Dragonfly

Flowcharts of the architecture of wizards

Object Research Systems (ORS), Inc.

November 2021

Table of contents

About wizards

Wizards are used in Dragonfly to guide the user through an orderly set of tasks. This is done by presenting different wizard pages, one at a time. It is expected that the user will fulfill the main task in each wizard page, to progress towards the completion of the workflow. Each wizard page is specialized: the tools presented are chosen to help the user accomplish the main task of that wizard page.

The responsibility of the wizard plugin class is to know the overall navigation of the workflow: what page should be shown when another one is completed or how to step back.

The responsibility of a wizard page is to show a set of tools to help the user accomplish a given task. A wizard page does not know the other wizard pages.

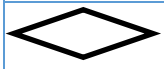
A workflow is created by making a plugin and a set of wizard pages. The plugin implementation (wizard model) should subclass the OrsAbstractWizardPlugin class and his mainform (wizard form) should subclass the OrsAbstractWizardContextWindow class. Each wizard page model should subclass the OrsAbstractWizardPageModel class and the associated wizard page form should subclass the OrsAbstractWizardPageForm class.

The following flowcharts describe the logic of the different mechanisms involved in the wizards. It relies on the communication between these 4 classes.

Note that many of the methods involved in this logic are intended to be overloaded in the concrete classes. This is why some of the methods shown in this document, for the abstract classes, are empty.

Flowcharts symbols

This is the meaning of the different symbols used in the flowcharts of this document. They follow the standard definition.

Shape	Name	Description
	Flowline	Order of operation
	Terminal	Beginning and ending of the method
	Process	Set of operations
	Decision	The process selects a path depending on a condition
	Predefined Process	Calling on another method

Color legend

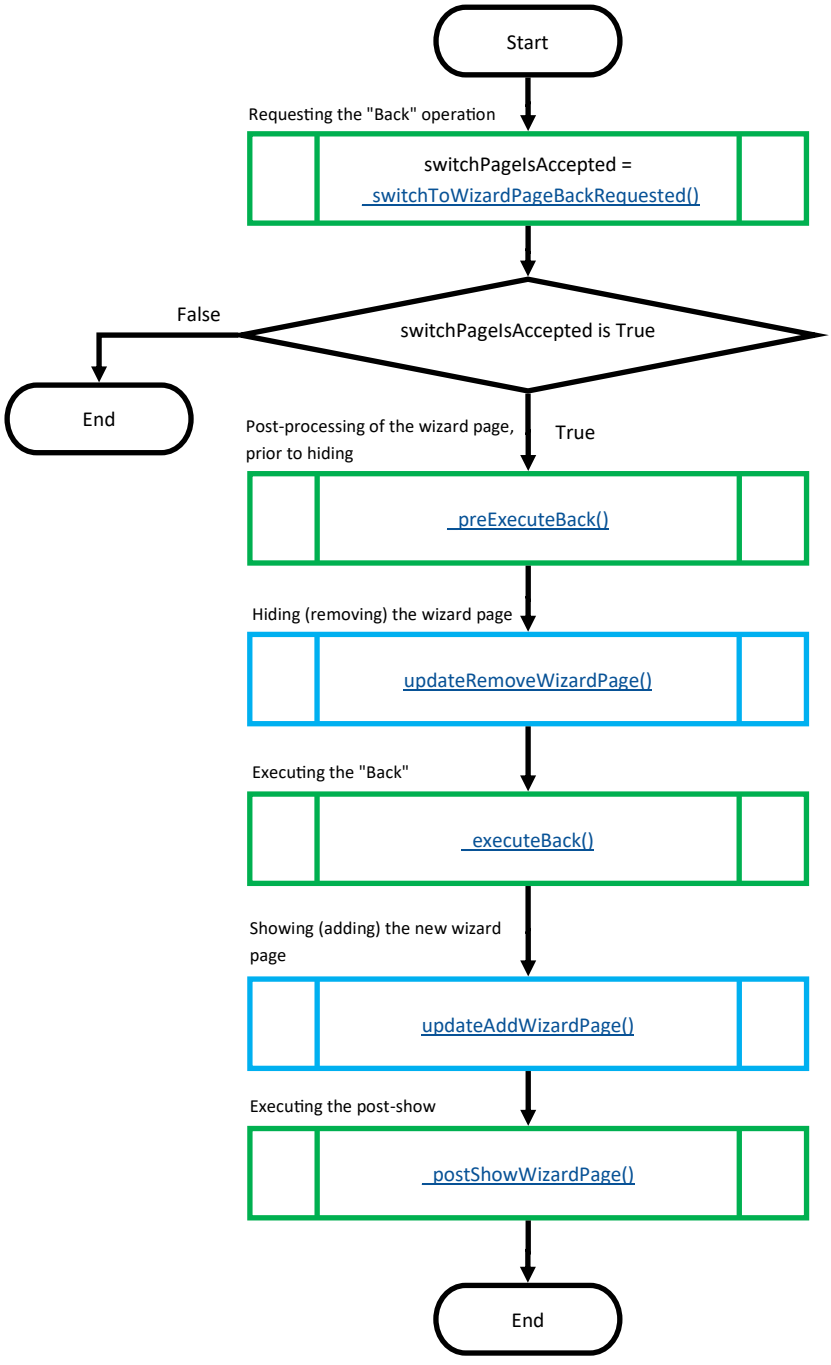
The shapes of the predefined processes are colored to facilitate the identification of the classes where they are defined:

Color	Class
	OrsAbstractWizardPlugin (wizard model)
	OrsAbstractWizardContextWindow (wizard form)
	OrsAbstractWizardPageModel (wizard page model)
	OrsAbstractWizardPageForm (wizard page form)

Transitions

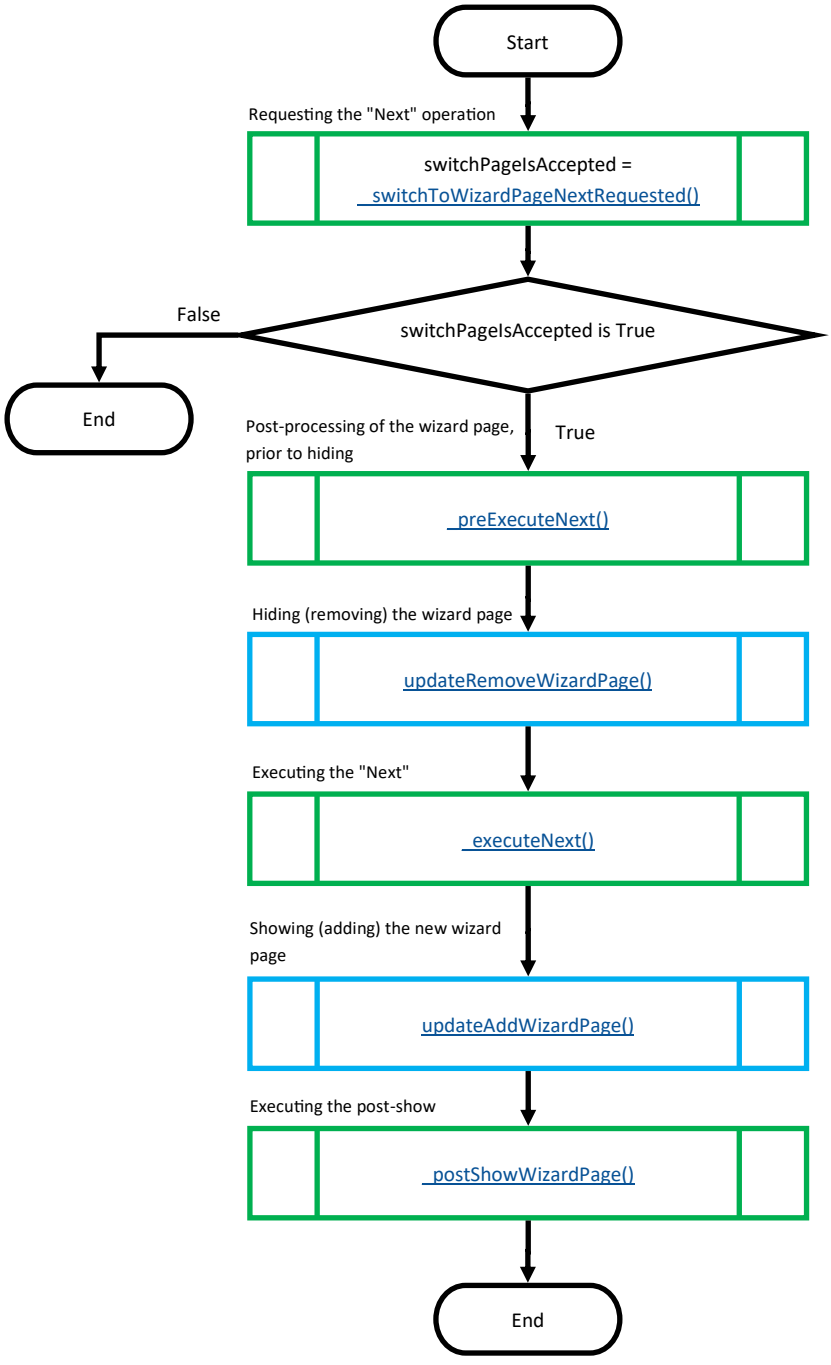
Switch Back Wizard Page (switchToWizardPageBack())

This method is called when pressing on the wizard button *Back*. It is used to switch to the previous wizard page.



