__className

Übersicht

Die Package-Sammlung besteht aus den folgenden Packages. Eine kurze Beschreibung folgt danach.

0.1 Beschreibung

1 Package ontologyFramework.OFDataMapping.primitiveData

1.1 Klassen-Liste

1.2 Package-Beschreibung

1.3 Klasse ontologyFramework.OFDataMapping.primitiveDataMapper.FileDataType

1.3.1 Übersicht

1.3.2 Inhaltsverzeichnis

1.3.3 Konstruktoren

FileDataType

1.3.4 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

1.4 Klasse ontologyFramework.OFDataMapping.primitiveDataMapper.IntegerMappe

1.4.1 Übersicht

1.4.2 Inhaltsverzeichnis

1.4.3 Konstruktoren

IntegerMapper

1.4.4 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

1.5 Klasse ontologyFramework.OFDataMapping.primitiveDataMapper.BooleanDatal

1.5.1 Übersicht

This class implements a mapping between and ontological individual «@literal:I» and a `Boolean`.

@author Buoncomapgni Luca

@version 1.0

1.5.2 Inhaltsverzeichnis

1.5.3 Variablen

BOOLEANMAPPING_dataProperty

1.5.4 Konstruktoren

BooleanDataMapper

1.5.5 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

1.6 Klasse ontologyFramework.OFDataMapping.primitiveDataMapper.StringDataMa

1.6.1 Übersicht

1.6.2 Inhaltsverzeichnis

1.6.3 Konstruktoren

StringDataMapper

1.6.4 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

1.7 Klasse ontologyFramework.OFDataMapping.primitiveDataMapper.FloatDataMa

1.7.1 Übersicht

This class implements a mapping between and ontological individual «@literal:I» and a `Boolean`.

@author Buoncomapgni Luca

@version 1.0

1.7.2 Inhaltsverzeichnis

1.7.3 Variablen

DoubleMAPPING_dataProperty

1.7.4 Konstruktoren

FloatDataMapper

1.7.5 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

1.8 Klasse ontologyFramework.OFDataMapping.primitiveDataMapper.TimeStampD

1.8.1 Übersicht

1.8.2 Inhaltsverzeichnis

1.8.3 Konstruktoren

TimeStampDataMapper

1.8.4 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

2 Package ontologyFramework.OFEventManagement.OFEvent

2.1 Klassen-Liste

AsString Given an input it returns it as a String. 7

AsOWLIndividual Given an input as a String it returns the OWLNamedIndividual associated to that name. 8

AsOWLDataProperty Given an input as a String it returns the OWLDataProperty associated to that name. 8

AsOWLObjectProperty Given an input as a String it returns the OWLObjectProperty associated to that name. 9

AsInteger This class get an input and return its value as integer. 9

AsOWLClass Given an input as a String it returns the OWLClass associated to that name. 10

2.2 Package-Beschreibung

2.3 Klasse ontologyFramework.OFEventManagement.OFEventParameter.AsString

2.3.1 Übersicht

Given an input it returns it as a String. If input is null it returns null.

@author Buoncomapgni Luca

@version 1.0

2.3.2 Inhaltsverzeichnis

2.3.3 Konstruktoren

AsString

2.3.4 Methoden

getParameter

2.4 Klasse ontologyFramework.OFEventManagement.OFEventParameter.AsOWLInc

2.4.1 Übersicht

Given an input as a String it returns the OWLNamedIndividual associated to that name.

@author Buoncomapgni Luca

@version 1.0

2.4.2 Inhaltsverzeichnis

2.4.3 Konstruktoren

AsOWLIndividual

2.4.4 Methoden

getParameter

2.5 Klasse ontologyFramework.OFEventManagement.OFEventParameter.AsOWLDa

2.5.1 Übersicht

Given an input as a String it returns the OWLDataProperty associated to that name.

@author Buoncomapgni Luca

@version 1.0

2.5.2 Inhaltsverzeichnis

2.5.3 Konstruktoren

AsOWLDataProperty

2.5.4 Methoden

getParameter

2.6 Klasse ontologyFramework.OFEventManagement.OFEventParameter.AsOWLOb

2.6.1 Übersicht

Given an input as a String it returns the OWLObjectProperty associated to that name.

@author Buoncomapgni Luca

@version 1.0

2.6.2 Inhaltsverzeichnis

2.6.3 Konstruktoren

AsOWLObjectProperty

2.6.4 Methoden

getParameter

2.7 Klasse ontologyFramework.OFEventManagement.OFEventParameter.AsInteger

2.7.1 Übersicht

This class get an input and return its value as integer. If the input is "null"it returns "null";

@author Buoncomapgni Luca

@version 1.0

2.7.2 Inhaltsverzeichnis

2.7.3 Konstruktoren

AsInteger

2.7.4 Methoden

getParameter

2.8 Klasse ontologyFramework.OFEventManagement.OFEventParameter.AsOWLCla

2.8.1 Übersicht

Given an input as a String it returns the OWLClass associated to that name.

@author Buoncomapgni Luca

@version 1.0

2.8.2 Inhaltsverzeichnis

2.8.3 Konstruktoren

AsOWLClass

2.8.4 Methoden

getParameter

3 Package ontologyFramework.OFErrorManagement.OFGUI.in

3.1 Klassen-Liste

3.2 Package-Beschreibung

3.3 Klasse ontologyFramework.OFErrorManagement.OFGUI.individualGui.Following

3.3.1 Übersicht

3.3.2 Inhaltsverzeichnis

3.3.3 Konstruktoren

FollowingManager

3.3.4 Methoden

addFollowText

addFollowText

addFollowText

addFollowText

addFollowText

removeFollowingText

removeFollowingText

removeFollowingText

removeFollowingText

removeFrameFollowing

removeFrameFollowing

getFollowTxt

getColor

getFrameId

getTableType

getAllInstances

isFollowed

isFollowed

printActiveIstances

printActiveIstances

isChoosedColor

allColorToFollow

3.4 Klasse ontologyFramework.OFErrorManagement.OFGUI.individualGui.ColorRend

3.4.1 Übersicht

3.4.2 Inhaltsverzeichnis

3.4.3 Konstruktoren

ColorRenderer

3.4.4 Methoden

getTableCellRendererComponent

3.5 Klasse ontologyFramework.OFErrorManagement.OFGUI.individualGui.EditorRun

3.5.1 Übersicht

3.5.2 Inhaltsverzeichnis

3.5.3 Konstruktoren

EditorRunner

3.5.4 Methoden

run

dispose

disposeAll

3.6 Klasse ontologyFramework.OFErrorManagement.OFGUI.individualGui.Individual(

3.6.1 Übersicht

3.6.2 Inhaltsverzeichnis

3.6.3 Konstruktoren

IndividualGuiRunner

3.6.4 Methoden

getThisThread

getAllframeid

removetoAlframeid

addtoAlframeid

removefromAllName

setInterrupt

run

3.7 Klasse ontologyFramework.OFErrorManagement.OFGUI.individualGui.ClassTable

3.7.1 Übersicht

This is like TableDemo, except that it substitutes a Favorite Color column for the Last Name column and specifies a custom cell renderer and editor for the color data.

3.7.2 Inhaltsverzeichnis

3.7.3 Konstruktoren

ClassTableIndividual

3.7.4 Methoden

saveSelection

restoreSelection

mouseReleased

renderText

renderItem

getTableType

getIndividualName

getTable

update

setD

getModel

mouseClicked

mouseEntered

mouseExited

mousePressed

3.8 Klasse ontologyFramework.OFErrorManagement.OFGUI.individualGui.ClassTable

3.8.1 Übersicht

3.8.2 Inhaltsverzeichnis

3.8.3 Konstruktoren

ClassTableIndividual.MyTableModel

3.8.4 Methoden

getColumnCount

getRowCount

getColumnName

getValueAt

getColumnClass

isCellEditable

setValueAt

4 Package ontologyFramework.OFContextManagement.synchr

4.1 Klassen-Liste

OFSynchroniserManagment	16
OFSerializeSynchroniserData	16
OFSynchroniserManagmentInterface	17
OFSynchroniserData	18
OFSynchroniserBuilder	18

4.2 Package-Beschreibung

4.3 Klasse ontologyFramework.OFContextManagement.synchronisingManager.OFSy

4.3.1 Übersicht

4.3.2 Inhaltsverzeichnis

4.3.3 Konstruktoren

OFSynchroniserManagment

4.3.4 Methoden

synchronise

4.4 Klasse ontologyFramework.OFContextManagement.synchronisingManager.OFSe

4.4.1 Übersicht

@author Buoncomapgni Luca

@version 1.0

get the essential data to load again a synchroniser using the class { @link OFSynchroniserData}. Actual serialization of synchroniser is done saving it into owl file and then reload into OFSynchroniserData thanks to the informations stored in this class.

4.4.2 Inhaltsverzeichnis

4.4.3 Konstruktoren

OFSerializeSynchroniserData Initialize all the field of this class

Parameter

- **syName** the synchronizer Name
- **size** the order of the synchronizer
- **manager** the manager attached to this synchronizer
- **ontoNames** the names associated to those OWLReferences

4.4.4 Methoden

getOntoNames

Rückgabewert the names associated to those OWLReferences

getSize

Rückgabewert the order of the synchronizer

getSyName

Rückgabewert the synchronizer Name

getManager

Rückgabewert the manager attached to this synchronizer

4.5 Interface ontologyFramework.OFContextManagement.synchronisingManager.OFSy

4.5.1 Übersicht

4.5.2 Inhaltsverzeichnis

4.5.3 Methoden

synchronise

4.6 Klasse ontologyFramework.OFContextManagement.synchronisingManager.OFSy

4.6.1 Übersicht

4.6.2 Inhaltsverzeichnis

4.6.3 Konstruktoren

OFSynchroniserData

OFSynchroniserData

4.6.4 Methoden

addToList

getFromList

getList

synchronise

getSerialisableData

4.7 Klasse ontologyFramework.OFContextManagement.synchronisingManager.OFSy

4.7.1 Übersicht

4.7.2 Inhaltsverzeichnis

4.7.3 Konstruktoren

OFSynchroniserBuilder

4.7.4 Methoden

buildInfo

getInitialisedClasses

clearInstanciateNotifier

5 Package ontologyFramework.OFDataMapping.complexDataT

5.1 Klassen-Liste

TimeLine	19
ProcedureDataMapper	21
ExceptionDataMapper	22
TimeWindowsDataMapper	22

5.2 Package-Beschreibung

5.3 Klasse ontologyFramework.OFDataMapping.complexDataType.TimeLine

5.3.1 Übersicht

5.3.2 Inhaltsverzeichnis

5.3.3 Variablen

SINCEEVER_windowsSize

CLEANER_nameAugmenter

5.3.4 Konstruktoren

TimeLine

TimeLine

5.3.5 Methoden

addToTimeLine

addToTimeLine

removeFromTimeLine

getIndividualName

getClassName

getTimeWindow

getTimeLine

Rückgabewert the timeLine

setTimeLine

Parameter

timeLine the timeLine to set

getIndividualBaseName

Rückgabewert the individualBaseName

setIndividualCName

Parameter

individualBaseName the individualBaseName to set

getCleanerIndividualName

Rückgabewert the CleanerindividualName

getCleanerClassName

Rückgabewert the CleanerClassName

getClassBaseName

Rückgabewert the classBaseName

setClassBaseName

Parameter

classBaseName the classBaseName to set

getRootClass

Rückgabewert the rootClass

setRootClass

Parameter

rootClass the rootClass to set

hasCleaner

Rückgabewert the hasCleaner

addCleaner

mapNames

mapNames

getCleaner

Rückgabewert the cleaner

setCleaner

Parameter

cleaner the cleaner to set

getListInvoker

Rückgabewert the listInvoker

setListInvoker

Parameter

listInvoker the listInvoker to set

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

setKeyWords

getConstantTimeLine

getSquareTimeLine

5.4 Klasse ontologyFramework.OFDataMapping.complexDataType.ProcedureDataM

5.4.1 Übersicht

5.4.2 Inhaltsverzeichnis

5.4.3 Konstruktoren

ProcedureDataMapper

5.4.4 Methoden

mapFromOntology

mapToOntology

setKeyWords

removeFromOntology

replaceIntoOntology

5.5 Klasse ontologyFramework.OFDataMapping.complexDataType.ExceptionDataM

5.5.1 Übersicht

5.5.2 Inhaltsverzeichnis

5.5.3 Konstruktoren

ExceptionDataMapper

5.5.4 Methoden

setKeyWords

mapFromOntology

mapToOntology

removeFromOntology

replaceIntoOntology

5.6 Klasse ontologyFramework.OFDataMapping.complexDataType.TimeWindowsDa

5.6.1 Übersicht

5.6.2 Inhaltsverzeichnis

5.6.3 Konstruktoren

TimeWindowsDataMapper

5.6.4 Methoden

mapFromOntology

mapToOntology

setKeyWords

removeFromOntology

replaceIntoOntology

6 Package ontologyFramework.OFErrorManagement.OFExcept

6.1 Klassen-Liste

6.2 Package-Beschreibung

6.3 Klasse ontologyFramework.OFErrorManagement.OFException.ExceptionNotifyer

6.3.1 Übersicht

6.3.2 Inhaltsverzeichnis

6.3.3 Konstruktoren

ExceptionNotifyer

6.3.4 Methoden

notifyException

6.4 Klasse ontologyFramework.OFErrorManagement.OFException.OFExceptionBuild

6.4.1 Übersicht

6.4.2 Inhaltsverzeichnis

6.4.3 Konstruktoren

OFExceptionBuilder

6.4.4 Methoden

buildInfo

getInitialisedClasses

clearInstanciateNotifier

6.5 Interface ontologyFramework.OFErrorManagement.OFException.OFExceptionNoti

6.5.1 Übersicht

6.5.2 Inhaltsverzeichnis

6.5.3 Methoden

notifyException

6.6 Klasse ontologyFramework.OFErrorManagement.OFException.ExceptionData

6.6.1 Übersicht

6.6.2 Inhaltsverzeichnis

6.6.3 Konstruktoren

ExceptionData

ExceptionData

6.6.4 Methoden

getIndName

Rückgabewert the indName

getMess

Rückgabewert the mess

isNotify

Rückgabewert the notify

isKill

Rückgabewert the kill

getBackStep

Rückgabewert the backStep

getNotifier

Rückgabewert the notifier

notifyException notify the exception using the set OFExceptionNotifier

setNotifier

Parameter

notifier the notifier to set

7 Package ontologyFramework.OFErrorManagement.OFGUI

7.1 Klassen-Liste

ClassExcange	27
EntryInfo	34
LoadOntology	34
MyFrame	35
ClassTree	35
GuiRunner	36
ClassFindManager	36
ClassRootManager	36

