

`cls:` structure class
`source:` filepath of xml file

_(*self*)

root(*self*)

child(*self*, *tag*, *name*=`'*_#'`, *content*=`None`, *_parentTag*=`None`, ***kwargs*)

This method sets a new item of the type tag into the current structure

Parameters

`tag:` structure type
`name:` structure name. Default value is formed by 'tag_position'
`content:` optional structure content
`kwargs:` other parameters @return : the new structure if content is none else the parent

save(*self*, *path*)

This method saves the structure as an xml file

Parameters

`path:` destination of the saved file

load(*self*, *path*)

This method loads the structure from an xml file

Parameters

`path:` path of the file

attributes(*self*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__init__(*self*, *source*=`None`)

A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:`	an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:`	specifies if a node with an existing path overwrite the value or append it.
`_validators:`	an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:`	all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
`_position:`	specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:`	path of the node.
`validator:`	the type of validation.
`parameterString:`	string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value
a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters
`encoding:` default is 'UTF-8'

Return Value
a formatted representation of the bag contents (ascii)

backref(*self*)

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value
a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value
a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters
`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars
`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

makePicklable(*self*)

This method make a Bag picklable.

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

- `path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

- `subscriberId:` an ID can be assigned for a subscription
- `update:` the eventhandler function linked to update event.
- `insert:` the eventhandler function linked to insert event.
- `delete:` the eventhandler function linked to delete event.
- `any:` the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback:` the function which is called.

17.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

17.1.3 Class Variables

Name	Description
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

17.2 Class GnrStructObj

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

17.2.1 Methods

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`
`parent:` @param structnode
`objclassdict:` dictionary of the classes
`kwargs:` return

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

buildChildren(*self*, *children*)

metadata(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

deleteChild(*self*, *name*)

onDelete(*self*)

deleteChildren(*self*)

root(*self*)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

get(*self*, *name*, *default*=`None`)

getResolver(*self*, *name*, *default*=`None`)

__len__(*self*)

__iter__(*self*)

__contains__(*self*, *name*)

items(*self*)

keys(*self*)

values(*self*)

init(*self*)

newChild(*self*, *child*)

afterChildrenCreation(*self*)

asBag(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

mixin(*self*, *cls*, ***kwargs*)

17.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

17.2.3 Class Variables

Name	Description
structnode	**Value:** `property(_get_structnode, _set_structnode)`
parent	**Value:** `property(_get_parent, _set_parent)`

17.3 Class StructObjResolver

object
gnr.core.gnrbag.BagResolver
gnr.core.gnrstructures.StructObjResolver

17.3.1 Methods

resolverDescription(*self*)
Overrides: gnr.core.gnrbag.BagResolver.resolverDescription

init(*self*, *obj*)
Overrides: gnr.core.gnrbag.BagResolver.init

expired(*self*)
Overrides: gnr.core.gnrbag.BagResolver.expired

__call__(*self*)
Overrides: gnr.core.gnrbag.BagResolver.__call__

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

getAttributes(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

load(*self*)

must be reimplemented

reset(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

17.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

17.3.3 Class Variables

Name	Description
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
classArgs	**Value:** `[]`
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True}`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

17.4 Class TestStructModule

object
└ **gnr.core.gnrstructures.TestStructModule**

17.4.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

buildOne(*self*, *name*, *path*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

17.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

17.5 Class GnrStructureError

exceptions.Exception
└ **gnr.core.gnrstructures.GnrStructureError**

17.5.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

18 Module gnr.core.gnrsys

sys

18.1 Functions

mkdir(*path*, *privileges*=`511`)

19 Package gnr.sql

19.1 Modules

- **adapters** *(Section 20, p. 148)*
- **gnrsqlutils**: gnrsqlutils.py
  (Section 28, p. 268)

20 Package gnr.sql.adapters

21 Module gnr.sql.adapters._gnrbaseadapter

21.1 Class SqlDbAdapter

object
└ **gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter**

Base class for sql adapters. All the methods of this class can be overwritten for specific db adapters, but only a few must be implemented in a specific adapter.

21.1.1 Methods

__init__(*self*, *dbroot*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

connect(*self*)

– IMPLEMENT THIS – Build and return a new connection object: ex. return dbapi.connect() The returned connection MUST provide cursors accessible by col number or col name (as list or as dict)

Return Value
a new connection object

cursor(*self*, *connection*, *cursorname*=`None`)

listen(*self*, *msg*, *timeout*=`None`, *onNotify*=`None`, *onTimeout*=`None`)

– IMPLEMENT THIS – Listen for interprocess message 'msg' onTimeout callbacks are executed on every timeout, onNotify on messages. Callbacks returns False to stop, or True to continue listening.

Parameters

`msg:`	name of the message to wait for
`timeout:`	seconds to wait for the message
`onNotify:`	function to execute on arrive of message
`onTimeout:`	function to execute on timeout

notify(*self*, *msg*, *autocommit*=`False`)

– IMPLEMENT THIS – Notify a message to listener processes.

Parameters

`msg:`	name of the message to notify
`autocommit:`	dafault False, if specific implementation of notify uses transactions, commit the current transaction

createdb(*self*, *name*, *encoding*=`None`)

– IMPLEMENT THIS – Create a new database

Parameters

`name:` db name
`encoding:` database text encoding

dropdb(*self*, *name*)

– IMPLEMENT THIS – Drop an existing database

Parameters

`name:` db name

defaultMainSchema(*self*)

– IMPLEMENT THIS – Drop an existing database

Return Value

the name of the default schema

listElements(*self*, *elType*, ***kwargs*)

– IMPLEMENT THIS – Get a list of element names: elements can be any kind of structure supported by a specific db. Usually an adapter accept as elType the following: schemata, tables, columns, views

Parameters

`elType:` type of structure element to list
`kwargs:` optional parameters, eg. for elType "columns" kwargs could be {'schema':'public', 'table':'mytable'}

Return Value

list of object names

relations(*self*)

– IMPLEMENT THIS – Get a list of all relations in the db. Each element of the list is a list (or tuple) with this elements: [foreign_constraint_name, many_schema, many_tbl, [many_col, ...], unique_constraint_name, one_schema, one_tbl, [one_col, ...]]

Return Value

list of relation's details

getPkey(*self*, *table*, *schema*)

– IMPLEMENT THIS –

Parameters

`table:` table name
`schema:` schema name

Return Value

list of columns wich are the primary key for the table

getColInfo(*self*, *table*, *schema*, *column*)

– IMPLEMENT THIS – Get a (list of) dict containing details about a column or all the columns of a table. Each dict has those info: name, position, default, dtype, length, notnull A specifica adapter can add to the dict other available infos

getIndexesForTable(*self*, *table*, *schema*)

– IMPLEMENT THIS – Get a (list of) dict containing details about all the indexes of a table. Each dict has those info: name, primary (bool), unique (bool), columns (comma separated string)

Parameters

`table:` table name
`schema:` schema name

Return Value

list of index infos

prepareSqlText(*self*, *sql*, *kwargs*)

Subclass in adapter if you want to change some sql syntax or params types. Example: for a search condition using regex, sqlite wants 'REGEXP', while postgres wants '~*'

Parameters

`sql:` the sql string to execute.
`kwargs:` the params dict

Return Value

tuple (sql, kwargs)

existsRecord(*self*, *dbtable*, *record_data*)

Test if a record yet exists in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object containing at least one entry for the pkey column of the table.

prepareRecordData(*self*, *record_data*)

Normalize a record_data object before actually execute an sql write command. Delete items which name starts with '@': eager loaded relations don't have to be written as fields. Convert Bag values to xml, to be stored in text or blob fields. Convert all fields names to lowercase ascii characters.

Parameters

`record_data:` a dict compatible object

insert(*self*, *dbtable*, *record_data*)

Insert a record in the db. All fields in record_data will be added: all keys must correspond to a column in the db.

Parameters

`dbtable:` an SqlTable object
`record_data:` a dict compatible object

update(*self*, *dbtable*, *record_data*)

Update a record in the db. All fields in record_data will be updated: all keys must correspond to a column in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object

delete(*self*, *dbtable*, *record_data*)

Delete a record from the db. All fields in record_data will be added: all keys must correspond to a column in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object containing at least one entry for the pkey column of the table.

analyze(*self*)

Perform analyze routines on the db

vacuum(*self*, *table*=`''`, *full*=`False`)

Perform analyze routines on the db

createSchema(*self*, *sqlschema*)

Create a new database schema

dropSchema(*self*, *sqlschema*)

Create a new database schema

createTableAs(*self*, *sqltable*, *query*, *sqlparams*)

addColumn(*self*, *sqltable*, *sqlname*, *dtype*=`'T'`, *size*=`None`, *notnull*=`None`, *pkey*=`None`)

columnSqlDefinition(*self*, *sqlname*, *dtype*, *size*, *notnull*, *pkey*)

returns the statement string for creating a table's column

dropTable(*self*, *sqltable*)

Create a new database schema

createIndex(*self*, *index_name*, *columns*, *table_sql*, *sqlschema*=`None`, *unique*=`None`)

create a new index

Parameters

`index_name:` name of the index (unique in schema)
`columns:` comma separated list of columns to include in the index
`table_sql:` actual sql name of the table

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

21.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

21.1.3 Class Variables

Name	Description
typesDict	**Value:** {`'character varying': 'A', 'character': 'A', 'text': 'T',...`
revTypesDict	**Value:** {`'A': 'character varying', 'C': 'character', 'T': 'text',...`

21.2 Class GnrDictRow

```
object ──┐
         list ──┐
gnr.core.gnrlist.GnrNamedList ──┐
                                gnr.sql.adapters._gnrbaseadapter.GnrDictRow
```

A row object that allow by-colun-name access to data, the capacity to add columns and alter data.

21.2.1 Methods

__init__(*self*, *cursor*, *values*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Return Value

`new list`

Overrides: gnr.core.gnrlist.GnrNamedList.__init__

__add__(*x*, *y*)

x+y

__contains__(*x*, *y*)

y in x

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*x*, *y*)

del x[y]

__delslice__(*x*, *i*, *j*)

del x[i:j]
Use of negative indices is not supported.

__eq__(*x*, *y*)

x==y

__ge__(*x*, *y*)

x>=y

__getattribute__(...)

x.__getattribute__('name') <==> x.name

Overrides: object.__getattribute__

__getitem__(*self*, *x*)
x[y]

Overrides: list.__getitem__ extit(inherited documentation)

__getslice__(*x*, *i*, *j*)

x[i:j]
Use of negative indices is not supported.

__gt__(*x*, *y*)

x>y

__hash__(*x*)

hash(x)

Overrides: object.__hash__

__iadd__(*x*, *y*)

x+=y

__imul__(*x*, *y*)

x*=y

__iter__(*x*)

iter(x)

__le__(*x*, *y*)

x<=y

__len__(*x*)

len(x)

__lt__(*x*, *y*)

x<y

__mul__(*x*, *n*)

x*n

__ne__(*x*, *y*)

x!=y

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

Overrides: object.__new__

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*self*)

repr(x)

Overrides: list.__repr__ extit(inherited documentation)

__reversed__(*L*)

return a reverse iterator over the list

__rmul__(*x*, *n*)

n*x

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *x*, *v*)

x[i]=y

Overrides: list.__setitem__ extit(inherited documentation)

__setslice__(*x*, *i*, *j*, *y*)

x[i:j]=y

Use of negative indices is not supported.

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

append(*L*, *object*)

append object to end

count(*L*, *value*)

return number of occurrences of value

Return Value
`integer`

extend(*L*, *iterable*)

extend list by appending elements from the iterable

extractItems(*self*, *columns*)

extractValues(*self*, *columns*)

get(*self*, *x*, *default*=`None`)

has_key(*self*, *x*)

index(...)

L.index(value, [start, [stop]]) -> integer – return first index of value

insert(*L*, *index*, *object*)

insert object before index

items(*self*)

iteritems(*self*)

keys(*self*)

pop(*L*, *index*=`...`)

remove and return item at index (default last)

Return Value
`item`

remove(*L*, *value*)

remove first occurrence of value

reverse(*L*)

reverse *IN PLACE*

sort(*L*, *cmp*=`None`, *key*=`None`, *reverse*=`False`)

stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1

update(*self*, *d*)

values(*self*)

21.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

21.3 Class NotImplementedException

exceptions.Exception
└
gnr.sql.adapters._gnrbaseadapter.NotImplementedException

21.3.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

22 Module gnr.sql.adapters.gnrpostgres

22.1 Variables

Name	Description
RE_SQL_PARAMS	**Value:** `re.compile(":(\w*)(\W|$)")`

22.2 Class SqlDbAdapter

object

gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter

gnr.sql.adapters.gnrpostgres.SqlDbAdapter

22.2.1 Methods

defaultMainSchema(*self*)
– IMPLEMENT THIS – Drop an existing database

Return Value
the name of the default schema

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.defaultMainSchema extit(inherited documentation)

connect(*self*)

Return a new connection object: provides cursors accessible by col number or col name

Return Value
a new connection object

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.connect

prepareSqlText(*self*, *sql*, *kwargs*)

Change the format of named arguments in the query from ':argname' to '%(argname)s'. Replace the 'REGEXP' operator with '~*'.

Parameters
`sql:` the sql string to execute.
`kwargs:` the params dict

Return Value
tuple (sql, kwargs)

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.prepareSqlText

createDb(*self*, *name*, *encoding*=`'unicode'`)

dropDb(*self*, *name*)

createTableAs(*self*, *sqltable*, *query*, *sqlparams*)
Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.createTableAs

vacuum(*self*, *table*=`''`, *full*=`False`)

Perform analyze routines on the db

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.vacuum

listen(*self*, *msg*, *timeout*=`10`, *onNotify*=`None`, *onTimeout*=`None`)

Listen for message 'msg' on the current connection using the Postgres LISTEN - NOTIFY method. onTimeout callbacks are executed on every timeout, onNotify on messages. Callbacks returns False to stop, or True to continue listening.

Parameters

`msg:` name of the message to wait for
`timeout:` seconds to wait for the message
`onNotify:` function to execute on arrive of message
`onTimeout:` function to execute on timeout

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.listen

notify(*self*, *msg*, *autocommit*=`False`)

Notify a message to listener processes using the Postgres LISTEN - NOTIFY method.

Parameters

`msg:` name of the message to notify
`autocommit:` if False (default) you have to commit transaction, and the message is actually sent on commit

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.notify

listElements(*self*, *elType*, ***kwargs*)

Get a list of element names.

Parameters

`elType:` one of the following: schemata, tables, columns, views.
`kwargs:` schema, table

Return Value

list of object names

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.listElements

relations(*self*)

Get a list of all relations in the db. Each element of the list is a list (or tuple) with this elements: [foreign_constraint_name, many_schema, many_tbl, [many_col, ...], unique_constraint_name, one_schema, one_tbl, [one_col, ...]]

Return Value

list of relation's details

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.relations

getPkey(*self*, *table*, *schema*)

– IMPLEMENT THIS –

Parameters

`table:` table name
`schema:` schema name

Return Value

list of columns wich are the primary key for the table

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.getPkey

getIndexesForTable(*self*, *table*, *schema*)

Get a (list of) dict containing details about all the indexes of a table. Each dict has those info: name, primary (bool), unique (bool), columns (comma separated string)

Parameters

`table:` table name
`schema:` schema name

Return Value

list of index infos

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.getIndexesForTable

getColInfo(*self*, *table*, *schema*, *column*=`None`)

Get a (list of) dict containing details about a column or all the columns of a table. Each dict has those info: name, position, default, dtype, length, notnull Every other info stored in information_schema.columns is available with the prefix '_pg_'

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.getColInfo

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *dbroot*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addColumn(*self*, *sqltable*, *sqlname*, *dtype*='T', *size*=None, *notnull*=None, *pkey*=None)

analyze(*self*)

Perform analyze routines on the db

columnSqlDefinition(*self*, *sqlname*, *dtype*, *size*, *notnull*, *pkey*)

returns the statement string for creating a table's column

createIndex(*self*, *index_name*, *columns*, *table_sql*, *sqlschema*=None, *unique*=None)

create a new index

Parameters

`index_name:`	name of the index (unique in schema)
`columns:`	comma separated list of columns to include in the index
`table_sql:`	actual sql name of the table

createSchema(*self*, *sqlschema*)

Create a new database schema

createdb(*self*, *name*, *encoding*=None)

– IMPLEMENT THIS – Create a new database

Parameters

`name:`	db name
`encoding:`	database text encoding

cursor(*self*, *connection*, *cursorname*=None)

delete(*self*, *dbtable*, *record_data*)

Delete a record from the db. All fields in record_data will be added: all keys must correspond to a column in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object containing at least one entry for the pkey column of the table.

dropSchema(*self*, *sqlschema*)

Create a new database schema

dropTable(*self*, *sqltable*)

Create a new database schema

dropdb(*self*, *name*)

– IMPLEMENT THIS – Drop an existing database

Parameters

`name:` db name

existsRecord(*self*, *dbtable*, *record_data*)

Test if a record yet exists in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object containing at least one entry for the pkey column of the table.

insert(*self*, *dbtable*, *record_data*)

Insert a record in the db. All fields in record_data will be added: all keys must correspond to a column in the db.

Parameters

`dbtable:` an SqlTable object
`record_data:` a dict compatible object

prepareRecordData(*self*, *record_data*)

Normalize a record_data object before actually execute an sql write command. Delete items which name starts with '@': eager loaded relations don't have to be written as fields. Convert Bag values to xml, to be stored in text or blob fields. Convert all fields names to lowercase ascii characters.

Parameters

`record_data:` a dict compatible object

update(*self*, *dbtable*, *record_data*)

Update a record in the db. All fields in record_data will be updated: all keys must correspond to a column in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object

22.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

22.2.3 Class Variables

Name	Description
typesDict	**Value:** `{'character varying': 'A', 'character': 'C', 'text': 'T',...`
revTypesDict	**Value:** `{'A': 'character varying', 'T': 'text', 'C': 'character',...`

22.3 Class GnrDictConnection

psycopg2.extensions.connection
psycopg2.extras.DictConnection
gnr.sql.adapters.gnrpostgres.GnrDictConnection

A connection that uses DictCursor automatically.

22.3.1 Methods

cursor(*self*, *name*=`None`)
Overrides: psycopg2.extras.DictConnection.cursor

22.4 Class GnrDictCursor

psycopg2.extensions.cursor
psycopg2.extras.DictCursor
gnr.sql.adapters.gnrpostgres.GnrDictCursor

22.4.1 Methods

__init__(*self*, **args*, ***kwargs*)

fetchmany(*self*, *size*=`None`)
Overrides: psycopg2.extras.DictCursor.fetchmany

callproc(*self*, *procname*, *vars*=`None`)

execute(*self*, *query*, *vars*=`None`, *async*=`0`)

fetchall(*self*)

fetchone(*self*)

next(*self*)

23 Module gnr.sql.adapters.gnrsqlite

23.1 Class SqlDbAdapter

object

gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter

gnr.sql.adapters.gnrsqlite.SqlDbAdapter

23.1.1 Methods

defaultMainSchema(*self*)
– IMPLEMENT THIS – Drop an existing database

Return Value
the name of the default schema

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.defaultMainSchema extit(inherited documentation)

regexp(*self*, *expr*, *item*)

connect(*self*)

Return a new connection object: provides cursors accessible by col number or col name

Return Value
a new connection object

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.connect

cursor(*self*, *connection*, *cursorname*=`None`)
Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.cursor

attach(*self*, *filepath*, *name*, *cursor*=`None`)

A special sqlite only method for attach external database file as a schema for the current one

Parameters

`filepath:`	external sqlite db file
`name:`	name of the schema containing the external db
`cursor:`	optional cursor object to use

prepareSqlText(*self*, *sql*, *kwargs*)

Replace the 'REGEXP' operator with '~*'. Replace the ILIKE operator with LIKE: sqlite LIKE is case insensitive

Return Value
tuple (sql, kwargs)

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.prepareSqlText

listElements(*self*, *elType*, ***kwargs*)

Get a list of element names.

Parameters

`elType:` one of the following: schemata, tables, columns, views.
`kwargs:` schema, table

Return Value

list of object names

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.listElements

relations(*self*)

Get a list of all relations in the db. Each element of the list is a list (or tuple) with this elements: [foreign_constraint_name, many_schema, many_tbl, [many_col, ...], unique_constraint_name, one_schema, one_tbl, [one_col, ...]]

Return Value

list of relation's details

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.relations

getPkey(*self*, *table*, *schema*)

– IMPLEMENT THIS –

Parameters

`table:` table name
`schema:` schema name

Return Value

list of columns wich are the primary key for the table

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.getPkey

getColInfo(*self*, *table*, *schema*, *column*=`None`)

Get a (list of) dict containing details about a column or all the columns of a table. Each dict has those info: name, position, default, dtype, length, notnull Every other info stored in PRAGMA table_info is available with the prefix '_sl_'

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.getColInfo

listen(*self*, *msg*, *timeout*=`None`, *onNotify*=`None`, *onTimeout*=`None`)

Actually sqlite has no message comunications: so simply sleep and executes onTimeout TODO: could be implemented with pyro to notify messages onTimeout callbacks are executed on every timeout, onNotify on messages. Callbacks returns False to stop, or True to continue listening.

Parameters

`msg:` name of the message to wait for
`timeout:` seconds to wait for the message
`onNotify:` function to execute on arrive of message
`onTimeout:` function to execute on timeout

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.listen

notify(*self*, *msg*, *autocommit*=`False`)

Actually sqlite has no message comunications: so simply pass

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.notify

createDb(*self*, *name*, *encoding*=`'unicode'`)

Create a new database file. Not really usefull with sqlite, just connecting will automatically create the database file WARNING: this method does not fail if the database jet exists, other adapters do.

Parameters

`name:` db name
`encoding:` database text encoding

dropDb(*self*, *name*)

Drop an existing database file (actually delete the file)

Parameters

`name:` db name

getIndexesForTable(*self*, *table*, *schema*)

Get a (list of) dict containing details about all the indexes of a table. Each dict has those info: name, primary (bool), unique (bool), columns (comma separated string)

Parameters

`table:` table name
`schema:` schema name

Return Value

list of index infos

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.getIndexesForTable

createSchema(*self*, *sqlschema*)

Create a new database schema. sqlite specific implementation actually attach an external db file.

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.createSchema

createIndex(*self*, *index_name*, *columns*, *table_sql*, *sqlschema*=`None`, *unique*=`None`)

create a new index sqlite specific implementation fix a naming difference: schema must be prepended to index name and not to table name.