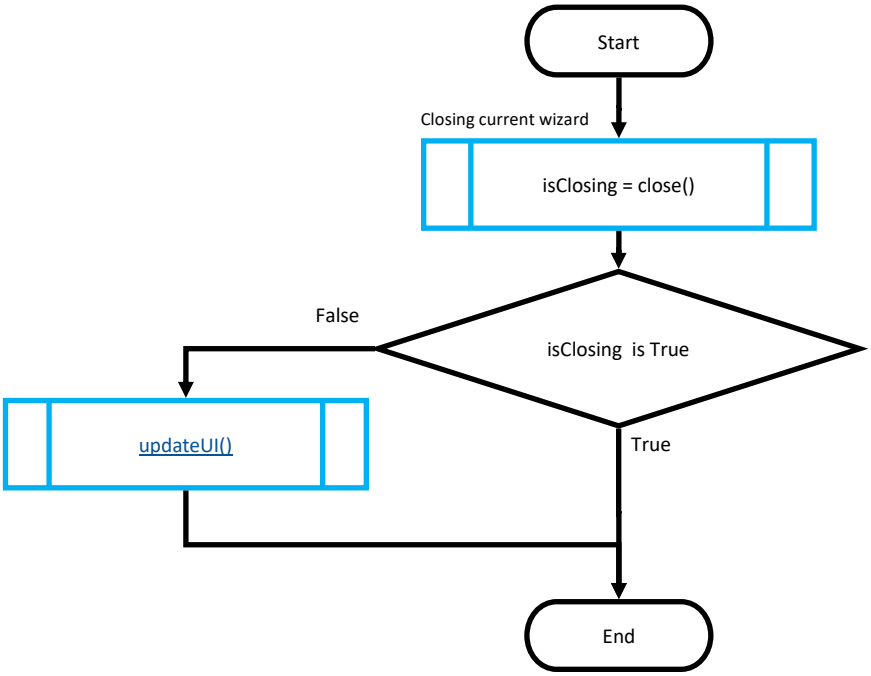
Switch Next Wizard Page (switchToWizardPageNext())

This method is called when pressing on the wizard button *Next*. It is used to switch to the next wizard page.



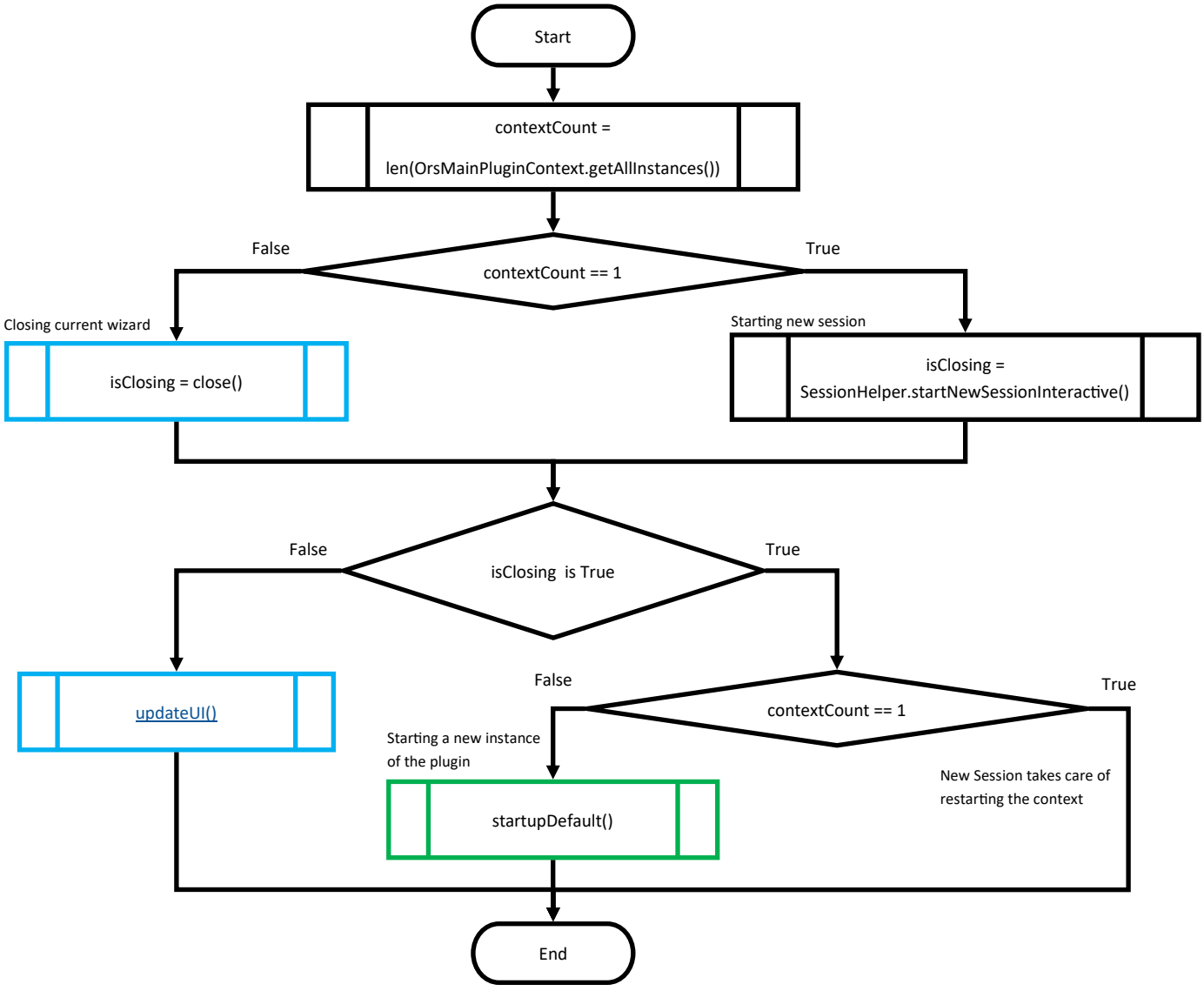
Close

This method is called when pressing on the wizard button *Close*. It is used to close the current wizard.



Restart

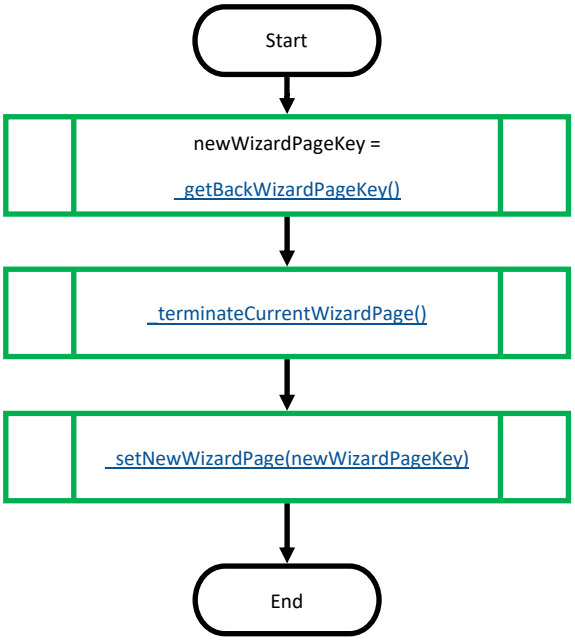
This method is called when pressing on the wizard button *Restart*. It is used to close the current wizard and start a new one.