7.2 Package-Beschreibung

7.3 Klasse ontologyFramework.OFErrorManagement.OFGUI.ClassExcange

7.3.1 Übersicht

7.3.2 Inhaltsverzeichnis

7.3.3 Variablen

labelTitle

allInstancesButtonLabel

serialiseFrameworkButtonLabel

legendButtonLabel

ontologyNameLabel

expandeXhekBoxTip

classRootLabel

findLabel

defaultRootClass

defaultSavingPath

indAssertLabel

classAssertLabel

defaultRootTree

nonSameIndividual

DEFAULTSERIALIZATIONPATHLABEL

ENTER

MIN

MAX

CANC

ESC

RIGTH

LEFT

UP

DOWN

classIcon

classInfIcon

individualcon

individualInfIcon

predClassIcon

predIndIcon

dataPropIcon

dataPropInfIcon

objPropIcon

objPropInfIcon

imClassIcon

imClassInfIcon

imIndividualIcon

imIndividualInfIcon

imIndividualPredIcon

imClassPredIcon

imDataPropIcon

imDataPropInfIcon

imObjPropIcon

imObjPropInfIcon

imAddColorIcon

imDeleteColorIcon

Things

dataPropertyTable

objectPropertyTable

classTable

sameIndividualTable

7.3.4 Konstruktoren

ClassExcange

7.3.5 Methoden

getTreePanelObj

addtoTreePanelObj

setNewTreeObj

refreshTreePanelObj

getOntoNameObj

getClassRootObj

getFindItemObj

getExpandAllObj

getFrameObj

getHoldTreeObj

setHoldTreeObj

getTreeObj

getIntestLabelObj

getProgressBar

setProgressBar

changeVisibilityProgressBar

getAllIndividualFrame

addAllIndividualFrame

removeAllIndividualFrame

removeAllIndividualFrame

getAllColorToFollow

setAllColorToFollow

getOntoRef

  Rückgabewert the ontoRef

setOntoRef

  Parameter

    ontoRef the ontoRef to set

getRenderer

getNullcolor

getAlreadyselectedcolor

isColorMatchSearch

setColorMatchSearch

getLoadState_btn

setLoadState_btn

getRootClassname

setRootClassname

getSavingPath

setSavingPath

getSavingName

setSavingName

getAllDataTable

addtoDataTable

removeFromDataTable

getAllObjTable

addtoObjTable

removeFromObjTable

getAllSameIndTable

addtoSameIndTable

removeFromSameIndTable

getAllClassTable

addtoClassTable

removeFromClassTable

getStringtoColor

addStringtoColor

removeStringtoColor

removeStringtoColor

addToSelectedOntoSet

removeFromSelectedOntoSet

removeFromSelectedOntoSet

clearSelectedOntoSet

getSelectedOntoSet

getTrueSelectedOntoSet

addToBuildedOntoSet

removeFromBuildedOntoSet

removeFromBuildedOntoSet

clearBuildedOntoSet

getBuildedOntoSet

getTrueBuildedOntoSet

getRunSchedulerFlag

 Rückgabewert the runScheduler_flag

setRunSchedulerFlag

 Parameter

  runScheduler_flag the runScheduler_flag to set

getExportAssertionFlag

 Rückgabewert the exportAssertion_flag

getSaveState_btn

 Rückgabewert the saveState_btn

setSaveState_btn

 Parameter

  saveState_btn the saveState_btn to set

setExportAssertionFlag

 Parameter

  exportAssertion_flag the exportAssertion_flag to set

getBroswareFrame

 Rückgabewert the brosware_Frame

setBroswareFrame

 Parameter

  brosware_Frame the brosware_Frame to set

getBroswarePathtextField

Rückgabewert the broswarePath_textField

getChosenLoadingPaths

Rückgabewert the chosenLoadingPaths

setChosenLoadingPaths

Parameter

paths

setBroswarePathtextField

Parameter

broswarePathtextField the broswarePath_textField to set

getFileChooser

Rückgabewert the fileChooser

setFileChooser

Parameter

fileChooser the fileChooser to set

getIndividualFramPeriod

Rückgabewert the individualFramPeriod

setIndividualFramPeriod

Parameter

individualFramPeriod the individualFramPeriod to set

getAllInstancesPeriod

Rückgabewert the allInstancesPeriod

setAllInstancesPeriod

Parameter

allInstancesPeriod the allInstancesPeriod to set

getTreePeriod

Rückgabewert the treePeriod

setTreePeriod

Parameter

treePeriod the treePeriod to set

getSAVINGPERIOD

Rückgabewert the sAVINGPERIOD

setSAVINGPERIOD

Parameter

sAVINGPERIOD the sAVINGPERIOD to set

7.4 Klasse ontologyFramework.OFErrorManagement.OFGUI.EntryInfo

7.4.1 Übersicht

7.4.2 Inhaltsverzeichnis

7.4.3 Variablen

name

icon

7.4.4 Konstruktoren

EntryInfo

7.4.5 Methoden

getIconNumber

getIcon

toString

setTopTree

getTopTree

path2icon

7.5 Klasse ontologyFramework.OFErrorManagement.OFGUI.LoadOntology

7.5.1 Übersicht

7.5.2 Inhaltsverzeichnis

7.5.3 Konstruktoren

LoadOntology

7.5.4 Methoden

updateOntology

saveOntology

getOntologyTokens

7.6 Klasse ontologyFramework.OFErrorManagement.OFGUI.MyFrame

7.6.1 Übersicht

7.6.2 Inhaltsverzeichnis

7.6.3 Konstruktoren

MyFrame

7.7 Klasse ontologyFramework.OFErrorManagement.OFGUI.ClassTree

7.7.1 Übersicht

7.7.2 Inhaltsverzeichnis

7.7.3 Konstruktoren

ClassTree

7.7.4 Methoden

valueChanged

doubleClick

find

isPath

expandAll

saveExpansionState

loadExpansionState

getTree

7.8 Klasse ontologyFramework.OFErrorManagement.OFGUI.GuiRunner

7.8.1 Übersicht

7.8.2 Inhaltsverzeichnis

7.8.3 Konstruktoren

GuiRunner

7.8.4 Methoden

run

7.9 Klasse ontologyFramework.OFErrorManagement.OFGUI.ClassFindManager

7.9.1 Übersicht

7.9.2 Inhaltsverzeichnis

7.9.3 Konstruktoren

ClassFindManager

7.9.4 Methoden

settFindWiev

updateComboBox

updateComboBox

7.10 Klasse ontologyFramework.OFErrorManagement.OFGUI.ClassRootManager

7.10.1 Übersicht

7.10.2 Inhaltsverzeichnis

7.10.3 Konstruktoren

ClassRootManager

7.10.4 Methoden

setRootClass

settRootWiev

updateComboBox

updateComboBox

updateExpandAll

8 Package ontologyFramework.OFRunning.OFInitialising

8.1 Klassen-Liste

8.2 Package-Beschreibung

8.3 Interface ontologyFramework.OFRunning.OFInitialising.OFBuilderInterface<T>

8.3.1 Übersicht

This interface is instantiated and called during the initialization phase of the software for the method `buildIndividual`. Its proposes is to be used to load classes into the framework. Than, they will be available trough the class: `OFBuildedListInvoker` as a Map where the initialized class from an implementation of this Procedure can be retrieved based on the list name specified by the builder ontological individual.

@author Buoncomapgni Luca

@version 1.0

8.3.2 Inhaltsverzeichnis

8.3.3 Methoden

buildInfo Given references to the ontology, already initialized classes and key words it discover the ontology to retrieve information and initialize other classes. A call to this method should clear all stored variables that are used in `getInitialised-Classes`

Parameter

keyWords retrieved by `getKeyWords`

ontoRef reference to the ontology which contains the builder individual

listInvoker references to already initialized classes from `OFInitialiser`

getInitialisedClasses During initialization phase, `OFInitialiser` calls `buildInfo` first and than retries the initialized Map from this method. Its returning value is add to the Map managed by `OFBuildedListInvoker` with key value given by `getBuildedListName`.

Rückgabewert initialisedMap a Map which contains the initialized classes linked to a key (by default it is of type String}

8.4 Klasse ontologyFramework.OFRunning.OFInitialising.OFInitialiser

8.4.1 Übersicht

This class is called during software startup. It porposes is to create an ontology reference (`OWLReferences`) and build up properties which do not change frequently during execution. Finally, them are organise inside a static and common HashMap accessible through the class `OFBuildedListInvoker`.

By default it calls all the implementation of the interface `OFBuilderInterface` which are described by an ontological individual as: `BuilderIndividual € OF-Builder          [ please refer to ... BUILDER_className]`

```
  hasKeyWords exactly 1 KeyWordInd          [HASKEYWORDS_objProp]
  hasListName exactly 1 ListName             [BUILDLISTNAME_objProp]
  implementsOFBuilderName exactly 1 PathName  [CLASSPACKAGE_objProp]
```

Than, when `buildInfo` has been called this class gets from `getInitialised-Classes` a Map of Objects that will be available on the static class `OFBuildedListIn-voker` for further usage.

@author Buoncomapgni Luca

@version 1.0

Siehe auch `OFBuilderCommon`, `OFBuilderInterface`, `OFBuildedListIn-voker`, `OWLReferences`

8.4.2 Inhaltsverzeichnis

8.4.3 Variablen

INVOKERNAME_separatorSymb Define the symbol for the name separator. Equal to «@value:INVOKERNAME_separatorSymb»

INVOKER_InstanceName IdentifiesgetFlag the name of the new instance of the class `OFBuildedListInvoker` which can be used to statically refer to the initialised list builded from this class. It is equal to `"listInvoker" + this.toString().substring( thi` `VOKERNAME\_separatorSymb ))`.

BUILDER_className It defines the name of the ontological class in which individuals must be located, to run up procedures for initialising classes. It is, by default, equal to «@value:BUILDER_className».

MAPPER_NameContains If the name of an individual inside the ontological class: «@value:BUILDER_className»; contains the String `MAPPER_NameContains` (by default equal to «@value:MAPPER_NameContains»). Than, is assured that the first occurrence of such individuals will fire the building mechanism (described in this class) always before of the other individual belong to the same ontological class.

PROCEDURE_NameContains If the name of an individual inside the ontological class: «@value:BUILDER_className»; contains the String `PROCEDURE_NameContains` (by default equal to «@value:PROCEDURE_NameContains»). Than, is assured that the first occurrence of such individuals will fire the building mechanism (described in this class) after that the other individual belong to the same ontological class has been initialised.

DEBUGGER_ClassName It defines the name of the ontological class in which the master debug configuration is belong to. By defaults it is set to «@value:DEBUGGER_ClassName». More detail on `buildDebugger`

BUILDERDEBUG_individualName Defines the name of an individual belong to the ontological class «@value:ontologyFramework.OFErrorManagement.DebuggingClassFlagDataDEBUG which describe, with the boolean value, if this class should produce logs or not. More detail on `DebuggingClassFlagData`

SERIALIZATORDEBUG_individualName

8.4.4 Konstruktoren

OFInitialiser Create new, without any effects.

It does not have any effects. It can be used to access only to the methods `buildIndividual` and `buildIndividual`.

OFInitialiser Create new reference to an ontology

Parameter

ontoName a key Name attached to this ontology reference

filepath directory absolute path to the owl file

ontologyPath IRI path associated to the relative ontology

command to create or load (from file or from web) an ontology

Exceptions

OWLOntologyCreationException Create new reference to ontology building a new `OWLReferences`; where inputs value are passed with the same meaning and order between those constructors.

8.4.5 Methoden

initialise Initialise individual belong to ontological class «@value:BUILDER_className».

Rückgabewert the manager of the map which contains all the initialised class (`staticList`).

It process all the ontological individuals belong to the class which has name «@value:BUILDER_className». For all of them it runs `buildIndividual` and update the list of initialised class. Moreover, to assure consistency, it look for an individual which as a name that contains the key word «@value:MAPPER_NameContains» If it exist it is processed for first. This methods return null and does not have any further computation if it is called from a class which has been created using the constructor `OFInitialiser`

buildIndividual Retrieve the OWLNameIndividual thought `getOWLIndividual` and call `buildIndividual`.

Parameter

individualName name of the ontological individual which belongs to «@value:BUILDER_classNa
class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert the same instance of the map with the up to date Object.

buildIndividual Build class through the interface `OFBuilderInterface`

Parameter

individual which belongs to «@value:BUILDER_className» class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert the same instance of the map with the up to date initialization (`OFBuildedListInvoker`).

To build classes for an ontological individual it goes across the following step:

```
  1 ->gets complete "Java.pakage.class" directory to aclass
that must implement the interface!
  OFBuilderInterface (this implements howdata should be
organized in classes during initialization). It uses! getImplementsName
  To do so, and returns null if even it does.
  2 -> uses Java reflection to instantiate the classde-
scribed from the String get in step (1). It uses:!
   instanciateOFBuilderByName

  3 ->get array of string which contains key words toin-
ject in the class instantiate in step (2). It uses:!
   getKeyWords

  4 ->call buildInfo

  5 ->get the name of the list which the individual wantsto
build using! getBuildedListName

  6 ->  add to the Map available in the class
   OFBuildedListInvoker
   a new element which has the name retrieved on step (5)as
a key. And the Map, returned from step (4), as value. !
```

buildIndividual as `buildIndividual` but instead of update the internal `OFBuildedListInvoker` it updates the parameter staticList

Parameter

individualName which belongs to «@value:BUILDER_className» class

ontoRef reference to the ontology in which the individual is belong to

staticList that will be update with the a new builded map

buildIndividual as `buildIndividual` but instead of update the internal `OFBuildedListInvoker` it updates the parameter staticList

Parameter

individual which belongs to «@value:BUILDER_className» class

ontoRef reference to the ontology in which the individual is belong to

staticList that will be update with the a new builded map

getInitialisedList

Rückgabewert initialisedList the manager of the map with the initialised class so far.

it returns empty Map if no class are initialised.

8.5 Klasse ontologyFramework.OFRunning.OFInitialising.OFBuilderCommon

8.5.1 Übersicht

This is a static class (set as non instanciable) which collects static methods useful during the system initialisation through builder mechanism. Especially in `buildIndividual`

@author Buoncomapgni Luca

@version 1.0

Siehe auch `OFInitialiser`, `OFBuilderInterface`, `KeyWordsMapper`, `NameMap-per`, `OFDebugLogger`

8.5.2 Inhaltsverzeichnis

8.5.3 Variablen

CLASSPACKAGE_objProp represents the name of the Object Property which link a builder individual to its Java class. It links to an individual which belongs to the "Name"ontological class. For this it represents a string which has the complete.java.class.package path to a class that implements `OFBuilderInterface`. By default it is equal to «@value:CLASSPACKAGE$_objProp >> representsthenameoftheObjectPro$
$@value : BUILDLISTNAME_objProp >>$

BUILDLISTNAME_objProp **HASKEYWORDS_objProp** represents the name of the Object Property which link a builder individual to the key words that we want inject inside the method `buildInfo`. It links to an individual which belongs to the "KeyWord"ontological class. For this, it represents an array of strings which. By default it is equal to «@value:HASKEYWORDS$_objProp >> Itisthenameofthedataproprietywhichindicatesthatthesy$
$@value : CONSOLEFLAG_dataProp >>$

CONSOLEFLAG_dataProp **FILEFLAG_dataProp** It is the name of the data propriety which indicates that the system should print on file. If it is attached to an individual which belongs to the ontological class "Debugger". By default it is set to: «@value:FILEFLAG$_dataProp >> Itisthen$
$@value : GUIFLAG_dataProp >>$

GUIFLAG_dataProp **PRINTRATE_dataProp** It is the name of the data propriety which indicates that the system should update the file and show log when their count reach this determinate threshold. If it is attached to an individual which belongs to the ontological class "Debugger". By default it is set to: «@value:PRINTRATE$_dataProp >>$
$.Notethatifthesystemisbrutallyshotteddown, someloggingtextcouldbestillbuffered; andtheywouldnotben$

8.5.4 Methoden

getKeyWords It retrieve the OWLNamedIndividual and calls `getKeyWords`

Parameter

individual name of the ontological individual which belongs to `BUILDER`$_{c}lassName$ class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert keyWords mapped keyWords from Literal to String[]

It retrieves the OWLNamedIndividual using `getOWLIndividual`, and it calls `getKeyWords`; transferring the returning value.

getKeyWords It maps the literal, described as a keyWord in the ontology, linked to the builder individual.

Parameter

individual ontological individual which belongs to `BUILDER`$_{c}lassName$ class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert keyWords mapped keyWords from Literal to String[]

Given an individual which represents the builder it gets the individual which describe the keyWord, through the Object Property defined by the name `HASKEYWORDS`$_{o}bjProp$. This, by default has the value «@value:HASKEYWORDS$_{o}bjProp >> .Than, thismethodsuses$`getKeyWordFromOntology`$tomaptheactualliteralintoanarrayofStr$

getImplementsName It retrieve the OWLNamedIndividual and calls `getImplementsName`.

Parameter

individualName name of the ontological individual which belongs to `BUILDER`$_{c}lassName$ class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert classDirectory the Java package.class path to the class which this individual represents

It computes the individual from its name; using `getOWLIndividual`. Then calls `getImplementsName`. And propagates its returning value.

getImplementsName Retrieve a reference to the class that, the builder individual addresses to

Parameter

individual ontological individual which belongs to `BUILDER`$_{c}lassName$ class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert classDirectory the Java package.class path to the class which this individual represents

It retrieves the individual which is linked trough the Object Property named: «@value:CLASSPACKAGE$_{objProp}>>$ $(definedinthefield$`CLASSPACKAGE`$_{objProp}).Finally,$

getBuildedListName It retrieve the OWLNamedIndividual and calls `getBuildedListName`.

Parameter

individualName name of the ontological individual which belongs to `BUILDER`$_{className}$ class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert listName the name that will be used as a key in the map `OFBuildedListInvoker` to address this data

It computes the individual from its name; using `getOWLIndividual`. Then calls `getImplementsName`. And propagates its returning value.

getBuildedListName It compute the name that will be used to stored the builded classes

Parameter

individual ontological individual which belongs to `BUILDER`$_{className}$ class

ontoRef reference to the ontology in which the individual is belong to

Rückgabewert listName the name that will be used as a key in the map `OFBuildedListInvoker` to address this data

It retrieves the individual which is linked trough the Object Property named: «@value:BUILDLISTNAME$_{objProp}>>$ $(definedinthefield$`BUILDLISTNAME`$_{objProp}).Finall$

buildDebugger It initialises the main debugging property

Parameter

individual which defines the debugging Property

listInvoker to retrieve already builded classes

ontoRef the reference to the ontology in which the individual belongs to.

It initialises the class `OFDebugLogger` with the information stored in the only (if more are available one will be picked up) individual belong to the ontological class «@value:ontologyFramework.OFErrorManagement.DebuggingClassFlagDa
$(whereitsnameisdescribedby:$ `DEBUGGER`$_{classFlags}).Inparticular,givenanindividualX,it$

```
 1 -> get the only individual attached to X trough  CONSOLEFLAG
```

9 Package ontologyFramework.OFDataMapping.ReservatedDa

9.1 Klassen-Liste

9.2 Package-Beschreibung

9.3 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.NameMapper

9.3.1 Übersicht

This static class is used to represent a Data Type mapper which is by default defined in the framework.

In particular, it is used to map the full classifier or a Java Class. So, given an ontological individual I which has only one Data-Property:

```
I {@value #propName} "full.classifier.toJava.Class"^^string
```

it returns a string which contains the pat. Namely: `full.classifier.toJava.Class`.

Note that this mechanism is equivalent to the String Mapper but has been divided from him to guarantee more maintainability. Moreover, it does not implement the mapping of a string into the ontology, but only the reading of such information. This is done to allow the usage of customisable Mapper for such data type without affect the initialization phase of the framework.

@author Buoncomapgni Luca

@version 1.0

9.3.2 Inhaltsverzeichnis

9.3.3 Variablen

NAME_propName Defines the default name of the ontologica Data Property to map java full qualifier between the system and the data structure.

9.3.4 Methoden

getNameFromOntology given the name of an individual as a String it retrieves the actual individual and use it to call `getNameFromOntology`. The returning value is than propagated.

Parameter

individualName name of the ontological individual

ontoRef OWL references to the ontology

Rückgabewert the name (full qualifier) of a class

getNameFromOntology Given an ontological individual which has a data property «@value:NAME$_{propName} >> itreturnsastringcontainingthenamewhichshouldpointtoaclassthrou$

Parameter

individual ontological individual from which retrieve the name

ontoRef OWL references to the ontology

Rückgabewert string stored in the property

9.4 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.EventDefinitio

9.4.1 Übersicht

9.4.2 Inhaltsverzeichnis

9.4.3 Methoden

getParameterFromOntology

getParameterFromOntology

9.5 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.EventComput

9.5.1 Übersicht

9.5.2 Inhaltsverzeichnis

9.5.3 Variablen

propName

9.5.4 Methoden

getNameFromOntology

getNameFromOntology

getVariablesName

getKeySymbols

Rückgabewert the keySymbols

setKeySymbols

Parameter

keySymbols the keySymbols to set

9.6 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.AbsoluteTime

9.6.1 Übersicht

This class defines the Object which implements an AbsoluteTimeWindows. It should be use to refer to a time windows which has its place in a time line. Basically it is just a data structure to store the state of a time windows in a particular instant. Note that in this framework time instances are describe has a Long which represents a Unix time stamp.

