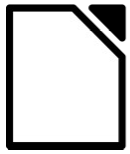


LibreOffice
The Document Foundation

UI Testing in LibreOffice

Xisco Fauli <xiscofauli@libreoffice.org>

Intro

- Implemented by Markus Mohrhard in 2016
- Consists of two parts:
  - C++ introspection library
    - find . -name "uiobject.hxx"
    - include/vcl/uitest/
  - Python framework. Inherits from python’s unitest framework
    - In uitest/

Limitations

- It doesn’t work on MacOS
- It works on Windows but it’s not enabled in CI
- On Linux it uses GEN enviroment.
- Slower than CppUnittest
- Base support
- File - Wizards

Makefile

```
$(eval $(call gb_UITest_UITest,sort))

$(eval $(call gb_UITest_add_modules,sort,$(SRCDIR)/sc/qa/uitest,\
	sort/ \
))

$(eval $(call gb_UITest_set_defs,sort, \
    TDOC="$(SRCDIR)/sc/qa/uitest/data" \
))
```

NOTE: oneprocess enabled by default
→ disable with $(eval $(call gb_UITest_avoid_oneprocess,search_replace))

Running the test (headless)

```
make UITest_sort
UITEST_TEST_NAME="sorting.CalcSorting.test_Sortingbuttons_detect_columnheaders"
```

```
make UITest_sort
```

Running the test

```
(cd sc && make -srj1 UITest_sort
UITEST_TEST_NAME="sorting.CalcSorting.test_Sortingbuttons_detect_columnheaders" SAL_USE_VCLPLUGIN=gen)
```

```
(cd sc && make -srj1 UITest_sort SAL_USE_VCLPLUGIN=gen)
```

More info:
https://wiki.documentfoundation.org/Development/UITests#Running_the_test

Running the test (bisect repos)

```
export SRCDIR=/home/xisco/bisect/bibisect-linux64-25.2
export PYTHONPATH=$SRCDIR/instdir/program:unotest/source/python
export URE_BOOTSTRAP=file://$SRCDIR/instdir/program/fundamentalrc
export TestUserDir=file:///tmp
export TDOC=$SRCDIR/sc/qa/uitest/data
export SAL_USE_VCLPLUGIN=gen
export LC_ALL=C

rm -rf /tmp/libreoffice/4

python3 /home/xisco/releases/uitest/test_main.py
--soffice=path:"$SRCDIR"/instdir/program/soffice --userdir=file:///tmp/libreoffice/4 --
file="/home/xisco/libreoffice/sc/qa/uitest/sort/sorting.py"
```

Logger

LO_COLLECT_UIINFO="test.log" SAL_USE_VCLPLUGIN=gen instdir/program/soffice

```
Start writer
Send UNO Command (".uno:UpdateInputFields")
Send UNO Command (".uno:InsertTable")
Open Modal InsertTableDialog
Action on element: nameedit with action : 16
Type on 'nameedit' {"KEYCODE": "CTRL+a"} from InsertTableDialog
Select in 'nameedit' {"FROM": "0", "TO": "6"} from InsertTableDialog
Type on 'nameedit' {"KEYCODE": "SHIFT+T"} from InsertTableDialog
Action on element: nameedit with action : TYPE
Action on element: nameedit with action : 16
Type on 'nameedit' {"TEXT": "e"} from InsertTableDialog
Action on element: nameedit with action : TYPE
Action on element: nameedit with action : 16
Type on 'nameedit' {"TEXT": "s"} from InsertTableDialog
Action on element: nameedit with action : TYPE
Action on element: nameedit with action : 16
Type on 'nameedit' {"TEXT": "t"} from InsertTableDialog
Action on element: nameedit with action : TYPE
Action on element: nameedit with action : 16
Type on 'nameedit' {"KEYCODE": "RETURN"} from InsertTableDialog
Create Table with  Columns : 2 , Rows : 2
Close Dialog
Open Modal QuerySaveDialog
Click on 'discard' from QuerySaveDialog
Close Dialog
```

Sample

```
def test_changeName(self):
    with self.ui_test.create_doc_in_start_center("writer") as document:

        with self.ui_test.execute_dialog_through_command(".uno:InsertTable") as xDialog:

            xNameEdit = xDialog.getChild("nameedit")

            xNameEdit.executeAction("TYPE", mkPropertyValues({"KEYCODE":"CTRL+A"}))
            xNameEdit.executeAction("TYPE", mkPropertyValues({"TEXT": "Test"}))

            xColSpin = xDialog.getChild("colspin")
            xColSpin.executeAction("UP", tuple())

            xRowSpin = xDialog.getChild("rowspin")
            xRowSpin.executeAction("UP", tuple())

            self.assertEqual("Test", get_state_as_dict(xNameEdit)["Text"])
            self.assertEqual("3", get_state_as_dict(xColSpin)["Text"])
            self.assertEqual("3", get_state_as_dict(xRowSpin)["Text"])

        tables = document.getTextTables()

        self.assertEqual("Test", tables[0].getName())
        self.assertEqual(3, len(tables[0].getRows()))
        self.assertEqual(3, len(tables[0].getColumns()))
```