Wizard model methods

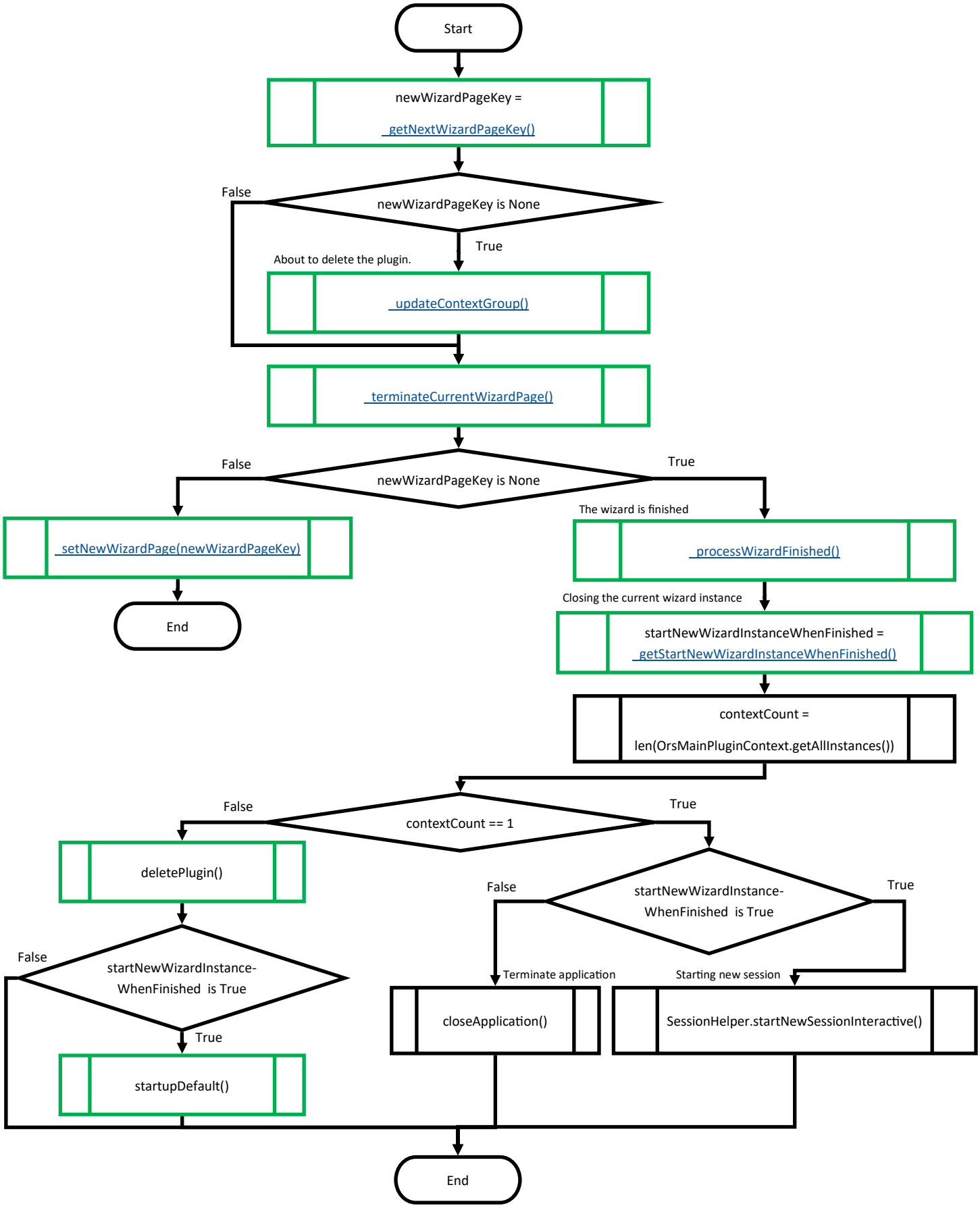
_executeBack()

Executes the operation related to the wizard "Back".



_executeNext()

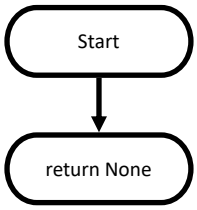
Executes the operation related to the wizard "Next" after the page has been hidden.



_getBackWizardPageKey()

Gets the wizard page key to go "back".

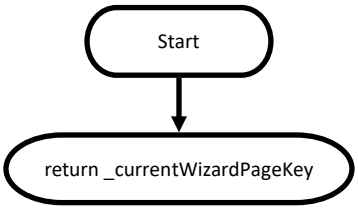
:return: a wizard page key
:rtype: str



getCurrentWizardPageKey()

Gets the current wizard page key.

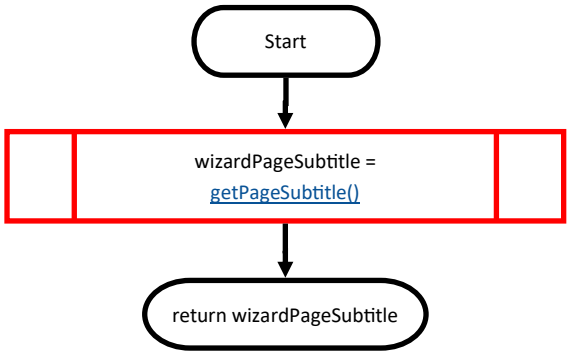
:return: a wizard page key
:rtype: str



getCurrentWizardPageSubtitle()

Gets the text used as subtitle on the page. Return carriage ("\n" character) may be used in the string.

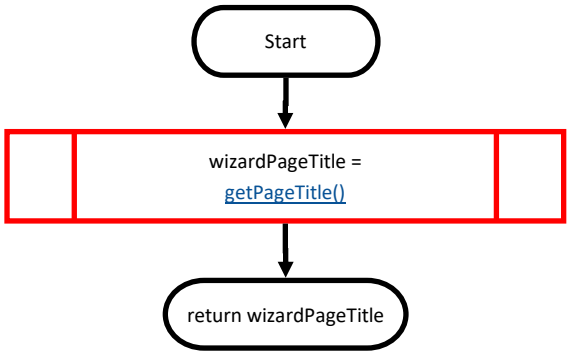
:return: subtitle
:rtype: str



getCurrentWizardPageTitle()

Gets the text used as title on the page.

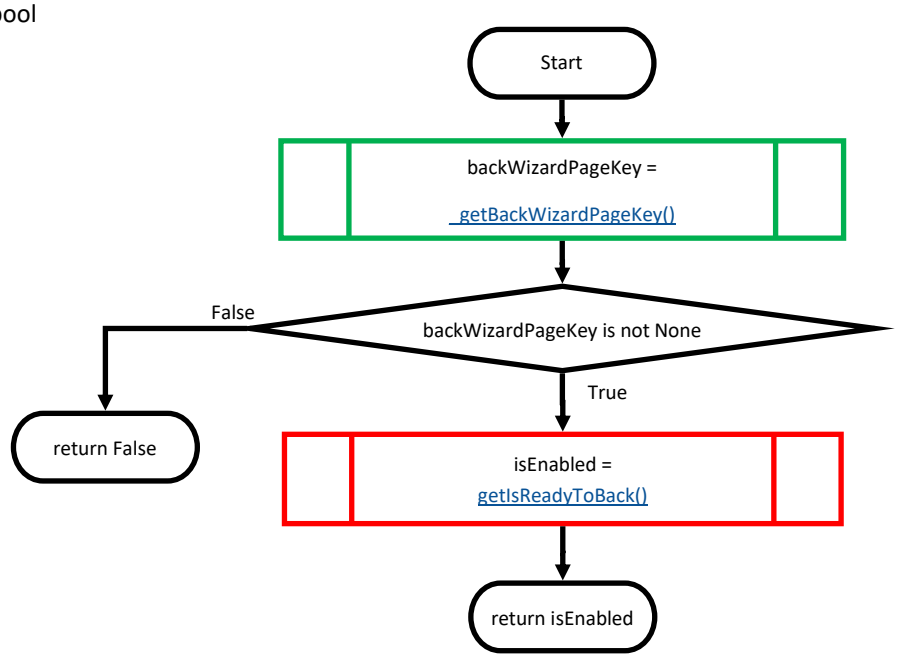
:return: title
:rtype: str



getIsWizardButtonBackEnabled()

Gets if the button “Back” should be enabled.

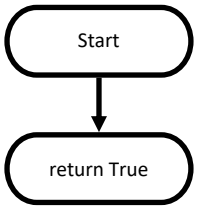
:return: True if the button should be enabled; False otherwise.
:rtype: bool



getIsWizardButtonCloseEnabled()

Gets if the button “Close” should be enabled.

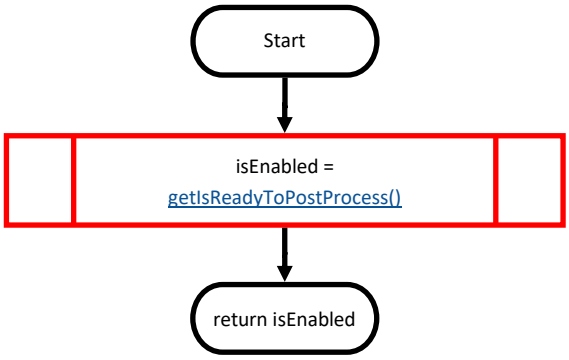
:return: True if the button should be enabled; False otherwise.
:rtype: bool



getIsWizardButtonNextEnabled()

Gets if the button “Next” should be enabled.

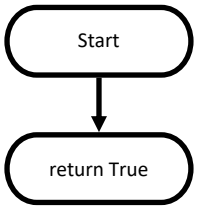
:return: True if the button should be enabled; False otherwise.
:rtype: bool



getIsWizardButtonRestartEnabled()

Gets if the button “Restart” should be enabled.

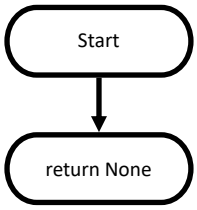
:return: True if the button should be enabled; False otherwise.
:rtype: bool



getLastWizardPageKey()

Gets the wizard page key associated to the last wizard page.

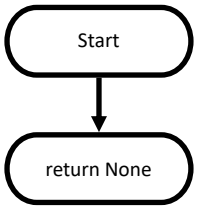
:return: a wizard page key
:rtype: str



_getNextWizardPageKey()

Gets the wizard page key to go "next".

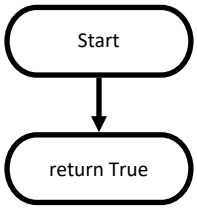
:return: a wizard page key. If None, the wizard will be deleted at the page switch.
:rtype: str



getShowWizardButtonBack()

Gets if the button “Back” should be shown.