Parameters

`index_name:` name of the index (unique in schema)
`columns:` comma separated list of columns to include in the index
`table_sql:` actual sql name of the table

Overrides: gnr.sql.adapters._gnrbaseadapter.SqlDbAdapter.createIndex

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *dbroot*, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addColumn(*self*, *sqltable*, *sqlname*, *dtype*='T', *size*=None, *notnull*=None, *pkey*=None)

analyze(*self*)

Perform analyze routines on the db

columnSqlDefinition(*self*, *sqlname*, *dtype*, *size*, *notnull*, *pkey*)

returns the statement string for creating a table's column

createTableAs(*self*, *sqltable*, *query*, *sqlparams*)

createdb(*self*, *name*, *encoding*=`None`)

– IMPLEMENT THIS – Create a new database

Parameters

`name:` db name
`encoding:` database text encoding

delete(*self*, *dbtable*, *record_data*)

Delete a record from the db. All fields in record_data will be added: all keys must correspond to a column in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object containing at least one entry for the pkey column of the table.

dropSchema(*self*, *sqlschema*)

Create a new database schema

dropTable(*self*, *sqltable*)

Create a new database schema

dropdb(*self*, *name*)

– IMPLEMENT THIS – Drop an existing database

Parameters

`name:` db name

existsRecord(*self*, *dbtable*, *record_data*)

Test if a record yet exists in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object containing at least one entry for the pkey column of the table.

insert(*self*, *dbtable*, *record_data*)

Insert a record in the db. All fields in record_data will be added: all keys must correspond to a column in the db.

Parameters

`dbtable:` an SqlTable object
`record_data:` a dict compatible object

prepareRecordData(*self*, *record_data*)

Normalize a record_data object before actually execute an sql write command. Delete items which name starts with '@': eager loaded relations don't have to be written as fields. Convert Bag values to xml, to be stored in text or blob fields. Convert all fields names to lowercase ascii characters.

Parameters

`record_data:` a dict compatible object

update(*self*, *dbtable*, *record_data*)

Update a record in the db. All fields in record_data will be updated: all keys must correspond to a column in the db.

Parameters

`dbtable:` a SqlTable object
`record_data:` a dict compatible object

vacuum(*self*, *table*=`''`, *full*=`False`)

Perform analyze routines on the db

23.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

23.1.3 Class Variables

Name	Description
typesDict	**Value:** `{'charactervarying': 'A', 'character varying': 'A', 'char...`
revTypesDict	**Value:** `{'A': 'character varying', 'T': 'text', 'C': 'character',...`

23.2 Class GnrSqliteCursor

pysqlite2.dbapi2.Cursor
└ **gnr.sql.adapters.gnrsqlite.GnrSqliteCursor**

23.2.1 Methods

index(*self*)

execute(*self*, **args*, ***kwargs*)

24 Module gnr.sql.gnrsql

24.1 Variables

Name	Description
__version__	**Value:** `'1.0b'`
gnrlogger	**Value:** `gnrlogging.getLogger('gnr.sql.gnrsql')`

24.2 Class GnrSqlDb

object
gnr.core.gnrlang.GnrObject
gnr.sql.gnrsql.GnrSqlDb

```
This is the main class of the gnrsql module.
A GnrSqlDb object has the following purposes:
  -manage the logical structure of a database, called database's model.
  -manage operations on db at high level, hiding adapter's layer and
  connections.
```

24.2.1 Methods

__init__(*self*, *implementation*=`'sqlite'`, *dbname*=`'mydb'`, *host*=`None`, *user*=`None`, *password*=`None`, *port*=`None`, *main_schema*=`None`)

This is the constructor method of the GnrSqlDb class.

Parameters

`implementation:`	'sqlite' or 'postgres' or other sql implementations.
`dbname:`	the name for your db.
`host:`	the database server host (for sqlite is None)
`user:`	a database user's name (for sqlite is None)
`password:`	the user's password (for sqlite is None)
`port:`	the connection port (for sqlite is None)
`main_schema:`	the database main_schema

Overrides: gnr.core.gnrlang.GnrObject.__init__

__del__(*self*)

startup(*self*)

Build the model.obj from the model.src

packageSrc(*self*, *name*)

Return a DbModelSrc corresponding to the required package

Parameters

`name:` the package name

packageMixin(*self*, *name*, *obj*)

Register a mixin for a package.

Parameters

`name:` the target package's name
`obj:` a class or an object to mixin

tableMixin(*self*, *tblpath*, *obj*)

Register an object or a class to mixin to a table.

Parameters

`name:` the target package's name
`obj:` a class or an object to mixin

loadModel(*self*, *source*=`None`)

Load the model.src from a xml source

Parameters

`source:` xml model (diskfile or text or url)

importModelFromDb(*self*)

Load the model.src extracting it from the database's information schema.

saveModel(*self*, *path*)

Save the current model as xml file at path

Parameters

`path:` the file path

checkDb(*self*, *applyChanges*=`False`)

Check if there the database structure is compatible with the current model

Parameters

`applyChanges:` boolean. If True, all the changes are executed and committed

closeConnection(*self*)

Close a connection

connection(*self*)

property .connection If there's not connection open and return connection to database

execute(*self*, *sql*, *sqlargs*=`None`, *cursor*=`None`, *cursorname*=`None`, *autocommit*=`False`)

Execute the sql statement using given kwargs

insert(*self*, *tblobj*, *record*)

Insert a record in the table.

Parameters

`tblobj:` the table object
`record:` an object implementing dict interface as colname, colvalue

update(*self*, *tblobj*, *record*)

Update a record of the table.

Parameters

`tblobj:` the table object
`record:` an object implementing dict interface as colname, colvalue

delete(*self*, *tblobj*, *record*)

Delete a record from the table.

Parameters

`tblobj:` the table object
`record:` an object implementing dict interface as colname, colvalue

commit(*self*)

Commit a transaction

rollback(*self*)

Rollback a transaction

listen(*self*, **args*, ***kwargs*)

Listen for a database event (postgres)

notify(*self*, **args*, ***kwargs*)

Database Notify

analyze(*self*)

Analyze db

vacuum(*self*)

Analyze db

package(*self*, *pkg*)

Returns a package object

Parameters

`pkw:` package name

packages(*self*)

Returns a package object

Parameters

`pkw:` package name

table(*self*, *tblname*, *pkg*=`None`)

returns a table object

Parameters

`table:` table name
`pkg:` package name

query(*self*, *table*, ***kwargs*)

See gnrsqltable.SqlTable.query

createDb(*self*, *name*, *encoding*=`'unicode'`)

Create a db with given name and encoding @param name @param encoding

dropDb(*self*, *name*)

Drop a db with given name @param name

createSchema(*self*, *name*)

Create a db with given name and encoding @param name @param encoding

importXmlData(*self*, *path*)

Populates a database from an xml file

Parameters

`path:` filepath

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

mixin(*self*, *cls*, ***kwargs*)

24.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

24.3 Class NotMatchingModelError

exceptions.Exception
└ **gnr.sql.gnrsql.NotMatchingModelError**

24.3.1 Methods

__getitem__(*...*)

__init__(*...*)

__str__(...)

25 Module gnr.sql.gnrsqldata

25.1 Variables

Name	Description
COLFINDER	**Value:** `re.compile(r"(\W|^)\$(\w+)")`
RELFINDER	**Value:** `re.compile(r"(\W|^)(\@([\w.@:]+))")`

25.2 Class SqlCompiledQuery

object
└ **gnr.sql.gnrsqldata.SqlCompiledQuery**

25.2.1 Methods

__init__(*self*, *maintable*, *relationDict*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

sqltext(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

25.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.3 Class SqlQueryCompiler

object
└
gnr.sql.gnrsqldata.SqlQueryCompiler

25.3.1 Methods

__init__(*self*, *tblobj*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

init(*self*, *lazy*=`None`, *eager*=`None`)

getFieldAlias(*self*, *fieldpath*)

getAlias(*self*, *attrs*, *path*, *basealias*)

colToAs(*self*, *col*)

updateFieldDict(*self*, *teststring*)

expandMultipleColumns(*self*, *flt*)

compiledQuery(*self*, *columns*=`''`, *where*=`''`, *order_by*=`''`, *distinct*=`''`, *limit*=`''`, *offset*=`''`, *group_by*=`''`, *having*=`''`, *relationDict*=`None`, *count*=`False`)

compiledRecordQuery(*self*, *lazy*=`None`, *eager*=`None`, *where*=`None`, *relationDict*=`None`)

recordFields(*self*, *fields*, *path*, *bagpath*, *basealias*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

25.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.4 Class SqlDataResolver

object
gnr.core.gnrbag.BagResolver
gnr.sql.gnrsqldata.SqlDataResolver

25.4.1 Methods

resolverSerialize(*self*)
Overrides: gnr.core.gnrbag.BagResolver.resolverSerialize

init(*self*)
Overrides: gnr.core.gnrbag.BagResolver.init

onCreate(*self*)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

load(*self*)

must be reimplemented

reset(*self*)

resolverDescription(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

25.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.4.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True, 'db': None}`
classArgs	**Value:** `['tablename']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

25.5 Class SqlRecordResolver

object

gnr.core.gnrbag.BagResolver

gnr.sql.gnrsqldata.SqlDataResolver

gnr.sql.gnrsqldata.SqlRecordResolver

25.5.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)
Overrides: gnr.core.gnrbag.BagResolver.init

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

onCreate(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)
Overrides: gnr.core.gnrbag.BagResolver.resolverSerialize

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

25.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.5.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True, 'where': None, 'lazy':...`
classArgs	**Value:** `['tablename']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

25.6 Class SqlQuery

object
└
gnr.sql.gnrsqldata.SqlQuery

SqlQuery object represent the way in which data can be extracted from a db. You can get data with these methods of SqlQuery: selection, with return a SqlSelection, fetch, count, cursor and serverCursor.

25.6.1 Methods

__init__(*self*, *dbtable*, *columns*=`None`, *where*=`None`, *order_by*=`None`, *distinct*=`None`, *limit*=`None`, *offset*=`None`, *group_by*=`None`, *having*=`None`, *relationDict*=`None`, *sqlparams*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Parameters

`dbtable:`	the table the query is focused on. (see the main doc)
`where:`	the same of sql one
`order_by:`	the same of sql one
`distinct:`	the same of sql one
`limit:`	the same of sql one
`offset:`	
`group_by:`	the same of sql one
`having:`	the same of sql one
`relationDict:`	a dictionary which associates relationPath names with an alias name. eg: {'$member_name':'@member_id.name'}
`sqlparams:`	a dictionary which associates sqlparams to their value

Overrides: object.__init__ extit(inherited documentation)

resolver(*self*, *mode*=`'bag'`)

sqltext(*self*)

compiled(*self*)

cursor(*self*)

get a cursor of current selection

fetch(*self*)

get a cursor of current selection and fetch it

selection(*self*, *pyWhere*=None, *workTable*=None, *key*=None)

```
Execute the query and return a SqlSelection
   or a SqlSelectionInTable.
@param workTable: when the selection it's too large to be kept in memory you can
                   give a name for a temporary table to store it. By the way
                   in this case the api to access to the selection is quite different.
@param pyWhere: a cb that can be used to reduce the selection during the fetch.
```

servercursor(*self*)

get a cursor on dbserver

serverfetch(*self*, *arraysize*=30)

get fetch of servercursor

count(*self*)

return rowcount. It does not save a selection

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

25.6.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.7 Class SqlSelection

object
└
gnr.sql.gnrsqldata.SqlSelection

Is the resulting data from the execution of a SqlQuery. Through SqlSelection you can get data into differents modes, using output method or freeze it into a file. You can sort and filter a SqlSelection.

25.7.1 Methods

__init__(*self*, *dbtable*, *data*, *index*=`None`, *colAttrs*=`None`, *key*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

allColumns(*self*)

db(*self*)

keyDict(*self*)

output(*self*, *mode*, *columns*=`None`, *offset*=`None`, *limit*=`None`, *flt*=`None`, *formats*=`None`, *locale*=`None`, *dfltFormats*=`None`, *recordResolver*=`None`, *asIterator*=`False`)

```
it return the selection into differents format
@param mode (optional):If you use this param.
    *list: return as a list of list
    *json: return as json representation
    *dictlist: returns the result as a list of dictionaries
    *bag: returns a result as a Bag divided into two parts 'headers'
          and 'rows'. headers contains the column names
           while rows contains the selection's rows as bags. Each row's column
          is a bag itself and has a SqlRecordResolver
```

__len__(*self*)

data(*self*)

freeze(*self*, *fpath*, *autocreate*=`False`)

getByKey(*self*, *k*)

record(*self*, *n*)

records(*self*, *offset*=`None`, *limit*=`None`)

sort(*self*, **args*)

filter(*self*, *flt*=`None`)

extend(*self*, *selection*, *merge*=`True`)

apply(*self*, *cb*)

insert(*self*, *i*, *values*)

newRow(*self*, *values*)

remove(*self*, *cb*)

analyze(*self*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`, ***kwargs*)

toTextGen(*self*, *outgen*, *formats*, *locale*, *dfltFormats*)

__iter__(*self*)

out_listItems(*self*, *outsource*)

out_count(*self*, *outsource*)

iter_data(*self*, *outsource*, *asIterator*=`False`)

out_data(*self*, *outsource*)

iter_dictlist(*self*, *outsource*, *asIterator*=`False`)

out_dictlist(*self*, *outsource*)

out_json(*self*, *outsource*)

out_list(*self*, *outsource*)

out_pkeylist(*self*, *outsource*)

out_tablebag(*self*, *outsource*)

out_bag(*self*, *outsource*, *recordResolver*=`False`)

bagHeaders_OLD(*self*, *outsource*)

buildAsBag(*self*, *outsource*, *recordResolver*)

buildAsBagAttr(*self*, *outsource*)

buildAsTableBag(*self*, *outsource*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

25.7.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

25.8 Class SqlSelectionInTable

object

gnr.sql.gnrsqldata.SqlSelection

gnr.sql.gnrsqldata.SqlSelectionInTable

25.8.1 Methods

__init__(*self*, *dbtable*, *workTable*, *compiled*, *sqlparams*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.sql.gnrsqldata.SqlSelection.__init__

__len__(*self*)
Overrides: gnr.sql.gnrsqldata.SqlSelection.__len__

sort(*self*, **args*)
Overrides: gnr.sql.gnrsqldata.SqlSelection.sort

filter(*self*, *flt*=None)
Overrides: gnr.sql.gnrsqldata.SqlSelection.filter

apply(*self*, *cb*)
Overrides: gnr.sql.gnrsqldata.SqlSelection.apply

output(*self*, *mode*, *columns*=None, *offset*=None, *limit*=None, *where*=None, *where_pars*=None)

```
it return the selection into differents format
@param mode (optional):If you use this param.
    *list: return as a list of list
    *json: return as json representation
    *dictlist: returns the result as a list of dictionaries
    *bag: returns a result as a Bag divided into two parts 'headers'
          and 'rows'. headers contains the column names
           while rows contains the selection's rows as bags. Each row's column
          is a bag itself and has a SqlRecordResolver
```

Overrides: gnr.sql.gnrsqldata.SqlSelection.output extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__iter__(*self*)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

allColumns(*self*)

analyze(*self*, *group_by*=None, *sum*=None, *distinct*=None, *key*=None, **kwargs*)

bagHeaders_OLD(*self*, *outsource*)

buildAsBag(*self*, *outsource*, *recordResolver*)

buildAsBagAttr(*self*, *outsource*)

buildAsTableBag(*self*, *outsource*)

data(*self*)

db(*self*)

extend(*self*, *selection*, *merge*=`True`)

freeze(*self*, *fpath*, *autocreate*=`False`)

getByKey(*self*, *k*)

insert(*self*, *i*, *values*)

iter_data(*self*, *outsource*, *asIterator*=`False`)

iter_dictlist(*self*, *outsource*, *asIterator*=`False`)

keyDict(*self*)

newRow(*self*, *values*)

out_bag(*self*, *outsource*, *recordResolver*=`False`)

out_count(*self*, *outsource*)

out_data(*self*, *outsource*)

out_dictlist(*self*, *outsource*)

out_json(*self*, *outsource*)

out_list(*self*, *outsource*)

out_listItems(*self*, *outsource*)

out_pkeylist(*self*, *outsource*)

out_tablebag(*self*, *outsource*)

record(*self*, *n*)

records(*self*, *offset*=`None`, *limit*=`None`)

remove(*self*, *cb*)

toTextGen(*self*, *outgen*, *formats*, *locale*, *dfltFormats*)

25.8.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.9 Class SqlSelectionResolver

object
gnr.core.gnrbag.BagResolver
gnr.sql.gnrsqldata.SqlDataResolver
gnr.sql.gnrsqldata.SqlSelectionResolver

25.9.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)
Overrides: gnr.core.gnrbag.BagResolver.init

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

onCreate(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)
Overrides: gnr.core.gnrbag.BagResolver.resolverSerialize

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

25.9.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.9.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True, 'db': None, 'columns':...`
classArgs	**Value:** `['tablename']`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

25.10 Class SqlRecord

object
└ **gnr.sql.gnrsqldata.SqlRecord**

25.10.1 Methods

__init__(*self*, *dbtable*, *pkey*=`None`, *where*=`None`, *lazy*=`None`, *eager*=`None`, *relationDict*=`None`, *sqlparams*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

resolver(*self*, *mode*=`'bag'`)

output(*self*, *mode*)

compiled(*self*)

result(*self*)

out_bag(*self*)

out_json(*self*)

out_dict(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

25.10.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.11 Class SqlRecordBag

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.sql.gnrsqldata.SqlRecordBag

25.11.1 Methods

__init__(*self*, *db*, *tablename*)
A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrbag.Bag.__init__ extit(inherited documentation)

save(*self*, ***kwargs*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters
`what:` the key path to test.

Return Value
a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path

`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
`_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:` path of the node.
`validator:` the type of validation.
`parameterString:` string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value

a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters

`encoding:` default is 'UTF-8'

Return Value

a formatted representation of the bag contents (ascii)

backref(*self*)

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value
a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value
a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters
`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters
`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters
`path:` path of the given item.

Return Value
the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars
`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

makePicklable(*self*)

This method make a Bag picklable.

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

- `path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

- `subscriberId:` an ID can be assigned for a subscription
- `update:` the eventhandler function linked to update event.
- `insert:` the eventhandler function linked to insert event.
- `delete:` the eventhandler function linked to delete event.
- `any:` the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback`: the function which is called.

25.11.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

25.11.3 Class Variables

Name	Description
db	**Value:** `property(_get_db, _set_db)`
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

25.12 Class SelectionExecutionError

exceptions.Exception
└ **gnr.sql.gnrsqldata.SelectionExecutionError**

25.12.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

25.13 Class RecordDuplicateError

exceptions.Exception
└ **gnr.sql.gnrsqldata.RecordDuplicateError**

25.13.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

25.14 Class RecordNotExistingError

exceptions.Exception
└
gnr.sql.gnrsqldata.RecordNotExistingError

25.14.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

25.15 Class RecordSelectionError

exceptions.Exception
└
gnr.sql.gnrsqldata.RecordSelectionError

25.15.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

26 Module gnr.sql.gnrsqlmodel

26.1 Variables

Name	Description
logger	**Value:** `logging.getLogger('gnr.sql.gnrsql')`

26.2 Class NotExistingTableError

exceptions.Exception
└
gnr.sql.gnrsqlmodel.NotExistingTableError

26.2.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

26.3 Class DbModel

object
└
gnr.sql.gnrsqlmodel.DbModel

26.3.1 Methods

__init__(*self*, *db*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

db(*self*)

build(*self*)

Db startup operations:
- prepares the GnrStructObj root
- loads all relations from Db structure

resolveAlias(*self*, *name*)

addRelation(*self*, *many_relation_tuple*, *oneColumn*, *name_one*=`None`, *name_many*=`None`, *mode*=`None`, *eager_one*=`None`, *eager_many*=`None`)

This method adds a relation in the current model.

Parameters

`manyColumn:`	the column of the "many table". Eg. 'video.movie.director_id'
`oneColumn:`	the column of the "one table". Eg. 'video.director.id'
`name_one:`	the one_to_many relation's name. Eg. 'movies'
`name_many:`	the many_to_one relation's name. Eg. 'director'
`case_insensitive:`	if True ('Y') the relation is case_insensitive
`eager_one:`	if True ('Y') the one_to_many relation is eager
`eager_many:`	if True ('Y') the many_to_one relation is eager

load(*self*, *source*=`None`)

Load the modelsrc from a xml source

Parameters

`source:` xml model (diskfile or text or url)

importFromDb(*self*)

save(*self*, *path*)

save the current modelsrc as xml file at path

Parameters

`path:` the file path

check(*self*, *applyChanges*=`False`)

This method verifies the compatibility between the database and the model. It saves sql statements that makes the database compatible with the model.

Parameters

`applyChanges:` if True applies the changes.

applyModelChanges(*self*)

packageMixin(*self*, *pkg*, *obj*)

tableMixin(*self*, *tblpath*, *obj*)

package(*self*, *pkg*)

Returns a package object

Parameters

`pkw:` package name

table(*self*, *tblname*, *pkg*=`None`)

returns a table object

Parameters

`table:` table name
`pkg:` package name

column(*self*, *colname*)

returns a column object

Parameters

`colname:` colname name

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

26.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.4 Class DbModelSrc

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.core.gnrstructures.GnrStructData
gnr.sql.gnrsqlmodel.DbModelSrc

this is a GnrStructData subclass of definition for the elements of a GenroDb TESTED in a_structure_load_test.py

26.4.1 Methods

package(*self*, *name*, *sqlschema*=`None`, *comment*=`None`, *name_short*=`None`, *name_long*=`None`, *name_full*=`None`, ***kwargs*)

Add a package to the structure. Child of root.

Parameters

`name:`	package name,
`comment:`	comment about package,
`name_short:`	name_short,
`name_long:`	name_long,
`name_full:`	name_full

externalPackage(*self*, *name*)

table(*self*, *name*, *pkey*=`None`, *rowcaption*=`None`, *sqlname*=`None`, *sqlschema*=`None`, *comment*=`None`, *name_short*=`None`, *name_long*=`None`, *name_full*=`None`, ***kwargs*)

Add a table to the structure. Child of package.

Parameters

`name:`	table name, @param sqlschema
`comment:`	comment about table,
`name_short:`	name_short,
`name_long:`	name_long,
`name_full:`	name_full @param pkey

column(*self*, *name*, *dtype*=`None`, *size*=`None`, *default*=`None`, *notnull*=`None`, *unique*=`None`, *indexed*=`None`, *sqlname*=`None`, *comment*=`None`, *name_short*=`None`, *name_long*=`None`, *name_full*=`None`, *group*=`None`, ***kwargs*)

insert a column into a table. Child of table.