@author Buoncomapgni Luca

@version 1.0

9.6.2 Inhaltsverzeichnis

9.6.3 Konstruktoren

AbsoluteTimeWindow Create absolute time windows with final property set.

Parameter

lowerBound minimum time stamp of the windows

centralTime central time stamp of the windows

upperBound maximum time stamp of the windows

ck time instant fixed in the representation. It should be the time when this windows has been frozen.

9.6.4 Methoden

getUpperBound

Rückgabewert the upperBound

getLowerBound

Rückgabewert the lowerBound

getCentralTime

Rückgabewert the centralTime

getActualClock

Rückgabewert the actualClock when the framework decide to froze the windows in this class

9.7 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.KeyWordsMap

9.7.1 Übersicht

This static class is used to represent a Data Type mapper which is by default defined in the framework.

In particular it is able to map an Array of String as:

`strings = ["A" "B" "C" ...]` w.r.t an ontological individual I which as the default DataProperty:

`I {@value #propName} "A B C ..."^^string`

If key words exist in an individual that are builded from the framework they are mapped with the porpuses to inject names in the builded classes. Basically to make thir coode more general with respect to different ontologies.

Note that this mapper does not store arrays in the ontology but only read them. In fact, due non trade-safe capability of the ontology, an ArrayMapper should be used for this kind of data types. Anyway this class permits the implementation of customizable approaches to describe the succession of data inside an array, without affect the initialization procedure of the framework.

@author Buoncomapgni Luca

@version 1.0

9.7.2 Inhaltsverzeichnis

9.7.3 Variablen

KEYWORD_propName Defines the default name of the ontologica Data Property to map KeyWords between the system and the data structure.

9.7.4 Methoden

getKeyWordFromOntology Given the name of an individual it retrieve that specific individual. Than, it calls `getKeyWordFromOntology` and the returning value is propagated.

Parameter

individualName name of the ontological individual for which get the key words.

ontoRef OWL references to the ontology.

Rückgabewert an Array of string where every cell contains a key word

getKeyWordFromOntology Given an ontological individual which has a data property «@value:KEYWORD$_{p}ropName >> itreturnsanarrayofstringcontaining, alltheword(ineachcellaw$

Parameter

individual ontological individual from where retrieve the key words

ontoRef OWL references to the ontology

Rückgabewert the key words collected in an array of strings.

9.8 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.Procedure

9.8.1 Übersicht

This class represents the default procedure mapping used from this framework. This requires that default procedure mast be represented in this way, other procedures may use customizable mappers. To allow so, this class should implement the java object to use to address to a procedure.

@author Buoncomapgni Luca

@version 1.0

9.8.2 Inhaltsverzeichnis

9.8.3 Konstruktoren

Procedure Create a new procedure object initialize with some needed quantities.

Parameter

- **scheduler** which address to the individual that describe a scheduler in the ontology
- **event** top abstraction individual that represent the event for this procedure. It will run only at timeTriggrt time & if its event result true at checking time.
- **timeTrigger** ontological individual that represents the quartz object to define the next trigger for this procedure.
- **synchronization** individual that represents if this procedure should run only when other procedures have done their work.
- **procedureName** which represent a string with the fully qualifier to the implementation of the procedure.
- **concurrrentPoolSize** which represents an Integer for the size of the quartz scheduler pool.
- **checkerFreqInMillisec** which represents a Long to describe the frequency too check the state of the events assign to this object.

9.8.4 Methoden

getScheduler

Rückgabewert the scheduler

getEvent

Rückgabewert the event

getTimeTrigger

Rückgabewert the timeTrigger

getSynchronization

Rückgabewert the synchronization

getProcedureNameLitteral

Rückgabewert the procedureNameLitteral

getProcedureName

Rückgabewert the procedureName

getConcurrrentPoolSizeLittteral

Rückgabewert the concurrrentPoolSizeLittteral

getConcurrentPoolSize

Rückgabewert the concurrentPoolSize

getCheckerFreqInMillisecLitteral

Rückgabewert the checkerFreqInMillisecLitteral

getCheckerFreqInMillisec

Rückgabewert the getCheckerFreqInMillisec

getSimpleCleaner ?? Static methods which returns a simple Procedure object

Parameter

ontoRef

Rückgabewert the Procedure

9.9 Klasse ontologyFramework.OFDataMapping.ReservatedDataType.TimeWindow

9.9.1 Übersicht

This class is the mapping representation of an ontological time windows.

A time windows is represented in the ontology with an individual and a bunch of SWRL rules more addressed in `TimeWindowsDataMapper`.

@author Buoncomapgni Luca

@version 1.0

9.9.2 Inhaltsverzeichnis

9.9.3 Konstruktoren

TimeWindow Create a time window whic has a size and a center value with respect to the centre (= 0) of an abstract time line always time invariant. For example give time windows as: «@literal:T1(10, 0), T2(10, -15)» «@literal:and T3(10, +15)» the representation will be distributed uniformally in an always fixed time line; as: «@literal:T2€[-15, -5) T1€[-5, 5) T3[5, 15)». During the running of the system this line will move during time and so the actual windows would be: «@literal:T2€[-15+t, -5+t) T1€[-5+t, 5+t) T3[5+t, 15+t)» Where «@literal:t» is a value close to the real time instance.

Parameter

size number of millisecond of the windows size

relativeCentre relative number of millisecond in which the windows is centered with respect to now.

TimeWindow It creates a time windows using the parameters: `size` and `relative center` as sow in `TimeWindow`. Moreover, it assign to this class names for ontological entities that are needed to map the windows from this framework to the ontology. In particolar they are the name of an Individual belong to the ontological class «@literal:DataType -> TimeWindow». And the name of the class in which other individual can be classified as belong to a give time window. «@literal:DataType -> TimeRepresentation» is the ontological path by default

Parameter

size number of millisecond of the windows size

relativeCentre relative number of millisecond in which the windows is centered with respect to now.

individualName name of the individual that describe this time window in the ontology

className name of the class that will behave as a time windows from the reasoning point of view. This class will collect all the other individual of the ontology which has a time stamp property that fall on this windows of time.

9.9.4 Methoden

getSize

Rückgabewert the size

getRelativeCentre

Rückgabewert the relativeCentre

getIndividualName

Rückgabewert the individual

setIndividualName

Parameter

individualName

getClassName

Rückgabewert the className

setClassName

Parameter

className the className to set

getRootClass

Rückgabewert the rootClass

setRootClass

Parameter

rootClass the rootClass to set

toString

getKeyWord

Rückgabewert the keyWord

setKeyWord

Parameter

keyWord the keyWord to set

getAbsoluteTimeWindows return the windows with its size and central instant computed with respect to an reference clock value. (long unix time stamp in milliseconds)

Parameter

actualCk time stamp of when compute the windows

Rückgabewert time windows compute with respect to an actual referiment.

getAbsoluteTimeWindows

Rückgabewert the absulute time windows computed for the time which this method is called

Return the result of: `getAbsoluteTimeWindows (System.currentTimeMillis())`

getAbsoluteTimeWindows return the windows with its size and central instant as are descripted in the ontology refered from `ontoRef`. It must contain the data property `keyWord[ 4] = "hasTypeTimeWindowsUpperBound` and `keyWord[ 5] = "hasTypeTimeWindowsLowerBound`.

Parameter

ontoRef time stamp of when compute the windows

Rückgabewert time windows compute with respect to an actual referiment.

10 Package ontologyFramework.OFEventManagement.OFLogi

10.1 Klassen-Liste

OFEventBuilder This class, as all the class that implements `OFBuilderInterface` has the proposes to initialize classes to be used during system evolution. 55

OFEventInterface This interface is used to define the event procedure. 57

OFEventDefinition This class contains the initialisated definition for all the events. 58

OFEventAggregation This class represent the event linked to every individual belong to the ontological class `"Event"`. 58

10.2 Package-Beschreibung

10.3 Klasse ontologyFramework.OFEventManagement.OFLogicalEventManagement

10.3.1 Übersicht

This class, as all the class that implements `OFBuilderInterface` has the proposes to initialize classes to be used during system evolution. In this case its initializes Events, in particular the classes: `OFEventParameterDefinition`, `OFEventDefinition` and `OFEventAggregation`.

A call to `buildInfo` causes the reset of the initialized classes Map, then all the individual inside the ontological class, named `keyWord[ 0]` (by default: ”Event”) are processed. Where, The definition of this class must be: `(hasTypeEventParameter min 1 string) and (`
`EventComputeLow exactly 1 string)`

For all of them it retrieves the computational low (ex: ”r1 && r2”) as a string and creates a new `OFEventAggregation`. Than, it gets the value of the data property named `keyWord[ 1]`} (by default: ”hasTypeEventParameter”) and it parse the incoming value (for example: ”r1 = OFEventProcedure_IndName”) using the symbol `ASSEGNATION`$_{s}ymb$; by default «@value:ASSEGNATION$_{s}ymb >> .Notethattheparameterisdiscardedift$
$implementsOFEventName).Finally, itgetsparameterstroughontologicalindividuallinkedbytheobjectpro$
`keyWord[ 3]`$(byDefault: ”hasEventDefinition”).ParametersareaddedtotheEventDefinitionthanksa$
`keyWord[ 4]`$(bydefault”hasTypeEventDefinition”).$

The call to the method `getInitialisedClasses` after called `initializeDefinition` returns a `HashMap<String, OFEventAggregation>` where, keys are the names of the individuals belong to the class named `keyWord[ 0]`. While the values are the classes which represent and allow to compute all the Events available during the calling of `initializeDefinition`

@author Buoncomapgni Luca

@version 1.0

10.3.2 Inhaltsverzeichnis

10.3.3 Variablen

ASSEGNATION_symb Symbol for divide parameters and accept tokens, used only in Event aggregation

ENDLine_symb System symbol to end a line, it represents the end of a command

ASSEGNATIONPARAMETER_symb Symbol to assign parameters to a variable. It represents an assegnation during Parameter definition

VARIABLE_symb Symbol which identify that the word used before than the next `SPLIT`$_{s}ymb$ is a local variable.

RETURN_symb Symbol which identify that the word used before than the next `SPLIT`$_{s}ymb$ is a the actual event instruction and no more a parameter.

ATONTOLOGY_symb It can be used after the declaration of a variable and identify the `OWLReferences` name i in whihc the parameter must be retrieved. If it is not specified than the corrent ontology is considered.

COMMAND_symb Symbol used the decide chains of computation to retrieve parameter, where they must to contains `SPLIT`$_{s}ymb$. An example is: `name.AsInteger.AsIntegerOWLDa` equivalent to write `AsIntegerOWLDataProperty( AsInteger( name.toString()));` in other languages.

NULL_symb Intercepts whenever null value should be given as input to computer parameter. It can be used only as a first element of parameter computation.

STARTPARAMETR_symb It defines the starting point in which parameter are used inside the event definition. It must be used in the returning line defined by `RETURN`$_{s}ymb$. Between this two symbol no check of the name is provided.

ENDPARAMETER_symb It defines the starting point in which parameter are used inside the event definition.

SPLIT_symb Symbol used compute tokens of every lines.

IMPORT_symb Symbol used to define the full identify package in which all the computational method to compute parameters are located. This string is added to the name of the name of the procedure (ex: ”in: java.package.”+ ”AsIntegerOWLDataProperty).

10.3.4 Konstruktoren

OFEventBuilder

10.3.5 Methoden

buildInfo

getInitialisedClasses

getText

10.4 Interface ontologyFramework.OFEventManagement.OFLogicalEventManagement

10.4.1 Übersicht

This interface is used to define the event procedure. It is called from `compute` which calls `isCorrectInput` first and, if the result is true it calls `evaluateEvent` and propagate the result to the `compute`.

@author Buoncomapgni Luca

@version 1.0

10.4.2 Inhaltsverzeichnis

10.4.3 Methoden

isCorrectInput sviluppated with safety pourposes it is called to check if the type of parameter in inputs are correct. If this return false the event result of `evaluateEvent` will be setted to null.

Parameter

inputs ordered in accord with the ontological definition of the events trhougth the object property `"hasTypeEventDefinition`

Rückgabewert true if the inputs are corrects. If return else, event computation dennied.

evaluateEvent implements how compute the event results starting from the inputs retrieved in `getParameter`

Parameter

> **inputs** parameter
>
> **invoker** access to a builded class duriing software initialization

Rückgabewert true f the event occurs, false otherwise.

10.5 Klasse ontologyFramework.OFEventManagement.OFLogicalEventManagement

10.5.1 Übersicht

This class contains the initialisated definition for all the events. It collets also references to `OFEventParameterDefinition`. And a method to compute the event result. Which is called by `compute` and calls `getParameter`.

@author Buoncomapgni Luca

@version 1.0

10.5.2 Inhaltsverzeichnis

10.5.3 Konstruktoren

OFEventDefinition

10.5.4 Methoden

compute

10.6 Klasse ontologyFramework.OFEventManagement.OFLogicalEventManagement

10.6.1 Übersicht

This class represent the event linked to every individual belong to the ontological class `"Event"`.

It is based on a boolean parameterized expression given as a String. And a list of parameters that can be added or removed. All parameters are expressed in therms of instances of `OFEventDefinition` and they must tagged with the name used in the expression.

@author Buoncomapgni Luca

@version 1.0

10.6.2 Inhaltsverzeichnis

10.6.3 Konstruktoren

OFEventAggregation Create new event with a specific boolean low as a String. This will be processed by the library `MVEL` on runtime and must return always a boolean value.

Parameter

aggregationLow parameterized logical relation

10.6.4 Methoden

addParameter Add a new parameter to the definition of this event tagged with the variable name used to define the aggregation low. This class must contains one OFEventDefinition for each name used in the `String aggregationLow`. (ex: "r1 r2"!) where r1 and r2 are variables.

Parameter

varName the name of the variable used in the String `aggregationLow`

eventDef initialized definition of the parameter

removeParameterMap Remove a parameter from definition of this event.

Parameter

varName the name of the variable used in the String `aggregationLow`

clearParameterMap Remove all the parameters from definition of this event.

compute Compute event result. It goes for all the parameter added to this class and calls `ofEventDefinition.compute( invoker)`. Finally, the returning boolean value is used to compute the aggregation low.

Parameter

invoker builded list of class during startup used by `evaluateEvent`

Rückgabewert true if the event occurs in this moment

11 Package ontologyFramework.OFErrorManagement

11.1 Klassen-Liste

11.2 Package-Beschreibung

11.3 Klasse ontologyFramework.OFErrorManagement.DebuggingClassFlagData

11.3.1 Übersicht

11.3.2 Inhaltsverzeichnis

11.3.3 Variablen

DEBUGGER_classFlags

DEBUGGINGFLAG_objectProperty

DEBUGGERLISTNAME_mapKey

11.3.4 Methoden

getFlag

getDebuggingMap

Rückgabewert the debuggingmap

setDebuggingMap

Parameter

debuggingMap the debuggingMap to set

rebuild

11.4 Klasse ontologyFramework.OFErrorManagement.OFDebugLogger

11.4.1 Übersicht

11.4.2 Inhaltsverzeichnis

11.4.3 Variablen

dataFormat

11.4.4 Konstruktoren

OFDebugLogger

OFDebugLogger

OFDebugLogger

OFDebugLogger

11.4.5 Methoden

addDebugStrign

addDebugStrign

setFlagToFollow

getFlagToFollow

getFollowedClass

getNamedInstance

getNamedClass

getDebugText

cleanDebugText

removeDebug

finalize

getAllInstances

printActiveIstances

getLogInfo

getTableInfo

printLogOnConsole

printLogOnFile

getPrintOnFile

Rückgabewert the printOnFile

setPrintOnFile

Parameter

printOnFile the printOnFile to set

getPrintOnConsole

Rückgabewert the printOnConsole

setPrintOnConsole

Parameter

printOnConsole the printOnConsole to set

getOrderPrintingRate

Rückgabewert the orderPrintingRate

setOrderPrintingRate

Parameter

orderPrintingRate the orderPrintingRate to set

getStartGui

Rückgabewert the startGui

setStartGui

Parameter

startGui the startGui to set

11.5 Klasse ontologyFramework.OFErrorManagement.ShowError

11.5.1 Übersicht

11.5.2 Inhaltsverzeichnis

11.5.3 Konstruktoren

ShowError

11.5.4 Methoden

catchCaller

11.6 Klasse ontologyFramework.OFErrorManagement.FileManager

11.6.1 Übersicht

11.6.2 Inhaltsverzeichnis

11.6.3 Variablen

keyTxt

keyLog

keyJava

keyOWL

delimiterText

defaultComand

11.6.4 Konstruktoren

FileManager

FileManager

11.6.5 Methoden

loadFile

closeFile

printOnFile

printOnFile

deleteFile

getFilePath

getFileWriter

getFileFormat

getDefaultBasePath

getPossibleFormat

getFormatFromAbsolutePath

12 Package ontologyFramework.OFDataMapping

12.1 Klassen-Liste

OFDataMapperBuilder This class implements the builder for Data Types. 65

OFDataMapperInterface This class defines the method that must be created into a class to define a one to one Mapper between an Ontological entity and a Data Type. 66

12.2 Package-Beschreibung

12.3 Klasse ontologyFramework.OFDataMapping.OFDataMapperBuilder

12.3.1 Übersicht

This class implements the builder for Data Types.

When `buildInfo` is called (by default from `OFInitialiser`) it creates and initialise a set of `OFBuilderInterface` that are colled in a hasMap with keys equals to the name of the keyWord at index 0 given from `getKeyWordFromOntology`. When the building process is complete the HasMap is added into the static map available througth `OFBuildedListInvoker` using the method: `getInitialisedClasses`.

@author Buoncomapgni Luca

@version 1.0

12.3.2 Inhaltsverzeichnis

12.3.3 Konstruktoren

OFDataMapperBuilder

12.3.4 Methoden

buildInfo

getInitialisedClasses

12.4 Interface ontologyFramework.OFDataMapping.OFDataMapperInterface<OntoEn

12.4.1 Übersicht

This class defines the method that must be created into a class to define a one to one Mapper between an Ontological entity and a Data Type. This allows to standardize the methods for all the mapper that can be used in the framework. By default they are created from `OFDataMapperBuilder` and stored in `OFBuildedListInvoker`.