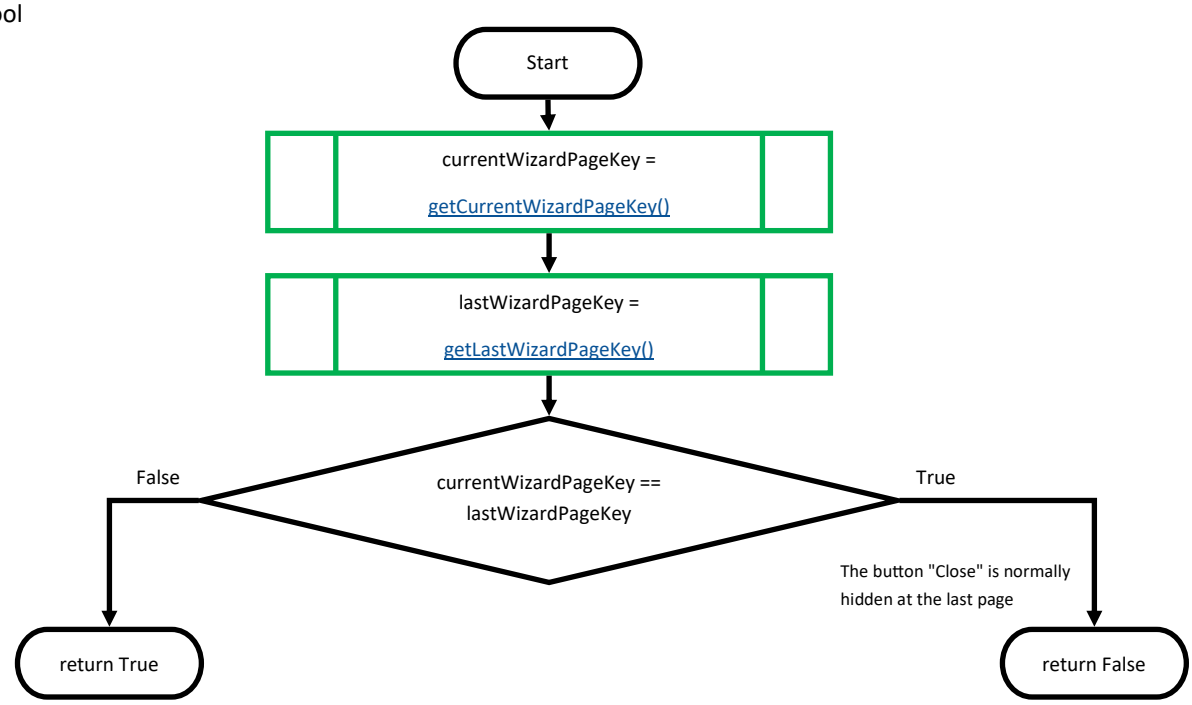
:return: True if the button should be shown; False otherwise.
:rtype: bool



getShowWizardButtonClose()

Gets if the button “Close” should be shown.

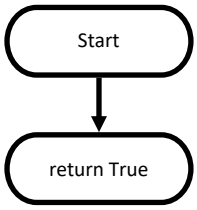
:return: True if the button should be shown; False otherwise.
:rtype: bool



getShowWizardButtonNext()

Gets if the button “Next” should be shown.

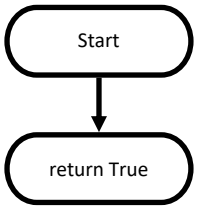
:return: True if the button should be shown; False otherwise.
:rtype: bool



getShowWizardButtonRestart()

Gets if the button “Restart” should be shown.

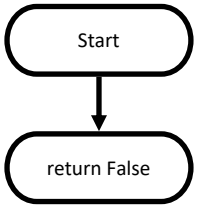
:return: True if the button should be shown; False otherwise.
:rtype: bool



_getStartNewWizardInstanceWhenFinished()

Gets to know if a new wizard instance should be started in the default configuration when the current instance is finished (when there is no more wizard page).

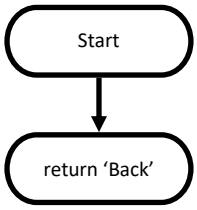
:return: True to have a new wizard instance; False otherwise.
:rtype: bool



getWizardButtonBackText()

Gets the text to write on the button “Back”.

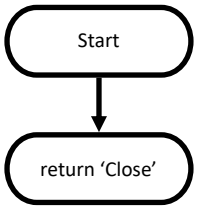
:return: text to write
:rtype: str



getWizardButtonCloseText()

Gets the text to write on the button “Close”.

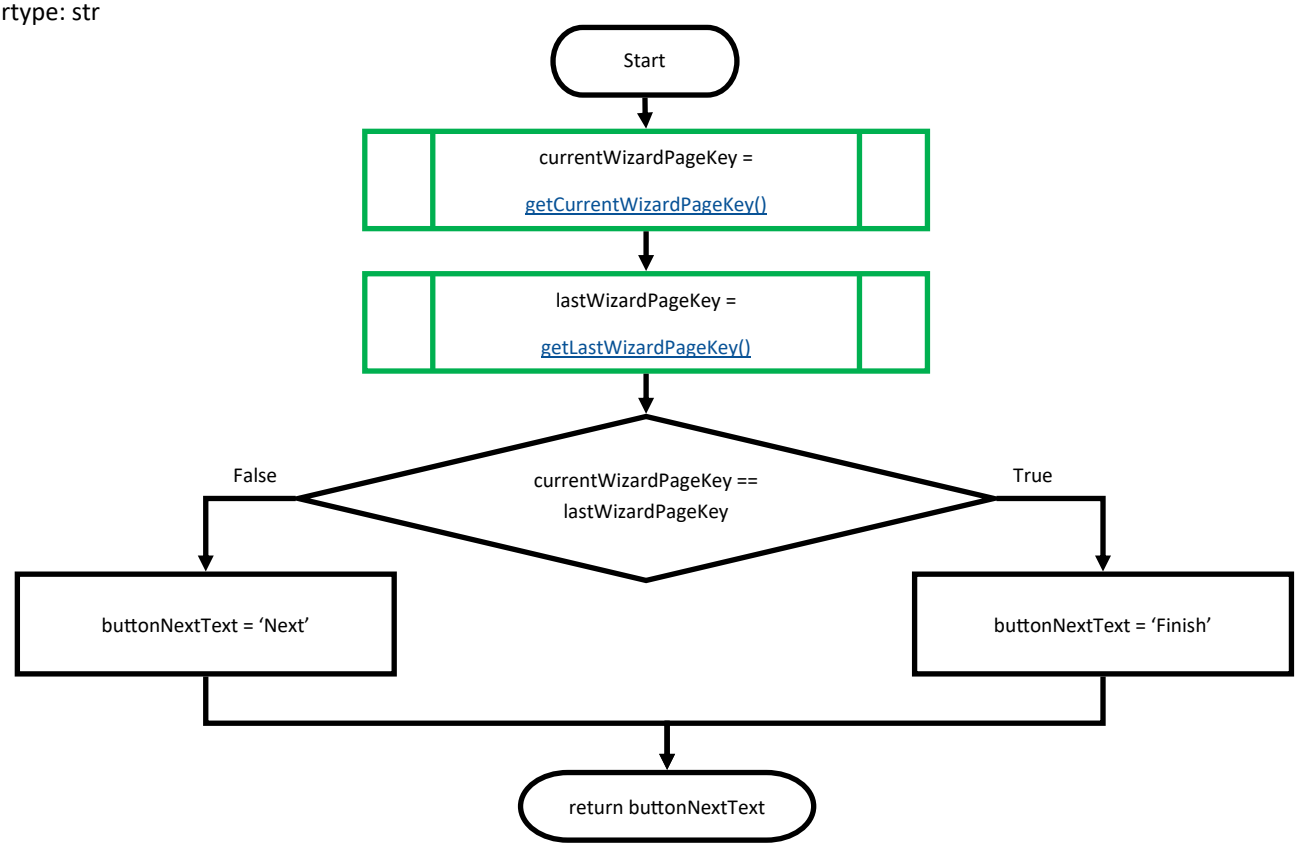
:return: text to write
:rtype: str



getWizardButtonNextText()

Gets the text to write on the button “Next”.

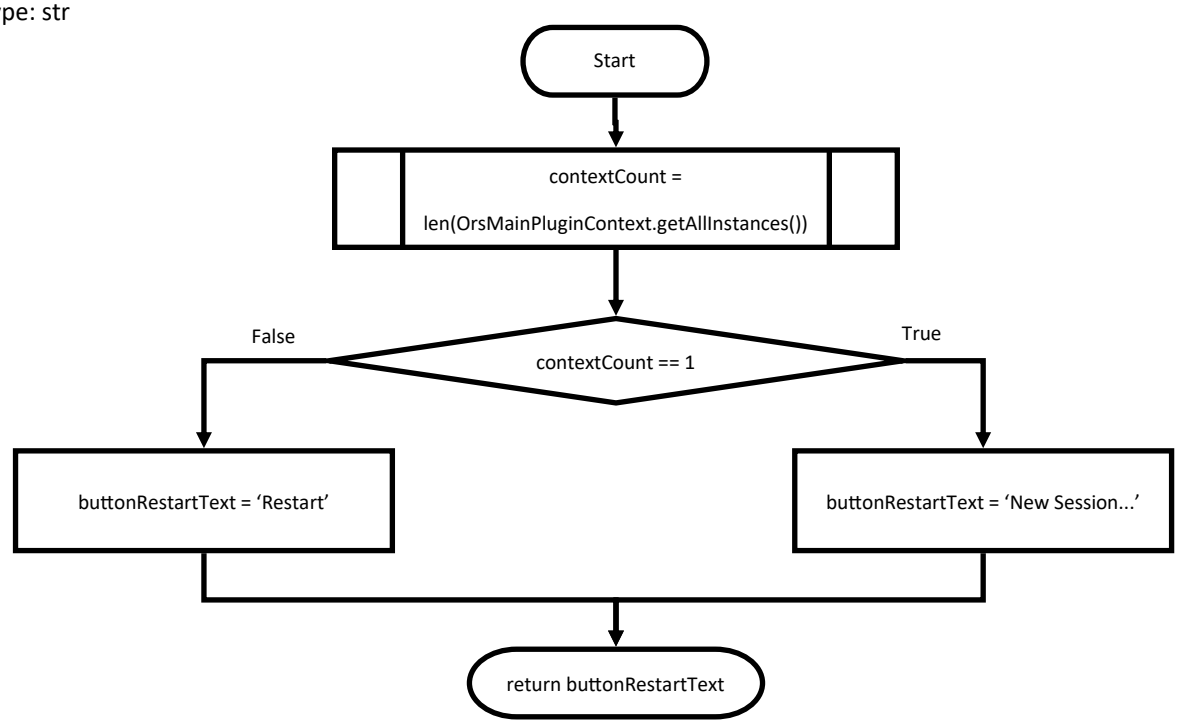
:return: text to write
:rtype: str



getWizardButtonRestartText()