Parameters

`name:`	column name, @param dtype
`size:`	string, 'min:max' or fixed lenght 'len'
`comment:`	comment about column, @param sqlname
`name_short:`	name_short,
`name_long:`	name_long,
`name_full:`	name_full @param default @param notnull @param unique @param indexed

index(*self*, *columns*=`None`, *name*=`None`, *unique*=`None`)

Add an index to a column. self must be a column src or an index_list

Parameters

`columns:`	list, or tuple, or string separated by commas
`name:`	index name

relation(*self*, *related_column*, *mode*=`'relation'`, *name_one*=`None`, *name_many*=`None`, *eager_one*=`None`, *eager_many*=`None`, ***kwargs*)

Parameters

`mode:`	relation (dflt), insensitive, foreignkey
`name_one:`	alias for one relation
`name_many:`	alias for many relation

_(*self*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path

`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__init__(*self*, *source*=`None`)

A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:`	an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:`	specifies if a node with an existing path overwrite the value or append it.
`_validators:`	an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:`	all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
`_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:` path of the node.
`validator:` the type of validation.
`parameterString:` string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value

a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters

`encoding:` default is 'UTF-8'

Return Value

a formatted representation of the bag contents (ascii)

attributes(*self*)

backref(*self*)

child(*self*, *tag*, *name*=`'*_#'`, *content*=`None`, *_parentTag*=`None`, ***kwargs*)

This method sets a new item of the type tag into the current structure

Parameters

`tag:` structure type
`name:` structure name. Default value is formed by 'tag_position'
`content:` optional structure content
`kwargs:` other parameters @return : the new structure if content is none else the parent

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value

a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value

a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters

`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars

`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

load(*self*, *path*)

This method loads the structure from an xml file

Parameters

`path:` path of the file

makePicklable(*self*)

This method make a Bag picklable.

makeRoot(*cls*, *source*=`None`, *protocls*=`None`)

This method builds the root instance for the given class.

Parameters

`cls:` structure class
`source:` filepath of xml file

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

root(*self*)

save(*self*, *path*)

This method saves the structure as an xml file

Parameters

`path:` destination of the saved file

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:` specifies if a node with an existing path overwrite the value or append it.
`_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

`path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function linked to update event.
`insert:` the eventhandler function linked to insert event.
`delete:` the eventhandler function linked to delete event.
`any:` the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback:` the function which is called.

26.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.4.3 Class Variables

Name	Description
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

26.5 Class DbModelObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj

Base class for all the StructObj in this module

26.5.1 Methods

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

doInit(*self*)

dbroot(*self*)

db(*self*)

adapter(*self*)

sqlname(*self*)

getTag(*self*)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

get(*self*, *name*, *default*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

values(*self*)

26.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.5.3 Class Variables

Name	Description
name_short	**Value:** `property(_get_name_short, _set_name_short)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.6 Class DbPackageObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj
gnr.sql.gnrsqlmodel.DbPackageObj

26.6.1 Methods

tables(*self*)

property. Returns a SqlTableList

table(*self*, *name*)

returns a table

tableSqlName(*self*, *tblobj*)

return the name of the given SqlTable

Parameters

`tblobj:` an instance of SqlTable

sqlschema(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doInit(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

sqlname(*self*)

values(*self*)

26.6.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.6.3 Class Variables

Name	Description
sqlclass	**Value:** `"package"`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.7 Class DbTableObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj
gnr.sql.gnrsqlmodel.DbTableObj

26.7.1 Methods

doInit(*self*)
Overrides: gnr.sql.gnrsqlmodel.DbModelObj.doInit

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

newRelationResolver(*self*, ***kwargs*)

pkg(*self*)

property. Returns the SqlPackage that contains the current table

fullname(*self*)

property. Returns the absolute table's name

sqlschema(*self*)

property. Returns the sqlschema

sqlname(*self*)

property. Returns the table's sqlname

Overrides: gnr.sql.gnrsqlmodel.DbModelObj.sqlname

sqlfullname(*self*)

property. Returns the table's sqlfullname

sqlnamemapper(*self*)

pkey(*self*)

property. Returns the table's pkey

rowcaption(*self*)

property. Returns the table's rowcaption

columns(*self*)

Returns an SqlColumnList

indexes(*self*)

Returns an SqlIndexedList

relations(*self*)

column(*self*, *name*)

Returns a column object.

Parameters

`name`: A column's name or a relation path starting from the current table.
Eg:@director_id.name

relations_one(*self*)

This method returns a bag containing all the ManyToOne relations that point to the current table

relations_many(*self*)

This method returns a bag containing all the OneToMany relations that starts from to the current table

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

values(*self*)

26.7.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.7.3 Class Variables

Name	Description
sqlclass	**Value:** `'table'`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.8 Class DbColumnObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj
gnr.sql.gnrsqlmodel.DbColumnObj

26.8.1 Methods

doInit(*self*)
Overrides: gnr.sql.gnrsqlmodel.DbModelObj.doInit

dtype(*self*)

property. Returns the data type

isReserved(*self*)

property. Returns the attribute reserved

readonly(*self*)

property. Returns the attribute readonly

pkg(*self*)

property. Returns the SqlPackage

table(*self*)

property. Returns the SqlTable

sqlschema(*self*)

property. Returns the sqlschema

sqlfullname(*self*)

property. Returns the sqlfullname

relatedTable(*self*)

Get the SqlTable that is related by the current column

relatedColumn(*self*)

Get the SqlColumn that is related by the current column

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

sqlname(*self*)

values(*self*)

26.8.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.8.3 Class Variables

Name	Description
sqlclass	**Value:** `'column'`

continued on next page

Name	Description
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.9 Class DbColumnListObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj
gnr.sql.gnrsqlmodel.DbColumnListObj

26.9.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, **kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doInit(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

sqlname(*self*)

values(*self*)

26.9.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.9.3 Class Variables

Name	Description
sqlclass	**Value:** `"column_list"`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.10 Class DbIndexListObj

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.sql.gnrsqlmodel.DbModelObj

gnr.sql.gnrsqlmodel.DbIndexListObj

26.10.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doInit(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

sqlname(*self*)

values(*self*)

26.10.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.10.3 Class Variables

Name	Description
sqlclass	**Value:** `"index_list"`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.11 Class DbPackageListObj

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.sql.gnrsqlmodel.DbModelObj

gnr.sql.gnrsqlmodel.DbPackageListObj

26.11.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doInit(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

sqlname(*self*)

values(*self*)

26.11.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.11.3 Class Variables

Name	Description
sqlclass	**Value:** `"package_list"`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.12 Class DbTableListObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj
gnr.sql.gnrsqlmodel.DbTableListObj

26.12.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doInit(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

sqlname(*self*)

values(*self*)

26.12.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.12.3 Class Variables

Name	Description
sqlclass	**Value:** `"table_list"`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.13 Class DbIndexObj

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.sql.gnrsqlmodel.DbModelObj
gnr.sql.gnrsqlmodel.DbIndexObj

26.13.1 Methods

sqlname(*self*)
Overrides: gnr.sql.gnrsqlmodel.DbModelObj.sqlname

table(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

adapter(*self*)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

db(*self*)

dbroot(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doInit(*self*)

get(*self*, *name*, *default*=`None`)

getAttr(*self*, *attr*=`None`, *dflt*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

values(*self*)

26.13.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.13.3 Class Variables

Name	Description
sqlclass	**Value:** `"index"`
name_full	**Value:** `property(_get_name_full, _set_name_full)`
name_long	**Value:** `property(_get_name_long, _set_name_long)`
name_short	**Value:** `property(_get_name_short, _set_name_short)`
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

26.14 Class RelationTreeResolver

object
gnr.core.gnrbag.BagResolver
gnr.sql.gnrsqlmodel.RelationTreeResolver

26.14.1 Methods

resolverSerialize(*self*)
Overrides: gnr.core.gnrbag.BagResolver.resolverSerialize

setDbroot(*self*, *dbroot*)

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

26.14.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.14.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True, 'main_tbl': None, 'tbl...`
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
classArgs	**Value:** `[]`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

26.15 Class ModelSrcResolver

object

gnr.core.gnrbag.BagResolver

gnr.sql.gnrsqlmodel.ModelSrcResolver

26.15.1 Methods

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

resolverSerialize(*self*)
Overrides: gnr.core.gnrbag.BagResolver.resolverSerialize

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

init(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

26.15.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

26.15.3 Class Variables

Name	Description
classKwargs	**Value:** `{'cacheTime': 300, 'readOnly': False, 'dbroot': None}`
classArgs	**Value:** `['dbId']`

continued on next page

Name	Description
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

26.16 Class ConfigureAfterStartError

exceptions.Exception
└
gnr.sql.gnrsqlmodel.ConfigureAfterStartError

26.16.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

27 Module gnr.sql.gnrsqltable

27.1 Variables

Name	Description
__version__	**Value:** `'1.0b'`
gnrlogger	**Value:** `gnrlogging.getLogger('gnr.sql.gnrsqltable')`

27.2 Class SqlTable

object
gnr.core.gnrlang.GnrObject
gnr.sql.gnrsqltable.SqlTable

27.2.1 Methods

__init__(*self*, *tblobj*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

model(*self*)

property model. Return the corresponding DbTableObj object

pkg(*self*)

property pkg. Return the DbPackageObj object that contains the current table

db(*self*)

property db Return the GnrSqlDb object

dbroot(*self*)

property db Return the GnrSqlDb object

column(*self*, *name*)

property db Return the DbColumnObj object

attributes(*self*)

pkey(*self*)

property db Return the DbColumnObj object

rowcaption(*self*)

property rowcaption. Returns the table's rowcaption

columns(*self*)

property columns Returns the DbColumnListObj object

indexes(*self*)

property indexes Returns the DbIndexListObj object

relations_one(*self*)

property relations_one Return a bag of relations that start from the current table

relations_many(*self*)

property relations_many Return a bag of relations that point to the current table

record(*self*, *pkey*=`None`, *where*=`None`, *lazy*=`None`, *eager*=`None`, *mode*=`'bag'`, *relationDict*=`None`, ***kwargs*)

This method is used to get a single record of the table. It returns a SqlRecordResolver. The record can be identified by

- its primary key
- one or more conditions passed as kwargs (e.g. username='foo')
- a where condition

Parameters

`pkey:`	record primary key.
`where(optional):`	This is the sql "WHERE" clause. We suggest not to use hardcoded values into the where clause, but refer to variables passed to the selection method as kwargs e.g. where="$date BETWEEN :mybirthday AND :christmas", mybirthday=mbd, christmas=xmas
`lazy:`	
`eager:`	
`mode:`	bag, dict, json
`relationDict(optional):`	this is a dictionary that contains couples composed by fieldName and relationPath e.g. {'$member_name':'@member_id.name'}

query(*self*, *columns*='*', *where*=None, *order_by*=None, *distinct*=None, *limit*=None, *offset*=None, *group_by*=None, *having*=None, *relationDict*=None, *sqlparams*=None, *workTable*=None, *pyWhere*=None, *mode*=None, ***kwargs*)

```
This method return an object SqlQuery object which represents a query that
can be executed with different modes.

   @param columns: it represents what stays between 'SELECT' and 'FROM' in the traditional SQL query.
        It is a string of column names and related fields separated by comma.
        Each column's name is prefixed with '$'. Related fields uses a syntax based on the char '@'
        and 'dot notation'.
        e.g. "@member_id.name".For selecting all columns use che char '*'.
        columns parameter accepts also special statements such as 'COUNT','DISTINCT' and 'SUM'.

   @param where (optional):This is the sql "WHERE" clause. We suggest not
              to use hardcoded values into the where clause, but
              refer to variables passed to the query method as kwargs
              e.g. where="$date BETWEEN :mybirthday AND :christmas", mybirthday=mbd, christmas=xmas

   @param order_by (optional): this param corrisponds to sql ORDER BY operator
   @param distinct (optional): this param corrisponds to sql DISTINCT operator
   @param limit (optional): number of result's rows.
   @param offset (optional): this param corrisponds to sql OFFSET operator
   @param group_by (optional): this param corrisponds to sql GROUP BY operator
   @param having (optional): this param corrisponds to sql HAVING operator
   @param relationDict (optional):a dictionary which associates relationPath names
                               with an alias name. eg: {'$member_name':'@member_id.name'}
   @param sqlparams (optional): an optional dictionary for sql query parameters.

   @param mode (optional):If you use this param.
   ATTENTION: this param makes the methond return a resolver in stand of a SqlQuery.
       *count: returns the number of rows
       *fetch: returns the same result of the adapter's fetch
       *dictlist: returns the result as a list of dictionaries
       *bag: returns a result as a Bag divided into two parts 'headers'
             and 'rows'. headers contains the column names
              while rows contains the selection's rows as bags. Each row's column
             is a bag itself and has a SqlRecordResolver
   @param **kwargs : another way to pass sql query parameters
```

frozenSelection(*self*, *fpath*)

it gets a pickled selection

existsRecord(*self*, *record*)

This method check if a record already exists in the table

insertOrUpdate(*self*, *record*)

This method inserts or updates a single record. If the record doesn't exist it inserts, else it updates.

Parameters

`record_data:` a dictionary that represent the record that must be updated

insert(*self*, *record*)

This method inserts a single record.

Parameters

`record_data:` a dictionary that represent the record that must be inserted

delete(*self*, *record*)

This method deletes a single record.

Parameters

`record_data:` a dictionary that represent the record that must be deleted

update(*self*, *record*)

This method updates a single record.

Parameters

`record_data:` a dictionary that represent the record that must be updated

newPkeyValue(*self*)

This method get a new univoque id to use as primary key on the current table

onIniting(*self*)

onInited(*self*)

trigger_onInserting(*self*, *record*)

trigger_onInserted(*self*, *record*)

trigger_onUpdating(*self*, *record*)

trigger_onUpdated(*self*, *record*)

trigger_onDeleting(*self*, *record*)

trigger_onDeleted(*self*, *record*)

rowcaptionDecode(*self*, *rowcaption*=`None`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

mixin(*self*, *cls*, ***kwargs*)

27.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

28 Module gnr.sql.gnrsqlutils

gnrsqlutils.py

28.1 Class ModelExtractor

object
└ **gnr.sql.gnrsqlutils.ModelExtractor**

28.1.1 Methods

__init__(*self*, *dbroot*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

extractModelSrc(*self*, *root*)

buildSchemata(*self*, *root*)

buildTables(*self*, *pkg*, *pkg_name*)

buildColumns(*self*, *tbl*, *pkg_name*, *tbl_name*)

buildIndexes(*self*, *tbl*, *pkg_name*, *tbl_name*)

buildRelations(*self*, *root*)

buildViews(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

28.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

28.2 Class SqlModelChecker

object
└
gnr.sql.gnrsqlutils.SqlModelChecker

This class has to keep a database aligned with its logical structure in the GnrSqlDb. If there is any change in the modelobj, database is automatically updated.

28.2.1 Methods

__init__(*self*, *db*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

checkDb(*self*)

prepares self.actual_tables, self.actual_schemata, self.actual_views and calls _checkPackage for each package. Returns a list of instructions for the database building.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

28.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

29 Package gnr.utils

29.1 Modules

- **gnrmail** *(Section 30, p. 272)*

30 Module gnr.utils.gnrmail

30.1 Functions

sendmail(*host*, *from_address*, *to_address*, *subject*, *body*, *user*='', *password*='')

31 Package gnr.web

32 Module gnr.web.gnrhtmlformatter

32.1 Class GnrFormatSkipRowException

exceptions.Exception
└ **gnr.web.gnrhtmlformatter.GnrFormatSkipRowException**

32.1.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

32.2 Class HtmlTable

object
└ **gnr.web.gnrhtmlformatter.HtmlTable**

32.2.1 Methods

__init__(*self*, *page*, *tblobj*, *skin*=`None`, *row_template*=`None`, *locale*=`None`, *colformats*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

page(*self*)

rownum(*self*)

headers(*self*, *title*=`''`, *header*=`''`, *tableclass*=`''`, *encoding*=`'utf-8'`, ***kwargs*)

footer(*self*)

getTHead(*self*, *fields*)

buildTableRow(*self*, *row*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

32.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

32.3 Class HtmlSelection

object

gnr.web.gnrhtmlformatter.HtmlSelection

32.3.1 Methods

__init__(*self*, *page*, *selection*, *skin*=`None`, *row_template*=`None`, *locale*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

page(*self*)

rownum(*self*)

rowGenerator(*self*)

get_html(*self*, *tblCssClass*=`''`, *tblId*=`''`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

32.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

32.4 Class HtmlFormatter

object
└
gnr.web.gnrhtmlformatter.HtmlFormatter

32.4.1 Methods

__init__(*self*, *page*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

page(*self*)

htmlTable(*self*)

htmlSelection(*self*, *selection*, *row_template*=`None`, *output*=`None`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

32.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

33 Module gnr.web.gnrsourcefragments

33.1 Functions

frg_periodPicker(*where*, *name_from*, *name_to*, *date_from*=`'*year.start'`, *date_to*=`'*year.end'`, *onValueChanged*=`''`)

date_builder(*value*)

frg_dblinker(*req*, *where*, *name*, *rpc*, *rpc_select*=`''`, *onValueChanged*=`''`)

encode_kwargs(*pageurl*, *url*, ***kwargs*)

34 Module gnr.web.gnrstandardpages

34.1 Functions

tablePage(*page*, *source*=None, *title*='', *tableclass*='', *thead*='', *row_template*='', *row_cb*=None, *skin*='gnr_blue', *excel*=None, *header*='', *footer*='', *filename*=None, *tot_cb*=None, *row_template_totals*=None, *locale*=None, *row_formats*=None, ***kwargs*)

tableHtml(*page*, *source*=None, *title*='', *tableclass*='', *thead*='', *row_template*='', *row_cb*=None, *skin*='gnr_blue', *excel*=None, *header*='', *footer*='', *filename*=None, *tot_cb*=None, *row_template_totals*=None, *locale*=None, *row_formats*=None, ***kwargs*)

34.2 Class TableBuilder

object
└ **gnr.web.gnrstandardpages.TableBuilder**

34.2.1 Methods

__init__(*self*, *page*, *source*=None, *title*='', *tableclass*='', *thead*='', *row_template*='', *row_cb*=None, *skin*='gnr_blue', *excel_nobr*=None, *header*='', *footer*='', *filename*=None, *tot_cb*=None, *row_template_totals*=None, *locale*=None, *row_formatDict*=None, *row_maskDict*=None, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

counter(*self*)

sendFile(*self*, *filename*, *mimetype*, *ext*, *content*)

requestWrite(*self*, *html*)

fromList(*self*)

fromServerCursor(*self*, *n*)

buildTableRow(*self*, *row*, *totals*=False)

templateReplaceRow(*self*, *row*, *totals*=False)

prepareTableRow(*self*, *row*, *totals*=False, *excel*=False, *excel_nobr*=False)

getColumnFormat(*self*, *k*, *v*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

34.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

34.3 Class StringTableBuilder

object

gnr.web.gnrstandardpages.TableBuilder

gnr.web.gnrstandardpages.StringTableBuilder

34.3.1 Methods

doTable(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *page*, *source*=`None`, *title*=`''`, *tableclass*=`''`, *thead*=`''`, *row_template*=`''`, *row_cb*=`None`, *skin*=`'gnr_blue'`, *excel_nobr*=`None`, *header*=`''`, *footer*=`''`, *filename*=`None`, *tot_cb*=`None`, *row_template_totals*=`None`, *locale*=`None`, *row_formatDict*=`None`, *row_maskDict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

buildTableRow(*self*, *row*, *totals*=`False`)

counter(*self*)

fromList(*self*)

fromServerCursor(*self*, *n*)

getColumnFormat(*self*, *k*, *v*)

prepareTableRow(*self*, *row*, *totals*=`False`, *excel*=`False`, *excel_nobr*=`False`)

requestWrite(*self*, *html*)

sendFile(*self*, *filename*, *mimetype*, *ext*, *content*)

templateReplaceRow(*self*, *row*, *totals*=`False`)

34.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

34.4 Class PageTableBuilder

object

gnr.web.gnrstandardpages.TableBuilder

gnr.web.gnrstandardpages.PageTableBuilder

34.4.1 Methods

doTable(*self*)

openPage(*self*)

closePage(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *page*, *source*=`None`, *title*=`''`, *tableclass*=`''`, *thead*=`''`, *row_template*=`''`, *row_cb*=`None`, *skin*=`'gnr_blue'`, *excel_nobr*=`None`, *header*=`''`, *footer*=`''`, *filename*=`None`, *tot_cb*=`None`, *row_template_totals*=`None`, *locale*=`None`, *row_formatDict*=`None`, *row_maskDict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

buildTableRow(*self*, *row*, *totals*=`False`)

counter(*self*)

fromList(*self*)

fromServerCursor(*self*, *n*)

getColumnFormat(*self*, *k*, *v*)

prepareTableRow(*self*, *row*, *totals*=`False`, *excel*=`False`, *excel_nobr*=`False`)

requestWrite(*self*, *html*)

sendFile(*self*, *filename*, *mimetype*, *ext*, *content*)

templateReplaceRow(*self*, *row*, *totals*=`False`)

34.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

34.5 Class ExcelTableBuilder

object
gnr.web.gnrstandardpages.TableBuilder
gnr.web.gnrstandardpages.ExcelTableBuilder

34.5.1 Methods

doTable(*self*)

buildTableRow(*self*, *row*, *totals*=`False`)
Overrides: gnr.web.gnrstandardpages.TableBuilder.buildTableRow

templatePageExcel(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *page*, *source*=`None`, *title*=`''`, *tableclass*=`''`, *thead*=`''`, *row_template*=`''`, *row_cb*=`None`, *skin*=`'gnr_blue'`, *excel_nobr*=`None`, *header*=`''`, *footer*=`''`, *filename*=`None`, *tot_cb*=`None`, *row_template_totals*=`None`, *locale*=`None`, *row_formatDict*=`None`, *row_maskDict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

counter(*self*)

fromList(*self*)

fromServerCursor(*self*, *n*)

getColumnFormat(*self*, *k*, *v*)

prepareTableRow(*self*, *row*, *totals*=`False`, *excel*=`False`, *excel_nobr*=`False`)

requestWrite(*self*, *html*)

sendFile(*self*, *filename*, *mimetype*, *ext*, *content*)

templateReplaceRow(*self*, *row*, *totals*=`False`)

34.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

35 Module gnr.web.gnrstandardpages_old

Created by Giovanni Porcari and Francesco Cavazzana on 2007-03-24.