@author Buoncomapgni Luca

@version 1.0

12.4.2 Inhaltsverzeichnis

12.4.3 Methoden

mapFromOntology Get informations from the ontology and returns a Java Object

Parameter

- **entity** ontological entity from which retrieve informations.
- **ontoRef** OWL reference to the ontology

Rückgabewert a Java Object which represent the ontological entity mapped in a Java data type

mapToOntology Store informations given as java Object into the ontology. It returns `true` if success.

Parameter

- **entity** ontological entity to create, delete or modify into the description
- **value** java Object which represent the data in a particular data-type to be added into the ontology
- **ontoRef** OWL references to the ontology

Rückgabewert true if the operation is successfully completed

removeFromOntology Delete from the ontology a particolar entity that is represented by the given Java Object.

Parameter

- **entity** ontological entity to delete (if it exists) from the description
- **value** java Object which represent the data in a particular data-type to be added into the ontology
- **ontoRef** OWL references to the ontology

Rückgabewert true if the operation is successfully completed

replaceIntoOntology Replace the value of an ontological entity in atomic way. Which means that, given an ontological entity E with a particular property A. The method will assign to A the new value and will remove the old one with no possibilities for the reasoner to update the data structure during those operations. Note that E is given as input while A should be encoded in the implementation of the interface.

Parameter

entity ontological entity for which replace the values

oldArg value to remove

newArg value to add

ontoRef OWL references to the ontology

Rückgabewert true if the operation is successfully completed

setKeyWords this method is called from `buildInfo` and should be used to store internally the name of interesting data represented in the ontology. Those data are setted in the ontology itself and are used all the time the mapper is called.

Parameter

kw words used in the ontology

13 Package ontologyFramework.OFEventManagement.OFLogi

13.1 Klassen-Liste

13.2 Package-Beschreibung

13.3 Klasse ontologyFramework.OFEventManagement.OFLogicalEventManagement

13.3.1 Übersicht

This class implement the event that takes as input : `...( OWLNamedIndividual ind, OWLClass cl)`. Which return true if the individual belongs to the class and false otherwise. In the ontology an event must be defined which belongs to the class `"OFEvent"` thus has the properties:

```
  implementsOFEventName "ontologyFramework.OFEventManagement.OFEventImplementa
  &
  hasEvent definition"in:ontologyFramework.OFEventManagement.OFEventParameter.
?a @ontoName(S1-3) Exception.AsOWLClass ?b @ontoName exc.AsOWLIndividual
!r IsInClass(?b ?a)"3trign!
```

So, `isCorrectInput` return true if `ts.get(0).getParameter() instanceof OWLNamedIndividual` and `inputs.get( 1).getParameter() instanceof OWLClass`, `inputs.get(1).getOntoRef()` = null! and `inputs.get(0).getOntoRef()` = null!, are true. The `evaluateEvent` just ask to the reasoner of the ontology named `"ontoName(S1-3)"` if in the class ”Exception”exist an individual called ”exc”and propagates the answare.

@author Buoncomapgni Luca

@version 1.0

13.3.2 Inhaltsverzeichnis

13.3.3 Konstruktoren

IsInClass

13.3.4 Methoden

isCorrectInput

evaluateEvent

13.4 Klasse ontologyFramework.OFEventManagement.OFLogicalEventManagement

13.4.1 Übersicht

This class implement the event that takes as input : `...( OWLNamedIndividual ind, OWLDataProperty prop)`. Which return true if the individual has that property and false otherwise. In the ontology an event must be defined which belongs to the class `"OFEvent"` thus has the properties:

```
  implementsOFEventName "ontologyFramework.OFEventManagement.OFEventImplementa
  &
  hasEvent definition"in:ontologyFramework.OFEventManagement.OFEventParameter.
?a  exc.AsOWLIndividual  ?b  hasExceptionNotify.AsOWLDataProperty
!r HasBooleanTrue( ?a ?b)"3trign!
```

So, `isCorrectInput` return true if `inputs.get( 0).getParameter() instanceof OWLNamedIndividual` and `inputs.get( 1).getParameter() instanceof OWLObjectProperty` and `inputs.get( 0).getOntoRef()` are true. The `evaluateEvent` just uses the `invoker.getClassFromList( "MappersList", "Boolean")` the get the boolean mapper and check if the value is true.

@author Buoncomapgni Luca

@version 1.0

13.4.2 Inhaltsverzeichnis

13.4.3 Konstruktoren

HasBooleanTrue

13.4.4 Methoden

isCorrectInput

evaluateEvent

13.5 Klasse ontologyFramework.OFEventManagement.OFLogicalEventManagement

13.5.1 Übersicht

13.5.2 Inhaltsverzeichnis

13.5.3 Konstruktoren

HasDifferentClassState

13.5.4 Methoden

isCorrectInput

evaluateEvent

14 Package ontologyFramework.OFEventManagement.OFTim

14.1 Klassen-Liste

TriggFrequently	This class create a new Quartz Trigger with particular parameter.	71
TriggerNow		72

14.2 Package-Beschreibung

14.3 Klasse ontologyFramework.OFEventManagement.OFTimeTriggerManagement.

14.3.1 Übersicht

This class create a new Quartz Trigger with particular parameter. The ontology must contain an individual which as those properties:

`implementsOFTimeTriggerName "ontologyFramework.OFEventManagement.OFTimeTrigg`
`&`
`in:ontologyFramework.OFEventManagement.OFEventParameter.?frequency`
`10.AsInteger ?couter` $\cdot AsInteger?priority6.AsInteger!rtriggFrequentlyInSeconds(?frequency?pr$

`So the method isCorrectInput returns true only if: inputs.get(0).getParameter(`
`stanceof Integer, inputs.get(1).getParameter() instanceof Inte-`
`ger and (inputs.get(2).getParameter() == null) || (inputs.get(2).getParameter()`
`stanceof Integer) are true. While the method getTrigger returns`
`a quartz trigger with the specified parameter. If count number`
`is equal to null than, the trigger has repeatForever() property.`
`if counter is 0 than the trigger is "fired now"only once`

@author `Buoncomapgni Luca`

@version `1.0`

14.3.2 Inhaltsverzeichnis

14.3.3 Konstruktoren

TriggFrequently

14.3.4 Methoden

isCorrectInput

getTrigger

14.4 Klasse ontologyFramework.OFEventManagement.OFTimeTriggerManagement.

14.4.1 Übersicht

14.4.2 Inhaltsverzeichnis

14.4.3 Konstruktoren

TriggerNow

14.4.4 Methoden

isCorrectInput

getTrigger

15 Package ontologyFramework.OFRunning

15.1 Klassen-Liste

15.2 Package-Beschreibung

15.3 Klasse ontologyFramework.OFRunning.OFSystemState

15.3.1 Übersicht

This class represent the state of the framework.

@author Buoncomapgni Luca

@version 1.0

15.3.2 Inhaltsverzeichnis

15.3.3 Variablen

OWLREFERENCES_keyWord Key with which the OWLReferences will be added into OFBuildedListInvoker

SYNCRHONISERLIST_keyWord Key with which the OFSynchroniser-Data has been added into OFBuildedListInvoker. Used since OFSynchroniserData not serializable.

DATAFORMAT Format of folder path in default usage

15.3.4 Konstruktoren

OFSystemState It calls OFSystemState. Where the last three parameter are set to null.

Parameter

builded map to be serialized

listInvokerInstanceName name of the Map to be initialized

exportInferd if true all the asserted axiom will be exported in the Ontology that will be saved

OFSystemState It calls OFSystemState. Where the two central parameter are set to null.

Parameter

builded map to be serialized

listInvokerInstanceName name of the Map to be initialized

ontoFilePath folder directory in which save the owl file created from the serialization mechanism. Null value loads in a path as: System.getProperty("user.dir") + "/files/Seriali
pleDateFormat(DATAFORMAT).format(date)}

exportInferd if true all the asserted axiom will be exported in the Ontology that will be saved

OFSystemState It calls OFSystemState. Where the last parameter is null and the other are propagated.

Parameter

builded map to be serialized

listInvokerInstanceName name of the Map to be initialized

ontoToSerializeName set of keys belong to Map that will be serialized. Null value loads in a serialization of all the map: getMap

listToSerializeName set of keys belong to the OWLReferences map that will be serialized. Null value loads in a serialization of all the map: getAllInstances

exportInferd if true all the asserted axiom will be exported in the Ontology that will be saved

OFSystemState Create a new System state relate to informations carried by builded. It ask for serializable representation of non serializable class and store them inside the Map. Than it adds additional data has debugging flags, carried by DebuggingClassFlagData; and OWLReferencesSerializable. In this last case the frameworks saves ontologies trough getAllSerializableInstances

Parameter

builded map to be serialized

listInvokerInstanceName name of the Map to be initialized

ontoToSerializeName set of keys belong to Map that will be serialized. Null value loads in a serialization of all the map: getMap

listToSerializeName set of keys belong to the OWLReferences map that will be serialized. Null value loads in a serialization of all the map: getAllInstances

ontoFilePath folder directory in which save the owl file created from the serialization mechanism. Null value loads in a path as: System.getProperty("user.dir") + "/files/Seriali
pleDateFormat(DATAFORMAT).format(date)}

exportInferd if true all the asserted axiom will be exported in the Ontology that will be saved

15.3.5 Methoden

getBuilderListName

Rückgabewert builderListName the name of the instance of OF-BuildedListInvoker which can been serialized

getSerialMap

Rückgabewert serialMap the instance of OFBuildedListInvoker which can been serialized

getOntologyFilePath

Rückgabewert ontologyFilePath the folder directory in which the ontologies are saved

getSerializableListName

Rückgabewert serializableListName get the keys of the map OF-BuildedListInvoker which are not serializable and need further computations.

setSerializableListName

Parameter

unserializableName set the keys of the map OFBuildedListInvoker which are not serializable and need further computations.

addToSerializableListName

Parameter

entry add the key of the map OFBuildedListInvoker which are not serializable and need further computations.

15.4 Klasse ontologyFramework.OFRunning.OFSerializator

15.4.1 Übersicht

This static class collects common methods to serialize and de-serialize the framework. Exception and errors are handled by OFDebugLogger

@author Buoncomapgni Luca

@version 1.0

15.4.2 Inhaltsverzeichnis

15.4.3 Variablen

SERIALIZATION_fileExtension the format of the file automatically added to directory/name".<<@value:SERIALIZATION$_{f}ileExtension>>$ " $Thenameoftheontologicalindividualwhichrepresenttheschedulerthatwillbeinitializeafterde-serializaition.$

SCHEDULER_individualNamePROCEDURE_individualName The name of the ontological individual which represent the algorithms that will be initialize after de-serializaition.

15.4.4 Methoden

saveFrameworkState It calls saveFrameworkState with all parameter equal to "null".

Parameter

exportInferd if true all the asserted axiom will be exported in the Ontology that will be saved

Rückgabewert OFSystemStates set of classes which represents the state of the OntologicalFramework

saveFrameworkState It calls saveFrameworkState with the first two parameter equal to "null"and the third equal to ontoFilePath.

Parameter

ontoFilePath the folder directory in which you want to store the ontologies

exportInferd if true all the asserted axiom will be exported in the Ontology that will be saved

Rückgabewert OFSystemStates set of classes which represents the state of the OntologicalFramework

saveFrameworkState It goes across all the instances of OFBuildedListInvoker and, for each of them Instantiates a new OFSystemState. Those are collected in a Set and given as output. All the parameter of this function are passed to constructor: OFSystemState

Parameter

ontoToSerializeName list of OWLReferences instance names that we want to serialize. If it is equal to "null"than all the instances are serialized

listToSerializeName list of Names of the Individual linked to the builder by Object Property <<@value:ontologyFramework.OFRunning.
$the folder directory in which you want to store the ontologies$

ontoFilePath **exportInferd** if true all the asserted axiom will be exported in the Ontology that will be saved

Rückgabewert OFSystemStates set of classes which represents the state of the OntologicalFramework

serializeObjectToFile It calls serializeObjectToFile where the first two parameters are "null"and the third is: toSerialize

Parameter

toSerialize set of classes which represents the state of the OntologicalFramework

Rückgabewert serializationPaths the set of paths in which objects has been serialized in a .<<@value:SERIALIZATION$_{fileExtension} >>$ $file.$

serializeObjectToFile It iterate over all the value of toSerialize. For each of them it retrieves the serializable Map (of type OFBuildedListInvoker) and writes it in a file.

If filePath is "null"than the base files path will be: oFBuildedListInvoker_SerialMap.getOntologyFilePath. Otherwise filePath it must be an absolute map to a folder, in which serialize the framework Java classes

If fileName is "null"than the name of a serialized Java Class will be getInstanceName. Otherwise it will be fileName + (count++).toString

In any case the complete path to a file will be: filePath + FileName + <<@value:SERIALIZATION$_{fileExtension}$ >>}

Parameter

filePath folder path in which save the serialized Classes

fileName base name of the serialized Classes belong to the folder linked by filePath

toSerialize set of classes which represents the state of the OntologicalFramework

Rückgabewert serializationPaths the set of paths in which objects has been serialized in a .<<@value:SERIALIZATION$_{fileExtension}$ >> *file.*

It load the frame status from files and re-instantiate it. It load all the files and retreive the related Map than, for each of it makes this steps: 1 -> get all OWLReferencesSerializable
load all the ontologies calling new OWLReferences.
eliminate them from the Map
2 -> for all the classes which are not Serializable.
Get serializable objects and re-instantiate them.
substitute those class between each other in the Map
3 -> re-build the scheduler since is not Serializable
substitute those class between each other in the Map
4 -> re-build Debugging Map
eliminate them from the Map
5 -> re-build static property of Map and return it

Parameter

filePaths the set of paths in which objects has been serialized in a .<<@value:SERIALIZATION$_{fileExtension}$ >> *file.*

Rückgabewert loadedList the set of list builded and stored during serialization.

16 Package ontologyFramework.OFEventManagement

16.1 Klassen-Liste

OFEventRepresentation This class contains a basic implementation of how store initializate mechanism from the OFlanguage in data property. 79

OFEventParameterDefinition This class defines the definition of a paramiter of a particular Event. 81

EventComputedData This class simply contains two field, moved during event parameter computation between the classes `OFEventParameterDefinition` and a class, called by name, which implements `OFEventInterface`. 82

OFEventParameterInterface This class is interface to implement the definition a parameter to be used during event computation. 83

16.2 Package-Beschreibung

16.3 Klasse ontologyFramework.OFEventManagement.OFEventRepresentation

16.3.1 Übersicht

This class contains a basic implementation of how store initializate mechanism from the OFlanguage in data property. Thasnks to this class it is possible to use the Event mapping mecchanism semply defining how to compute them.

@author Buoncomapgni Luca

@version 1.0

16.3.2 Inhaltsverzeichnis

16.3.3 Konstruktoren

OFEventRepresentation Create a new definition of event

Parameter

packageClassName the full qualify to a class that represent the event provedure implementing `OFEventInterface`

16.3.4 Methoden

getOrder

Rückgabewert varNameOrder the ordered variable names to compute parameter for the event.

setOrder

Parameter

varNameOrder the ordered variable names to compute parameter for the event.

getClassName

Rückgabewert the fully java quilifier to the event class that implements `OFEventInterface`

getParameterMap

Rückgabewert the parameterMap. It contains an unordered set of `OFEventParameterDefinition` linked by variableName string value.

setParameterMap

Parameter

parameterMap set the parameterMap. It contains an unordered set of `OFEventParameterDefinition` linked by variableName string value.

addToParameterMap add a parameter into the event tagged by its variable name. Those names must be coherent with the one retrieved during the event building; managed by `OFEventBuilder`

Parameter

varName the name of the parameter

epd a parameter to inject as input into the Event implementstion (interface of `OFEventInterface`)

addToParameterMap add parameters into the event as a map where keys are variable names and value initialized parameter. The names, must be coherent with the one retrieved during the event building managed by `OFEventBuilder`

Parameter

map of varName and parameter to inject as input into the Event implementation (interface of `OFEventInterface`)

removeFromParameterMap

Parameter

varName name of the variable which define the parameter to remove from this event.

getComputedParameterList It goes trough all the parameter following the ordered name of variables. For each of them it instantiates an new `EventComputedData` with the correspondent parameter (computed each time using `getParameter`) and its ontology reference. All of them are than collected in a ordered List.

Rückgabewert update computed list of parameter results

compute /** Here the creation of a new instance of the event implementation should be done (by default for `OFTimeTriggerInterface` and `OFEventInterface`. Using `getClassName` is possible to load a new instance of such Interface (you must define your own methods to do so). Than the ordered and update parameter values can be retrieved using `getComputedParameterList`, to have the inputs to check your own event implementation.

Parameter

invoker lsit of builded class during initialization to be used by `evaluateEvent`

Rückgabewert the event result

16.4 Klasse ontologyFramework.OFEventManagement.OFEventParameterDefinition

16.4.1 Übersicht

This class defines the definition of a paramiter of a particular Event.

@author Buoncomapgni Luca

@version 1.0

16.4.2 Inhaltsverzeichnis

16.4.3 Konstruktoren

OFEventParameterDefinition create new parameter definition

Parameter

classpackageName full java qualifyer of the parameter implementstion. Which must implement `OFEventParameterInterface`

parameterInput initial input to the parameter implementation

eventOntoRef ontological reference of this parameter

16.4.4 Methoden

getClassPackageName

Rückgabewert the java full qualifyer of the parameter implementation

getInput

Rückgabewert the input to the parameter implementation

setInput

Parameter

input set the input to the parameter implementation

getOWLReferences

Rückgabewert the ontological reference of this parameter

getParameter Compute the value of the parameter. It instanciate the parameter implentation using `getClassPackageName` and call `getParameter`, Where `getInput` and `getOWLReferences` are the inputs, respectivaly.