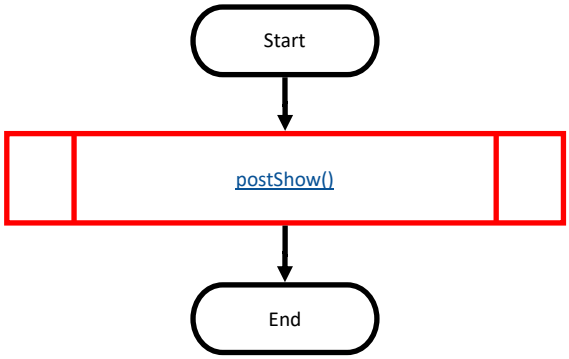
Gets the text to write on the button “Restart”.

:return: text to write
:rtype: str



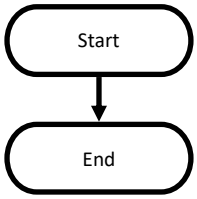
_postShowWizardPage()

Executes the operation related to the wizard "Back" or "Next" after the page has been shown.



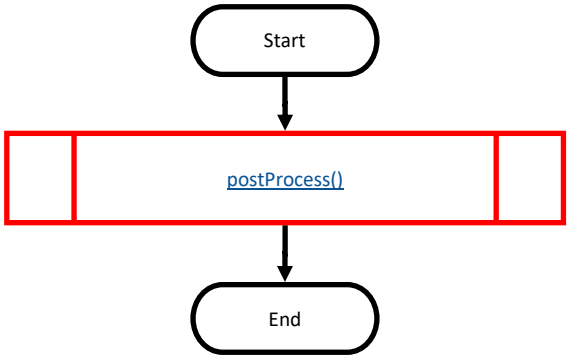
_preExecuteBack()

Executes the operation related to the wizard "Back" while the page is still visible.



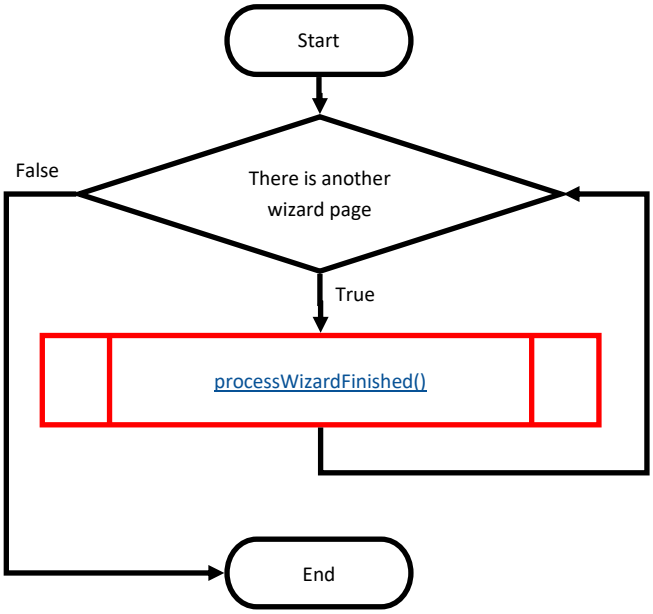
_preExecuteNext()

Executes the operation related to the wizard "Next" while the page is still visible.



_processWizardFinished()

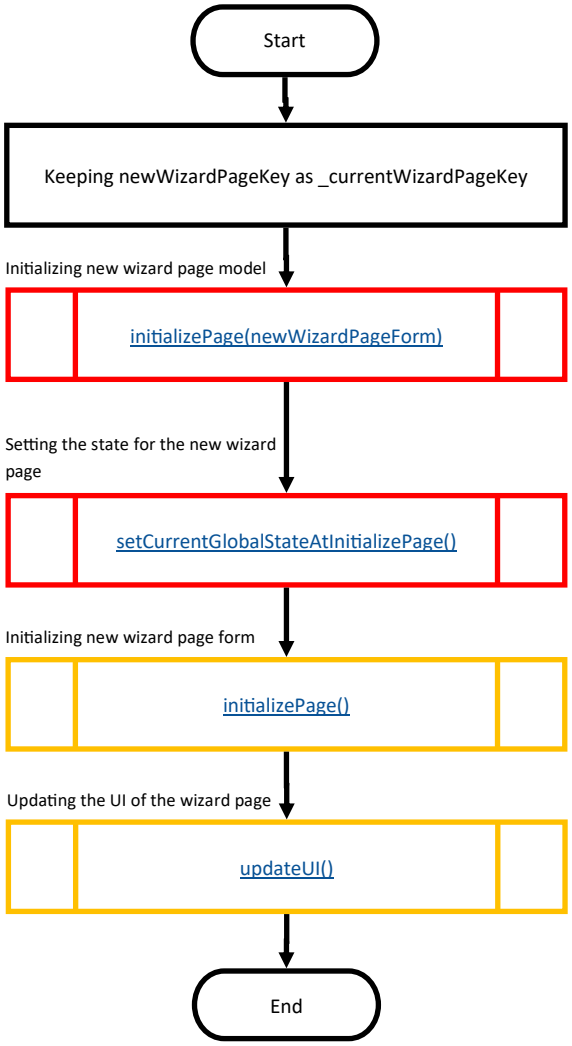
Method called to perform any task when the wizard is finished.



_setNewWizardPage(newWizardPageKey)

Initializes the wizard page.

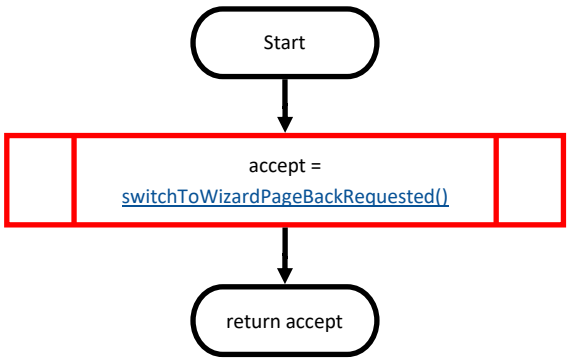
:param newWizardPageKey: new wizard page key
:type newWizardPageKey: str



_switchToWizardPageBackRequested()

Method called to request the "Back" operation. It is the place to query the user for any information prior to start a switch page.

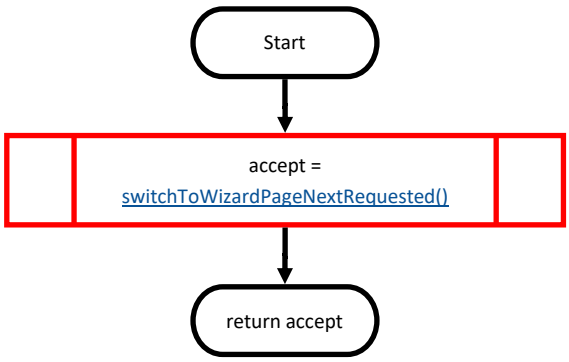
:return: True to accept the page switch request; False to reject it.
:rtype: bool



_switchToWizardPageNextRequested()

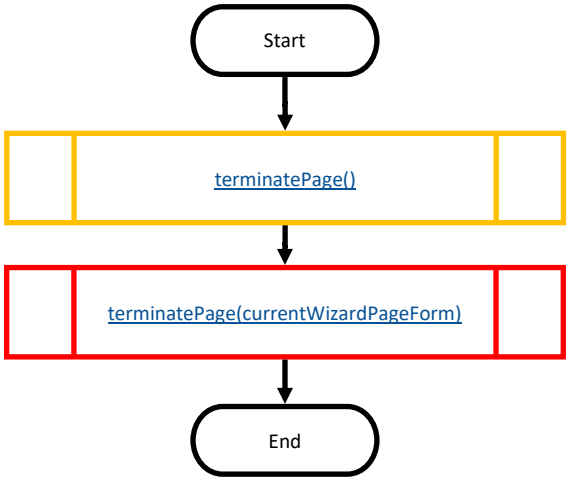
Method called to request the "Next" operation. It is the place to query the user for any information prior to start a switch page.