35.1 Functions

buildTableRow(*counter*, *row*, *row_template*, *page*, *locale*, *row_formats*, *row_cb*=`None`, *excel*=`False`, *excel_nobr*=`False`)

tablePage(*page*, *source*=`None`, *title*=`''`, *tableclass*=`''`, *thead*=`''`, *row_template*=`''`, *row_cb*=`None`, *skin*=`'gnr_blue'`, *excel*=`None`, *header*=`''`, *footer*=`''`, *filename*=`None`, *tot_cb*=`None`, *row_template_totals*=`None`, *locale*=`None`, *row_formats*=`None`, ***kwargs*)

excelTablePage(*page*, *source*, *thead*=`''`, *row_template*=`''`, *row_cb*=`None`, *header*=`''`, *footer*=`''`, *filename*=`None`, *locale*=`None`, *excel_nobr*=`False`, *row_formats*=`None`, ***kwargs*)

tableHtml(*page*, *source*=`None`, *tableclass*=`''`, *thead*=`''`, *row_template*=`''`, *row_cb*=`None`, *skin*=`'gnr_blue'`, *excel*=`None`, *header*=`''`, *footer*=`''`, *filename*=`None`, *tot_cb*=`None`, *row_template_totals*=`None`, *locale*=`None`, *row_formats*=`None`, ***kwargs*)

35.2 Class Counter

object
└ **gnr.web.gnrstandardpages_old.Counter**

35.2.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

add(*self*, *n*=`1`)

get(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, *...*)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

35.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36 Module gnr.web.gnrwebcore

core.py

Created by Giovanni Porcari on 2007-03-24.

36.1 Functions

indexFolder(*req*, ***kwargs*)

36.2 Variables

Name	Description
CONNECTION_TIMEOUT	**Value:** `1800`
CONNECTION_REFRESH	**Value:** `300`
AUTH_OK	**Value:** `0`
AUTH_NOT_LOGGED	**Value:** `1`
AUTH_FORBIDDEN	**Value:** `-1`

36.3 Class GrowlStub

object
└
gnr.web.gnrwebcore.GrowlStub

36.3.1 Methods

notify(*self*, **args*, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.4 Class GnrWebClientError

exceptions.Exception
└ **gnr.web.gnrwebcore.GnrWebClientError**

36.4.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

36.5 Class GnrWebServerError

exceptions.Exception
└
gnr.web.gnrwebcore.GnrWebServerError

36.5.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

36.6 Class Dojo1_mixin

object
└
gnr.web.gnrwebcore.Dojo1_mixin

36.6.1 Methods

rootLayoutContainer(*self*, *root*, *menues*=`'Browse'`, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.6.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.7 Class GnrWebPage

object

gnr.core.gnrlang.GnrObject

gnr.web.gnrwebcore.GnrWebPage

36.7.1 Methods

__init__(*self*, *req*, *customclass*, *filepath*, *page_id*=`None`, *xxcnt*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

index(*self*)

rpc_eval(*self*, *expr*)

onEnd(*self*)

mixins(*self*)

Implement this method in your page for mixin the page with methods from the local _component folder

Return Value
list of mixin names, moduleName:className

onInit(*self*)

requestWrite(*self*, *txt*, *encoding*=`'utf-8'`)

gnotify(*self*, *title*, *description*)

growl(*self*)

log(*self*, *msg*)

user(*self*)

Get the user from hidden _user attribute.

userTags(*self*)

avatar(*self*)

pageAuthTags(*self*, *method*=`None`, ***kwargs*)

rpc_doLogin(*self*, *login*=`None`, *guestName*=`None`, ***kwargs*)

Service method that set user's avatar into its connection if

- The user exists and his password is correct.
- The user is guest

pageLocalDocument(*self*, *docname*)

freezeSelection(*self*, *selection*, *name*)

unfreezeSelection(*self*, *dbtable*, *name*)

connectionFolder(*self*)

connection(*self*)

utils(*self*)

rpc(*self*)

html(*self*)

session(*self*)

catalog(*self*)

config(*self*)

siteUri(*self*)

siteFolder(*self*)

siteName(*self*)

app(*self*)

db(*self*)

application(*self*)

packages(*self*)

package(*self*)

packageId(*self*)

dbform(*self*)

logfile(*self*)

eagers(*self*)

subclass to define page local eager relations

siteHandler(*self*)

rootLayoutContainer(*self*, *root*, *menues*=`'Browse'`, ***kwargs*)

rpc_menu_browse(*self*, *path*=`None`)

toJson(*self*, *obj*)

rpc_main(*self*, *_auth*=`AUTH_OK`, ***kwargs*)

main(*self*, *root*, ***kwargs*)

forbiddenPage(*self*, *root*, ***kwargs*)

loginPage(*self*, *root*, ***kwargs*)

windowTitle(*self*)

newSourceRoot(*self*)

rpc_resolverRecall(*self*, *resolverPars*=`None`, ***auxkwargs*)

kidTemplate(*self*, *path*, ***kwargs*)

cheetahTemplate(*self*, *req*, *path*, ***kwargs*)

makoTemplate(*self*, *path*, ***kwargs*)

staticFile(*self*, *path*, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

mixin(*self*, *cls*, ***kwargs*)

36.7.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.7.3 Class Variables

Name	Description
locale	**Value:** `property(_get_locale, _set_locale)`

36.8 Class GnrWebRpc

object
└
gnr.web.gnrwebcore.GnrWebRpc

36.8.1 Methods

__call__(*self*, *page*, *method*=`None`, *mode*=`'bag'`, *_auth*=`AUTH_FORBIDDEN`, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.8.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.9 Class GnrWebConnection

object
└
gnr.web.gnrwebcore.GnrWebConnection

36.9.1 Methods

__init__(*self*, *page*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

data(*self*)

writedata(*self*)

Write immediatly the disk file, not the cookie: use it for update data during a long process

write(*self*)

page(*self*)

appSlot(*self*)

updateAvatar(*self*, *avatar*)

connectionFolder(*self*)

connectionFile(*self*)

rpc_logout(*self*, ***kwargs*)

close(*self*)

connFolderRemove(*self*, *path*=`None`)

verify(*self*, *cookie*)

cleanExpiredConnections(*self*, *rnd*=`None`)

connectionList(*self*, *cleanExpired*=`False`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.9.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.10 Class GnrWebSession

object
└
gnr.web.gnrwebcore.GnrWebSession

36.10.1 Methods

__init__(*self*, *page*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

getSessionData(*self*, *page_id*)

save(*self*, *page*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.10.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.11 Class GnrWebUtils

object
└
gnr.web.gnrwebcore.GnrWebUtils

36.11.1 Methods

__init__(*self*, *page*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

siteFolder(*self*, **args*, ***kwargs*)

rootFolder(*self*, **args*, ***kwargs*)

pageFolder(*self*, **args*, ***kwargs*)

relativePageFolder(*self*, **args*, ***kwargs*)

abspath(*self*, *path*)

absoluteDiskPath(*self*, *path*)

diskPathToUri(*self*, *url*)

readFile(*self*, *path*)

filename(*self*)

dirbag(*self*, *path*=`''`, *base*=`''`, *include*=`''`, *exclude*=`None`, *ext*=`''`)

pageTitle(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.11.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.12 Class GnrProcessHandler

object

gnr.web.gnrwebcore.GnrProcessHandler

36.12.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

getSiteHandler(*self*, *page*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.12.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.13 Class GnrWebAppHandler

object
┬
└ **gnr.web.gnrwebcore.GnrWebAppHandler**

36.13.1 Methods

__init__(*self*, *page*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

getDb(*self*, *dbId*=`None`)

__getitem__(*self*, *dbId*=`None`)

page(*self*)

appId(*self*)

getPackages(*self*)

rpc_getPackages(*self*)

getTables(*self*, *pkg*=`None`)

rpc_getTables(*self*, *pkg*=`None`)

getTablesTree(*self*)

rpc_getTablesTree(*self*)

getTableFields(*self*, *pkg*=`''`, *table*=`''`, ***kwargs*)

rpc_getTableFields(*self*, *pkg*=`''`, *table*=`''`, ***kwargs*)

dbStructure(*self*, *path*=`''`, ***kwargs*)

rpc_dbStructure(*self*, *path*=`''`, ***kwargs*)

rpc_selection(*self*, *pkg*=`''`, *dbtable*=`''`, *columns*=`''`, *relationDict*=`None`, *where*=`None`, *limit*=`None`, *resultmode*=`'dictlist'`, ***kwargs*)

---

returns a list of dict: use for feed a filteringTable with json data

selection_as_default(*self*, *tblobj*, *selectionResolver*, *resultmode*, ***kwargs*)

selection_as_html(*self*, *tblobj*, *selection*, *resultmode*, ***kwargs*)

rpc_getRecord(*self*, *dbtable*=`None`, *pkg*=`None`, *pkey*=`None`, ***kwargs*)

rpc_saveRecord(*self*, *dbtable*=`None`, *recordBag*=`None`)

rpc_dbSelect(*self*, *dbtable*=`None`, *dbfield*=`None`, *keyfield*=`'pkey'`, *condition*=`None`, *data*=`None`, *limit*=`10`, ***kwargs*)

rpc_combo_menu(*self*, *dbtable*=`None`, *dbfield*=`None`, *keyfield*=`None`, *condition*=`None`, *data*=`None`, *limit*=`10`, ***kwargs*)

rpc_db_table_filter(*self*, *columns*=`None`, *value*=`None`, *dbtable*=`None`, *dbfield*=`None`, *condition*=`None`, *recmax*=`None`, *searchmode*=`'wordstart'`, *charmin*=`None`, ***kwargs*)

rpc_getRecordForm(*self*, *dbtable*=`None`, *fields*=`None`, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.13.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

36.14 Class GnrIndexWebPage

object
└
gnr.web.gnrwebcore.GnrIndexWebPage

36.14.1 Methods

main(*self*, *root*, ***kwargs*)

windowTitle(*self*)

walkDirectory(*self*, *bag*, *where*, *back*=`None`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

36.14.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

37 Module gnr.web.gnrwebdbtables

37.1 Class GnrWebDbForm

object
└ **gnr.web.gnrwebdbtables.GnrWebDbForm**

37.1.1 Methods

__init__(*self*, *page*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

page(*self*)

getTableId(*self*, *dbtable*)

topPane(*self*, *pane*, *dbtable*, *listform*=`None`, *columns*=`None`, *filterOn*=`None`)

bodyPane(*self*, *pane*, *dbtable*)

formColsToSqlCols(*self*, *fieldlist*, *tblobj*)

listForm(*self*, *filterwhere*, *tablewhere*, *dbtable*=`None`, *gnrId*=`None`, *columns*=`None`, *filterOn*=`None`, ***kwargs*)

recordForm(*self*, *pane*, *dbtable*=`''`, *gnrId*=`None`)

getRecordForm(*self*, *where*, *dbtable*=`None`, *datasource*=`None`, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

37.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

38 Module gnr.web.gnrwebstart

gnrwebtemplates.py

38.1 Functions

indexPage(*req*, *skin*=`'gnr_blue'`, *customCss*=`''`, *page_id*=`None`, *charset*=`'utf-8'`, ***startArgs*)

getCssBag(*req*, *skin*)

findInheritedFile(*req*, *skin*, *filename*, *fileext*, *onlyFirst*=`False`)

relocateUrl(*req*, *url*)

mainFile(*req*)

makeRelative(*req*, *url*)

39 Module gnr.web.gnrwebstruct

39.1 Class GnrDomSrcError

exceptions.Exception
└ **gnr.web.gnrwebstruct.GnrDomSrcError**

39.1.1 Methods

__getitem__(...)

__init__(...)

__str__(...)

39.2 Class GnrDomElem

object
└ **gnr.web.gnrwebstruct.GnrDomElem**

39.2.1 Methods

__init__(*self*, *obj*, *tag*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__call__(*self*, **args*, ***kwargs*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

39.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

39.3 Class GnrDomSrc

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.core.gnrstructures.GnrStructData
gnr.web.gnrwebstruct.GnrDomSrc

39.3.1 Methods

makeRoot(*cls*, *page*, *source*=`None`)
This method builds the root instance for the given class.

Overrides: gnr.core.gnrstructures.GnrStructData.makeRoot extit(inherited documentation)

page(*self*)

__getattr__(*self*, *fname*)

child(*self*, *tag*, *name*=`None`, *envelope*=`None`, ***kwargs*)
This method sets a new item of the type tag into the current structure

Overrides: gnr.core.gnrstructures.GnrStructData.child extit(inherited documentation)

h1(*self*, *content*=`None`, ***kwargs*)

h2(*self*, *content*=`None`, ***kwargs*)

h3(*self*, *content*=`None`, ***kwargs*)

h4(*self*, *content*=`None`, ***kwargs*)

h5(*self*, *content*=`None`, ***kwargs*)

h6(*self*, *content*=`None`, ***kwargs*)

li(*self*, *content*=`None`, ***kwargs*)

td(*self*, *content*=`None`, ***kwargs*)

th(*self*, *content*=`None`, ***kwargs*)

span(*self*, *content*=`None`, ***kwargs*)

pre(*self*, *content*=`None`, ***kwargs*)

div(*self*, *content*=`None`, ***kwargs*)

dt(*self*, *content*=`None`, ***kwargs*)

option(*self*, *content*=`None`, ***kwargs*)

caption(*self*, *content*=`None`, ***kwargs*)

button(*self*, *caption*=`None`, ***kwargs*)

column(*self*, *label*=`''`, *field*=`''`, *expr*=`''`, *name*=`''`, ***kwargs*)

data(*self*, **args*, ***kwargs*)

script(*self*, *content*=`''`, ***kwargs*)

remote(*self*, *method*=`''`, ***kwargs*)

func(*self*, *name*, *pars*=`''`, *funcbody*=`None`, ***kwargs*)

subscribe(*self*, *what*, *event*, *func*=`None`, ***kwargs*)
This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Overrides: gnr.core.gnrbag.Bag.subscribe extit(inherited documentation)

css(*self*, *rule*, *styleRule*=`''`)

macro(*self*, *name*=`''`, *source*=`''`, ***kwargs*)

formbuilder(*self*, *cols*=`None`, *dbtable*=`None`, *lblclass*=`'gnrfieldlabel'`, *lblpos*=`'L'`, *lblalign*=`None`, *fldalign*=`None`, *lblvalign*=`'middle'`, *fldvalign*=`'middle'`, ***kwargs*)

place(*self*, *fields*)

getField(*self*, *fld*)

_(*self*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters
`what:` the key path to test.

Return Value
a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters
`path:` path of the given item.

Return Value
the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__init__(*self*, *source*=`None`)

A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:`	an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:`	specifies if a node with an existing path overwrite the value or append it.
`_validators:`	an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:`	all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
`_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:` path of the node.
`validator:` the type of validation.
`parameterString:` string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value

a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters

`encoding:` default is 'UTF-8'

Return Value

a formatted representation of the bag contents (ascii)

attributes(*self*)

backref(*self*)

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value
a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value
a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters
`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters
`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters
`path:` path of the given item.

Return Value
the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars
`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

load(*self*, *path*)

This method loads the structure from an xml file

Parameters

`path:` path of the file

makePicklable(*self*)

This method make a Bag picklable.

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.

`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

root(*self*)

save(*self*, *path*)

This method saves the structure as an xml file

Parameters

`path:` destination of the saved file

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:` specifies if a node with an existing path overwrite the value or append it.
`_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

`path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

sum(*self*, *what*='#v')

toXml(*self*, *filename*=None, *encoding*='UTF-8', *typeattrs*=True, *unresolved*=False, *autocreate*=False)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=None, *insert*=None, *delete*=None, *any*=None)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback`: the function which is called.

39.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

39.3.3 Class Variables

Name	Description
htmlNS	**Value:** `['a', 'abbr', 'acronym', 'address', 'area', 'b', 'base', ...`
dojoNS	**Value:** `['AccordionContainer', 'AnimatedPng', 'Button', 'Chart', ...`
gnrNS	**Value:** `['menu', 'menuBar', 'menuItem', 'Tree', 'Select', 'DbSele...`
genroNameSpace	**Value:** `dict([(name.lower(), name) for name in htmlNS])`
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

39.4 Class GnrFormBuilder

object
└ **gnr.web.gnrwebstruct.GnrFormBuilder**

39.4.1 Methods

__init__(*self*, *tbl*, *cols*=`None`, *dbtable*=`None`, *lblclass*=`'gnrfieldlabel'`, *lblpos*=`'L'`, *lblalign*=`None`, *fldalign*=`None`, *lblvalign*=`'middle'`, *fldvalign*=`'middle'`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

page(*self*)

tbl(*self*)

place(*self*, ***fieldpars*)

setField(*self*, *field*, *row*=`None`, *col*=`None`)

setFields(*self*, *fields*, *rowstart*=`0`, *colstart*=`0`)

getRow(*self*, *r*)

nextCell(*self*, *r*, *c*)

setRow(*self*, *fields*, *row*=`None`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

39.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

40 Package gnr.wx

41 Module gnr.wx.gnrdevtools

Spare.py is a starting point for a wxPython program.

41.1 Class DeveloperFrame

object
└ **gnr.wx.gnrdevtools.DeveloperFrame**

41.1.1 Methods

configure(*self*)

conf_logger(*self*, *pages*)

conf_browser(*self*, *pages*)

conf_wxexplorer(*self*, *pages*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

41.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

41.2 Class GnrWxInspector

object

gnr.core.gnrlang.GnrObject

gnr.wx.gnrwxapp.GnrModule

gnr.wx.gnrdevtools.GnrWxInspector

41.2.1 Methods

configure(*self*)
This methods configures the all the GnrWidgets inside the module

Overrides: gnr.wx.gnrwxapp.GnrModule.configure extit(inherited documentation)

conf_column(*self*, *pages*)

conf_pane(*self*, *pages*)

newInspector(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *application*, *name*, *path*=`None`, *startframes*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

data(*self*)

frameData(*self*, *source*=`None`)

frameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getById(*self*, *source*, *id*)

getDataNode(*self*, *source*, *id*=`None`)

getFrameData(*self*, *source*=`None`)

getFrameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getSelectedNode(*self*, *id*=`None`)

getWidget(*self*, *id*=`None`, *source*=`None`)

get_activeFrame(*self*)

init(*self*)

mixin(*self*, *cls*, ***kwargs*)

newWidget(*self*)

onFrameStarted(*self*, *name*)

onFrameStarting(*self*, *name*, *data*)

onModuleStarting(*self*)

on_menu_selected(*self*, ***kwargs*)

on_popup_selected(*self*, *event*)

run(*self*)

This method executes some startup actions: -calls the method start -set the sturtupitems into widgets attribute -for each startup item calls its handler function

saveToXml(*self*, *param*)

setFrameData(*self*, *data*, *name*=`None`)

set_activeFrame(*self*, *frame*)

stop(*self*)

widgets(*self*)

41.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

41.2.3 Class Variables

Name	Description
modulename	**Value:** `'WxInspector'`
activeFrame	**Value:** `property(get_activeFrame, set_activeFrame)`
application	**Value:** `property(_get_application, _set_application)`

41.3 Class GnrPyDocFrame

object
└ **gnr.wx.gnrdevtools.GnrPyDocFrame**

41.3.1 Methods

onBuild(*self*)

configure(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

41.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

41.4 Class GnrDevTools

object
gnr.core.gnrlang.GnrObject
gnr.wx.gnrwxapp.GnrModule
gnr.wx.gnrdevtools.GnrDevTools

41.4.1 Methods

init(*self*)
Overrides: gnr.wx.gnrwxapp.GnrModule.init

devModeOn(*self*, *obj*, *evt*=`None`)

devModeOff(*self*, *obj*, *evt*=`None`)

getRect(*self*, *obj*)

getAbsPosition(*self*, *obj*)

on_enter_window(*self*, *evt*)

on_leave_window(*self*, *evt*)

popup_inspector(*self*, *obj*, *evt*=`None`)

popup_browser(*self*, *obj*, *evt*=`None`)

popup_pydoc(*self*, *obj*, *evt*=`None`)

getDevMenu(*self*)

setFontDialog(*self*, *obj*, *evt*=`None`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *application*, *name*, *path*=`None`, *startframes*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

configure(*self*)

This methods configures the all the GnrWidgets inside the module

data(*self*)

frameData(*self*, *source*=`None`)

frameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getById(*self*, *source*, *id*)

getDataNode(*self*, *source*, *id*=`None`)

getFrameData(*self*, *source*=`None`)

getFrameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getSelectedNode(*self*, *id*=`None`)

getWidget(*self*, *id*=`None`, *source*=`None`)

get_activeFrame(*self*)

mixin(*self*, *cls*, ***kwargs*)

newWidget(*self*)

onFrameStarted(*self*, *name*)

onFrameStarting(*self*, *name*, *data*)

onModuleStarting(*self*)

on_menu_selected(*self*, ***kwargs*)

on_popup_selected(*self*, *event*)

run(*self*)

This method executes some startup actions: -calls the method start -set the sturtupitems into widgets attribute -for each startup item calls its handler function

saveToXml(*self*, *param*)

setFrameData(*self*, *data*, *name*=`None`)

set_activeFrame(*self*, *frame*)

stop(*self*)

widgets(*self*)

41.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

41.4.3 Class Variables

Name	Description
modulename	**Value:** `'_devtools'`
activeFrame	**Value:** `property(get_activeFrame, set_activeFrame)`
application	**Value:** `property(_get_application, _set_application)`

42 Module gnr.wx.gnrwidgets

gnrwx core

42.1 Variables

Name	Description
widgetlist	**Value:** `[getattr(gnrwx, x) for x in dir(gnrwx) if hasattr(getattr...`
widgetdict	**Value:** `dict([(x.shortname.lower(), x) for x in widgetlist])`