Rückgabewert the object returned by `getParameter`

16.5 Klasse ontologyFramework.OFEventManagement.EventComputedData

16.5.1 Übersicht

This class simply contains two field, moved during event parameter computation between the classes `OFEventParameterDefinition` and a class, called by name, which implements `OFEventInterface`.

@author Buoncomapgni Luca

@version 1.0

16.5.2 Inhaltsverzeichnis

16.5.3 Konstruktoren

EventComputedData create new EventComputedData

Parameter

parameter computed parameter for the Event

ontoRef ontological referents of the parameter

16.5.4 Methoden

getParameter

Rückgabewert the parameter

getOntoRef

Rückgabewert the ontological reference of the parameter;

toString

16.6 Interface ontologyFramework.OFEventManagement.OFEventParameterInterface

16.6.1 Übersicht

This class is interface to implement the definition a parameter to be used during event computation. It is instanciated and called by `getParameter`

@author Buoncomapgni Luca

@version 1.0

16.6.2 Inhaltsverzeichnis

16.6.3 Methoden

getParameter It compute and return a new value of the parameter to be used during event computation. Inputs are defined by `OFEventBuilder` and `OFEventDefinition` and are the first rigth token of a parameter chain starting from the name of the class that implements this interface. Example1: `exc.AsString` if the implementation is a class called ”AsString”than, on this method `input = "exc"`. (String by default.) Example2: `@OntoName exc.AsOWLNamedIndividual.Exist` if the implementation is a class with name ”Exist”than, `input = OWLNAMEDINDIVIDUAL_withName_exc` and ”OntoName”is the name associated to the ontology in with the individual ”exc”should be retrieved.

Parameter

input of the parameter coming from ontological definition

ontoRef ontological reference of this input

Rückgabewert the actual parameter for event computation.

17 Package ontologyFramework.OFEventManagement.OFTim

17.1 Klassen-Liste

OFTimeTriggerDefinition This class contains the initialisated definition for all the temporal trigger. 84

OFTimeTriggerBuilder This class, as all the class that implements `OFBuilderInterface` has the proposes to initialize classes to be used during system evolution. 85

OFTimeTriggerInterface This class is used to initialise, store and compute Temporal Trigger. 86

17.2 Package-Beschreibung

17.3 Klasse ontologyFramework.OFEventManagement.OFTimeTriggerManagement.

17.3.1 Übersicht

This class contains the initialisated definition for all the temporal trigger. It collets also references to `OFEventParameterDefinition`. And a method to get the actual trigger object.

@author Buoncomapgni Luca

@version 1.0

17.3.2 Inhaltsverzeichnis

17.3.3 Konstruktoren

OFTimeTriggerDefinition

17.3.4 Methoden

compute

17.4 Klasse ontologyFramework.OFEventManagement.OFTimeTriggerManagement.

17.4.1 Übersicht

This class, as all the class that implements `OFBuilderInterface` has the proposes to initialize classes to be used during system evolution. In this case its initializes Trigger relate to Time, in particular the classes: `OFTimeTriggerDefinition`.

A call to `buildInfo` causes the reset of the initialized classes Map, then all the individual inside the ontological class, named `keyWord[ 0]` (by default: ”OFTimeTrigger”) are processed. Where, The definition of this class must be: `(hasTimeTriggerDefinition exactly 1 TimeTriggerDefinition) and (implementsOFTimeTriggerName exactly 1 Name)`

For all of them it gets the name of the trigger implementation as the fully qualifyer if the class that implement it. This retrieved thanks to the object property named `keyWord[ 1]`, (by default: ”implementsOFTimeTriggerName”). Than, the method retrieve the definiton of the trigger as the string value of the object property named: `keyWord[ 3]` (by default, ”hasTypeTimeTriggerDefinition”) attached to an individual that is linked to this one trhougth the object property named `keyWord[ 2]` (by default: ”hasTimeTriggerDefinition”). Where, the parsing procedure of the text are hinnerated from `OFEventBuilder`. As well as the managament of its parameter are managed by `OFEventParameterDefinition`

The call to the method `getInitialisedClasses` after called `initializeDefinition`} returns a `HashMap<String, OFTimeTriggerDefinition>` where, keys are the names of the individuals belong to the class named `keyWord[ 0]`. While the values are the classes which represent and allow to compute all the temporal triggers available during the calling of `initializeDefinition`.

@author Buoncomapgni Luca

@version 1.0

17.4.2 Inhaltsverzeichnis

17.4.3 Konstruktoren

OFTimeTriggerBuilder

17.4.4 Methoden

buildInfo

getInitialisedClasses

17.5 Interface ontologyFramework.OFEventManagement.OFTimeTriggerManagement.

17.5.1 Übersicht

This class is used to initialise, store and compute Temporal Trigger. `isCorrectInput` is called frist and if it returns true than `getTrigger`} is called with the same inputs. This is by default done from `compute`

@author Buoncomapgni Luca

@version 1.0

17.5.2 Inhaltsverzeichnis

17.5.3 Methoden

isCorrectInput sviluppated with safety pourposes it is called to check if the type of parameter in inputs are correct. If this return false the event result of `getTrigger` will be setted to null.

Parameter

inputs ordered in accord with the ontological definition of the events trhougth the object property `"hasTypeTimeTriggereDefinition`

Rückgabewert true if the inputs are corrects. If return else, event computation dennied.

getTrigger implements how to get the temporal trigger starting from the inputs retrieved in `getParameter`

Parameter

inputs parameter

invoker access to a builded class duriing software initialization

Rückgabewert true f the event occurs, false otherwise.

18 Package ontologyFramework.OFRunning.OFInvokingManag

18.1 Klassen-Liste

18.2 Package-Beschreibung

18.3 Klasse ontologyFramework.OFRunning.OFInvokingManager.ReflactionInstancia

18.3.1 Übersicht

This is a static class which collects common methods to instantiate OFInterfaces using Java Reflection. It use neither generic nor dynamic usage of the Reflection API to decrease the computational complexity.

@author Buoncomapgni Luca

@version 1.0

18.3.2 Inhaltsverzeichnis

18.3.3 Variablen

REFLACTIONDERDEBUG_individualName Name of an ontological individual which must exist belong to the class `DebuggedClass` and has to have an object property `logsDebuggingData exactly 1 Boolean`

18.3.4 Methoden

instanciateOFDataMapperByName Given a class name as string it creates a new instances of `OFDataMapperInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert mapperInst an new instance of the named class which implements `OFDataMapperInterface`.

instanciateOFBuilderByName Given a class name as string it creates a new instances of `OFBuilderInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert builderInst an new instance of the named class which implements `OFBuilderInterface`.

instanciateOFExceptionNotifierByName Given a class name as string it creates a new instances of `OFExceptionNotifierInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert exectNotifyInst an new instance of the named class which implements `OFExceptionNotifierInterface`.

instanciateOFEventParameterByName Given a class name as string it creates a new instances of `OFEventParameterInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert eventParamInst an new instance of the named class which implements `OFEventParameterInterface`.

instanciateOFSynchroniseerManagerByName Given a class name as string it creates a new instances of `OFSynchroniserManagmentInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert synchInst an new instance of the named class which implements `OFSynchroniserManagmentInterface`.

instanciateOFEventByName Given a class name as string it creates a new instances of `OFEventInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert eventInst an new instance of the named class which implements `OFEventInterface`.

instanciateOFTimeTriggrtByName Given a class name as string it creates a new instances of `OFTimeTriggerInterface`, and returns it. It returns null value if an exception is thrown; in this case the message is handled by `OFDebugLogger`

Parameter

packageClassName full class qualifier

Rückgabewert timeTriggetInst an new instance of the named class which implements `OFTimeTriggerInterface`.

18.4 Klasse ontologyFramework.OFRunning.OFInvokingManager.OFBuildedListInvol

18.4.1 Übersicht

This class is the manager of a synchornized `HashMap`. This has String keys equal to the value of the objectProperty named as `BUILDLISTNAME`$_objProp$ for each individuals belong to the ontological class `BUILDER`$_className$. The relative value linked to this unique name is the returning value of the method `getInitialisedClasses` which still is an hashMap with String keys.

It is used to initialize classes during the initialization phase of the framework trough the Interface `OFBuilderInterface`. Than, during system evolution, those classes can be retrieved using this class for further computation.

Access to this class is returned by `OFInitialiser`. Static access to this individual can been done also by name. Since every instances are collected in an static Map by name.

@author Buoncomapgni Luca

@version 1.0

Siehe auch `OFInitialiser`, `OFBuilderInterface`

18.4.2 Inhaltsverzeichnis

18.4.3 Konstruktoren

OFBuildedListInvoker Create a new map and add this instance to the static map of OFBuildedListInvoker. (see `getAllInstances` to track an instance).

Parameter

individualName the name of this instance of this class. If it is null this instance will not be tracked.

18.4.4 Methoden

getInstanceName

Rückgabewert the name of this instances saved in the static map retrievable from `getOFBuildedListInvoker`

addTobuildedList Add a value to the hashMap calling: `buildedList.put( key, ( Map< String, Ob ject>) object)`. It returns false if `buildedList.containsKey( key)`, and does not add the map. By default, during framework initialization keys are the value of the objectProperty named as `BUILDLISTNAME`$_{o}bjProp$ for each individuals belong to the ontological class `BUILDER`$_{c}lassName$. Example : ”MapperList”, ”EventList”...

Parameter

key of the main hasMap

object added to the main hashMap

Rückgabewert key already used flag

addTobuildedList It calls `addTobuildedList`. If overwrite is true the system forces OFBuildedListInvoker to substitute the value to the key if this already exist.

Parameter

key of the main hasMap

object added to the main hashMap

overwrite force system to overwrite if the key already exist

Rückgabewert key already used flag

removeTobuildedList Remove a value to the hashMap calling: `buildedList.remove( key)`. It returns false if buildedList.containsKey(key)!.

Parameter

key of the main hasMap

Rückgabewert operation successful flag

clearbuildedList clears all the builded list calling: `buildedList.clear()`

getMap

Rückgabewert returns the overall builded Map

getMap returns a field of the overall builded Map calling `buildedList.get(key)`.

Parameter

key of the main hasMap

Rückgabewert the object of the map relate to key

getStaticListFromName it calls { @link `getMap`}. and returns its value as an Object.

Parameter

key of the main hasMap

Rückgabewert the object of the map relate to key

getClassFromList It returns the value of `buildedList.get( listName).get( key)`, which the actually builded class during the initialization phase.

Parameter

listName key of the main hasMap

key of the map returned from `getInitialisedClasses`

Rückgabewert the initialized class

addToAllInstances Add an instance of OFBuildedListInvoker to the static HasMap which collect them by `getInstanceName`. This method is used when the instance has been created with a null name and later we want that it appears in the static instance tracked. It adds the instance in according with the name retrieved with: `inv.getInstanceName()`; if this is null the instance inv will no be added.

Parameter

inv instance to track.

getAllInstances

Rückgabewert all the Map where instances of this class are tracked by instance-Name

getOFBuildedListInvoker Returns the instance of this class calling : `return( llInstances.get( ref erenceName))`. If no instance exist with this name than, the methods returns null.

Parameter

referenceName the name of the instance to retrieve

Rückgabewert the instance with the speciefied name.

isInAllInstances It simply uses: `return( allInstances.containsKey( key));`

Parameter

key the instance name

Rückgabewert true if it exist

19 Package ontologyFramework.OFErrorManagement.OFGUI.a

19.1 Klassen-Liste

19.2 Package-Beschreibung

19.3 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.ClassT

19.3.1 Übersicht

This is like TableDemo, except that it substitutes a Favorite Color column for the Last Name column and specifies a custom cell renderer and editor for the color data.

19.3.2 Inhaltsverzeichnis

19.3.3 Konstruktoren

ClassTableLog

19.3.4 Methoden

setTableDimensions

setTableDimensions

getTable

update

saveSelection

restoreSelection

clear

getModel

19.4 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.ClassT

19.4.1 Übersicht

19.4.2 Inhaltsverzeichnis

19.4.3 Konstruktoren

ClassTableLog.MyTableModel

19.4.4 Methoden

getColumnCount

getRowCount

getColumnName

getValueAt

getColumnClass

isCellEditable

setValueAt

19.5 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.Legen

19.5.1 Übersicht

19.5.2 Inhaltsverzeichnis

19.5.3 Konstruktoren

LegendRunner

19.5.4 Methoden

actionPerformed

19.6 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.ColorR

19.6.1 Übersicht

19.6.2 Inhaltsverzeichnis

19.6.3 Konstruktoren

ColorRenderer

19.6.4 Methoden

getTableCellRendererComponent

19.7 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.ClassT

19.7.1 Übersicht

19.7.2 Inhaltsverzeichnis

19.7.3 Konstruktoren

ClassTableInstance

19.7.4 Methoden

setTableDimensions

getModel

mouseReleased

saveSelection

restoreSelection

getTable

update

mouseClicked

mouseEntered

mouseExited

mousePressed

19.8 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.ClassT

19.8.1 Übersicht

19.8.2 Inhaltsverzeichnis

19.8.3 Konstruktoren

ClassTableInstance.MyTableModel

19.8.4 Methoden

getColumnCount

getRowCount

getColumnName

getValueAt

getColumnClass

isCellEditable

setValueAt

19.9 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.Allinst

19.9.1 Übersicht

19.9.2 Inhaltsverzeichnis

19.9.3 Konstruktoren

AllinstancesRunner

19.10 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.Bros

19.10.1 Übersicht

19.10.2 Inhaltsverzeichnis

19.10.3 Konstruktoren

Broswere Create the frame.

19.10.4 Methoden

openBroswer Launch the application.

19.11 Klasse ontologyFramework.OFErrorManagement.OFGUI.allInstancesGUI.Fram

19.11.1 Übersicht

19.11.2 Inhaltsverzeichnis

19.11.3 Variablen

stop

19.11.4 Konstruktoren

FrameworkSerializator

19.11.5 Methoden

stopRun

run

20 Package ontologyFramework.OFProcedureManagment.OFP

20.1 Klassen-Liste

AlgorithmCheckerJob	97
AlgorithmMainJob	98
ClockUpdater	98
OFJobAbstract	98
ReasonerUpdater	99
Cleaner	100

20.2 Package-Beschreibung

20.3 Klasse ontologyFramework.OFProcedureManagment.OFProcedureImplementat

20.3.1 Übersicht

20.3.2 Inhaltsverzeichnis

20.3.3 Konstruktoren

AlgorithmCheckerJob

20.3.4 Methoden

execute

20.4 Klasse ontologyFramework.OFProcedureManagment.OFProcedureImplementat

20.4.1 Übersicht

20.4.2 Inhaltsverzeichnis

20.4.3 Konstruktoren

AlgorithmMainJob

20.5 Klasse ontologyFramework.OFProcedureManagment.OFProcedureImplementat

20.5.1 Übersicht

20.5.2 Inhaltsverzeichnis

20.5.3 Konstruktoren

ClockUpdater

20.6 Klasse ontologyFramework.OFProcedureManagment.OFProcedureImplementat

20.6.1 Übersicht

20.6.2 Inhaltsverzeichnis

20.6.3 Variablen

CONCURRENCYPOOLSYZE_varName

ontoName

invokerName

callerName

procedureName

SYNCHRONIZATION_objPropertyName

20.6.4 Konstruktoren

OFJobAbstract

20.6.5 Methoden

execute

setRunning

setScheduled

getConcurrencePoolSize

Rückgabewert the concurrencePoolSize

getOWLOntologyRefeferences

Rückgabewert the ontoRef

getProcedureIndividualName

Rückgabewert the procedureIndividualName

getInvoker

Rückgabewert the invoker

getAlgorithmCaller

Rückgabewert the algorithmCaller

getAlgorithmInstanceNameBase

Rückgabewert the algorithmInstanceNameBase

getAlgorithmInstanceName

Rückgabewert the algorithmInstanceName

addLogStrign

addLogStrign

20.7 Klasse ontologyFramework.OFProcedureManagment.OFProcedureImplementat

20.7.1 Übersicht

20.7.2 Inhaltsverzeichnis

20.7.3 Konstruktoren

ReasonerUpdater

20.8 Klasse ontologyFramework.OFProcedureManagment.OFProcedureImplementat

20.8.1 Übersicht

20.8.2 Inhaltsverzeichnis

20.8.3 Konstruktoren

Cleaner

21 Package ontologyFramework.OFContextManagement

21.1 Klassen-Liste

21.2 Package-Beschreibung

21.3 Klasse ontologyFramework.OFContextManagement.OWLLibrary

21.3.1 Übersicht

This static class implement several common procedure for manipulating entity inside an ontology, using OWL api 3.0

@author Buoncomapgni Luca

@version 1.0

21.3.2 Inhaltsverzeichnis

21.3.3 Variablen

PELLET_reasonerFactoryQualifier Full qualifier of the Pellet reasoner Factory. String to be called by Java reflection to instantiate a Reasoner.

SNOROCKET_reasonerFactoryQualifier Full qualifier of the Snorocket reasoner Factory. String to be called by Java reflection to instantiate a Reasoner.

HERMIT_reasonerFactoryQualifier Full qualifier of the Hermit reasoner Factory. String to be called by Java reflection to instantiate a Reasoner.

FACTPLUSPLUS_reasonerFactoryQualifier Full qualifier of the Fact++ reasoner Factory. String to be called by Java reflection to instantiate a Reasoner.

21.3.4 Methoden

createOntologyManager creates and returns a new OWLOntologyManager. If the parameter has a not null `getIriFilePath` and `getOntologyPath` that this method set this Iri mapper to the manager using: `manager.addIRIMapper( new SimpleIRIMapper( ontoPath, filePath))`

Parameter

OWLReferences reference to the ontology.

Rückgabewert the manager of the ontology refereed by the parameter.

createOntology Creates an new empty ontology in accord to the `getIriOntologyPath`. It will return null if the ontology path associate to the parameter is null.

Parameter

OWLReferences reference to the ontology.