:return: True to accept the page switch request; False to reject it.
:rtype: bool



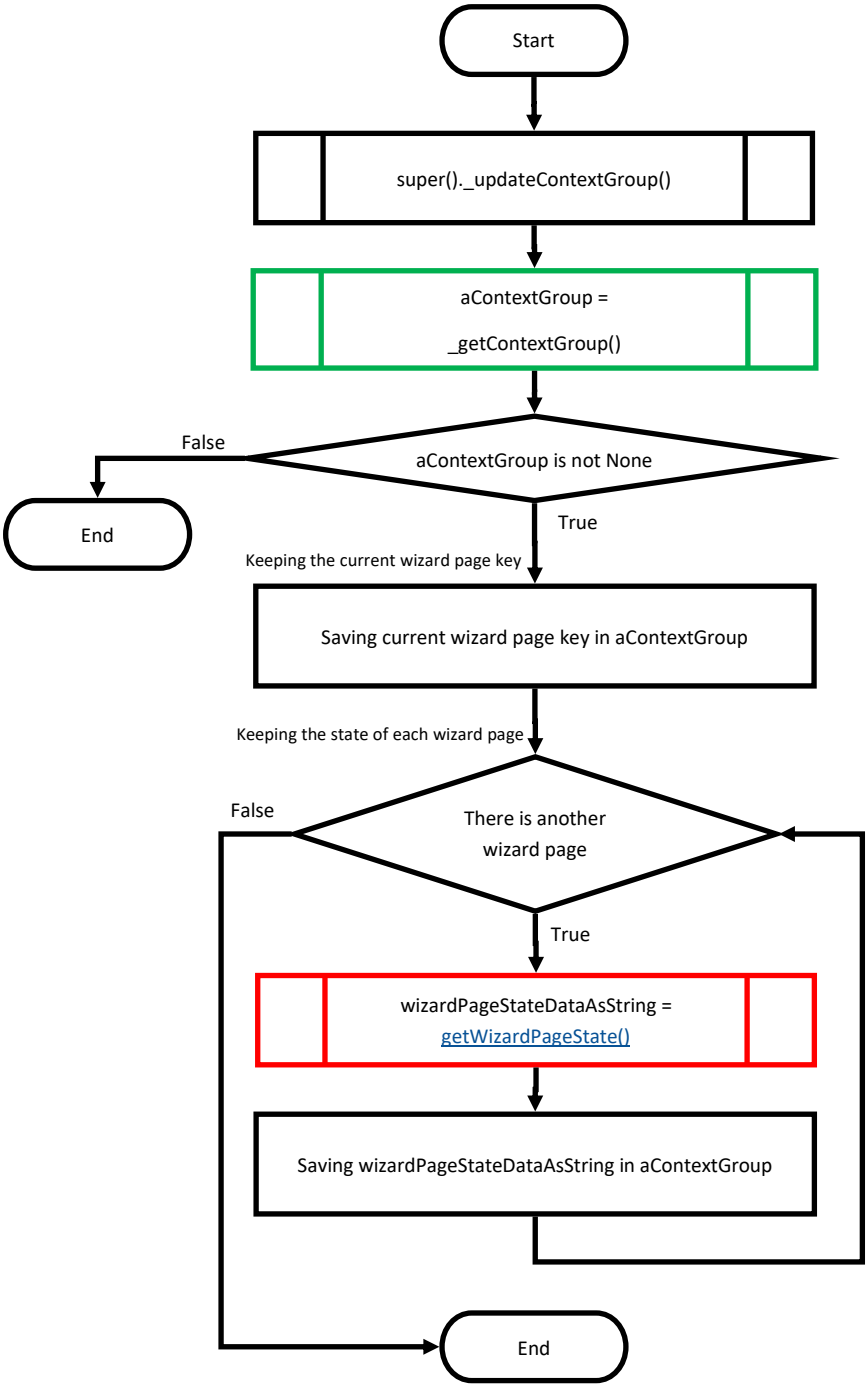
_terminateCurrentWizardPage()

Method called to terminate the page after it is hidden.



_updateContextGroup()

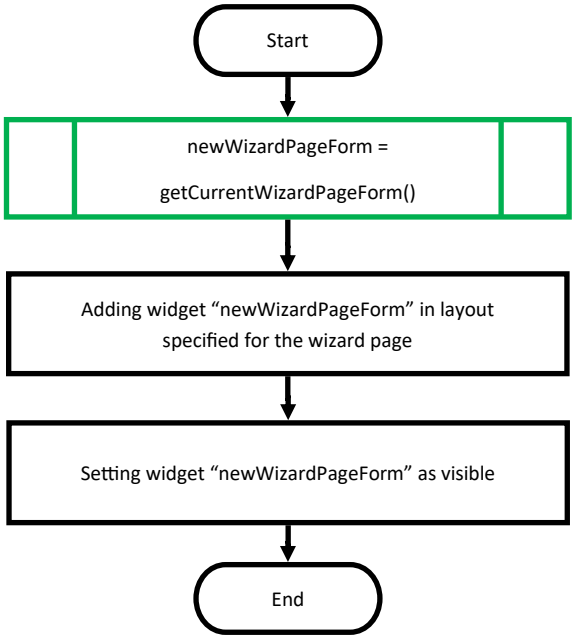
Updates the contents of the context group in preparation for saving.



Wizard form methods

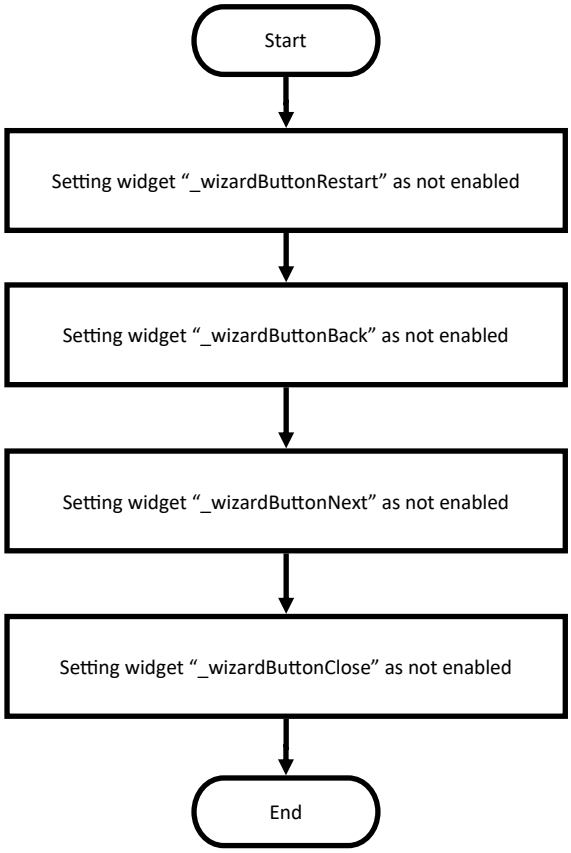
_addWizardPageForm()

Adds the current wizard page form in the layout specified for the wizard page.



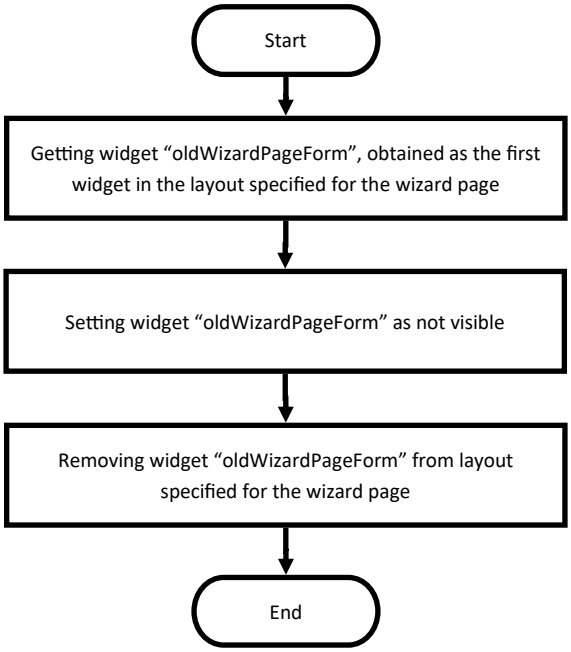
_disableWizardButtons()

Disable the wizard buttons used for the navigation.



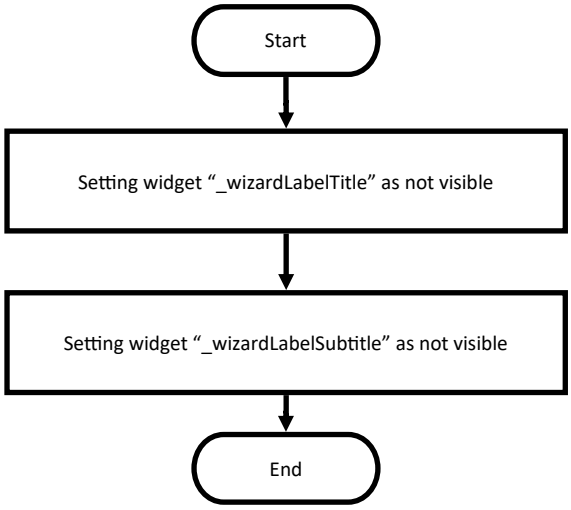
_removeWizardPageForm()

Removes the current wizard page form from the layout specified for the wizard page.