42.2 Class GnrValidator

object
└
gnr.wx.gnrwidgets.GnrValidator

42.2.1 Methods

__init__(*self*, *onbj*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

42.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

42.3 Class GnrBaseValidator

object
└
gnr.wx.gnrwidgets.GnrBaseValidator

42.3.1 Methods

validate(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

42.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

42.4 Class GnrModule

object
└
gnr.wx.gnrwidgets.GnrModule

42.4.1 Methods

__init__(*self*, *application*, *name*, *autostart*=`False`, *path*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

init(*self*)

configure(*self*)

child(*self*, *wdgtype*, *name*, ***kwargs*)

frame(*self*, *name*, ***kwargs*)

dialog(*self*, *name*, *ok*=`None`, *cancel*=`None`, ***kwargs*)

request(*self*, *message*=`None`, *default*=`None`, *title*=`None`, *invokedBy*=`None`, ***kwargs*)

message(*self*, *message*=`None`, *title*=`None`, *style*=`None`, *invokedBy*=`None`, ***kwargs*)

colorpicker(*self*, *color*=`None`, *invokedBy*=`None`, ***kwargs*)

dirchooser(*self*, *message*=`None`, *directory*=`None`, *invokedBy*=`None`, ***kwargs*)

filechooser(*self*, *message*=`''`, *default*=`None`, *directory*=`None`, *wildcard*=`None`, *invokedBy*=`None`, ***kwargs*)

fontchooser(*self*, *font*=`None`, *invokedBy*=`None`, ***kwargs*)

alert(*self*, *message*=`None`, *invokedBy*=`None`)

loadstruct(*self*, *path*)

savestruct(*self*, *path*)

wdgdata(*self*)

start(*self*)

run(*self*)

build(*self*)

stop(*self*)

buildDialog(*self*, *name*, *invokedBy*=`None`)

buildFrame(*self*, *name*, *invokedBy*=`None`)

showDialog(*self*, *name*, *invokedBy*)

on_menu_selected(*self*, *event*)

on_popup_selected(*self*, *event*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

42.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

42.4.3 Class Variables

Name	Description
modulename	**Value:** `'unnamed module'`
autostart	**Value:** `False`
wxobj	**Value:** `property(_get_wxobj, _set_wxobj)`
application	**Value:** `property(_get_application, _set_application)`

42.5 Class GnrMainModule

object

gnr.wx.gnrwidgets.GnrModule

gnr.wx.gnrwidgets.GnrMainModule

42.5.1 Methods

configure(*self*)
Overrides: gnr.wx.gnrwidgets.GnrModule.configure

splashScreen(*self*, *name*, *image*=`None`, *timeout*=`'5000'`, ***kwargs*)

taskBar(*self*, *name*, *icon*=`None`, ***kwargs*)

popup_aaa(*self*, *line*, *event*=`None`)

start(*self*)
Overrides: gnr.wx.gnrwidgets.GnrModule.start

menu_open(*self*, *event*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *application*, *name*, *autostart*=`False`, *path*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

alert(*self*, *message*=`None`, *invokedBy*=`None`)

build(*self*)

buildDialog(*self*, *name*, *invokedBy*=`None`)

buildFrame(*self*, *name*, *invokedBy*=`None`)

child(*self*, *wdgtype*, *name*, ***kwargs*)

colorpicker(*self*, *color*=`None`, *invokedBy*=`None`, ***kwargs*)

dialog(*self*, *name*, *ok*=`None`, *cancel*=`None`, ***kwargs*)

dirchooser(*self*, *message*=`None`, *directory*=`None`, *invokedBy*=`None`, ***kwargs*)

filechooser(*self*, *message*=`''`, *default*=`None`, *directory*=`None`, *wildcard*=`None`, *invokedBy*=`None`, ***kwargs*)

fontchooser(*self*, *font*=`None`, *invokedBy*=`None`, ***kwargs*)

frame(*self*, *name*, ***kwargs*)

init(*self*)

loadstruct(*self*, *path*)

message(*self*, *message*=`None`, *title*=`None`, *style*=`None`, *invokedBy*=`None`, ***kwargs*)

on_menu_selected(*self*, *event*)

on_popup_selected(*self*, *event*)

request(*self*, *message*=`None`, *default*=`None`, *title*=`None`, *invokedBy*=`None`, ***kwargs*)

run(*self*)

savestruct(*self*, *path*)

showDialog(*self*, *name*, *invokedBy*)

stop(*self*)

wdgdata(*self*)

42.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

42.5.3 Class Variables

Name	Description
modulename	**Value:** `'_main'`
autostart	**Value:** `True`
application	**Value:** `property(_get_application, _set_application)`
wxobj	**Value:** `property(_get_wxobj, _set_wxobj)`

42.6 Class GnrWidget

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.wx.gnrwidgets.GnrWidget

42.6.1 Methods

__init__(*self*, **args*, ***kwargs*)
A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrbag.Bag.__init__ extit(inherited documentation)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

child(*self*, *wdgtype*, *name*, ***kwargs*)

panel(*self*, *name*, ***kwargs*)

button(*self*, *name*, *label*=`None`, *default*=`'N'`, *action*=`None`, ***kwargs*)

checkbox(*self*, *name*, *dflt*=None, *label*=None, ***kwargs*)

checklistbox(*self*, *name*, *choices*='', *dflt*=None, *lbl*=None, *validator*=None, ***kwargs*)

field(*self*, *name*, *lbl*=None, *len*='20', *rows*='1', *datasource*=None, *value*=None, ***kwargs*)

choice(*self*, *name*, *choices*='', *dflt*=None, *lbl*=None, *validator*=None, ***kwargs*)

listbox(*self*, *name*, *choices*='', *dflt*=None, *lbl*=None, *validator*=None, ***kwargs*)

statictext(*self*, *name*, *value*=None, ***kwargs*)

notebook(*self*, *name*, *label*=None, ***kwargs*)

listbook(*self*, *name*, *label*=None, ***kwargs*)

choicebook(*self*, *name*, *label*=None, ***kwargs*)

tree(*self*, *name*, *datasource*=None, ***kwargs*)

menubar(*self*, *name*='*', ***kwargs*)

menu(*self*, *name*, *label*=None, *title*=None, ***kwargs*)

menuline(*self*, *name*, *label*=None, *_sub*=False, ***kwargs*)

lister(*self*, *name*, *datasource*=None, ***kwargs*)

gridsizer(*self*, *name*='*', *rows*=0, *cols*=0, *vgap*=0, *hgap*=0, ***kwargs*)

flexsizer(*self*, *name*='*', *rows*=0, *cols*=0, *vgap*=0, *hgap*=0, ***kwargs*)

bagsizer(*self*, *name*='*', *hgap*=0, *vgap*=0, *cols*=0, ***kwargs*)

boxsizer(*self*, *name*='*', *orient*='H', *label*=None, ***kwargs*)

hsizer(*self*, *name*='*', *label*=None, ***kwargs*)

vsizer(*self*, *name*='*', *label*=None, ***kwargs*)

multisplitter(*self*, *name*='*', *orient*='H', ***kwargs*)

splitter(*self*, *name*='*', *orient*='H', *ratio*='50', ***kwargs*)

sashwindow(*self*, *name*='*', ***kwargs*)

toolbar(*self*, *name*=`'*'`, ***kwargs*)

styledtext(*self*, *name*, *datasource*=`None`, ***kwargs*)

pythoneditor(*self*, *name*=`'*'`, *datasource*=`None`, ***kwargs*)

pycrust(*self*, *name*=`'*'`, *datasource*=`None`, ***kwargs*)

build(*self*, *path*=`None`, *invokedBy*=`None`)

buildChildren(*self*, *widgets*=`None`)

datasource(*self*)

createWidgets(*self*, *widgets*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
- `_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

- `path:` path of the node.
- `validator:` the type of validation.
- `parameterString:` string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value
a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters
`encoding:` default is 'UTF-8'

Return Value
a formatted representation of the bag contents (ascii)

backref(*self*)

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value
a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value
a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters
`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars
`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

makePicklable(*self*)

This method make a Bag picklable.

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

- `path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

- `subscriberId:` an ID can be assigned for a subscription
- `update:` the eventhandler function linked to update event.
- `insert:` the eventhandler function linked to insert event.
- `delete:` the eventhandler function linked to delete event.
- `any:` the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback`: the function which is called.

42.6.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

42.6.3 Class Variables

Name	Description
wxobj	**Value:** `property(_get_wxobj, _set_wxobj)`
application	**Value:** `property(_get_application, _set_application)`
module	**Value:** `property(_get_module, _set_module)`
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

42.7 Class GnrApplication

object

gnr.core.gnrlang.GnrObject

gnr.wx.gnrwidgets.GnrApplication

42.7.1 Methods

__init__(*self*, *modules*=`None`, *mainmodule*=`False`, *userlevel*=`'user'`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

run(*self*)

start(*self*)

build(*self*, *wxapp*)

stop(*self*)

loadModule(*self*, *modulename*, *as*=`None`, *autostart*=`None`)

getModules(*self*, *onlyPublic*=`True`)

on_popupmenu(*self*, *line*, *event*=`None`)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

mixin(*self*, *cls*, ***kwargs*)

42.7.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43 Module gnr.wx.gnrwx

gnrwx core

43.1 Functions

gnrWxDict(*m*='gnr.wx.gnrwx')

43.2 Variables

Name	Description
GNRWXDIR	**Value:** `os.path.dirname(__file__)`
CONVERTER	**Value:** `GnrClassCatalog()`
ARTDICT	**Value:** `dict([(x [4:].lower(), getattr(wx, x)) for x in dir(wx) i...`

43.3 Class GnrWxObject

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.wx.gnrwx.GnrWxObject

This is an ancestor class that defines the basic functions of GnrWx objects

43.3.1 Methods

getTag(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

parentwindow(*self*)

parentdatanode(*self*)

application(*self*)

data(*self*)

module(*self*)

rootname(*self*)

parentframe(*self*)

fullname(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

afterChildrenCreation(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

get(*self*, *name*, *default*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

init(*self*)

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

newChild(*self*, *child*)

onDelete(*self*)

root(*self*)

values(*self*)

43.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.3.3 Class Variables

Name	Description
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.4 Class GnrWxWidget

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget

43.4.1 Methods

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

subscribeDataChanges(*self*)

getDynAttributes(*self*)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

move(*self*, *pos*=`None`)

attribute_int(*self*, *v*)

attribute_wxid(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

windowAfter(*self*)

setFromDatasource(*self*, *node*=`None`)

loadValue(*self*, *dflt*=`''`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

killFocus(*self*, *event*)

setFocus(*self*, *event*)

getValue(*self*)

setValue(*self*, *value*)

getEventWidget(*self*, *event*)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

getDataNode(*self*, *source*=`None`)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

addAfterShowCall(*self*, *action*)

doAfterShowCalls(*self*)

convertedAttribute(*self*, *attr*, *default*=`None`)

setStyles(*self*, *currstyle*, *styles*)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

dynAttrCalls(*self*)

setFont(*self*, *font*, *own*=`False`)

setOwnFont(*self*, *font*)

setGnrEvents(*self*, *events*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

timerStart(*self*, *value*=`1000`)

timerOn(*self*)

timerStop(*self*)

onMouse(*self*, *evt*)

onDrop(*self*, *result*)

setDragCodes(*self*, *info*)

getFromDataObject(*self*, *dataObject*)

dataToDrag(*self*)

setDropTarget(*self*, *format*, *dropinfo*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

onDropFiles(*self*, *obj*, *paths*)

createBitmap(*self*)

setAuiInfo(*self*, *aui*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

application(*self*)

asBag(*self*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

data(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getById(*self*, *id*)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

items(*self*)

keys(*self*)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

onDelete(*self*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

root(*self*)

rootname(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

43.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.4.3 Class Variables

Name	Description
dragformats	**Value:** `'text,unicode,filename,bitmap'`
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_handlers	**Value:** {}
class_events	**Value:** {}
class_styles	**Value:** {}
parent	**Value:** `property(_get_parent, _set_parent)`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.5 Class GnrWxControl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxControl

43.5.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.5.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

continued on next page

Name	Description

43.5.3 Class Variables

Name	Description
datasourcedefault	**Value:** `':*'`
defaultvalue	**Value:** `''`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.6 Class GnrWxSizer

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSizer

43.6.1 Methods

attachToParent(*self*)

getOrient(*self*)

orient(*self*)

addSpacer(*self*, *spacer*)

newChild(*self*, *obj*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

calcBorder(*self*, *obj*)

calcFlag(*self*, *obj*, *borderwhere*)

calcFlagInner(*self*, *expand*, *align*, *borderwhere*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.6.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.6.3 Class Variables

Name	Description
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.7 Class GnrWxSplashScreen

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSplashScreen

43.7.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=None, *static*=False)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.7.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.7.3 Class Variables

Name	Description
wdgtname	**Value:** 'splashScreen'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.8 Class GnrWxTaskBarIcon

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTaskBarIcon

43.8.1 Methods

CreatePopupMenu(*self*)

This method is called by the base class when it needs to popup the menu for the default EVT_RIGHT_DOWN event. Just create the menu how you want it and return it from this function, the base class takes care of the rest.

makeIcon(*self*, *img*)

The various platforms have different requirements for the icon size...

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.8.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.8.3 Class Variables

Name	Description
wdgtname	**Value:** `'taskBar'`
TBMENU_ACTIVATE	**Value:** `wx.NewId()`
TBMENU_QUIT	**Value:** `wx.NewId()`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.9 Class GnrWxMenuBar

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxMenuBar

43.9.1 Methods

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

buildBar(*self*)

subscribeDataChanges(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.subscribeDataChanges

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.9.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.9.3 Class Variables

Name	Description
wdgtname	**Value:** `'menubar'`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.10 Class GnrWxMenu

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxMenu

43.10.1 Methods

menuSelected(*self*, *event*)

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

removeItem(*self*, *item*)

removeMenu(*self*, *menu*)

onDelete(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.onDelete

onMenuOpen(*self*, *event*)

onMenuClose(*self*, *event*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.10.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.10.3 Class Variables

Name	Description
wdgtname	**Value:** `'menu'`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.11 Class GnrWxMenuItem

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxMenuItem

43.11.1 Methods

onDelete(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.onDelete

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.11.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.11.3 Class Variables

Name	Description
wdgtname	**Value:** 'menuitem'
itemtypes	**Value:** {'normal': wx.ITEM_NORMAL, 'radio': wx.ITEM_RADIO, 'check...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)

continued on next page

Name	Description
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.12 Class GnrWxSashWindow

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSashWindow

43.12.1 Methods

sashDragged(*self*, *event*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.12.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.12.3 Class Variables

Name	Description
wdgtname	**Value:** `'sashwindow'`
class_styles	**Value:** `{'default': wx.SW_3D, '3d': wx.SW_3D, '3d_border': wx.SW_...`
sashpos	**Value:** `{'T': wx.SASH_TOP, 'B': wx.SASH_BOTTOM, 'L': wx.SASH_LEFT...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** `{}`
class_handlers	**Value:** `{}`
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.13 Class GnrWxSplitter

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSplitter

43.13.1 Methods

split(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.13.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.13.3 Class Variables

Name	Description
wdgtname	**Value:** `'splitter'`
class_styles	**Value:** `{'default': wx.SP_NOBORDER, 'noborder': wx.SP_NOBORDER, '...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.14 Class GnrWxMultiSplitter

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxMultiSplitter

43.14.1 Methods

split(*self*)

newChild(*self*, *window*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.14.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.14.3 Class Variables

Name	Description
wdgtname	**Value:** `'multiSplitter'`
class_styles	**Value:** `{'default': wx.SP_NOBORDER, 'noborder': wx.SP_NOBORDER, '...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`

continued on next page

Name	Description
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.15 Class GnrWxMediaCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxMediaCtrl

43.15.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=None, *static*=False)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.15.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.15.3 Class Variables

Name	Description
wdgtname	**Value:** 'mediacontrol'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.16 Class GnrWxPanel

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxPanel

43.16.1 Methods

afterChildrenCreation(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.afterChildrenCreation

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.16.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.16.3 Class Variables

Name	Description
wdgtname	**Value:** `'panel'`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.17 Class GnrWxToolbar

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxToolbar

non funziona

43.17.1 Methods

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.17.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.17.3 Class Variables

Name	Description
wdgtname	**Value:** 'toolbar'
class_styles	**Value:** {'default': wx.TB_HORIZONTAL | wx.NO_BORDER, 'horizontal'...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None

continued on next page

Name	Description
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.18 Class GnrWxStyledText

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxStyledText

43.18.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.18.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.18.3 Class Variables

Name	Description
wdgtname	**Value:** `'styledText'`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.19 Class GnrWxPythonEditor

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.wx.gnrwx.GnrWxObject

gnr.wx.gnrwx.GnrWxWidget

gnr.wx.gnrwx.GnrWxPythonEditor

43.19.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.19.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.19.3 Class Variables

Name	Description
wdgtname	**Value:** `'pythonEditor'`
class_handlers	**Value:** `{'margin': 'setMargin'}`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`

continued on next page

Name	Description
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.20 Class GnrWxPyCrust

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.wx.gnrwx.GnrWxObject

gnr.wx.gnrwx.GnrWxWidget

gnr.wx.gnrwx.GnrWxPyCrust

43.20.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=None, *static*=False)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.20.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.20.3 Class Variables

Name	Description
wdgtname	**Value:** 'pycrust'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.21 Class GnrWxHtmlWindow

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxHtmlWindow

43.21.1 Methods

setURL(*self*, *url*)

setPage(*self*, *html*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.21.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.21.3 Class Variables

Name	Description
wdgtname	**Value:** `'htmlwindow'`
class_styles	**Value:** `{'noscrollbar': wx.html.HW_SCROLLBAR_NEVER, 'scrollbaraut...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.22 Class GnrWxStaticText

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxStaticText

43.22.1 Methods

setValue(*self*, *value*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

getValue(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.getValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.22.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.22.3 Class Variables

Name	Description
defaultvalue	**Value:** `''`
wdgtname	**Value:** `'staticText'`
class_styles	**Value:** `{'default': wx.ALIGN_LEFT, 'left': wx.ALIGN_LEFT, 'right'...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`

continued on next page

Name	Description
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.23 Class GnrWxListbox

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxControl
gnr.wx.gnrwx.GnrWxListbox

43.23.1 Methods

setValue(*self*, *values*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

killFocus(*self*, *event*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.killFocus

getValue(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.getValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.23.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.23.3 Class Variables

Name	Description
wdgtname	**Value:** 'listbox'
class_styles	**Value:** {'default': wx.LB_DEFAULT, 'single': wx.LB_SINGLE, 'multi...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** ':*'
defaultvalue	**Value:** ''
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.24 Class GnrWxChoice

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxControl
gnr.wx.gnrwx.GnrWxChoice

43.24.1 Methods

setValue(*self*, *value*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.24.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.24.3 Class Variables

Name	Description
wdgtname	**Value:** `'choice'`
class_events	**Value:** `{'choice': wx.EVT_CHOICE}`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `':*'`
defaultvalue	**Value:** `''`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.25 Class GnrWxCheckBox

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxCheckBox

43.25.1 Methods

setValue(*self*, *value*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.25.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.25.3 Class Variables

Name	Description
wdgtname	**Value:** 'checkbox'
class_styles	**Value:** {'2state': wx.CHK_2STATE, '3state': wx.CHK_3STATE, '3user...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None

continued on next page

Name	Description
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.26 Class GnrWxCheckListBox

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxCheckListBox

43.26.1 Methods

setValue(*self*, *values*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.26.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.26.3 Class Variables

Name	Description
wdgtname	**Value:** `'checklistbox'`
class_styles	**Value:** `{'default': wx.LB_DEFAULT, 'single': wx.LB_SINGLE, 'multi...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.27 Class GnrWxButton

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.wx.gnrwx.GnrWxObject

gnr.wx.gnrwx.GnrWxWidget

gnr.wx.gnrwx.GnrWxButton

43.27.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

__init__(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.27.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.27.3 Class Variables

Name	Description
wdgtname	**Value:** `'button'`
class_styles	**Value:** `{'left': wx.BU_LEFT, 'top': wx.BU_TOP, 'right': wx.BU_RIG...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`

continued on next page

Name	Description
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.28 Class GnrWxRichTextCtrl

object

gnr.core.gnrlang.GnrObject

gnr.core.gnrstructures.GnrStructObj

gnr.wx.gnrwx.GnrWxObject

gnr.wx.gnrwx.GnrWxWidget

gnr.wx.gnrwx.GnrWxControl

gnr.wx.gnrwx.GnrWxRichTextCtrl

43.28.1 Methods

setValue(*self*, *value*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

getValue(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.getValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.28.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.28.3 Class Variables

Name	Description
wdgtname	**Value:** 'richtext'
class_styles	**Value:** {'enterkey': wx.TE_PROCESS_ENTER, 'tabkey': wx.TE_PROCESS...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** ':*'
defaultvalue	**Value:** ''
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.29 Class GnrWxDatePickerCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxControl
gnr.wx.gnrwx.GnrWxDatePickerCtrl

43.29.1 Methods

killFocus(*self*, *event*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.killFocus

setFocus(*self*, *event*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setFocus

attribute_default(*self*, *value*)

setValue(*self*, *value*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

getValue(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.getValue

setRange(*self*, *dt1*, *dt2*)

getRange(*self*, *dt1*, *dt2*)

setFormat(*self*, *format*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.29.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.29.3 Class Variables

Name	Description
wdgtname	**Value:** 'datepicker'
class_styles	**Value:** {'spin': wx.DP_SPIN, 'dropdown': wx.DP_DROPDOWN, 'default...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** ':*'
defaultvalue	**Value:** ''
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.30 Class GnrWxTextCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxControl
gnr.wx.gnrwx.GnrWxTextCtrl

revised-1

43.30.1 Methods

killFocus(*self*, *event*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.killFocus

setFocus(*self*, *event*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setFocus

attribute_default(*self*, *value*)

setValue(*self*, *value*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setValue

adjustSize(*self*, *event*=`None`)

getValue(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.getValue

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.30.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.30.3 Class Variables

Name	Description
wdgtname	**Value:** 'field'
defaultvalue	**Value:** ''
class_styles	**Value:** {'enterkey': wx.TE_PROCESS_ENTER, 'tabkey': wx.TE_PROCESS...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** ':*'
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.31 Class GnrWxBookCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBookCtrl

43.31.1 Methods

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.31.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.31.3 Class Variables

Name	Description
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.32 Class GnrWxChoicebook

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBookCtrl
gnr.wx.gnrwx.GnrWxChoicebook

43.32.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.32.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.32.3 Class Variables

Name	Description
wdgtname	**Value:** 'choicebook'
wxclass	**Value:** wx.Choicebook
class_styles	**Value:** {'default': wx.CHB_DEFAULT, 'top': wx.CHB_TOP, 'left': wx...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}

continued on next page

Name	Description
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.33 Class GnrWxListbook

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBookCtrl
gnr.wx.gnrwx.GnrWxListbook

43.33.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.33.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.33.3 Class Variables