Exceptions

OWLOntologyCreationException

Rückgabewert a new empty ontology in accord with the parameter.

loadOntologyFromFile It loads an ontology from file in accord with the function parameter; to do so the method uses the ontology manager from: `getManager`. It will return null if `getIriOntologyPath` is null.

Parameter

OWLReferences reference to the ontology.

Exceptions

OWLOntologyCreationException

Rückgabewert a pointer to the ontology refered by the parameter,

loadOntologyFromWeb It loads an ontology where its `getIriOntologyPath` defines an path to be browsed into the web. It returns null if the IRI ontology Path is null.

Parameter

ontoRef reference to the ontology.

Exceptions

OWLOntologyCreationException

Rückgabewert a pointer to the ontology refered by the parameter,

getPrefixFormat Returns a prefix manager to be attached into an ontolofy manager to simplify IRI definition and usage

Parameter

OWLReferences a reference to an OWL ontology.

Rückgabewert a prefix manager format.

getOWLDataFactory Returns the OWLDataFactory associate to the OWLManager associate to the parameter.

Parameter

OWLReferences

Rückgabewert an OWL data factory

getReasoner It creates and returns a Reasoner instance. The type of reasoner is defined by the reasoner name factory, which could be: `PELLET`$_reasonerFactoryQualifier$, `SNOROCKET`$_reasonerFactoryQualifier$, `HERMIT`$_reasonerFactoryQualifier$ or `FACTPLUSPLUS`$_reas$ The created reasoner, will be attached to the ontology references given as parameter. If buffering flag is true than the reasoner will update its state only if `reasoner.flush()` is called. Otherwise this reasoner will synchronizes itself at any ontological changes. The system will return null if a Reflaction error occurs in instancing the class defined by the parameter reasonerFactoryName.

Parameter

reasonerFactoryName full qualifier to the reasoner factory.

ontoRef references to the OWL ontology.

buffering flag.

Rückgabewert a new instance to the specified reasoner.

getPelletReasoner Returns an instance of the Pellet reasoner. If buffering is true than the reasoner is update only when `reasoner.flush()` is called. Otherwise returns a reasoner which synchronizes itself at any ontological changes.

Parameter

ontoRef references to the OWL ontology

buffering flag

Rückgabewert a new Pellet reasoner instance

getSnorocketReasoner Returns an instance of the Snorocket reasoner. If buffering is true than the reasoner is update only when `reasoner.flush()` is called. Otherwise returns a reasoner which synchronizes itself at any ontological changes.

Parameter

ontoRef references to the OWL ontology

buffering flag

Rückgabewert a new Snorocket reasoner instance

getHermitReasoner Returns an instance of the Hermit reasoner. If buffering is true than the reasoner is update only when `reasoner.flush()` is called. Otherwise returns a reasoner which synchronizes itself at any ontological changes.

Parameter

ontoRef references to the OWL ontology

buffering flag

Rückgabewert a new Hermit reasoner instance

getFactReasoner Returns an instance of the Fact++ reasoner. If buffering is true than the reasoner is update only when `reasoner.flush()` is called. Otherwise returns a reasoner which synchronizes itself at any ontological changes.

Parameter

ontoRef references to the OWL ontology

buffering flag

Rückgabewert a new Fact++ reasoner instance

printOntonolyOnConsole It prints the ontology over console using Manchester formatting.

Parameter

ontoRef reference to an OWL ontology

Exceptions

OWLOntologyStorageException

synchroniseReasoner If the Ontology is consistent it will synchronize a buffering reasoner calling `reasoner.flush()`; if the reasoner has a false buffering flag, than this method has no effects. If an inconsistency error occurs than this method will print over console an explanation of the error. Note that if the ontology is inconsistent than all the methods in this class may return a null value.

Parameter

ontoRef references to an OWL ontology.

getOWLClass Returns an Object which represents an ontological class with a given name and specifics IRI paths. If the entity already exists in the entology than the object will refer to it, otherwise the method will create a new ontological entity.

Parameter

className string to define the name of the ontological class

ontoRef reference to an OWL ontology.

Rückgabewert the OWL class with the given name and IRI paths in accord to the OWLReference

getOWLIndividual Returns an Object which represents an onological individual with a given name and specific IRI paths. If the entity already exists in the entology than the object will refer to it, otherwise the method will create a new ontological entity.

Parameter

individualName string to define the name of the ontological individual

ontoRef reference to an OWL ontology.

Rückgabewert the OWL individual with the given name and IRI paths in accord to the OWLReference

getOWLDataProperty Returns an Object which represents an onological data property with a given name and specific IRI paths. If the entity already exists in the entology than the object will refer to it, otherwise the method will create a new ontological entity.

Parameter

dataPropertyName string to define the name of the ontological data property

ontoRef reference to an OWL ontology.

Rückgabewert the OWL data property with the given name and IRI paths in accord to the OWLReference

getOWLObjectProperty Returns an Object which represents an onological object property with a given name and specific IRI paths. If the entity already exists in the entology than the object will refer to it, otherwise the method will create a new ontological entity.

Parameter

objPropertyName string to define the name of the ontological object property

ontoRef reference to an OWL ontology.

Rückgabewert the OWL object property with the given name and IRI paths in accord to the OWLReference

getOWLLiteral Returns an Object which represents an onological literal with a given value and specific IRI paths. Indeed it calls: `OWLLibrary.getOWLLiteral( value, null, ontoRef)`.

Parameter

> **value** object to define the value of the ontological literal
>
> **ontoRef** reference to an OWL ontology.

Rückgabewert the OWL literal with the given value, type and IRI paths in accord to the OWLReference

getOWLLiteral Given an Object value this method returns the OWLLiteral in accord with the actual type of value. The parameter Type can be null if value is of type: String, Integer, Boolean, Float, Long; otherwise this method will returns null. For more specific data type this methods require to give in input the rigth OWLDataType parameter. Generally it will return null if the data type of the parameter value is unknown.

Parameter

> **value** object to define the value of the ontological literal
>
> **type** the OWL data type to define the literal
>
> **ontoRef** reference to an OWL ontology.

Rückgabewert the OWL literal with the given value, type and IRI paths in accord to the OWLReference

getIndividualB2Class It returns all the ontological individual which are defined in the refereed ontology and which are belonging to the calss with name defined by the parameter. Indeed this method will call `getOWLClass`, to get the actual OWL class Object and than it use it to call `getIndividualB2Class`. Than the returning value is propagated, so it returns null if no individual are classified in that class or if such class does not exist in the refereed ontology.

Parameter

> **className** name of the ontological calss
>
> **ontoRef** reference to an OWL ontology.

Rückgabewert an not ordered set of individual belong to such class.

getIndividualB2Class It returns all the ontological individual which are defined in the refereed ontology and which are belonging to the calss defined by the parameter. It returns null if no individual are classified in that class or if such class does not exist in the refereed ontology.

Parameter

> **ontoClass** OWL class for which the individual are asked.
>
> **ontoRef** reference to an OWL ontology.

Rückgabewert an not ordered set of individual belong to such class.

getOnlyIndividualB2Class It returns one ontological individual which are defined in the refereed ontology and which are belonging to the calss with name defined by the parameter. Indeed this method will call `getOWLClass`, to get the actual OWL class Object and than it use it to call `getIndividualB2Class`. Than, using `getOnlyElement` it will return one individual that are belongign to the class. It returns null if no individual are classified in that class, if such class does not exist in the refereed ontology or if the individual set returned by `OWLLibrary.getIndividualB2Class( .. )` has `size > 1`.

Parameter

className name of the ontological calss

ontoRef reference to an OWL ontology.

Rückgabewert an individual belong to such class.

getOnlyIndividualB2Class It returns an ontological individual which are defined in the refereed ontology and which are belonging to the calss defined by the parameter. It returns null if no individual are classified in that class, if such class does not exist in the refereed ontology or if there are more than one individual classified in that class (since it uses `getOnlyElement`).

Parameter

ontoClass OWL class for which the individual are asked.

ontoRef reference to an OWL ontology.

Rückgabewert an individual belong to such class.

getIndividualClasses It returns the set of classes in which an individual has been classified.

Parameter

individual ontological individual object

ontoRef reference to an OWL ontology.

Rückgabewert a not ordered set of all the classes where the individual is belonging to.

getDataPropertyB2Individual Returns the set of literal value relate to an OWL Data Property which has a specific name and which is assign to a given individual. Indeed it retrieves OWL object from strings and calls: `getDataPropertyB2Individual`. Than its returning value is propagated.

Parameter

individualName name to the ontological individual belonging to the refering ontology

propertyName data property name applied to the ontological individual belonging to the refering ontology

ontoRef reference to an OWL ontology.

Rückgabewert a not ordered set of literal value of such property applied to a given individual

getDataPropertyB2Individual Returns the set of literal value relate to an OWL Data Property and assigned to a given individual. It returns null if such data property or individual doesn not exist. Also if the individual has not such proprerty.

Parameter

individual the OWL individual belonging to the refering ontology

property the OWL data property applied to the ontological individual belonging to the refering ontology

ontoRef reference to an OWL ontology.

Rückgabewert a not ordered set of literal value of such property applied to a given individual

getOnlyDataPropertyB2Individual Returns one literal value attached to a given individual througth a specific data property. Here both, individual and property, are given by name, than the system calls `getOnlyDataPropertyB2Individual` and its returning value is used with `getOnlyElement`.

Parameter

individualName name to the ontological individual belonging to the refering ontology

propertyName data property name applied to the ontological individual belonging to the refering ontology

ontoRef reference to an OWL ontology.

Rückgabewert a literal value of such property applied to a given individual

getOnlyDataPropertyB2Individual Returns one litteral value attached to a given OWL individual througth an OWL data property. This returns null if `getDataPropertyB2Individual` or `getOnlyElement` return null.

Parameter

individual the OWL individual belonging to the refering ontology

property the OWL data property applied to the ontological individual belonging to the refering ontology

ontoRef reference to an OWL ontology.

Rückgabewert a literal value of such property applied to a given individual

getObjectPropertyB2Individual Returns all the values (individuals) to an Object property, given by name, linked to an individual, given by name as well. Indeed it retrueve the OWL Objects by name using `getOWLObjectProperty` and `getOWLIndividual`. Than it calls `getObjectPropertyB2Individual` propagating its returning value.

Parameter

individualName the name of an ontological individual

propertyName the name of an ontological object property

ontoRef reference to an OWL ontology.

Rückgabewert a not ordered set of all the values (OWLNamedIndividual) that the individual has w.r.t. such object property.

getObjectPropertyB2Individual Returns all the values (individuals) to an Object property, given by name, linked to an individual, given by name as well. It will return null if such object property or individual does not exist.

Parameter

individual an OWL individual

property an OWL object property

ontoRef reference to an OWL ontology.

Rückgabewert a not ordered set of all the values (OWLNamedIndividual) that the individual has w.r.t. such object property.

getOnlyObjectPropertyB2Individual Returns a value (individual) to an Object property, given by name, linked to an individual, given by name as well. Indeed it retrueve the OWL Objects by name using `OWLLibrary` and `getOWLIndividual`. Than it calls `getObjectPropertyB2Individual` and its returning value is used to call `getOnlyElement` which define the actual returning value of this method.

Parameter

individualName the name of an ontological individual

propertyName the name of an ontological object property

ontoRef reference to an OWL ontology.

Rückgabewert a value (OWLNamedIndividual) that the individual has w.r.t. such object property.

getOnlyObjectPropertyB2Individual Returns a value (individual) to an Object property, given by name, linked to an individual, given by name as well. It will return null if such object property or individual does not exist. Finally it can return null if `getOnlyElement` returns null.

Parameter

individual an OWL individual

property an OWL object property

ontoRef reference to an OWL ontology.

Rückgabewert a value (OWLNamedIndividual) that the individual has w.r.t. such object property.

getSubClassOf Returns all the classes that are sub classes of the given parameter. Here class is defined by name, so this method uses: `getOWLClass` to get an OWLClass and than it calls `getSubClassOf` propagating its returning value.

Parameter

className name of the ontological class to find sub classes

ontoRef reference to an OWL ontology.

Rückgabewert a not order set of all the sub classes of cl parameter.

getSubClassOf Returns all the classes that are sub classes of the given class parameter. It returns null if no sub classes are defined in the ontology.

Parameter

cl OWL class to find sub classes

ontoRef reference to an OWL ontology.

Rückgabewert a not order set of all the sub-classes of cl parameter.

getSuperClassOf Returns all the classes that are super classes of the given parameter. Here class is defined by name, so this method uses: `getOWLClass` to get an OWLClass and than it calls `getSuperClassOf` propagating its returning value.

Parameter

className name of the ontological class to find super classes

ontoRef reference to an OWL ontology.

Rückgabewert a not order set of all the super classes of cl parameter.

getSuperClassOf Returns all the classes that are super classes of the given class parameter. It returns null if no super classes are defined in the ontology.

Parameter

cl OWL class to find super classes

ontoRef reference to an OWL ontology.

Rückgabewert a not order set of all the super classes of cl parameter.

setSubClassOf Set the parameter subClass to be a sub class of the parameter superClass. Here classes are given by name, the method uses `getOWLClass` to cal `setSubClassOf` and propagate its returning value.

Parameter

superClassName the name of the ontological super class

subClassName the name of the ontological sub class

ontoRef reference to an OWL ontology.

Rückgabewert an ontologial axiom to describe this hyerarchly dependece between classes.

setSubClassOf Set the parameter subClass to be a sub class of the parameter superClass. Here classes are given by name, the method uses `getOWLClass` to cal `setSubClassOf` and propagate its returning value. If the boolean value addAxiom is true, than the axioms to add to describe those dependencies are stored in an internal buffer. it will be not added to the buffer if it is false.

Parameter

superClassName the name of the ontological super class

subClassName the name of the ontological sub class

addAxiom flag to store the adding axioms into a buffer managed in this class.

ontoRef reference to an OWL ontology.

Rückgabewert an ontologial axiom to describe this hyerarchly dependece between classes.

setSubClassOf Set the parameter subClass to be a sub class of the parameter superClass. Here classes are given by name, the method uses `getOWLClass` to cal `setSubClassOf` and propagate its returning value. If the boolean value addAxiom is true, than the axioms to add to describe those dependencies are stored in an internal buffer. it will be not added to the buffer if it is false. On the other hand if the parameter applyChanges is true than those changes are also immidiately apllied, otherwise a call to apply them is required.

Parameter

superClassName the name of the ontological super class

subClassName the name of the ontological sub class

addAxiom flag to store the adding axioms into a buffer managed in this class.

applyChanges flag to decide if applyng those change immediatly or not.

ontoRef reference to an OWL ontology.

Rückgabewert an ontologial axiom to describe this hyerarchly dependece between classes.

setSubClassOf Set the parameter subClass to be a sub class of the parameter superClass.

Parameter

superClass the OWL super class

subClass the OWL sub class

ontoRef reference to an OWL ontology.

Rückgabewert an ontologial axiom to describe this hyerarchly dependece between classes.

setSubClassOf Set the parameter subClass to be a sub class of the parameter superClass. If addAxiom flag is true than, this axioms will be stored inside an internal buffer, otherwise no. This is done by calling `getAddAxiom`.

Parameter

superClass the OWL super class

subClass the OWL sub class

addAxiom flag to store the adding axioms into a buffer managed in this class. * @param ontoRef

ontoRef reference to an OWL ontology.

Rückgabewert an ontologial axiom to describe this hyerarchly dependece between classes.

setSubClassOf Set the parameter subClass to be a sub class of the parameter superClass. If addAxiom flag is true than, this axioms will be stored inside an internal buffer, otherwise no. This is done by calling `getAddAxiom`. On the other hand if applyChanges id true than the changes are immediately moved into the otology, otherwise no. This is done by calling `applyChanges`.

Parameter

superClass the OWL super class

subClass the OWL sub class

addAxiom flag to store the adding axioms into a buffer managed in this class.

applyChanges flag to decide if applyng those change immediatly or not.

ontoRef reference to an OWL ontology.

Rückgabewert an ontologial axiom to describe this hyerarchly dependece between classes.

getAddAxiom It returns a list of ontology changes to be done to build a given axiom into the ontology. Indeed it calls: `getAddAxiom` with the flag value always set to `true`.

Parameter

axiom to describe relationsheps between ontological entities.

ontoRef reference to an OWL ontology.

Rückgabewert the order set of changes to build a given axiom.

getAddAxiom It returns a list of ontology changes to be done to build a given axiom into the ontology. If the flag `addToChangeList` is true than those changes will be stored inside an internul buffer, otherwise no.

Parameter

axiom o describe relationsheps between ontological entities.

addToChangeList flag to decide if add them into the internal buffer of changes

ontoRef reference to an OWL ontology.

Rückgabewert the order set of changes to build a given axiom.

getRemoveAxiom It returns a list of ontology changes to be done to remove a given axiom from the ontology. Indeed it calls: `getRemoveAxiom` with the flag value always set to `true`.

Parameter

axiom to describe relationsheps between ontological entities.

ontoRef reference to an OWL ontology.

Rückgabewert the order set of changes to remove a given axiom.

getRemoveAxiom It returns a list of ontology changes to be done to remove a given axiom from the ontology. If the flag `addToChangeList` is true than those changes will be stored inside an internul buffer, otherwise no.

Parameter

axiom o describe relationsheps between ontological entities.

addToChangeList flag to decide if add them into the internal buffer of changes

ontoRef reference to an OWL ontology.

Rückgabewert the order set of changes to remove a given axiom.

applyChanges It applies all the changes and axioms stored in the internal buffer into the ontology. After its work, it will clean up this buffer.

Parameter

ontoRef reference to an OWL ontology.

applyChanges It applies, into the ontology, only the change given as parameter.

Parameter

addAxiom change to apply in the ontology

ontoRef param ontoRef reference to an OWL ontology.

applyChanges It applies, into the ontology, all the changes given as parameter.

Parameter

addAxiom list of ontological changes.

ontoRef reference to an OWL ontology.

getSubObjectProperty Return all the sub object property relate do a property given as an input parameter. Indeed, it retrieve the object from their names using `getOWLObjectProperty`. Than it calls `getSubObjectProperty` and propagate its returning value.