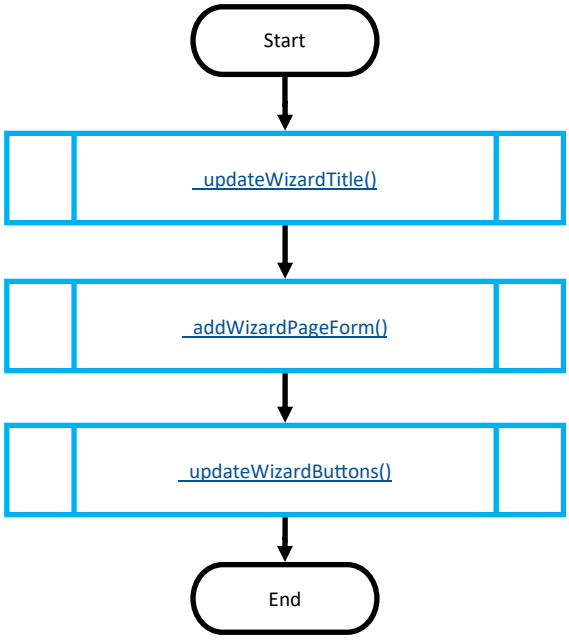
_removeWizardTitle()

Hides the widgets used for the title and subtitle.



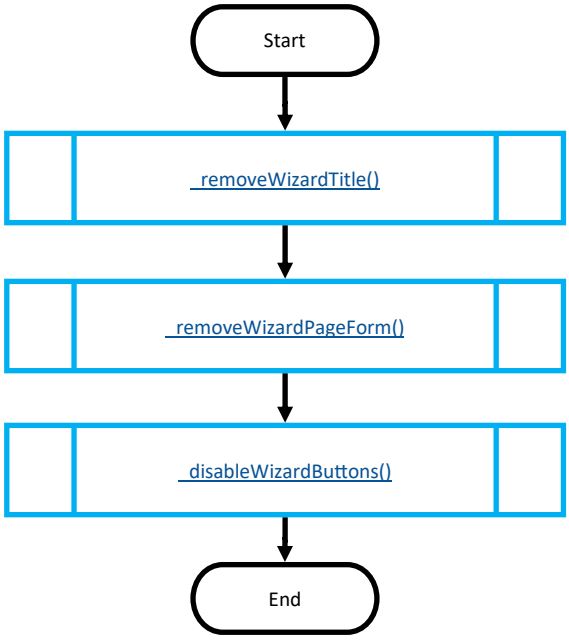
updateAddWizardPage()

Updates the UI by adding the wizard page form.



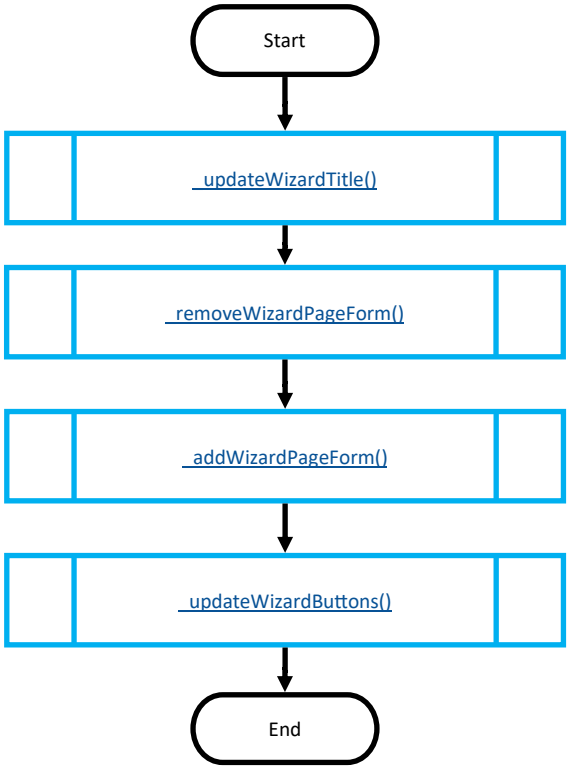
updateRemoveWizardPage()

Updates the UI by removing the wizard page form.



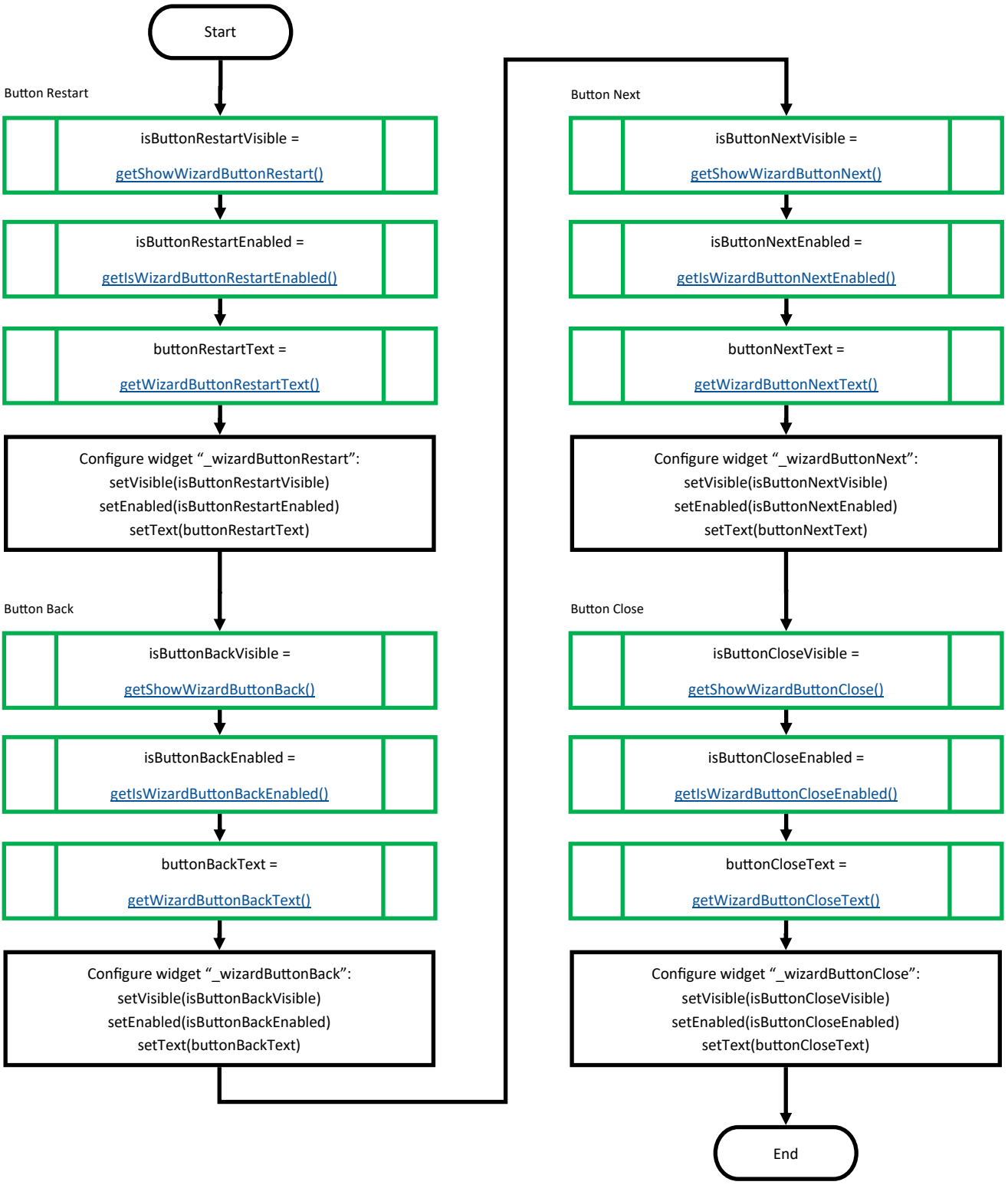
updateUI()

Puts the proper wizard page in the wizard page layout. Updates the visibility and state of the buttons used for the navigation.



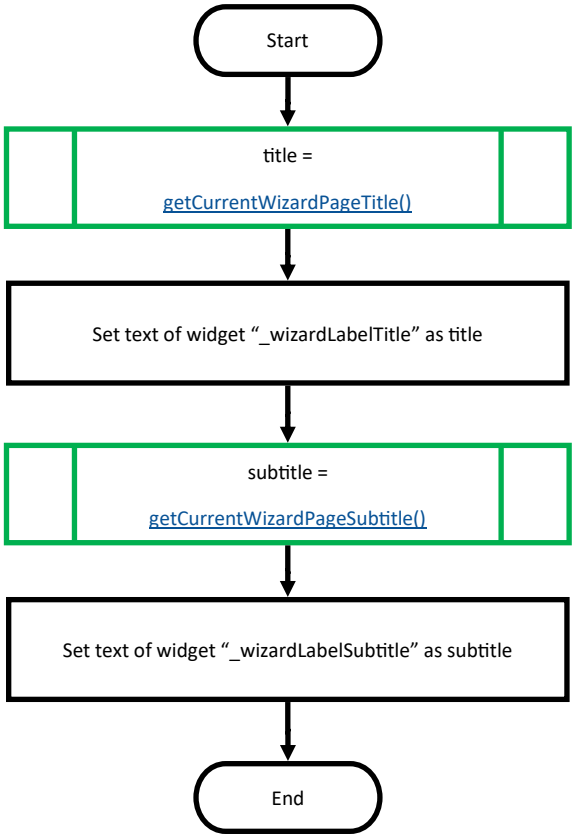
_updateWizardButtons()

Updates the visibility and state of the buttons used for the navigation.