Name	Description
wdgtname	**Value:** `'listbook'`
wxclass	**Value:** `wx.Listbook`
class_styles	**Value:** `{'default': wx.LB_DEFAULT, 'top': wx.LB_TOP, 'left': wx.L...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** `{}`
class_handlers	**Value:** `{}`
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.34 Class GnrWxNotebook

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBookCtrl
gnr.wx.gnrwx.GnrWxNotebook

43.34.1 Methods

__contains__(*self*, *name*)

__delattr__(*...*)

x.__delattr__('name') <==> del x.name

__getattribute__(*...*)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.34.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.34.3 Class Variables

Name	Description
wdgtname	**Value:** `'notebook'`
wxclass	**Value:** `wx.Notebook`
class_styles	**Value:** `{'default': wx.NB_TOP, 'top': wx.NB_TOP, 'left': wx.NB_LE...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.35 Class GnrWxListCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxListCtrl

43.35.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.35.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.35.3 Class Variables

Name	Description
wdgtname	**Value:** 'lister'
class_styles	**Value:** {'list': wx.LC_LIST, 'report': wx.LC_REPORT, 'virtual': w...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}

continued on next page

Name	Description
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** {`'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.36 Class GnrWxBagListCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBagListCtrl

43.36.1 Methods

onGetItemText(*self*, *item*, *col*)

subscribeDataChanges(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.subscribeDataChanges

onHeadersUpdate(*self*, *obj*, *bag*, *node*, *oldvalue*)

onHeadersInsert(*self*, *obj*, *bag*, *node*, *idx*)

onHeadersDelete(*self*, *obj*, *bag*, *node*, *idx*)

onRowsChanged(*self*, *obj*, *bag*, *node*, *idx*)

rebuildColumns(*self*, *columns*)

setColumn(*self*, *label*, *col*=`None`)

delColumn(*self*, *col*=`None`)

setCache(*self*, *event*)

fillCache(*self*, *cache*, *rows*, *items*)

datasourceSize(*self*, *datasource*)

datasourceHeaders(*self*, *datasource*)

datasourceNumCol(*self*, *datasource*)

setAttrs(*self*)

OnGetItemImage(*self*, *item*)

OnGetItemAttr(*self*, *item*)

SortItems(*self*, *sorter*=`cmp`)

GetListCtrl(*self*)

GetSortImages(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.36.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.36.3 Class Variables

Name	Description
wdgtname	**Value:** `'baglist'`
class_styles	**Value:** `{'list': wx.LC_LIST, 'report': wx.LC_REPORT, 'icon': wx.L...`
class_events	**Value:** `{'list_begin_drag': wx.EVT_LIST_BEGIN_DRAG, 'list_begin_r...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_handlers	**Value:** `{}`
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.37 Class GnrWxTableBrowser

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBagListCtrl
gnr.wx.gnrwx.GnrWxTableBrowser

43.37.1 Methods

browser(*self*)

onGetItemText(*self*, *item*, *col*)
Overrides: gnr.wx.gnrwx.GnrWxBagListCtrl.onGetItemText

subscribeDataChanges(*self*)
Overrides: gnr.wx.gnrwx.GnrWxBagListCtrl.subscribeDataChanges

onRowcountChanged(*self*, *node*, *oldvalue*)

setCache(*self*, *event*)
Overrides: gnr.wx.gnrwx.GnrWxBagListCtrl.setCache

GetListCtrl(*self*)

GetSortImages(*self*)

OnGetItemAttr(*self*, *item*)

OnGetItemImage(*self*, *item*)

SortItems(*self*, *sorter*=`cmp`)

__contains__(*self*, *name*)

__delattr__(*...*)

x.__delattr__('name') <==> del x.name

__getattribute__(*...*)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

datasourceHeaders(*self*, *datasource*)

datasourceNumCol(*self*, *datasource*)

datasourceSize(*self*, *datasource*)

delColumn(*self*, *col*=`None`)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fillCache(*self*, *cache*, *rows*, *items*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onHeadersDelete(*self*, *obj*, *bag*, *node*, *idx*)

onHeadersInsert(*self*, *obj*, *bag*, *node*, *idx*)

onHeadersUpdate(*self*, *obj*, *bag*, *node*, *oldvalue*)

onMouse(*self*, *evt*)

onRowsChanged(*self*, *obj*, *bag*, *node*, *idx*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

rebuildColumns(*self*, *columns*)

root(*self*)

rootname(*self*)

setAttrs(*self*)

setAuiInfo(*self*, *aui*)

setColumn(*self*, *label*, *col*=`None`)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.37.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.37.3 Class Variables

Name	Description
wdgtname	**Value:** `'tablebrowser'`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`

continued on next page

Name	Description
class_events	**Value:** {'list_begin_drag': wx.EVT_LIST_BEGIN_DRAG, 'list_begin_r...
class_handlers	**Value:** {}
class_styles	**Value:** {'list': wx.LC_LIST, 'report': wx.LC_REPORT, 'icon': wx.L...
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.38 Class GnrWxTreeListCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTreeListCtrl

43.38.1 Methods

addChild(*self*, *parent*, *label*, *datasource*=None, *attributes*=None)

onExpandNode(*self*, *event*)

doExpandNode(*self*, *node*, *data*)

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.38.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.38.3 Class Variables

Name	Description
class_styles	**Value:** {'edit_labels': wx.TR_EDIT_LABELS, 'no_buttons': wx.TR_NO...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.39 Class GnrWxTreeCtrl

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTreeCtrl

43.39.1 Methods

addChild(*self*, *parent*, *label*, *datasource*=`None`)

OnExpandNode(*self*, *event*)

doExpand(*self*, *node*)

rebuild(*self*, *item*=`None`)

selectedItem(*self*)

itemLabelAndData(*self*, *item*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.39.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.39.3 Class Variables

Name	Description
wdgtname	**Value:** `'tree'`
class_styles	**Value:** `{'edit_labels': wx.TR_EDIT_LABELS, 'no_buttons': wx.TR_NO...`
class_events	**Value:** `{'tree_begin_drag': wx.EVT_TREE_BEGIN_DRAG, 'tree_begin_r...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_handlers	**Value:** `{}`
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.40 Class GnrWxBagTree

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxBagTree

43.40.1 Methods

setTreeStyle(*self*, *styles*)

isExpandable(*self*, *datanode*)

getNodeViewer(*self*, *datanode*)

linkedItem(*self*, *node*)

unlinkItem(*self*, *node*)

linkItem(*self*, *item*, *node*)

onDatasourceUpdate(*self*, *node*=`None`, ***kwargs*)

onDatasourceInsert(*self*, *node*=`None`, *ind*=`0`, ***kwargs*)

onDatasourceDelete(*self*, *node*=`None`, ***kwargs*)

getBagNode(*self*, *item*=`None`)

setDragCodes(*self*, *codes*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setDragCodes

setDropCodes(*self*, *codes*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setDropCodes

createItemBitmap(*self*, *item*)

showItem(*self*, *node*)

getNodeTreePath(*self*, *node*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.40.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.40.3 Class Variables

Name	Description
wdgtname	**Value:** 'bagtree'
class_styles	**Value:** {'autocheck': ct.TR_AUTO_CHECK_CHILD, 'autotoggle': ct.TR...
class_events	**Value:** {'tree_begin_drag': ct.EVT_TREE_BEGIN_DRAG, 'tree_begin_r...
ctrl_types	**Value:** {'no': 0, 'cb': 1, 'rb': 2}
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...

continued on next page

Name	Description
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** {`'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.41 Class GnrWxBoxSizer

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSizer
gnr.wx.gnrwx.GnrWxBoxSizer

43.41.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

addSpacer(*self*, *spacer*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attachToParent(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calcBorder(*self*, *obj*)

calcFlag(*self*, *obj*, *borderwhere*)

calcFlagInner(*self*, *expand*, *align*, *borderwhere*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getOrient(*self*)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *obj*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

orient(*self*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.41.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.41.3 Class Variables

Name	Description
wdgtname	**Value:** 'boxsizer'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None

continued on next page

Name	Description
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.42 Class GnrWxStaticBoxSizer

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSizer
gnr.wx.gnrwx.GnrWxStaticBoxSizer

43.42.1 Methods

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

addSpacer(*self*, *spacer*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attachToParent(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calcBorder(*self*, *obj*)

calcFlag(*self*, *obj*, *borderwhere*)

calcFlagInner(*self*, *expand*, *align*, *borderwhere*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getOrient(*self*)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *obj*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

orient(*self*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.42.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.42.3 Class Variables

Name	Description
wdgtname	**Value:** 'staticboxsizer'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.43 Class GnrWxGridSizer

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSizer
gnr.wx.gnrwx.GnrWxGridSizer

43.43.1 Methods

__contains__(*self*, *name*)

__delattr__(*...*)

x.__delattr__('name') <==> del x.name

__getattribute__(*...*)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

addSpacer(*self*, *spacer*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attachToParent(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calcBorder(*self*, *obj*)

calcFlag(*self*, *obj*, *borderwhere*)

calcFlagInner(*self*, *expand*, *align*, *borderwhere*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getOrient(*self*)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *obj*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

orient(*self*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.43.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.43.3 Class Variables

Name	Description
wdgtname	**Value:** 'gridsizer'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.44 Class GnrWxFlexGridSizer

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSizer
gnr.wx.gnrwx.GnrWxFlexGridSizer

43.44.1 Methods

__contains__(*self*, *name*)

__delattr__(*...*)

x.__delattr__('name') <==> del x.name

__getattribute__(*...*)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

addSpacer(*self*, *spacer*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attachToParent(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calcBorder(*self*, *obj*)

calcFlag(*self*, *obj*, *borderwhere*)

calcFlagInner(*self*, *expand*, *align*, *borderwhere*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getOrient(*self*)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *obj*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

orient(*self*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.44.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.44.3 Class Variables

Name	Description
wdgtname	**Value:** 'flexsizer'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.45 Class GnrWxGridBagSizer

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxSizer
gnr.wx.gnrwx.GnrWxGridBagSizer

43.45.1 Methods

newChild(*self*, *obj*)
Overrides: gnr.wx.gnrwx.GnrWxSizer.newChild

getLbl(*self*, *obj*)

findPosition(*self*, *at*, *span*)

place(*self*, *obj*, *at*, *span*, *flag*, *border*, *userData*=`None`, *same*=`False`, *rowheight*=`None`)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

addSpacer(*self*, *spacer*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attachToParent(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calcBorder(*self*, *obj*)

calcFlag(*self*, *obj*, *borderwhere*)

calcFlagInner(*self*, *expand*, *align*, *borderwhere*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getOrient(*self*)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

orient(*self*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.45.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.45.3 Class Variables

Name	Description
wdgtname	**Value:** 'bagsizer'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.46 Class GnrWxTopWindow

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTopWindow

43.46.1 Methods

gui(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.46.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.46.3 Class Variables

Name	Description
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.47 Class GnrWxDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTopWindow
gnr.wx.gnrwx.GnrWxDialog

43.47.1 Methods

afterChildrenCreation(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.afterChildrenCreation

extraStyles(*self*, *s*)

onClose(*self*, *evt*)

show(*self*, *modal*=`'Y'`, *keepAlive*=`False`)

loadFields(*self*, *wxobj*)

getFields_(*self*, *node*)

setFields_(*self*, *node*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

gui(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.47.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

continued on next page

Name	Description

43.47.3 Class Variables

Name	Description
wdgtname	**Value:** `'dialog'`
class_styles	**Value:** {`'default': wx.DEFAULT_DIALOG_STYLE, 'title': wx.CAPTION,...`
attribute_types	**Value:** {`'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** {`'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** {`'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** {`'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** {`'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.48 Class GnrWxFrame

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTopWindow
gnr.wx.gnrwx.GnrWxFrame

43.48.1 Methods

setAuiInfo(*self*, *child*, *aui*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.setAuiInfo

afterChildrenCreation(*self*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.afterChildrenCreation

setCustomMenubar(*self*)

extraStyles(*self*, *s*)

onClose(*self*, *evt*)

newChild(*self*, *child*)
Overrides: gnr.wx.gnrwx.GnrWxWidget.newChild

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

gui(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.48.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.48.3 Class Variables

Name	Description
wdgtname	**Value:** `'frame'`
class_styles	**Value:** {`'default': wx.DEFAULT_FRAME_STYLE, 'title': wx.CAPTION, ...`
attribute_types	**Value:** {`'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** {`'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** {`'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** {`'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** {`'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.49 Class GnrWxDirDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxDirDialog

43.49.1 Methods

show(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.49.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.49.3 Class Variables

Name	Description
wdgtname	**Value:** 'dirchooser'
class_styles	**Value:** {'default': wx.DD_DEFAULT_STYLE, 'newdir': wx.DD_NEW_DIR_...
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)

continued on next page

Name	Description
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.50 Class GnrWxFileDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxFileDialog

43.50.1 Methods

show(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.50.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.50.3 Class Variables

Name	Description
wdgtname	**Value:** `'filechooser'`
class_styles	**Value:** `{'open': wx.OPEN, 'save': wx.SAVE, 'confirm_overwrite': w...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.51 Class GnrWxMessageDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxMessageDialog

43.51.1 Methods

show(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.51.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.51.3 Class Variables

Name	Description
wdgtname	**Value:** `'message'`
class_styles	**Value:** `{'ok': wx.OK, 'cancel': wx.CANCEL, 'yes_no': wx.YES_NO, '...`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`

continued on next page

Name	Description
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.52 Class GnrWxTextEntryDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxTextEntryDialog

43.52.1 Methods

show(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=None, *static*=False)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.52.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.52.3 Class Variables

Name	Description
wdgtname	**Value:** 'request'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** None
defaultvalue	**Value:** None
dragformats	**Value:** 'text,unicode,filename,bitmap'
parent	**Value:** property(_get_parent, _set_parent)
standard_wxid	**Value:** {'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....
structnode	**Value:** property(_get_structnode, _set_structnode)

43.53 Class GnrWxFontDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxFontDialog

43.53.1 Methods

show(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*=`'base'`, *module*=`None`)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=`1000`)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.53.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

43.53.3 Class Variables

Name	Description
wdgtname	**Value:** `'fontchooser'`
attribute_types	**Value:** `{'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...`
base_events	**Value:** `{'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...`
base_handlers	**Value:** `{'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...`
base_styles	**Value:** `{'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...`
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

43.54 Class GnrWxColorDialog

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrstructures.GnrStructObj
gnr.wx.gnrwx.GnrWxObject
gnr.wx.gnrwx.GnrWxWidget
gnr.wx.gnrwx.GnrWxColorDialog

43.54.1 Methods

show(*self*)

__contains__(*self*, *name*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *static*=`False`)

__hash__(*x*)

hash(x)

__init__(*self*, *tag*=`None`, *structnode*=`None`, *parent*=`None`, *name*=`None`, *attrs*=`None`, *children*=`None`, *objclassdict*=`None`, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

addAfterShowCall(*self*, *action*)

afterChildrenCreation(*self*)
Overrides: gnr.core.gnrstructures.GnrStructObj.afterChildrenCreation

application(*self*)

asBag(*self*)

attribute_int(*self*, *v*)

attribute_pos(*self*, *v*)

attribute_size(*self*, *v*)

attribute_wxid(*self*, *v*)

bindEvent(*self*, *evt*, *handlername*, *handlerdefault*)

buildChild(*self*, *childnode*, ***kwargs*)

buildChildren(*self*, *children*)

calculateStyle(*self*, *style*=`None`, *default*=`'default'`)

convertedAttribute(*self*, *attr*, *default*=`None`)

createBitmap(*self*)

data(*self*)

dataToDrag(*self*)

deleteChild(*self*, *name*)

deleteChildren(*self*)

doAfterShowCalls(*self*)

dynAttrCalls(*self*)

fullname(*self*)

get(*self*, *name*, *default*=`None`)

getAttribute(*self*, *attr*=`None`, *default*=`None`)

getById(*self*, *id*)

getDataNode(*self*, *source*=`None`)

getDatasource(*self*, *datasourcename*=`'datasource'`, *dflt*=`None`)

getDynAttributes(*self*)

getEventWidget(*self*, *event*)

getFromDataObject(*self*, *dataObject*)

getHandler(*self*, *hname*, *dflt*=`None`, *module*=`None`)

getItem(*self*, *path*, *default*=`None`, *static*=`False`)

getResolver(*self*, *name*, *default*=`None`)

getTag(*self*)

getValue(*self*)

init(*self*, *__children*=`None`, ***kwargs*)
Overrides: gnr.core.gnrstructures.GnrStructObj.init

items(*self*)

keys(*self*)

killFocus(*self*, *event*)

loadValue(*self*, *dflt*=`''`)

makeRoot(*cls*, *parent*, *structnode*, *objclassdict*, ***kwargs*)

This class method instatiates the first element (root)

Parameters

`cls:`	
`parent:`	@param structnode
`objclassdict:`	dictionary of the classes
`kwargs:`	return

metadata(*self*)

mixin(*self*, *cls*, ***kwargs*)

module(*self*)

moreSettings(*self*, *obj*=`None`, *attributes*=`None`)

move(*self*, *pos*=`None`)

newChild(*self*, *child*)
Overrides: gnr.core.gnrstructures.GnrStructObj.newChild

onDatanodeUpdate(*self*, *node*, *oldvalue*)

onDelete(*self*)

onDrop(*self*, *result*)

onDropFiles(*self*, *obj*, *paths*)

onMouse(*self*, *evt*)

parentdatanode(*self*)

parentframe(*self*)

parentwindow(*self*)

popUpOpen(*self*, *evt*)

popUpSelected(*self*, *event*)

root(*self*)

rootname(*self*)

setAuiInfo(*self*, *aui*)

setDragCodes(*self*, *info*)

setDropCodes(*self*, *dropInfo*)

setDropFile(*self*, *dropInfo*)

setDropFile_(*self*, *pars*)

setDropTarget(*self*, *format*, *dropinfo*)

setFocus(*self*, *event*)

setFont(*self*, *font*, *own*=`False`)

setFromDatasource(*self*, *node*=`None`)

setGnrEvents(*self*, *events*)

setOwnFont(*self*, *font*)

setPopUpMenu(*self*, *lines*, *mode*='base', *module*=None)

setStyles(*self*, *currstyle*, *styles*)

setValue(*self*, *value*)

subscribeDataChanges(*self*)

timerOn(*self*)

timerStart(*self*, *value*=1000)

timerStop(*self*)

values(*self*)

window(*self*)

Return wx.Window corresponding to the current GnrWxObject

windowAfter(*self*)

43.54.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

43.54.3 Class Variables

Name	Description
wdgtname	**Value:** 'colorpicker'
attribute_types	**Value:** {'minsize': 'size', 'maxsize': 'size', 'rows': 'int', 'co...
base_events	**Value:** {'activate': wx.EVT_ACTIVATE, 'activate_app': wx.EVT_ACTI...
base_handlers	**Value:** {'minsize': 'SetMinSize', 'maxsize': 'SetMaxSize', 'font'...
base_styles	**Value:** {'simpleborder': wx.SIMPLE_BORDER, 'doubleborder': wx.DOU...
class_events	**Value:** {}
class_handlers	**Value:** {}
class_styles	**Value:** {}

continued on next page

Name	Description
datasourcedefault	**Value:** `None`
defaultvalue	**Value:** `None`
dragformats	**Value:** `'text,unicode,filename,bitmap'`
parent	**Value:** `property(_get_parent, _set_parent)`
standard_wxid	**Value:** `{'OK': wx.ID_OK, 'CANCEL': wx.ID_CANCEL, 'SEPARATOR': wx....`
structnode	**Value:** `property(_get_structnode, _set_structnode)`

44 Module gnr.wx.gnrwxapp

xxxxxxxx

44.1 Class GnrWidget

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.core.gnrstructures.GnrStructData
gnr.wx.gnrwxapp.GnrWidget

44.1.1 Methods

wxobjdict(*self*)

getWxobj(*self*, *name*)

This method returns the GnrWxObject with the given name

frame(*self*, *name*=`'*_#'`, *start*=`'Y'`, *datasource*=`':*'`, ***kwargs*)

Set a frame into the current module

Parameters

`name:` frame's name
`start:` if Y the frame is built during the module startup phase

dialog(*self*, *name*=`None`, *ok*=`None`, *cancel*=`None`, ***kwargs*)

This method set a dialog into the current module

Parameters

`name:` dialog's name
`ok:` ok button
`cancel:` cancel button

panel(*self*, *name*=`None`, ***kwargs*)

This method puts a new panel into the current GnrWidget.

Parameters

`name:` panel's name
`kwargs:` other configuration parameters

button(*self*, *name*, *label*=`None`, *default*=`'N'`, *action*=`None`, ***kwargs*)

This method puts a new button into the current GnrWidget.

Parameters

`name:` button's name
`label:` displayed label
`default:`
`kwargs:` other configuration parameters

checkbox(*self*, *name*, *dflt*=`None`, *label*=`None`, ***kwargs*)

This method puts a new checkbox into the current GnrWidget.

Parameters

`name:` checkbox name
`label:` displayed label
`dflt:`
`kwargs:` other configuration parameters

checklistbox(*self*, *name*, *choices*=`''`, *dflt*=`None`, *lbl*=`None`, *validator*=`None`, ***kwargs*)

This method puts a new checklistbox into the current GnrWidget.

Parameters

`name:` checkbox name
`choices:` selectable options
`lbl:` displayed label
`dflt:`
`kwargs:` other configuration parameters

datepicker(*self*, *name*, *lbl*=`None`, ***kwargs*)

revised-1

field(*self*, *name*, *lbl*=`None`, *datatype*=`None`, *len*=`'20'`, *rows*=`'1'`, *datasource*=`None`, *value*=`None`, ***kwargs*)

This method puts a new text field into the current GnrWidget.

Parameters

`name:` field's name
`label:` displayed label
`dflt:`
`kwargs:` other configuration parameters revised-1

choice(*self*, *name*, *choices*=`''`, *default*=`None`, *lbl*=`None`, *validator*=`None`, ***kwargs*)

This method puts a new choice widget into the current GnrWidget.

Parameters

`name:` choice name
`choices:`
`lbl:`
`dflt:`
`validator:`
`kwargs:` other configuration parameters revised-1

listbox(*self*, *name*, *choices*=`''`, *dflt*=`None`, *lbl*=`None`, *validator*=`None`, ***kwargs*)

This method puts a new listbox into the current GnrWidget.

Parameters

`name:` checkbox name
`lbl:` displayed label
`dflt:`
`kwargs:` other configuration parameters

statictext(*self*, *name*, *value*=`None`, *default*=`''`, ***kwargs*)

This method puts a new static text into the current GnrWidget.