Parameter

objectPropName the name of the ontological object property to check for its sub property

ontoRef reference to an OWL ontology.

Rückgabewert an unordered set of Expression to define this hyererchly relations.

getSubObjectProperty eturn all the sub object property relate do a property given as an input parameter.It can return null if no sub object property are defined into the ontology for the input parameter.

Parameter

objectProp the OWL object property to check for its sub property

ontoRef reference to an OWL ontology.

Rückgabewert an unordered set of Expression to define this hyererchly relations.

getOnlyElement Its pourpuses is to be used when an entity of the ontology can have only one element by construction. In particolar this method returns true `if( set.size() > 1)`. Otherwise it will iterate over the set and return just the first value. Note that a set does not guarantee that its order is always the same.

Parameter

set a generic set of object

Rückgabewert an element of the set

getOnlyString Its pourpuses is to be used when an entity of the ontology can have only one literal by construction. In particolar this method returns true `if( set.size() > 1)`. Otherwise it will iterate over the set and return just the first value. Note that a set does not guarantee that its order is always the same.

Parameter

set set of literals

Rückgabewert an element of the set as a string

renameEntity It returns the changes that must be done into the ontology to rename an entity, they should be applied by calling `applyChanges(renameChanges, ontoRef)`.

Parameter

entity ontological object to rename

newIRI new name as ontological IRI path

ontoRef reference to an OWL ontology.

Rückgabewert the changesa to be apllyed into the ontology to rename an entity with a new IRI.

renameEntity It returns the changes that must be done into the ontology to rename an entity. If the flag appyChanges is true than the entity will be immediately renamed into the ontology.

Parameter

entity ontological object to rename

newIRI new name as ontological IRI path

applyChanges flag to apply immediatly those changes,

ontoRef reference to an OWL ontology.

Rückgabewert the changesa to be apllyed into the ontology to rename an entity with a new IRI.

getOWLObjectName It uses a render defined as `OWLObjectRenderer renderer = new DLSyntaxObjectRenderer();` to get the name of an ontological object from its IRI path. It returns null if the input parametere is null.

Parameter

o the objet for which get the ontological name

Rückgabewert the name of the ontological object given as input parameter.

getOWLSetAsString Returns a set of names given a set of ontological objects. It uses a renderer: `OWLObjectRenderer renderer = new DLSyntaxObjectRenderer();` to do this work. The inoput set cannot have null value. It will riturn null if the input set is empty.

Parameter

set of ontological object from which retrieve names.

Rückgabewert set of names of the object contained in the input parameter.

getOWLLiteralAsString It converts all the literal inside a set into a set of string using `literal.getLiteral()`. The input set cannot contain null values. This method returns null if the input set is empty.

Parameter

set set of ontological individual.

Rückgabewert the input set converterd to string.

saveOntology It will save an ontology into a file. The files path is retrieved from the OWLReferences class using: `ontoRef.getIriFilePath();`. Note that this procedure may replace an already existing file. The exporting of the asserted relation is done by: `exportOntology` and my be an expencive procedure.

Parameter

exportInf flag to export inferences as fixed relations.

ontoRef reference to an OWL ontology.

saveOntology It will save an ontology into a file. The files path is given as input parameter, and this method does not update: `ontoRef.getIriFilePath();`. Note that this procedure may replace an already existing file. The exporting of the asserted relation is done by: `exportOntology` and my be an expencive procedure.

Parameter

exportInf flag to export inferences as fixed relations.

filePath directiory in which save the ontology.

ontoRef reference to an OWL ontology.

addObjectPropertyB2Individual Returns a list of changes to be applied into the ontology to add a new object property (with its value) into an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

ind individual that have to have a new object property.

prop object property to be added.

value individual which is the value of the given object property.

bufferize flag to bufferize changes inside an internal buffer.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to add this specific object property.

addObjectPropertyB2Individual Returns a list of changes to be applied into the ontology to add a new object property (with its value) into an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology. Indeed it retrieve the ontological object from name and than it calls: `addObjectPropertyB2Individual`

Parameter

individualName tha name of an ontological individual that havo to have a new object property

propName name of the object property inside the ontology refered by ontoRef.

valueName individual name inside te refereed ontology to be the value of the given object property

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to add this specific object property.

addDataPropertyB2Individual Returns a list of changes to be applied into the ontology to add a new datat property (with its value) into an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

ind individual that have to have a new data property.

prop data property to be added.

value literal which is the value of the given data property.

bufferize flag to bufferize changes inside an internal buffer.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to add this specific data property.

addDataPropertyB2Individual Returns a list of changes to be applied into the ontology to add a new data property (with its value) into an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology. Indeed it retrieve the ontological object from name and than it calls: `addDataPropertyB2Individual`

Parameter

individualName tha name of an ontological individual that havo to have a new data property

propertyName name of the data property inside the ontology refered by ontoRef.

value literal to be added as the value of a data property.

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to add the given data property.

addIndividualB2Class Returns the ontological changes to be applyed to put an individual inside an ontological class. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

ind individual to add into an ontological class

c ontological class that than will conteind this individual

bufferize flag to bufferize changes inside an internal buffer.

ontoRef reference to an OWL ontology.

Rückgabewert changes to be done into the refereed ontology to set an individual to be belonging to a specific class.

addIndividualB2Class Returns a list of changes to be applied into the ontology to set an individual to belonging to a class. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology. Indeed it retrieve the ontological object from name inside the refering ontology and than it calls: `addIndividualB2Class`

Parameter

individualName tha name of an ontological individual that have to be beloging to a given class.

className the name of an ontological class that will contains the input individual parameter.

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to set an individual belong to a class.

removeObjectPropertyB2Individual Returns a list of changes to be applied into the ontology to remove an object property (with its value) from an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

ind individual from which remove a given object property.

prop object property to be removed.

value individual which is the value of the given object property.

bufferize flag to bufferize changes inside an internal buffer.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to remove this specific object property.

removeObjectPropertyB2Individual Returns a list of changes to be applied into the ontology to remove a given object property (with its value) from an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology. Indeed it retrieve the ontological object from name and than it calls: `removeObjectPropertyB2Individual`

Parameter

individualName tha name of an ontological individual from which remove the object property

propName name of the object property inside the ontology refered by ontoRef.

valueName individual name inside te refereed ontology to be the value of the given object property

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to remove this specific object property.

removeDataPropertyB2Individual Returns a list of changes to be applied into the ontology to remove datat property (with its value) from an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

ind individual from which remove the given data property.

prop data property to be removed.

value literal which is the value of the given data property.

bufferize flag to bufferize changes inside an internal buffer.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to remove this specific data property.

removeDataPropertyB2Individual Returns a list of changes to be applied into the ontology to remove a data property (with its value) from an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology. Indeed it retrieve the ontological object from name and than it calls: `removeDataPropertyB2Individual`

Parameter

individualName tha name of an ontological individual from which remove the data property

propertyName name of the data property inside the ontology refered by ontoRef.

value literal to be removed as the value of a data property.

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to remove this specific data property.

removeIndividualB2Class Returns the ontological changes to be applyed to remove an individual from an ontological class. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

ind individual to remove from an ontological class

c ontological class that than was conteind this individual

bufferize flag to bufferize changes inside an internal buffer.

ontoRef reference to an OWL ontology.

Rückgabewert changes to be done into the refereed ontology to set an individual to not be anymore belonging to a specific class.

removeIndividualB2Class Returns a list of changes to be applied into the ontology to remove an individual to belonging to a class. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology. Indeed it retrieve the ontological object from name inside the refering ontology and than it calls: `removeIndividualB2Class`

Parameter

individualName tha name of an ontological individual that have not to be beloging to a given class.

className the name of an ontological class that will no more contains the input individual parameter.

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert the changes to be done into the refereed ontology to set an individual to do not belong to a class anymore.

removeIndividual Returns the changes to be apllied into the refering ontology for removing an individual. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

individual to be removed from the ontology.

bufferize flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert tha changes to be done into the refereed ontology to remove a given individual.

removeIndividual Returns the changes to be applied into the refering ontology for removing a set of individuals. If the bufferize flag is true than those changes will be saved inside at the internal buffer of this class which can be applied by calling: `applyChanges`. If this flag is false than only this changes will be immediately applied to the refering ontology.

Parameter

individuals set of individuals to be removed.

bufferised flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

Rückgabewert tha changes to be done into the refereed ontology to remove a given set individuals.

replaceDataProperty Atomically (with respect to reasoner update) replacing of a data property. Indeed, it will remove all the possible data property with a given values using `removeDataPropertyB2Individual`. Than, it add the new value calling `addDataPropertyB2Individual`. Refer to this last two for how the flag bufferized is used.

Parameter

ind individual for which a data property will be replaced.

prop property to replace

oldValue set of old values to remove

newValue new value to add

buffered flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

replaceDataProperty Atimically (with respect to reasoner update) replacing of a data property. Indeed, it will remove the possible data property with a given value using `removeDataPropertyB2Individual`. Than, it add the new value calling `addDataPropertyB2Individual`. Refer to this last two for how the flag bufferized is used.

Parameter

ind individual for which a data property will be replaced.

prop property to replace

oldValue value to remove

newValue new value to add

buffered flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

replaceObjectProperty Atomically (with respect to reasoner update) replacing of a object property. Indeed, it will remove the possible object property with a given values using `removeObjectPropertyB2Individual`. Than, it add the new value calling `addObjectPropertyB2Individual`. Refer to this last two for how the flag bufferized is used.

Parameter

ind individual for which a object property will be replaced.

prop property to replace

oldValue set of old values to remove

newValue new value to add

buffered flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

replaceIndividualClass Atomically (with respect to reasoner update) replacing of individual type. Which means to remove an individual from a class and add it to belong to another calss. Indeed, it will remove the possible type with a given values using `removeIndividualB2Class`. Than, it add the new value calling `addIndividualB2Class`. Refer to this last two for how the flag bufferized is used.

Parameter

ind individual to change its classification.

oldValue old class in which the individual is belonging to

newValue new class in which the individual will belonging to

buffered flag to buffering changes internally to this class.

ontoRef reference to an OWL ontology.

getPelletExplanation It uses Manchester syntax to explain possible inconsistencies.

Parameter

ontoRef reference to an OWL ontology.

Rückgabewert an inconcistency explanation as a string of text.

21.4 Klasse ontologyFramework.OFContextManagement.OWLReferences

21.4.1 Übersicht

This class define a complete reference to a OWL ontology. In particular, it should be create to introduce an ontology into the framework. This class is compatible in all the part of the framework and it helps in moving ontologies through the computational flow.

@author Buoncomapgni Luca

@version 1.0

21.4.2 Inhaltsverzeichnis

21.4.3 Variablen

CREATEcommand Value to describe the create ontology command during class construction. It will create a new ontology as a file considering a given filePath.

LOADFROMFILEcommand Value to describe the load from file command during class construction It will load an ontology w.r.t. filePath and ontoPath.

LOADFROMWEBcommand Value to describe the load ontology from web command during class construction. In this case filePath will be set to `null`.

21.4.4 Konstruktoren

OWLReferences Constructor to resume this class from its serialization variable. It can be retrieved using `getSerialisableData`. This is done since the reasoner and the ontology are not serializable trougth the interface `Serializable`

Parameter

serial

OWLReferences Create a new references to an ontology using the standard reasoner. By default it is set to Pellet reasoner with a buffering synchronisation.

Parameter

ontologyName name of this OWLReferences instances, used to refer to this instance.

filePath IRI path to the file where the ontology is stored.

ontologyPath IRI path of the ontology.

command value to define the create or load from file or web comand.

OWLReferences Create a new references to an ontology.

Parameter

ontologyName name of this OWLReferences instances, used to refer to this instance.

filePath IRI path to the file where the ontology is stored.

ontologyPath IRI path of the ontology.

reasonerInstance instance to the reasoner to attach to this ontology.

command value to define the create or load from file or web comand.

21.4.5 Methoden

setPelletReasoner Set the reasoner attached to this ontology as Pellet. If @param buffering is true than the ontology buffers changes that will be synchronized by the reasoner in one call to `reasoner.flush();`. If it is false than the reasoner will be synchronized at every changes of the ontology structure.

Parameter

buffering flag to set a buffering, or not buffering Pellet.

setHermitReasoner Set the reasoner attached to this ontology as Hermit. If @param buffering is true than the ontology buffers changes that will be synchronized by the reasoner in one call to `reasoner.flush();`. If it is false than the reasoner will be synchronized at every changes of the ontology structure.

Parameter

buffering reasoner buffering flag

setSnorocketReasoner Set the reasoner attached to this ontology as Snorocket. If @param buffering is true than the ontology buffers changes that will be synchronized by the reasoner in one call to `reasoner.flush();`. If it is false than the reasoner will be synchronized at every changes of the ontology structure.

Parameter

buffering reasoner buffering flag

setFactReasoner Set the reasoner attached to this ontology as Fact++. If @param buffering is true than the ontology buffers changes that will be synchronized by the reasoner in one call to `reasoner.flush();`. If it is false than the reasoner will be synchronized at every changes of the ontology structure.

Parameter

buffering reasoner buffering flag

isBufferingReasoner Returns true if the reasoner has a buffering synchronisation. false if the reasoner is apdated at every ontological changes.

Rückgabewert the bufferingReasoner flag

getOntoName Every OWLReferences has only one ontology attached to it. Moreover, every of them has a name used to statically refer to the different OWLReferences inside the framework. This name must be different for every OWLReferences since they are stored in an `HashMap<String, OWLReferences>`.

Rückgabewert the name of the OWLReferences.

getIriFilePath Returns the IRI path to the file where the ontology is.

Rückgabewert the iriFilePath

getFilePath Returns the IRI path to the file where the ontology is, as a String.

Rückgabewert the iriFilePath as a String

setFilePath Set the file Path of the ontology refereed by this instance. As long as is not the case in which an ontology should be locally saved from web is recommended to set this variable in during class constructors.

Parameter

filePath the filePath to set.

getUsedCommand Returns the command used to initialize this instance.

Rückgabewert the used command.

getOntologyPath Returns the IRI path associate to the ontology refereed by this class as a String.

Rückgabewert the IRI ontology Path as a String

getIriOntologyPath Returns the IRI path associate to the ontology refereed by this class.

Rückgabewert the IRI ontology Path

getManager Returns the OWL manager associate only to the ontology refereed by this instance.

Rückgabewert the OWL manager

getFactory Returns the OWL data factory, used to get object used for ontological changes.

Rückgabewert the factory

setReasoner Set an external reasoner instance to this ontology. The old relation to a reasoner will be deleted.

Parameter

reasoner the reasoner to set

getOntology Get the ontology refereed by this class.

Rückgabewert the ontology

getPm Returns a prefix manager to semply the IRI representation.

Rückgabewert the prefix manager

getReasoner

Rückgabewert the reasoner instance associate to this ontology

getSerialisableData Serialize this class saving important quantities and using a special constructor: `OWLReferences`. Basically, it calls `getSerialisableData` with input parameter `filePath = this.getIriFilePath()`.

Rückgabewert serializable Data relate to this OWLReferences

getSerialisableData Serialize this class saving important quantities and using a special constructor: `OWLReferences`.

Parameter

filePath new file Path for the serializated class

Rückgabewert serializable Data relate to this OWLReferences

getAllInstances Returns a map that contains all the instances of OWLReferences class create into the framework. Instances are organized w.r.t the ontoName attached to them

Rückgabewert Map between ontoName and OWKReferences

getOWLReferences Return a particular OWLReferences, given its ontoName. Basically it just calls: `return( this.getAllInstances().get(referenceName))`.

Parameter

referenceName the name attached to a particolar OWLReferences (ontoName).

Rückgabewert the instance of this class attached to a particular name

isInAllInstances check if exist an OWLReferences with a particolar name already stored in the Map (`getAllInstances`). Basically it just calls: `return( this.getAllInstance.cont`

Parameter

key ontoName used to store a OWLReferences.

Rückgabewert true if it exist, false otherwise.

checkConsistent call the reasoner to check ontology consistency and synchronizes the consistency flag

getAllSerializableInstances It goes trough all the named instances of this class and for each of them calls `getSerialisableData`. Note that to properly serialize an OWLOntology that has to be saved in owl format.

Parameter

basePath folder path in which save the ontologies

nameToSerialize name of the ontologies to save. If this is null then all keys of `getAllInstances` will be considered.

exportInfer if true all the asserted axiom will be exported in the Ontology

Rückgabewert the map of serializable data.

21.5 Klasse ontologyFramework.OFContextManagement.OFReasonerProgressMonit

21.5.1 Übersicht

21.5.2 Inhaltsverzeichnis

21.5.3 Konstruktoren

OFReasonerProgressMonitor

21.5.4 Methoden

setReasonerName

reasonerTaskStarted

reasonerTaskStopped

reasonerTaskProgressChanged

reasonerTaskBusy

21.6 Klasse ontologyFramework.OFContextManagement.OWLReferencesSerializable

21.6.1 Übersicht

@author Buoncomapgni Luca

@version 1.0

get the essential data to load again a specific OWLOnotology using the class { @link OWLReferences}. Actual serialization of ontology is done saving it into owl file and then reload into OWLReferences thanks to the informations stored in this class.

21.6.2 Inhaltsverzeichnis

21.6.3 Konstruktoren

OWLReferencesSerializable initializes all the field of the class

Parameter

ontoName the name associated to this OWLReferences

filePath the directory path to the file

ontologyPath the ontologyPath (IRI path)

usedCommand the usedCommand to load (from file or web) or create

21.6.4 Methoden

getOntoName

Rückgabewert the name associated to this OWLReferences

getFilePath

Rückgabewert the directory path to the file

getOntologyPath

Rückgabewert the ontologyPath (IRI path)

getUsedCommand

Rückgabewert the usedCommand to load (from file or web) or create

21.7 Klasse ontologyFramework.OFContextManagement.InferedAxiomExporter

21.7.1 Übersicht

This static class is used to export an ontology, making all the asserted property as fixed one.