_updateWizardTitle()

Updates the text of the widgets used for the title and subtitle.

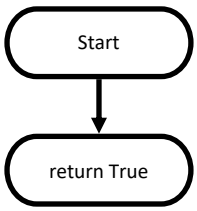


Wizard page model methods

getIsReadyToBack()

Method called to know if the wizard page can be left by going backward with "Back".

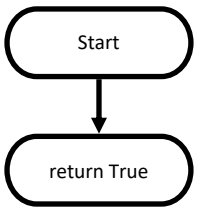
:return: True if the page can be left by going backward with "Back"; False otherwise.
:rtype: bool



getIsReadyToPostProcess()

Method called to know if the wizard page can be left by going forward with "Next".

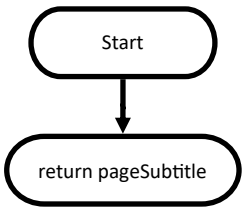
:return: True if the page can be left by going forward with "Next"; False otherwise.
:rtype: bool



getPageSubtitle()

Gets the text to write as the page subtitle.

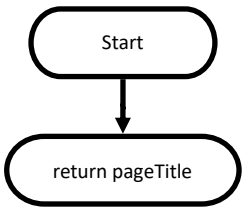
:return: text to write
:rtype: str



getPageTitle()

Gets the text to write as the page title.

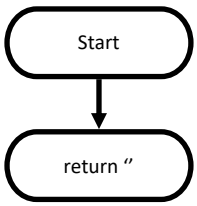
:return: text to write
:rtype: str



getWizardPageState()

Gets the wizard page state. This is the information of the wizard page to save with the context group.

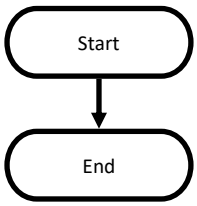
:return: wizard page state
:rtype: str



initializePage(newWizardPageForm)

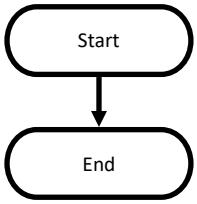
Initializes the wizard page. This method is called to prepare the page just before it is shown. It should put the forms of the plugins in their correct location.

:param newWizardPageForm: new wizard page form
:type newWizardPageForm: OrsAbstractWizardPageForm



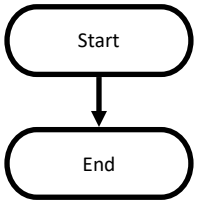
postProcess()

Method called just before the page is left (still visible). It is the place to do computation at the exit of the page (going forward with "Next").



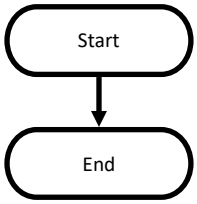
postShow()

This method is called to prepare the page just after it is shown.



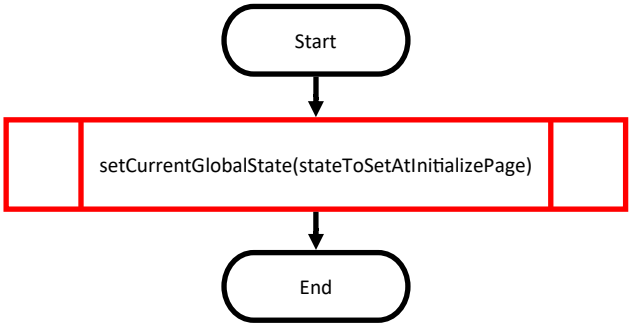
processWizardFinished()

This method is called to perform any task when the wizard is finished.



setCurrentGlobalStateAtInitializePage()

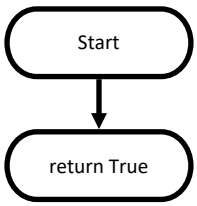
This method sets the global state to agree with one defined by the model or the default one. This is done essentially to avoid getting into a wizard page with the cursor of a preceding page.



switchToWizardPageBackRequested()

Method called to request the "Back" operation. It is the place to query the user for any information prior to start a switch page.

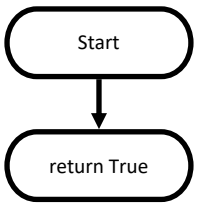
:return: True to accept the page switch request; False to reject it.
:rtype: bool



switchToWizardPageNextRequested()

Method called to request the "Next" operation. It is the place to query the user for any information prior to start a switch page.

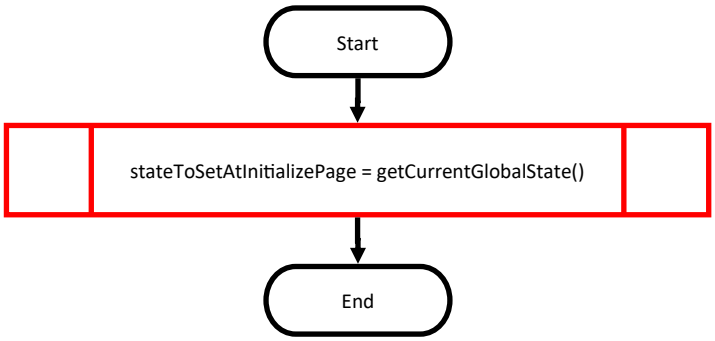
:return: True to accept the page switch request; False to reject it.
:rtype: bool



terminatePage(aWizardPageForm)

Method called to terminate the page after it is hidden.

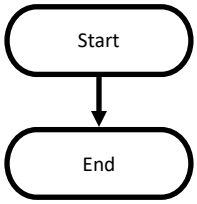
:param aWizardPageForm: wizard page form
:type aWizardPageForm: OrsAbstractWizardPageForm



Wizard page form methods

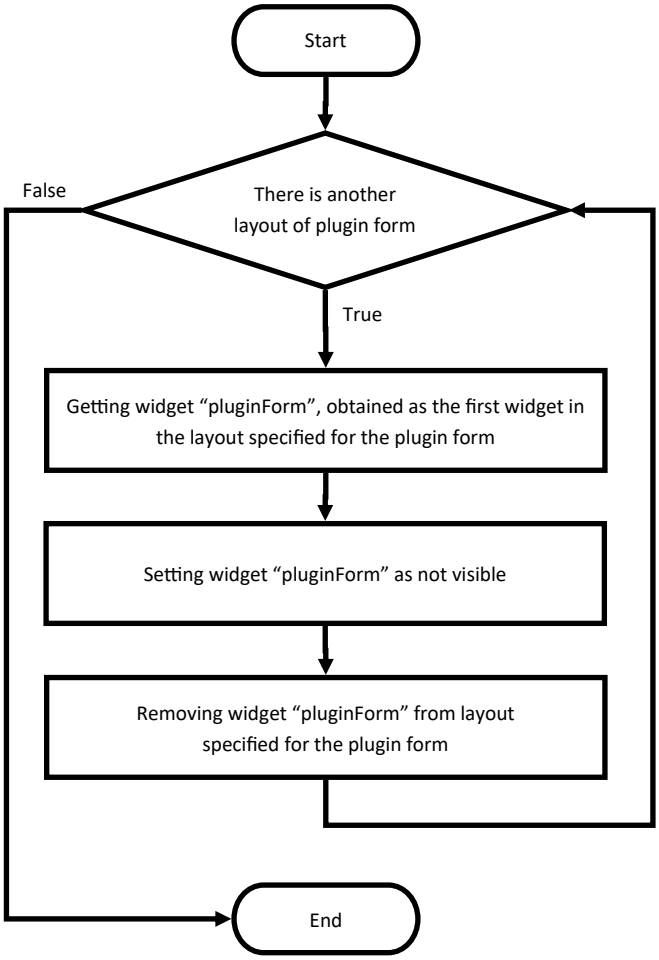
initializePage()

This method is called to prepare the wizard page form just before it is shown.



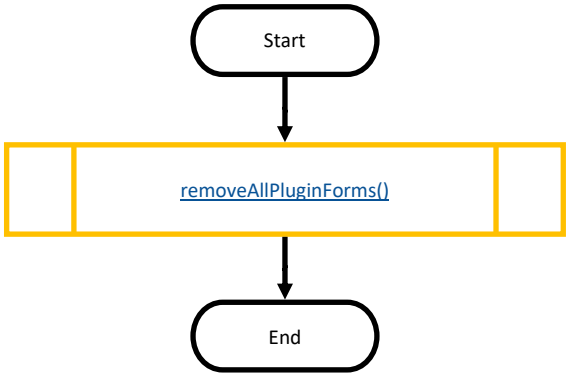
removeAllPluginForms()

This method is called to remove all plugin forms from the wizard page form.



terminatePage()

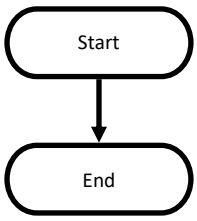
Method called to terminate the page after it is hidden.



updateStateButtons(newState)

Method called to update the checked status (pressed or not) of the state buttons.

:param newState: name of the new (current) state
:type newState: str



updateUI()

Updates the UI of the wizard page with the information contained in the wizard page model.