Parameters

`name:` text name
`value:` text value
`kwargs:` other configuration parameters

richtext(*self*, *name*, *value*=`None`, *default*=`''`, ***kwargs*)

This method puts a new rich text into the current GnrWidget.

Parameters

`name:` text name
`value:` text value
`kwargs:` other configuration parameters

notebook(*self*, *name*, *label*=`None`, ***kwargs*)

This method puts a new notebook widget into the current GnrWidget.

Parameters

`name:` notebook's name
`label:` displayed label
`kwargs:` other configuration parameters

listbook(*self*, *name*, *label*=`None`, ***kwargs*)

This method puts a new listbook widget into the current GnrWidget.

Parameters

`name:` listbook's name
`label:` displayed label
`kwargs:` other configuration parameters

choicebook(*self*, *name*, *label*=`None`, ***kwargs*)

This method puts a new choicebook widget into the current GnrWidget.

Parameters

`name:` choicebook's name
`label:` displayed label
`kwargs:` other configuration parameters

tree(*self*, *name*, *datasource*=`None`, ***kwargs*)

This method puts a new tree widget into the current GnrWidget.

Parameters

`name:` tree's name
`datasource:` object that contains tree's data
`kwargs:` other configuration parameters

bagtree(*self*, *name*, *datasource*=`None`, ***kwargs*)

This method puts a new tree widget into the current GnrWidget.

Parameters

`name:` tree's name
`datasource:` object that contains tree's data
`kwargs:` other configuration parameters

menubar(*self*, *datasource*=`None`, ***kwargs*)

menu(*self*, *name*=`None`, *label*=`None`, *title*=`None`, ***kwargs*)

This method puts a new menu widget into the current GnrWidget.

Parameters

`name:` menu name
`label:` displayed label @param title
`kwargs:` other configuration parameters

menuline(*self*, *name*=`None`, *label*=`None`, *_sub*=`False`, ***kwargs*)

This method puts a new menuline into the current GnrWidget.

Parameters

`name:` menuline name
`label:` displayed label
`_sub:` boolean, it's true if the line is a submenu
`kwargs:` other configuration parameters

lister(*self*, *name*, *datasource*=`None`, ***kwargs*)

This method puts a new lister into the current GnrWidget.

Parameters

`name:` lister name @param datoasource
`kwargs:` other configuration parameters

baglist(*self*, *name*, *datasource*=`None`, ***kwargs*)

This method puts a new baglist into the current GnrWidget.

Parameters

`name:` lister name @param datoasource
`kwargs:` other configuration parameters

tablebrowser(*self*, *name*, ***kwargs*)

This method puts a new tablebrowser into the current GnrWidget.

gridsizer(*self*, *name*=`'*'`, *rows*=`0`, *cols*=`0`, *vgap*=`0`, *hgap*=`0`, ***kwargs*)

This method puts a new gridSizer into the current GnrWidget.

Parameters

`name:` gridsizer name
`rows:` number of rows
`cols:` number of cols
`vgap:` vertical gap
`hgap:` horizontal gap
`kwargs:` other configuration parameters

flexsizer(*self*, *name*=`'*'`, *rows*=`0`, *cols*=`0`, *vgap*=`0`, *hgap*=`0`, ***kwargs*)

This method puts a new flexSizer into the current GnrWidget.

Parameters

`name:` flexsizer name
`rows:` number of rows
`cols:` number of cols
`vgap:` vertical gap
`hgap:` horizontal gap
`kwargs:` other configuration parameters

bagsizer(*self*, *name*=`'*'`, *hgap*=`0`, *vgap*=`0`, *cols*=`0`, ***kwargs*)

This method puts a new bagSizer into the current GnrWidget.

Parameters

`name:` bagsizer name
`cols:` number of cols
`vgap:` vertical gap
`hgap:` horizontal gap
`kwargs:` other configuration parameters

boxsizer(*self*, *name*=`None`, *orient*=`'H'`, *label*=`None`, ***kwargs*)

This method puts a new boxSizer into the current GnrWidget.

Parameters

`name:` boxSizer name
`orient:` sizer's orientation
`kwargs:` other configuration parameters

hsizer(*self*, *name*=`None`, *label*=`None`, ***kwargs*)

This method puts a horizontal boxSizer into the current GnrWidget.

Parameters

`name:` sizer name @param label
`kwargs:` other configuration parameters

vsizer(*self*, *name*=`None`, *label*=`None`, ***kwargs*)

This method puts a vertical boxSizer into the current GnrWidget.

Parameters

`name:` sizer name @param label
`kwargs:` other configuration parameters

multisplitter(*self*, *name*=`None`, *orient*=`'H'`, ***kwargs*)

This method puts a multisplitter into the current GnrWidget.

Parameters

`name:` multisplitter's name
`orient:` splitter's orienation
`kwargs:` other configuration parameters

splitter(*self*, *name*=`'*'`, *orient*=`'H'`, *ratio*=`'50'`, ***kwargs*)

This method puts a splitter into the current GnrWidget.

Parameters

`name:` splitter's name
`orient:` splitter's orienation
`ratio:` size percentage
`kwargs:` other configuration parameters

sashwindow(*self*, *name*=`'*'`, ***kwargs*)

toolbar(*self*, *name*=`'*'`, ***kwargs*)

styledtext(*self*, *name*, *datasource*=`None`, ***kwargs*)

mediacontrol(*self*, *name*=`'*_#'`, *filepath*=`''`, *datasource*=`None`, ***kwargs*)

pythoneditor(*self*, *name*=`'*'`, *datasource*=`None`, ***kwargs*)

pycrust(*self*, *name*=`'*'`, *datasource*=`None`, ***kwargs*)

htmlwindow(*self*, *name*=`'*'`, *datasource*=`None`, ***kwargs*)

bagInspector(*self*, *name*=`'datatree'`, *datasource*=`'^'`, *label*=`'Data Tree'`, *aui*=`'left,pos=99,caption=Data Tree Panel,closebtn=n'`, ***kwargs*)

createWidgets(*self*, *widgets*)

This method puts several widgets into the current GnrWidget

Parameters

`widgets:` string that contains the widgets as couples name:type separated by ',' if a widget couple starts with '/' it will be set in another row.

_(*self*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path

`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__init__(*self*, *source*=`None`)

A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:`	an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:`	specifies if a node with an existing path overwrite the value or append it.
`_validators:`	an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:`	all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
`_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:` path of the node.
`validator:` the type of validation.
`parameterString:` string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value

a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters

`encoding:` default is 'UTF-8'

Return Value

a formatted representation of the bag contents (ascii)

attributes(*self*)

backref(*self*)

child(*self*, *tag*, *name*='*_#', *content*=None, *_parentTag*=None, ***kwargs*)

This method sets a new item of the type tag into the current structure

Parameters

- `tag:` structure type
- `name:` structure name. Default value is formed by 'tag_position'
- `content:` optional structure content
- `kwargs:` other parameters @return : the new structure if content is none else the parent

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value

a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value

a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters

- `kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

- `kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=None, *attr*=None)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars

`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

load(*self*, *path*)

This method loads the structure from an xml file

Parameters

`path:` path of the file

makePicklable(*self*)

This method make a Bag picklable.

makeRoot(*cls*, *source*=`None`, *protocls*=`None`)

This method builds the root instance for the given class.

Parameters

`cls:` structure class
`source:` filepath of xml file

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.

`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

root(*self*)

save(*self*, *path*)

This method saves the structure as an xml file

Parameters

`path:` destination of the saved file

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.

`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:` the path of the given item.
`item_value:` the value to set.
`_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:` specifies if a node with an existing path overwrite the value or append it.
`_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

`path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function linked to update event.
`insert:` the eventhandler function linked to insert event.
`delete:` the eventhandler function linked to delete event.
`any:` the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback:` the function which is called.

44.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

44.1.3 Class Variables

Name	Description
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

44.2 Class GnrWxGui

object
└ **gnr.wx.gnrwxapp.GnrWxGui**

44.2.1 Methods

init(*self*, *module*, *path*)

loadStruct(*self*, *path*)

This method loads module structure from an XML file @param path

saveStruct(*self*, *path*)

This method saves the module structure in an XML file @param path

setup(*self*)

buildFrames(*self*)

directbuild(*self*, *tag*, *parent*=`None`, ***attrs*)

newFrame(*self*, *frame*, *name*=`None`, ***kwargs*)

This method sets a new frame in the frames dict of the current module

Parameters

`frame:` frame identifier
`name:` frame name
`kwargs:` other parameters

newDialog(*self*, *dialog*, *name*=`None`, *pos*=`'center'`, ***kwargs*)

This method sets a new frame in the frames dict of the current module

Parameters

`frame:` frame identifier
`name:` frame name
`kwargs:` other parameters

showDialog(*self*, *name*)

This method displays a dialog box @param name

request(*self*, *message*=`None`, *default*=`None`, *title*=`None`, *pos*=`'center'`, ***kwargs*)

This method set a request message into the current module

Parameters

`message:` request message @param default
`title:` displayed title
`kwargs:` other parameters

message(*self*, *message*=`''`, *title*=`''`, *style*=`None`, *pos*=`'center'`, ***kwargs*)

This method set a message dialog into the current module

Parameters

`message:` message text
`title:` displayed title
`style:`
`kwargs:` other parameters

alert(*self*, *message*=`None`, *pos*=`'center'`)

This method set an alert dialog into the current module @param message

colorpicker(*self*, *color*=`None`, *pos*=`'center'`, ***kwargs*)

This method set a colorpicker dialog into the current module @param color

Parameters

`kwargs:` other parameters

dirchooser(*self*, *message*=`None`, *directory*=`None`, *pos*=`'center'`, ***kwargs*)

This method set a directory-choice dialog into the current module @param message @param directory

Parameters

`kwargs:` other parameters

fileOpen(*self*, *message*=`'Open'`, *default*=`''`, *directory*=`''`, *wildcard*=`''`, *pos*=`'center'`, ***kwargs*)

fileSave(*self*, *message*=`'Save as'`, *default*=`''`, *directory*=`''`, *wildcard*=`''`, *pos*=`'center'`, ***kwargs*)

filechooser(*self*, *message*=`''`, *default*=`''`, *directory*=`''`, *wildcard*=`''`, *pos*=`'center'`, ***kwargs*)

This method set a file-choice dialog into the current module @param message @param directory @param default @param wildcard

Parameters

`kwargs:` other parameters

fontchooser(*self*, *srcfont*=`''`, *pos*=`'center'`, ***kwargs*)

This method set a font-choice dialog into the current module @param font

Parameters

`kwargs:` other parameters

makeBitMap(*self*, *textdata*, *size*=`None`)

startBusy(*self*)

endBusy(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

44.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

44.3 Class GnrModule

object
gnr.core.gnrlang.GnrObject
gnr.wx.gnrwxapp.GnrModule

A GnrModule is a GnrObjects that handles the GUI of an application -modulename is the module identifier

44.3.1 Methods

__init__(*self*, *application*, *name*, *path*=`None`, *startframes*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

data(*self*)

set_activeFrame(*self*, *frame*)

get_activeFrame(*self*)

init(*self*)

configure(*self*)

This methods configures the all the GnrWidgets inside the module

newWidget(*self*)

onModuleStarting(*self*)

run(*self*)

This method executes some startup actions: -calls the method start -set the sturtupitems into widgets attribute -for each startup item calls its handler function

stop(*self*)

getSelectedNode(*self*, *id*=`None`)

getById(*self*, *source*, *id*)

getWidget(*self*, *id*=`None`, *source*=`None`)

getDataNode(*self*, *source*, *id*=`None`)

getFrameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

frameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getFrameData(*self*, *source*=`None`)

frameData(*self*, *source*=`None`)

on_menu_selected(*self*, ***kwargs*)

on_popup_selected(*self*, *event*)

widgets(*self*)

onFrameStarting(*self*, *name*, *data*)

onFrameStarted(*self*, *name*)

setFrameData(*self*, *data*, *name*=`None`)

saveToXml(*self*, *param*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

mixin(*self*, *cls*, ***kwargs*)

44.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

44.3.3 Class Variables

Name	Description
modulename	**Value:** `'unnamed module'`
activeFrame	**Value:** `property(get_activeFrame, set_activeFrame)`
application	**Value:** `property(_get_application, _set_application)`

44.4 Class GnrBaseModule

object

gnr.core.gnrlang.GnrObject

gnr.wx.gnrwxapp.GnrModule

gnr.wx.gnrwxapp.GnrBaseModule

44.4.1 Methods

init(*self*)
Overrides: gnr.wx.gnrwxapp.GnrModule.init

configure(*self*)
This methods configures the all the GnrWidgets inside the module

Overrides: gnr.wx.gnrwxapp.GnrModule.configure extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *application*, *name*, *path*=`None`, *startframes*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

data(*self*)

frameData(*self*, *source*=`None`)

frameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getById(*self*, *source*, *id*)

getDataNode(*self*, *source*, *id*=`None`)

getFrameData(*self*, *source*=`None`)

getFrameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters

`obj:` event

getSelectedNode(*self*, *id*=`None`)

getWidget(*self*, *id*=`None`, *source*=`None`)

get_activeFrame(*self*)

mixin(*self*, *cls*, ***kwargs*)

newWidget(*self*)

onFrameStarted(*self*, *name*)

onFrameStarting(*self*, *name*, *data*)

onModuleStarting(*self*)

on_menu_selected(*self*, ***kwargs*)

on_popup_selected(*self*, *event*)

run(*self*)

This method executes some startup actions: -calls the method start -set the sturtupitems into widgets attribute -for each startup item calls its handler function

saveToXml(*self*, *param*)

setFrameData(*self*, *data*, *name*=`None`)

set_activeFrame(*self*, *frame*)

stop(*self*)

widgets(*self*)

44.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

44.4.3 Class Variables

Name	Description
modulename	**Value:** `'basemodule'`
activeFrame	**Value:** `property(get_activeFrame, set_activeFrame)`
application	**Value:** `property(_get_application, _set_application)`

44.5 Class GnrApplication

gnr.wx.App
└
gnr.wx.gnrwxapp.GnrApplication

A GnrApplication is composed by one or more GnrModules...

44.5.1 Methods

__init__(*self*, *modules*=`None`, *frames*=`None`, *userlevel*=`'user'`, *log*=`None`, ***kwargs*)

run(*self*)

This method starts the application

OnInit(*self*)

this is the implementation of the wxapp method OnInit. It is called from MainLoop and handles some setup functions before making run all loaded modules. When all modules are started returs true

OnExit(*self*)

onApplicationStarting(*self*)

onApplicationStopping(*self*)

onApplicationStarted(*self*)

onApplicationStopped(*self*)

loadModule(*self*, *modulename*, *name*=`None`, *path*=`None`, *startframes*=`None`)

This method loads the module that has name modulename, or modulename itself if the argument is instance of GnrModule

getModules(*self*, *onlyPublic*=`True`)

This method collects all the modules that must be loaded into the application

saveModules(*self*)

Utility to save application's modules to Xml

runModule(*self*, *event*, *module*)

startDrag(*self*, *obj*, *codes*, *item*=`None`)

endDrag(*self*)

dragged(*self*)

log(*self*, *cls*, *level*, *msg*)

44.5.2 Class Variables

Name	Description
devmenu	**Value:** `['Export to xml: saveToXml', 'PyDoc:popup_pydoc', 'Browse...`
activeModule	**Value:** `property(_get_activeModule, _set_activeModule)`

45 Module gnr.wx.moduletest

Esempio di applicazione GenroWx

45.1 Class TestModule

object
gnr.wx.gnrwidgets.GnrModule
gnr.wx.moduletest.TestModule

45.1.1 Methods

configure(*self*)
Overrides: gnr.wx.gnrwidgets.GnrModule.configure

conf_menu(*self*, *frame*)

conf_LeftColumn(*self*, *column*)

conf_Fields(*self*, *pages*)

conf_List(*self*, *pages*)

conf_Tree(*self*, *pages*)

conf_Multisplitters(*self*, *pages*)

conf_Test(*self*, *pages*)

conf_Styledtext(*self*, *pages*)

conf_Pythoneditor(*self*, *pages*)

conf_PyCrust(*self*, *pages*)

closeFrame(*self*, *evt*)

popup_stc(*self*, *label*)

popup_beta(*self*, *label*)

fontsel(*self*, *x*)

start(*self*)
Overrides: gnr.wx.gnrwidgets.GnrModule.start

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *application*, *name*, *autostart*=`False`, *path*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

alert(*self*, *message*=`None`, *invokedBy*=`None`)

build(*self*)

buildDialog(*self*, *name*, *invokedBy*=`None`)

buildFrame(*self*, *name*, *invokedBy*=`None`)

child(*self*, *wdgtype*, *name*, ***kwargs*)

colorpicker(*self*, *color*=`None`, *invokedBy*=`None`, ***kwargs*)

dialog(*self*, *name*, *ok*=`None`, *cancel*=`None`, ***kwargs*)

dirchooser(*self*, *message*=`None`, *directory*=`None`, *invokedBy*=`None`, ***kwargs*)

filechooser(*self*, *message*=`''`, *default*=`None`, *directory*=`None`, *wildcard*=`None`, *invokedBy*=`None`, ***kwargs*)

fontchooser(*self*, *font*=`None`, *invokedBy*=`None`, ***kwargs*)

frame(*self*, *name*, ***kwargs*)

init(*self*)

loadstruct(*self*, *path*)

message(*self*, *message*=`None`, *title*=`None`, *style*=`None`, *invokedBy*=`None`, ***kwargs*)

on_menu_selected(*self*, *event*)

on_popup_selected(*self*, *event*)

request(*self*, *message*=`None`, *default*=`None`, *title*=`None`, *invokedBy*=`None`, ***kwargs*)

run(*self*)

savestruct(*self*, *path*)

showDialog(*self*, *name*, *invokedBy*)

stop(*self*)

wdgdata(*self*)

45.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

45.1.3 Class Variables

Name	Description
modulename	**Value:** `'Test Module'`
autostart	**Value:** `True`
application	**Value:** `property(_get_application, _set_application)`
wxobj	**Value:** `property(_get_wxobj, _set_wxobj)`

46 Module gnr.wx.test_drag

46.1 Functions

runTest(*frame*, *nb*, *log*)

46.2 Variables

Name	Description
ID_CopyBtn	**Value:** `wx.NewId()`
ID_PasteBtn	**Value:** `wx.NewId()`
ID_BitmapBtn	**Value:** `wx.NewId()`
overview	**Value:** `"...`

46.3 Class EventManager

object

gnr.core.gnrlang.GnrObject

gnr.wx.gnrwxapp.GnrModule

gnr.wx.test_drag.EventManager

46.3.1 Methods

configure(*self*)
This methods configures the all the GnrWidgets inside the module

Overrides: gnr.wx.gnrwxapp.GnrModule.configure extit(inherited documentation)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *application*, *name*, *path*=`None`, *startframes*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: gnr.core.gnrlang.GnrObject.__init__

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

data(*self*)

frameData(*self*, *source*=`None`)

frameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters
`obj:` event

getById(*self*, *source*, *id*)

getDataNode(*self*, *source*, *id*=`None`)

getFrameData(*self*, *source*=`None`)

getFrameName(*self*, *obj*=`None`)

This method receives an event object and returns the name of the frame that is target of the event

Parameters
`obj:` event

getSelectedNode(*self*, *id*=`None`)

getWidget(*self*, *id*=`None`, *source*=`None`)

get_activeFrame(*self*)

init(*self*)

mixin(*self*, *cls*, ***kwargs*)

newWidget(*self*)

onFrameStarted(*self*, *name*)

onFrameStarting(*self*, *name*, *data*)

onModuleStarting(*self*)

on_menu_selected(*self*, ***kwargs*)

on_popup_selected(*self*, *event*)

run(*self*)

This method executes some startup actions: -calls the method start -set the sturtupitems into widgets attribute -for each startup item calls its handler function

saveToXml(*self*, *param*)

setFrameData(*self*, *data*, *name*=`None`)

set_activeFrame(*self*, *frame*)

stop(*self*)

widgets(*self*)

46.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

46.3.3 Class Variables

Name	Description
modulename	**Value:** `'Event Manager'`
activeFrame	**Value:** `property(get_activeFrame, set_activeFrame)`

continued on next page

Name	Description
application	**Value:** `property(_get_application, _set_application)`

46.4 Class ClipTextPanel

gnr.wx.Panel
└ **gnr.wx.test_drag.ClipTextPanel**

46.4.1 Methods

__init__(*self*, *parent*, *log*)

OnCopy(*self*, *evt*)

OnPaste(*self*, *evt*)

OnCopyBitmap(*self*, *evt*)

46.5 Class OtherDropTarget

gnr.wx.PyDropTarget
└ **gnr.wx.test_drag.OtherDropTarget**

46.5.1 Methods

__init__(*self*, *window*, *log*)

OnEnter(*self*, *x*, *y*, *d*)

OnLeave(*self*)

OnDrop(*self*, *x*, *y*)

OnData(*self*, *x*, *y*, *d*)

46.6 Class MyFileDropTarget

gnr.wx.FileDropTarget
└ **gnr.wx.test_drag.MyFileDropTarget**

46.6.1 Methods

__init__(*self*, *window*, *log*)

OnDropFiles(*self*, *x*, *y*, *filenames*)

46.7 Class MyTextDropTarget

gnr.wx.TextDropTarget
└ **gnr.wx.test_drag.MyTextDropTarget**

46.7.1 Methods

__init__(*self*, *window*, *log*)

OnDropText(*self*, *x*, *y*, *text*)

OnDragOver(*self*, *x*, *y*, *d*)

46.8 Class FileDropPanel

gnr.wx.Panel
└ **gnr.wx.test_drag.FileDropPanel**

46.8.1 Methods

__init__(*self*, *parent*, *log*)

WriteText(*self*, *text*)

SetInsertionPointEnd(*self*)

46.9 Class TestPanel

gnr.wx.Panel
└ **gnr.wx.test_drag.TestPanel**

46.9.1 Methods

__init__(*self*, *parent*, *log*)

47 Module gnr.wx.testresolver

Esempio di applicazione EzWx

47.1 Class GnrWidgetResolver

object
gnr.core.gnrbag.BagResolver
gnr.wx.testresolver.GnrWidgetResolver

47.1.1 Methods

__init__(*self*, *template*)
Overrides: gnr.core.gnrbag.BagResolver.__init__

load(*self*)
must be reimplemented

Overrides: gnr.core.gnrbag.BagResolver.load extit(inherited documentation)

__call__(*self*, ***kwargs*)

__contains__(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *k*)

__hash__(*x*)

hash(x)

__init__(*self*, **args*, ***kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*self*)
str(x)

Overrides: object.__str__ extit(inherited documentation)

digest(*self*, *k*=`None`)

expired(*self*)

getAttributes(*self*)

instanceKwargs(*self*)

items(*self*)

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

reset(*self*)

resolverDescription(*self*)

resolverSerialize(*self*)

setAttributes(*self*, *attributes*)

sum(*self*, *k*=`None`)

values(*self*)

47.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

47.1.3 Class Variables

Name	Description
attributes	**Value:** `property(getAttributes, setAttributes)`
cacheTime	**Value:** `property(_get_cacheTime, _set_cacheTime)`
classArgs	**Value:** `[]`
classKwargs	**Value:** `{'cacheTime': 0, 'readOnly': True}`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`

47.2 Class GnrFormTemplate

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.wx.testresolver.GnrFormTemplate

47.2.1 Methods

__**init**__(*self*, *wdgtype*, ***kwargs*)
A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrbag.Bag.__init__ extit(inherited documentation)

child(*self*, *wdgtype*, *name*, ***kwargs*)

frame(*self*, *name*, ***kwargs*)

dialog(*self*, *name*, *ok*=`None`, *cancel*=`None`, ***kwargs*)

root(*self*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:`	an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:`	specifies if a node with an existing path overwrite the value or append it.
`_validators:`	an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:`	all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
`_position:`	specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:`	path of the node.
`validator:`	the type of validation.
`parameterString:`	string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value
a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters
`encoding:` default is 'UTF-8'

Return Value
a formatted representation of the bag contents (ascii)

backref(*self*)

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value
a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value
a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters
`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars
`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

makePicklable(*self*)

This method make a Bag picklable.