@author Buoncomapgni Luca

@version 1.0

21.7.2 Inhaltsverzeichnis

21.7.3 Methoden

exportOntology Given an ontology it changes all the asserted property into a fixed one. This method just ask for all the asserted entity and make a copy of them. Note that this procedure may be computational expensive.

Parameter

ontoRef the ontology to export

Rückgabewert the input ontology with asserted entity exported

isImportingClosure

Rückgabewert the importingClosure

setImportingClosure

Parameter

importingClosure the importingClosure to set

22 Package ontologyFramework.OFProcedureManagment

22.1 Klassen-Liste

Algorithm This class is an implementation of `OFProcedureInterface` which is designed to run procedure using Quartz API; it works with data initialized from `OFProcedureBuilder`. 130

OFProcedureBuilder This class is design to initialize (build) a Procedure object with respect to the ontology into the framework. 133

OFProcedureSynchronisation This class contains, for a given procedure individual name, a list of names related to other ontological individuals that represent procedures. 135

OFProcedureInterface This interface represents a Procedure object. 137

ProcedureConcurrenceData This class describes a procedure while is running to manage dead line and concurrent pool monitoring. 139

OFSchedulingBuilder This class is design to initialize (build) a Quartz scheduler object with respect to the ontology into the framework. 140

ProcedureConcurrenceManager This class is used to initialize and manage a Procedure with its ID which is necessary to manage a concurrent pool approach. 141

22.2 Package-Beschreibung

22.3 Klasse ontologyFramework.OFProcedureManagment.Algorithm

22.3.1 Übersicht

This class is an implementation of `OFProcedureInterface` which is designed to run procedure using Quartz API; it works with data initialized from `OFProcedure-Builder`. It is design with the following behaviors: a Procedure must be linked to a scheduler and a tread pool. Also, it has a checker job to update the state of the ontology that is describing the procedure. This job is automatically created and runs with a specific frequency defined in the ontology. The checker implementation is given by `AlgorithmCheckerJob`. Also, a procedure can contains an Event, if not is considered by default that the it has an Event always true. Furthermore, a procedure can be synchronized with another individual that describe a procedure. In this last case the event is not considered. When the individual that synchronize this procedure ends, or when an event is true than the procedure is ready to run. Anyway, it will actually run in accord with its TimeTrigger ontological definition that describe its behavior when the previous consideration are favorable to run this procedure. Remember that the checker job will always run on background and its frequency affect the velocity of the changes that the system can appreciated. Is recommended to build checker that are not computational complex and make their frequency high enough. Synchronization properties are not affected by the checker frequency. Finally, the always need features for a Procedure are: Checker, TimeTrigger, thread pool size, full qualifier to the procedure implementation and an Event or a Synchronization property. If both of them exist than the system will consider only one of them in accord with the considerations above.

In particular this class initialize a procedure in accord with `OFProcedureBuilder` during building time. This will set inside this class all the interesting characteristics of procedure and also it created the related CheckerJob and MainJob (which contains the runnable implementation of the procedure). Than the CheckerJob is run and will be stopped only by `shotdown`. The checker set the quantity relate to Events and TimeTrigger and this changes make this class calling `stop` or `run` in a way to implement the above considerations. Moreover, it will care to give to the implementing procedural job the most up to date information required as input througth the Quartz JobDataMap.

@author Buoncomapgni Luca

@version 1.0

22.3.2 Inhaltsverzeichnis

22.3.3 Variablen

chekerName

22.3.4 Konstruktoren

Algorithm

22.3.5 Methoden

initialise

run

stop

getEventResult

getEventName

setEventResult

getTimeTrigger

getTimeTriggerName

setTimeTrigger

shotdown

getScheduler

getCheckerJob

getMainJob

getAllInstances All the time that this class is created its instance is also collected into a HashMap with its ontological individual name as key. This method returns the map of the create instances so far created for this class.

Rückgabewert all the instance of this class created so far and collected into a map.

getOFProcedureInterface This methods calls: `Algorithm#getAllInstances().get( referenceName)` to return an instance of this class which is relate to an ontological individual with name equal `tokens` the input parameter.

Parameter

referenceName

Rückgabewert The instance of this individual which has individual named as the input parameter.

isInAllInstances It calls: `Algorithm#getAllInstances().containsKey( key)` to return true if an instance with the key name has been already created for this class, false otherwise.

Parameter

key the name of the instance of this class (name of the ontological individual relate to this procedure).

Rückgabewert true if an instance of this class already exist with such name; false if not.

22.4 Klasse ontologyFramework.OFProcedureManagment.OFProcedureBuilder

22.4.1 Übersicht

This class is design to initialize (build) a Procedure object with respect to the ontology into the framework. It will build an HashMap between the name of an ontological individual and a `OFProcedureInterface`. This map will be available than into the static map manager: `OFBuildedListInvoker`.

By definition the individual which reflect the building mechanism used in this implementation is: `B_OFProcedureBuilder € OFBuilder`

```
hasTypeName "ontologyFramework.OFProcedureManagment.OFProcedureBuilder"^^str
 buildList "ProcedureList"^^string
hasTypeKeyWord "OFProcedure hasOFProcedureCheckerFrequencyInMilliSeconds
hasOFProcedureScheduler hasOFProcedureEvent hasOFProcedureTrigger
hasOFProcedureConcurrentPoolSize hasOFProcedureSynchronisationWith"^^string
```

Where the first key word is relate to the name of the ontological class to looking for procedure individuals. While the other are the name of Object Property which link an individual to its properties. This basically means that to define an individual which will be one to one relate with a Procedure object just create it as: `P_Procedure1 € OFProcedure`

```
[1] hasTypeName "fullQualifier.ToImplementationOF.OFJobAbstract"
 [2] hasOFProcedureConcurrentPoolSize "4"^^integer[]
 [3] hasOFProcedureCheckerFrequencyInMillisecond "200"^^long
 [4] hasOFProcedureTimeTrigger TT_TriggerNow
 [5] hasOFProcedureEvent Ev_EventTrue
 [6] hasOFProcedureSynchronizationWith P_ReasonerUpdater
```

Where [1] refer to the full Java qualifier to the object that actually implements an procedure; it should extend the class `OFJobAbstract`. [2] defines the maximum number of instance of the same procedure that can run at the same time. Moreover, [3] indicates the running period of the checker object, aimed to synchronize the individual P_Procedure1 between its ontological representation and a relate istance of `OFProcedureInterface`. [4] and [5] define the only individual that, in turns, defines the TimeTrigger, and Event respectively associate to this procedure. Finally, [5] define if a procedure should wait the end of another befaure to run.

@author Buoncomapgni Luca

@version 1.0

22.4.2 Inhaltsverzeichnis

22.4.3 Konstruktoren

OFProcedureBuilder

22.4.4 Methoden

buildInfo

getInitialisedClasses

runAllProcedure It will call `run` for all the classes Initialized during the last building time.

runAllProcedure It will call `run` for all the individual given as an input parameter `Map< String, OFProcedureInterface>`.

Parameter

toRun map of procedure to run.

runProcedure It will look by keys over the last builded map to find the relate procedure to run. Than it will call `run`.

Parameter

individualName key name of the class OFProcedureInterface to run.

runProcedure It will look by keys over the map given as input parameter to find the relate procedure to run. Than it will call `run`.

Parameter

individualName key of the procedure to run.

toRun map between string (name) and OFProcedureInterface.

stopAllProcedure It will call `stop` for all the classes Initialized during the last building time.

stopAllProcedure It will call `stop` for all the individual given as an input parameter `Map< String, OFProcedureInterface>`.

Parameter

toStop map of procedure to stop.

stopProcedure It will look by keys over the last builded map to find the relate procedure to stop. Than it will call `stop`.

Parameter

individualName key name of the class OFProcedureInterface to stop.

stopProcedure It will look by keys over the map given as input parameter to find the relate procedure to stop. Than it will call `stop`.

Parameter

individualName key of the procedure to stop.

toStop map between string (name) and OFProcedureInterface.

22.5 Klasse ontologyFramework.OFProcedureManagment.OFProcedureSynchronisat

22.5.1 Übersicht

This class contains, for a given procedure individual name, a list of names related to other ontological individuals that represent procedures. In particular, the list represents the name of ontological procedures that have to finish its computation before to run this particular procedure.

@author Buoncomapgni Luca

@version 1.0

22.5.2 Inhaltsverzeichnis

22.5.3 Konstruktoren

OFProcedureSynchronisation Create a new synchronization object relate to an procedure individual.

Parameter

individualName of the ontological procedure for this instance.

OFProcedureSynchronisation Create a new synchronization object relate to a procedure individual. Also it initializes it with some synchronization relations with respect to other procedure. Practically this method will call `this.addSynchronisation( i)` for all `i` into the parameter `synchInd`. If this class has been already created for the procedure defined by individualName, than this method does not create a new instance and update the synchronization individuals to the already existing one, avoiding to duplicate data.

Parameter

individualName of the ontological procedure for this instance.

synchInd set of individual to synchronize

22.5.4 Methoden

addSynchronisation Add an individual to the synchronization list. This means that the procedure with the individual name defined by `getSyncName` before to run has to wait until also the procedure refereed to the individual given as parameter ends it work.

Parameter

ind procedure individual to be synchronized with.

addSynchronisation It calls `addSynchronisation` for all component inside the set given as input parameter.

Parameter

ind set of procedure individual to be synchronized with.

getSynchIndividuals

Rückgabewert the actual list of synchronization individuals

toString

getAllInstances static method that returns all instance of this class created by the framework. Those are collected in an HashMap with keys equal to the ontological name of the procedure.

Rückgabewert all created instance of this class.

getOFProcedureSynchronisation It looks inside the returning value of `getAllInstances` and return an istance of this class in accord with its name inside the map. Basically it just return; `OFProcedureSynchronisation.getAllInstances.get( referenceName)`

Parameter

referenceName the individual name of the procedure for which refer to its synchronization list.

Rückgabewert an instance of this class relate to the procedure defined by an individual with name: `referenceName`

removeSynchronisation It remove the instance with individual name equal to the input parameter from the map that collect all the active instance of this class. Practically, it just call: `OFProcedureSynchronisation.getAllInstances.remove( synchName)`

Parameter

synchName individualName for which the instance of this class must be removed.

removeSynchronisation It remove the instance given as input parameter from the map that collect all the active instance of this class. Practically, it just call: `OFProcedureSynchronisation.getAllInstances.remove( synchObj.getSyncName()`

Parameter

synchObj

22.6 Interface ontologyFramework.OFProcedureManagment.OFProcedureInterface

22.6.1 Übersicht

This interface represents a Procedure object. Instances of this interface will be create in accord from `OFProcedureBuilder`. An implementation of this Interface is `Algorithm` which uses Quartz engine.

@author Buoncomapgni Luca

@version 1.0

22.6.2 Inhaltsverzeichnis

22.6.3 Methoden

initialise This method will be called just after all building times. It should be used to initialize variables that need to get data from the ontology.

Parameter

procedureInd ontological individual which is reflacting a procedure

ontoRef reference to an OWL ontology

listInvoker static list manager of builded entity

keyWords coming from the builder ontological definition througth data type: `hasTypeKeyWor exactly 1 string`

run Should run this procedure immediately.

stop Should stop this procedure immediately (or just after is computation, if it is running).

shotdown After a call to this method the procedure should be no more schedulable (at least, as long as a new building time occurs).

getEventName Get the individual name of the Event attached to the ontological representation of a procedure. It should be able to update themselves without call a building mechanism.

Parameter

ontoRef reference to an OWL ontology

Rückgabewert the individual name of the Event attached to this procedure.

setEventResult The name returned by: `getEventName` will be used during checking (as an example: `AlgorithmCheckerJob`) time to retrieve their result through the Event list (`OFBuildedListInvoker`). When its boolean result is available this method is called to deal with a possible change of event result. So, it should use `run` `stop` or `shotdown` to change the state of the procedure.

Parameter

result the boolean value associate to the Event which name is given by `getEventName`

getEventResult should return the most up to date Event result given by; `setEventResult`

Rückgabewert the boolean value associate to the Event which name is given by `getEventName`

getTimeTriggerName Get the individual name of the TimeTrigger attached to the ontological representation of a procedure. It should be able to update themselves without call a building mechanism.

Parameter

ontoRef reference to an OWL ontology

Rückgabewert the individual name of the TimeTrigger attached to this procedure.

setTimeTrigger The name returned by: `getTimeTriggerName` will be used during checking (as an example: `AlgorithmCheckerJob`) time to retrieve their result through the TimeTrigger list (`OFBuildedListInvoker`). When its returnig variable is available this method is called to deal with a possible change of trigger. So, it should use `run` `stop` or `shotdown` to change the state of the procedure.

Parameter

timeTrigger the Trigger Object associate to the TimeTrigger individual which name is given by `getEventName`

getTimeTrigger should return the most up to date Trigger object given by; `setEventResult`

Rückgabewert the Trigger Object associate to the TimeTrigger individual which name is given by `getEventName`

getScheduler If a scheduler mechanism is used it should return the instance to scheduler object associate to this procedure

Rückgabewert the scheduler

getCheckerJob Since a procedure is always linked to a Checker procedure, which has the objective to synchronize ontological changes of the procedure individual it is convenient to represent the relate Checker object inside this implementation giving an external access to it througth this method.

Rückgabewert the checker procedure (must be a runnable implementation)

getMainJob It may be useful to describe a procedure in a way that its computational steps are in an separate runnable class. This allows more flexibility in scheduling and building phases. If this is the case than, this implementaion will define the shape of a generic algorithm. Than different Instances of this class will be related to particolar scripts pointed by the returning value of this method.

Rückgabewert the runnable implementation of this specific procedure.

22.7 Klasse ontologyFramework.OFProcedureManagment.ProcedureConcurrenceDa

22.7.1 Übersicht

This class describes a procedure while is running to manage dead line and concurrent pool monitoring. Base on this information the system is aware about trigger missing and also it helps in concurrency managing describing a thread pool implementation. This means that every procedure need an integer value to describe how many instances of the same procedure can run concurrently.

@author Buoncomapgni Luca

@version 1.0

Siehe auch `ProcedureConcurrenceManager`

22.7.2 Inhaltsverzeichnis

22.7.3 Variablen

NAMEID_symbSeparator

22.7.4 Konstruktoren

ProcedureConcurrenceData Create a new Procedure time tracking. Useful just when the procedure is running.

Parameter

procedureName name of the ontological individual wich describe a procedure

ID an integer number between 0 and the value defined by a particular concurrent pool size

22.7.5 Methoden

getProcedureName

Rückgabewert the procedureName

getID

Rückgabewert the procedure ID

toString

getTriggerNow it just returns `new Date();`

Rückgabewert the triggerNow

getDeadline

Rückgabewert the deadline

isDeadlineSetted

Rückgabewert true if the dead line is set, false otherwise.

setDeadline

Parameter

deadline the deadline to set

22.8 Klasse ontologyFramework.OFProcedureManagment.OFSchedulingBuilder

22.8.1 Übersicht

This class is design to initialize (build) a Quartz scheduler object with respect to the ontology into the framework. It will build an HashMap between the name of an ontological individual and a `Scheduler` which is given by: `sch = new StdSchedulerFactory( propertyPath).getScheduler();`. This map will be available than into the static map manager: `OFBuildedListInvoker`.

By definition the individual which reflect the building mechanism used in this implementation is: `B_SchedulerBuilding € OFBuilder`

```
 hasTypeName "ontologyFramework.OFProcedureManagment.OFSchedulingBuilder"^^st
  buildList "SchedulerList"^^string
 hasTypeKeyWord "QuartzScheduler hasQuartzSchedulerProperty"^^string
```

Where the first key word is relate to the name of the ontological class to looking for scheduling individuals. While the second is the name of Object Property which link an individual to its .properties file. This basically means that to define an individual which will be one to one relate with a Quartz scheduler object just create it as:

```
S_Scheduler1 € QuartzScheduler
  hasQuartzSchedulerProperty exactly 1 FileInd
  where:FileInd € File
  KeyFileInd hasTypeFile "/src/sempleSch.property"^^string
```

@author Buoncomapgni Luca

@version 1.0

22.8.2 Inhaltsverzeichnis

22.8.3 Konstruktoren

OFSchedulingBuilder

22.8.4 Methoden

buildInfo

getInitialisedClasses

shotDownAllScheduler shouts down (`scheduler.shoutdown()`) all the scheduler builded from this class during previous building time.

22.9 Klasse ontologyFramework.OFProcedureManagment.ProcedureConcurrenceMa

22.9.1 Übersicht

This class is used to initialize and manage a Procedure with its ID which is necessary to manage a concurrent pool approach. Practically, this class represent a List of `ProcedureConcurrenceData` with fixed size. Initially all places of the list are null and this means that the pool is empty. Where an instance is running, the first place of the list will be associate to a ProcedureConcurrenceData object and it will means that that place inside the pool is unusable. Than, when the instance Finishes its work its relate place inside the list will go back to null. So, if this list does not contain null places means that the pool is full, namely the procedure cannot run even if all its conditions are satisfied. The ID associate to the procedure will be equal to the index inside the list represented by this class.

@author Buoncomapgni Luca

@version 1.0

22.9.2 Inhaltsverzeichnis

22.9.3 Konstruktoren

ProcedureConcurrenceManager Create a new ProcedureConcurrenceManager associate to a particular Procedure individual.

Parameter

- **procedureName** ontological name of the procedure
- **concurrencyOrder** the concurrency pool size for this procedure.

22.9.4 Methoden

generateID Returns an new instance of `ProcedureConcurrenceData` initialized with procedureName and the index (ID) inside the concurrent pool. If the pool is full than this method will return null.

Rückgabewert the procedure data in terms of concurrency if the thread pool is not full.

removeID remove a procedure from the concurrency pool. It should be called as soon as the procedure ends.

Parameter

pcd concurrency data with respect to a procedure ended that should be removed from the thread pool.

getProcedureName

Rückgabewert the procedureName

getProcedureActivated

Rückgabewert the list of all the instances running inside the pool related to this procedure.