Jython JTextComponent 1

[JTextComponent 1][Jython Swing][Swing][Jython][Python]

方針

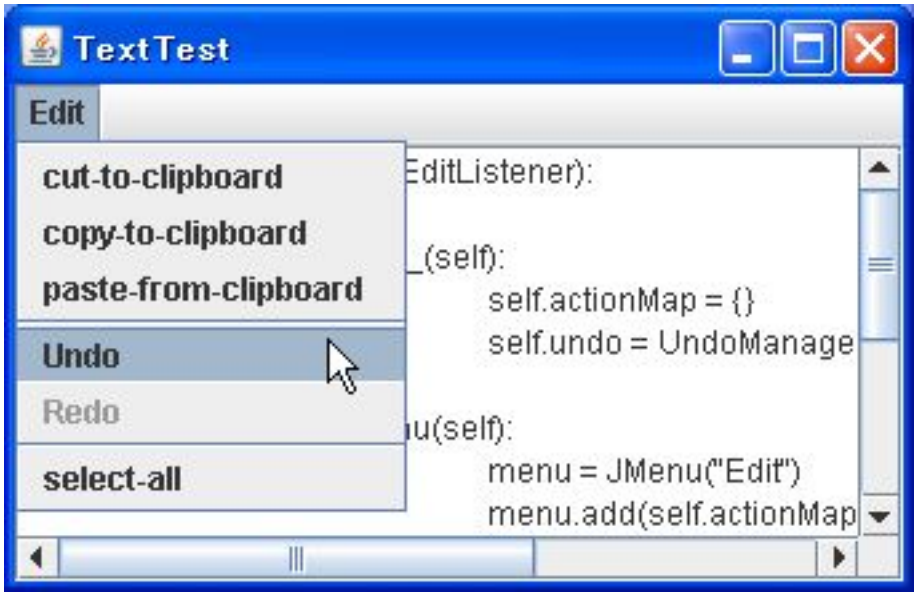
・半ば強引に、Swing で作成したサンプルを Jython に書き換えてみる。

内容

・メニューとアクションの割付
・キーボード押下との割付
・Undo、Redo の実装
・Document の利用
・Document のイベント感知

ソースコード

実行例



ソースコード

```
# -*- coding: utf-8 -*-

from java.lang import *
from java.awt import BorderLayout,Event
from java.awt.event import ActionEvent, KeyEvent
from java.util import HashMap, Map

from javax.swing import AbstractAction, Action, InputMap
from javax.swing import SwingUtilities, JFrame, JMenu, JMenuBar, JScrollPane, JTextArea, KeyStroke
from javax.swing.event import UndoableEditEvent, UndoableEditListener
from javax.swing.text import DefaultEditorKit, Document, JTextComponent
from javax.swing.undo import UndoManager

class JTextTest(object):

    def createUI(self):
        frame = JFrame("TextTest")
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)

        txtArea = JTextArea(10, 30)
        frame.contentPane.add(JScrollPane(txtArea), BorderLayout.CENTER)
```

```
        # TextComponent が持つ Action を Map に格納
        actionMap = {}
        for action in txtArea.actions:
            actionMap[action.getValue(Action.NAME)] = action

        undo = UndoManager()
        action_param = {}
        action_param['undo'] = undo
        redoAction = RedoAction(**action_param)
        action_param['redoAction'] = redoAction
        undoAction = UndoAction(**action_param)
        action_param['undoAction'] = undoAction

        # メニューを作成
        menu = JMenu("Edit")
        menu.add(actionMap.get(DefaultEditorKit.cutAction))
        menu.add(actionMap.get(DefaultEditorKit.copyAction))
        menu.add(actionMap.get(DefaultEditorKit.pasteAction))
        menu.addSeparator()
        menu.add(undoAction)
        menu.add(redoAction)
        menu.addSeparator()
        menu.add(actionMap.get(DefaultEditorKit.selectAllAction))

        mb = JMenuBar()
        mb.add(menu)
        frame.JMenuBar = mb

        # Document のイベント感知 (Undo Redo 管理を行う )
        doc = txtArea.document
        doc.addUndoableEditListener(UndoEventHandler(**action_param))

        # テキストの Undo、Redo にキーを割り当てる (Ctrl+z,Ctrl+y)
        inputMap = txtArea.inputMap
        undoKey = KeyStroke.getKeyStroke(KeyEvent.VK_Z,Event.CTRL_MASK)
        redoKey = KeyStroke.getKeyStroke(KeyEvent.VK_Y,Event.CTRL_MASK)
        inputMap.put(undoKey, undoAction)
        inputMap.put(redoKey, redoAction)

        frame.pack()
        frame.visible = True

class UndoEventHandler(UndoableEditListener):
        def __init__(self, undo, undoAction, redoAction):
            self.undo = undo
            self.undoAction = undoAction
            self.redoAction = redoAction

        def undoableEditHappened(self, e):
            self.undo.addEdit(e.edit)
            self.undoAction.updateState()
            self.redoAction.updateState()

class UndoAction(AbstractAction):
    def __init__(self, undo, redoAction):
        super(UndoAction, self).__init__("Undo")
        self.undo = undo
        self.redoAction = redoAction
        self.enabled = False

    def actionPerformed(self, e):
        try:
            self.undo.undo()
        except Exception, ex:
            ex.printStackTrace()
        self.updateState()
        # Redo を可能に
        self.redoAction.updateState()

    def updateState(self):
        self.setEnabled(self.undo.canUndo())

class RedoAction(AbstractAction):
    def __init__(self, undo):
        super(RedoAction, self).__init__("Redo")
        self.undo = undo
        self.enabled = False

    def actionPerformed(self, e):
        try:
            self.undo.redo()
```

```
            except Exception, ex:
                ex.printStackTrace()
            self.updateState()

        def updateState(self):
            self.setEnabled(self.undo.canRedo())

    class Invoker(Runnable):
        def run(self):
            jtt = JTextTest()
            jtt.createUI()

    SwingUtilities.invokeLater(Invoker())
```