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

`item_path:`	the path of the given item.
`item_value:`	the value to set.
`_attributes:`	an optional parameter, it specified the attributes of the value to set. Default is 'None'.
`_position:`	an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
`_duplicate:`	specifies if a node with an existing path overwrite the value or append it.
`_validators:`	an optional parameter, it specified the validarors of the value to set. Default is 'None'.
`kwargs:`	all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

`path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

`subscriberId:`	an ID can be assigned for a subscription
`update:`	the eventhandler function linked to update event.
`insert:`	the eventhandler function linked to insert event.
`delete:`	the eventhandler function linked to delete event.
`any:`	the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback`: the function which is called.

47.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

47.2.3 Class Variables

Name	Description
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

47.3 Class GnrFormItem

object
gnr.core.gnrlang.GnrObject
gnr.core.gnrbag.Bag
gnr.wx.testresolver.GnrFormItem

47.3.1 Methods

child(*self*, *wdgtype*, *name*, ***kwargs*)

field(*self*, *name*, *len*=`20`, *rows*=`1`, *label*=`None`, *default*=`''`, *ftype*=`'T'`, *labelpos*=`None`, ***kwargs*)

gridsizer(*self*, *name*=`'*'`, *rows*=`0`, *cols*=`0`, *vgap*=`0`, *hgap*=`0`, ***kwargs*)

flexsizer(*self*, *name*=`'*'`, *rows*=`0`, *cols*=`0`, *vgap*=`0`, *hgap*=`0`, ***kwargs*)

bagsizer(*self*, *name*=`'*'`, *hgap*=`0`, *vgap*=`0`, *cols*=`0`, ***kwargs*)

boxsizer(*self*, *name*=`'*'`, *orient*=`'H'`, *label*=`None`, ***kwargs*)

hsizer(*self*, *name*=`'*'`, *label*=`None`, ***kwargs*)

vsizer(*self*, *name*=`'*'`, *label*=`None`, ***kwargs*)

button(*self*, *name*, *label*=`None`, *default*=`'N'`, *action*=`None`, ***kwargs*)

__call__(*self*, *what*=`None`)

__contains__(*self*, *what*)

The "in" operator can be used to test the existence of a key in a bag. Also nested keys are allowed.

Parameters

`what:` the key path to test.

Return Value

a boolean value, True if the key exists in the bag, False otherwise.

__delattr__(...)

x.__delattr__('name') <==> del x.name

__delitem__(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

__eq__(*self*, *other*)

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__getitem__(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

__hash__(*x*)

hash(x)

__init__(*self*, *source*=`None`)

A new bag can be created in various ways:

- parsing a local file, a remote url or a text string (see fromXml)
- converting a dictionary into a Bag
- passing a list or a tuple just like for the builtin dict() command

Overrides: gnr.core.gnrlang.GnrObject.__init__

__iter__(*self*)

__len__(*self*)

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__setitem__(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

__str__(*self*, *exploredNodes*=`None`, *mode*=`'static,weak'`)

This method returns a formatted representation of the bag contents.

Return Value

a formatted representation of the bag contents (unicode)

Overrides: object.__str__

addItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`">"`, *_validators*=`None`, ***kwargs*)

This method adds an item to the current Bag using a path in the form "label1.label2...labelN"; it returns the current bag. If the path already exists, this method replicates the path keeping old values and the new value.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specifies the attributes of the value to set. Default is 'None'.
- `_position:` specifies the position where to add the new item. It can be "<" or ">" followed by "#n" or "label". Default is append after last item.

Return Value

the current bag.

addValidator(*self*, *path*, *validator*, *parameterString*)

This method add a validator into the node at the given path

Parameters

`path:` path of the node.
`validator:` the type of validation.
`parameterString:` string which contains the params for validation.

analyze(*self*, *data*, *group_by*=`None`, *sum*=`None`, *distinct*=`None`, *key*=`None`)

comment analyze

asDict(*self*, *ascii*=`False`, *lower*=`False`)

This method converts a Bag in a Dictionary.

Return Value

a Dictionary equivalent to the given Bag.

asString(*self*, *encoding*=`'UTF-8'`, *mode*=`'weak'`)

This method calls the __str__ method: asString() returns an ascii encoded formatted representation of the bag.

Parameters

`encoding:` default is 'UTF-8'

Return Value

a formatted representation of the bag contents (ascii)

backref(*self*)

clear(*self*)

This method clears the Bag.

clearBackRef(*self*)

This method clear all the setBackRef() assumption.

copy(*self*)

This method returns a copy of the Bag.

Return Value

a copy of the Bag.

deepcopy(*self*)

This method returns a deep copy of the Bag.

Return Value

a deep copy of the Bag.

Deprecated: IT DOESN'T WORK

defineFormula(*self*, ***kwargs*)

Define a formula that uses defined symbols.

Parameters

`kwargs:` a pair of key-value which rapresent the formula and the string that describes it.

defineSymbol(*self*, ***kwargs*)

Define a variable and link it to a value at the specified path. The value linked is a BagFormula Resolver.

Parameters

`kwargs:` a dict of symbol to define for a formula.

delAttr(*self*, *path*=`None`, *attr*=`None`)

delItem(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

delParentRef(*self*)

This method set false the reference to the ParentBag of this Bag.

digest(*self*, *what*=`None`, *condition*=`None`)

Extracts multiple values from a Bag. It can be used with special keys that are applied to all the nodes. Then digest() returns a list as long as the Bag containing the requested values.

Parameters

`what:` this param is a comma separated string of special keys. Special keys are:

- #k: the label of each node
- #v: the value of each node
- #__v: the value of each node in 'static' mode
- #a: the attributes of each node
- #a.attrname: the attribute 'attrname' of each node
- subpath: the value of this subpath of each node this parameter can start with a path before the list of special keys to apply the digest to a subpath of this Bag. Path and special keys are separated by ':'.

`condition:` set a condition for digest process

fillFrom(*self*, *source*)

This method fills a void Bag from: basestring, bag, list.

Parameters

`source:` the source for the Bag.

formula(*self*, *formula*, ***kwargs*)

Sets a BagFormula resolver.

Parameters

`formula:` a string that represents the expression with symbolic vars
`kwargs:` links between symbols and paths associated to their values

fromXml(*self*, *source*, *catalog*=`None`, *bagcls*=`None`, *empty*=`None`)

This method fills the Bag with values read from an XML string or file or URL.

Parameters

`source:` the XML source to be loaded in the Bag.
`catalog:`
`bagcls:` bagcls empty:

fullpath(*self*)

get(*self*, *label*, *default*=`None`, *mode*=`None`)

getAttr(*self*, *path*=`None`, *attr*=`None`, *default*=`None`)

This method get the value of the attribute of the node at the given path

Parameters

`path:` path of the given item.
`_atts:` the label of the attribute to get.

getFormula(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

getIndex(*self*)

This method return the index of the Bag with all the internal address.

getIndexList(*self*, *asText*=`False`)

This method return the index of the Bag as a plan list of the Nodes paths.

getItem(*self*, *path*, *default*=`None`, *mode*=`None`)

This method reimplements the list's __getitem__(). Usually a path is a string formed by the labels of the nested items, joined by the char '.' but several different path notations have been implemented to offer some useful features. If a path segment starts with '#' is followed by a number, it means that for that level, the item will by identified by its index position, as a list element. If a path ends with '.?', function returns the item's keys. If at the last path-level the label contains '#', what follows the '#' is considered the key of an item's attribute and the function will return that attribute's value. If a path starts with '?' the path is interpreted as a digest. A path can also ba a list of keys.

Parameters

`path:` the item's path
`default:` an optional default value, default is 'None'.

Return Value

the value of the given item

```
>>> mybag=Bag()
>>> mybag['aa.bb.cc']=1234
>>> mybag['aa.bb.cc']
1234
```

getNode(*self*, *path*=`None`, *asTuple*=`False`, *autocreate*=`False`, *default*=`None`)

This method returns the BagNode stored at this path.

Parameters

`path:` path of the given item.

getNodeByAttr(*self*, *attr*, *value*, *path*=`None`)

This method returns the first found node which has an attribute named 'attr' equal to 'value'. E.g. searching a node with a given 'id' in a Bag build from html.

Parameters

`attr:` path of the given item.
`value:` path of the given item.
`path:` optional, an empty list that will be filled with the path of the found node.

Return Value

a BagNode with the requested attribute

getNodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

getResolver(*self*, *path*)

This method get the resolver of the node at the given path.

Parameters

`path:` path of the node.

has_key(*self*, *path*)

This method is analog to dictionary's has_key() method.

Parameters

`path:` path of the given item.

Return Value

a boolean value: True if the given item has a key, False otherwise.

items(*self*)

This method returns a list of tuples containing all key,value pairs.

Return Value

a list of tuples containing all key,value pairs of the Bag.

iteritems(*self*)

iterkeys(*self*)

itervalues(*self*)

keys(*self*)

This method returns a list containing all the keys of the Bag.

Return Value

a list containing all the keys of the Bag.

makePicklable(*self*)

This method make a Bag picklable.

merge(*self*, *otherbag*, *upd_values*=`True`, *add_values*=`True`, *upd_attr*=`True`, *add_attr*=`True`)

Create a new Bag by the merging of this Bag and another one.

mixin(*self*, *cls*, ***kwargs*)

nodes(*self*, *condition*=`None`)

Get the actual list of nodes contained in the Bag

pickle(*self*, *destination*=`None`, *bin*=`True`)

This method returns a pickled Bag.

Parameters

`destination:` an optional parameter; it is the destination path; default is 'None'.
`bin:` a boolean optional parameter, if set to 'False' the Bag is pickled in ASCII code, if set to 'True' is pickled in binary format. Default is 'True'.

Return Value

the pickled Bag.

pop(*self*, *path*)

This method is analog to dictionary's pop() method. It pops the given item from the Bag; it returns the given item.

Parameters

`path:` path of the given item.

Return Value

the given item.

popNode(*self*, *path*)

removeValidator(*self*, *path*, *validator*)

This method add a validator into the node at the given path

restoreFromPicklable(*self*)

This method restore a Bag to its original form from its picklable.

setAttr(*self*, *_path*=`None`, *_attributes*=`None`, ***kwargs*)

This method set attributes into the node at the given path

Parameters

`_path:` path of the target item.
`_attributes:` a dict of attributes to set into the node.

setBackRef(*self*, *node*=`None`, *parent*=`None`)

This method set a stricter hypotesis about the structure of a bag. It make it more similar to a tree-leaf model: a Bag can have only one Parent and it knows has a reference to its Parent.

Parameters

`node:` not required
`parent:` not required

setCallBackItem(*self*, *path*, *callback*, ***kwargs*)

setCallable(*self*, *name*, *argstring*=`None`, *func*=`'pass'`)

review

setItem(*self*, *item_path*, *item_value*, *_attributes*=`None`, *_position*=`None`, *_duplicate*=`False`, *_updattr*=`False`, *_validators*=`None`, ***kwargs*)

This method sets an item in the Bag using a path in the form "label1.label2...labelN".It returns the current bag. If the path already exists, it overwrites the value at the given path.

Parameters

- `item_path:` the path of the given item.
- `item_value:` the value to set.
- `_attributes:` an optional parameter, it specified the attributes of the value to set. Default is 'None'.
- `_position:` an optional parameter, if specified the method setItem() behaves like addItem(). Default is 'None'.
- `_duplicate:` specifies if a node with an existing path overwrite the value or append it.
- `_validators:` an optional parameter, it specified the validarors of the value to set. Default is 'None'.
- `kwargs:` all remaining kwargs can be attributes AND/OR validators .

Return Value

the current bag.

setResolver(*self*, *path*, *resolver*)

This method set a resolver into the node at the given path.

Parameters

- `path:` path of the node.

sort(*self*, *pars*=`'#k:a'`)

pars None: label ascending pars "

subscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

This method provides a subscribing of a function to an event. Subscribing an event on a Bag means that every time that it happens, it is propagated along the bag hierarchy and is triggered by its eventhandler. A subscription can be seen as a couple event-function, this means that I can define many eventhandlers for the same event.

Parameters

- `subscriberId:` an ID can be assigned for a subscription
- `update:` the eventhandler function linked to update event.
- `insert:` the eventhandler function linked to insert event.
- `delete:` the eventhandler function linked to delete event.
- `any:` the eventhandler function linked to do whenever something happens.

sum(*self*, *what*=`'#v'`)

toXml(*self*, *filename*=`None`, *encoding*=`'UTF-8'`, *typeattrs*=`True`, *unresolved*=`False`, *autocreate*=`False`)

This method returns a complete standard XML version of the Bag, including the encoding tag <?xml version='1.0' encoding='UTF-8'?> the content of the Bag is hierarchically represented as an XML block sub-element of the node <GenRoBag> (see the toXmlBlock() documentation for more details about type representation). Is also possible to write the result on a file, passing the path of the file as the 'filename' parameter.

Parameters

`filename:` an optional parameter, it is the path of the output file; default value is 'None'
`encoding:` an optional parameter, is used to set the XML encoding; default value is UTF-8.

Return Value

an XML version of the bag.

```
>>> mybag=Bag()
>>> mybag['aa.bb']=4567
>>> mybag.toXml()
'<?xml version='1.0' encoding='iso-8859-15'?>
 <GenRoBag>
 <aa><bb T="L">
 4567</bb></aa></GenRoBag>'
```

unpickle(*self*, *source*)

This method unpickles a pickled Bag.

Parameters

`source:` the source path.

Return Value

the unpickled Bag.

unsubscribe(*self*, *subscriberId*, *update*=`None`, *insert*=`None`, *delete*=`None`, *any*=`None`)

delete a subscription of an event of given subscriberId.

Parameters

`subscriberId:` an ID can be assigned for a subscription
`update:` the eventhandler function to remove
`insert:` the eventhandler function to remove
`delete:` the eventhandler function to remove
`any:` the eventhandler function to remove

update(*self*, *otherbag*)

this method merge a Bag into the current one.

Parameters

`otherbag:` a Bag to merge into.

values(*self*)

This method returns a list containing all values of the Bag.

Return Value

a list containing all the values of the Bag.

walk(*self*, *callback*)

Calls a function for each node of the Bag.

Parameters

`callback`: the function which is called.

47.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

47.3.3 Class Variables

Name	Description
modified	**Value:** `property(_get_modified, _set_modified)`
node	**Value:** `property(_get_node, _set_node)`
parent	**Value:** `property(_get_parent, _set_parent)`
parentNode	**Value:** `property(_get_parentNode, _set_parentNode)`
rootattributes	**Value:** `property(_get_rootattributes, _set_rootattributes)`

48 Package gnr.xtnd

49 Module gnr.xtnd.gnrstats

49.1 Class TotalizeSelection

object
gnr.xtnd.gnrstats.TotalizeSelection

49.1.1 Methods

setParameters(*self*, *db*, *mainTable*, *relationDict*, *columns*, *where*, *queryArgs*, *fields*, *anagfields*, *sortfields*, *grcol*, *colgroups*, *subtotals*, *total_col*=`True`, *nfetch*=`1000`)

calculate(*self*, *prevperiod*=`False`)

getRowCursor(*self*)

rowsTotalize(*self*, *serverfetch*, *prevperiod*=`False`)

rowPreprocess(*self*, *r*)

getOneBlock(*self*, *row*, *grouplist*, *prevperiod*=`False`)

sortKey(*self*, *row*, *grouplist*)

readRow(*self*, *row*, *prevperiod*=`False`)

getAnagFields(*self*, *row*, *totals*)

anag_base(*self*, *row*, *totals*, *fld*)

getRowValues(*self*, *row*, *colnum*, *totals*)

addToGroup(*self*, *values*, *colnum*, *totals*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

49.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

50 Module gnr.xtnd.gnrtimestamp

50.1 Variables

Name	Description
BASE36	**Value:** `'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'`

50.2 Class GnrTimeStamp

object
└ **gnr.xtnd.gnrtimestamp.GnrTimeStamp**

50.2.1 Methods

__init__(*self*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

get(*self*, *station*=1, *base*=1)

encode(*self*, *number*, *nChars*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

50.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

51 Module gnr.xtnd.soap4D

52 Module gnr.xtnd.sync4Dapp

52.1 Class Struct4D

object
└ **gnr.xtnd.sync4Dapp.Struct4D**

52.1.1 Methods

__init__(*self*, *instance_folder*, *packages_folder*=`None`)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

folder4d(*self*)

folder4dstruct(*self*)

areaFolder(*self*, *area*)

modelFolder(*self*, *area*)

fromNameAs(*self*, *name*)

nameConverter(*self*, *table*=`''`, *field*=`''`, *fullname*=`None`, *mode*=`'4D'`)

get4dTables(*self*)

buildNames4d(*self*)

build(*self*, *configBag*)

buildArea(*self*, *sname*, *sbag*)

buildTable(*self*, *pkg*, *tbag*, *mode*=`'4D'`)

buildField(*self*, *table*, *fldbag*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

52.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

52.2 Class GnrAppSync4D

object
gnr.app.gnrapp.GnrApp
gnr.xtnd.sync4Dapp.GnrAppSync4D

52.2.1 Methods

onIniting(*self*)
Overrides: gnr.app.gnrapp.GnrApp.onIniting

onInited(*self*)
Overrides: gnr.app.gnrapp.GnrApp.onInited

folder4d(*self*)

folderdialog4d(*self*)

folder4dDataIn(*self*)

folder4dDataOut(*self*)

folder4dTodayDataOut(*self*)

beforeLoop(*self*)

loop(*self*)

do(*self*)

lookForBackSync(*self*)

syncOutTransaction(*self*, *transaction*)

lookFor4dFiles(*self*)

errorLog(*self*)

importFolder(*self*, *folder*, *fileBag*)

importFile(*self*, *fullname*)

writeTransaction(*self*, *data*, *attr*, *file_name*=`None`)

writeImport(*self*, *b*, *file_name*=`None`)

setSubTriggerSchemata(*self*, *data*)

rebuildRecipe(*self*)

importTable(*self*, *tbl*)

firstImport(*self*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(*self*, *instanceFolder*, *custom_config*=`None`, **kwargs*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

applyChangesToDb(*self*)

auth_py(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

auth_sql(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

auth_xml(*self*, *node*, *username*, *password*=`None`, *authenticate*=`False`)

checkDb(*self*)

checkPassword(*self*, *login_pwd*, *authenticate*=`False`, *defaultTags*=`None`, **kwargs*)

checkResourcePermission(*self*, *pageTags*, *userTags*)

getAvatar(*self*, *username*, *password*=`None`, *authenticate*=`False`)

getResource(*self*, *pkg*, *path*, *locale*=`None`)

guestLogin(*self*)

init(*self*)

makeAvatar(*self*, ***kwargs*)

newUserUrl(*self*)

onWebPageCreation(*self*, *page*)

realPath(*self*, *path*)

52.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

53 Module gnr.xtnd.sync4Dpyro

53.1 Class Sync4DCommander

Pyro.core.ObjBase
└
gnr.xtnd.sync4Dpyro.Sync4DCommander

53.1.1 Methods

__init__(*self*, *daemon*, *instancefolder*)

loopCondition(*self*)

run(*self*)

stop(*self*)

54 Module gnr.xtnd.sync4Dtransaction

54.1 Class TransactionManager4D

object
└ **gnr.xtnd.sync4Dtransaction.TransactionManager4D**

54.1.1 Methods

__init__(*self*, *app*, *pkgid*)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

db(*self*)

writeTransaction(*self*, *mode*, *action*, *maintable*, *data*, *request_id*=`None`, *request_ts*=`None`, *user_id*=`None`, *session_id*=`None`, *user_ip*=`None`, *queue_id*=`None`, *file_name*=`None`)

do_import(*self*, *data*, *tablepath*)

do_sync_trigger(*self*, *data*, *pkg*, *table*, *action*)

do_trigger(*self*, *data*, *tablepath*, *action*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

54.1.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

54.1.3 Class Variables

Name	Description
app	**Value:** `property(_get_app, _set_app)`

55 Module gnr.xtnd.sync4Dutils

55.1 Functions

gnr4dNetBag(*host4D*, *method*, *params*=`None`)

Call a 4D method via 4D WebService Server and GnrNetBag

Parameters

`host4D:` host (and port) of the 4D webserver
`method:` name of the method to invoke on 4D in the form 4dMethod.$1:$2
`params:` a Bag containing all needed params: 4D receive it as $3 (string: name of a GnrViVa BLOB)

55.2 Class Utils4D

object
└ **gnr.xtnd.sync4Dutils.Utils4D**

55.2.1 Methods

bag4dTableToListDict(*self*, *b*)

listDictTobag4dTable(*self*, *listdict*)

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)

Return Value

`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

55.2.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

55.3 Class Pkg4D

object
└
gnr.xtnd.sync4Dutils.Pkg4D

55.3.1 Methods

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

__new__(*T*, *S*, ...)
Return Value
`a new object with type S, a subtype of T`

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

55.3.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

Index