Debugging

- print(xDialog.getChildren())

```
('3', '4', 'InsertTableDialog', 'box3', 'cancel', 'colspin', 'dialog-action_area1', 'dialog-vbox1',
'dontsplitcb', 'formatlbinstable', 'frame2', 'frame3', 'grid1', 'grid2', 'grid4', 'headercb', 'help', 'label1',
'label2', 'label3', 'lbTableStyle', 'lbwarning', 'nameedit', 'ok', 'previewinstable', 'repeatcb',
'repeatgroup', 'repeatheaderafter', 'repeatheaderspin', 'rowspin', 'stylesframe')
```

- print(get_state_as_dict(xNameEdit))

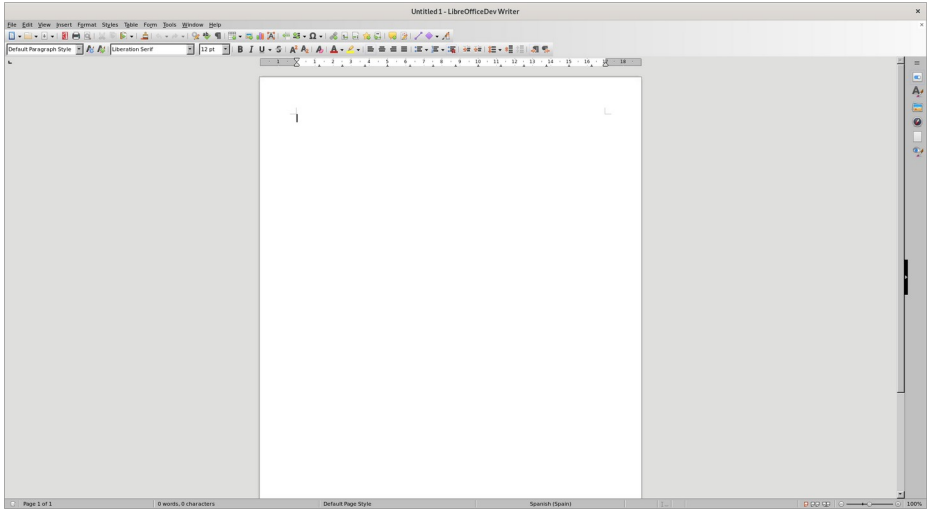
```
{'AbsPosition': '832x382', 'DisplayText': '', 'Enabled': 'true', 'HasFocus': 'true', 'ID': 'nameedit',
'MaxTextLength': '2147483647', 'Parent': 'grid2', 'QuickHelpText': '', 'ReallyVisible': 'true',
'RelPosition': '62x0', 'SelectedText': '', 'Size': '327x23', 'Text': 'Test', 'Visible': 'true', 'WindowType':
'147'}
```

Create empty document

```
with self.ui_test.create_doc_in_start_center("writer") as document:
```

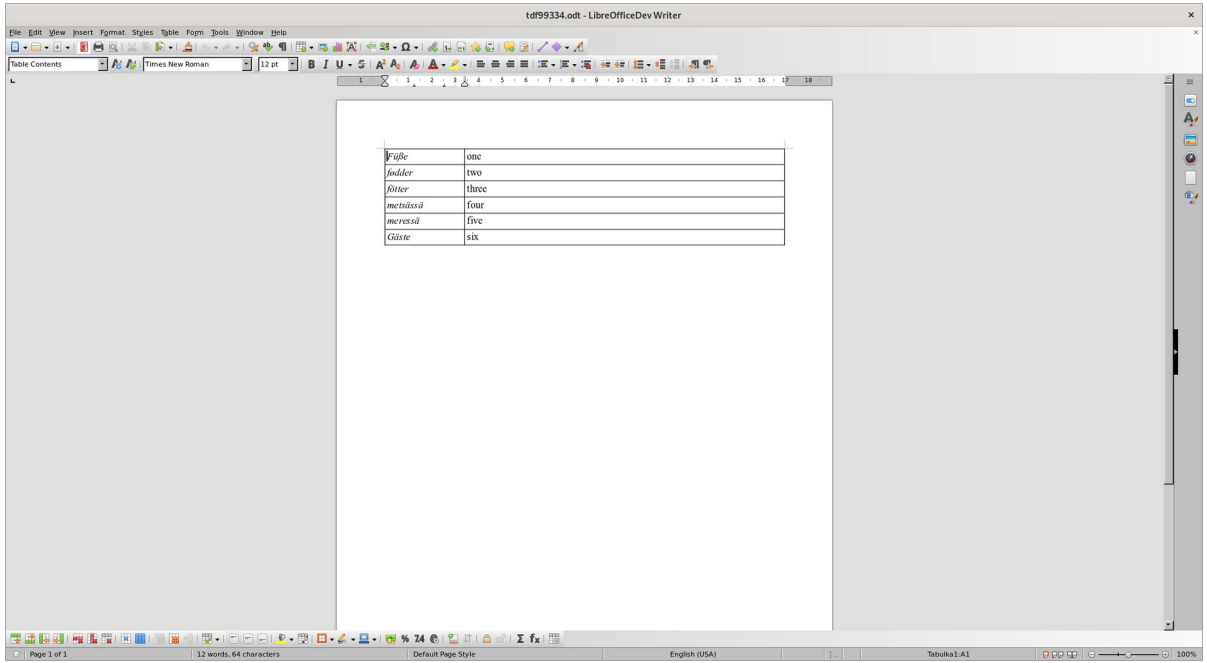
```
with self.ui_test.create_doc_in_start_center("calc") as document:
```

```
with self.ui_test.create_doc_in_start_center("impress") as document:
```



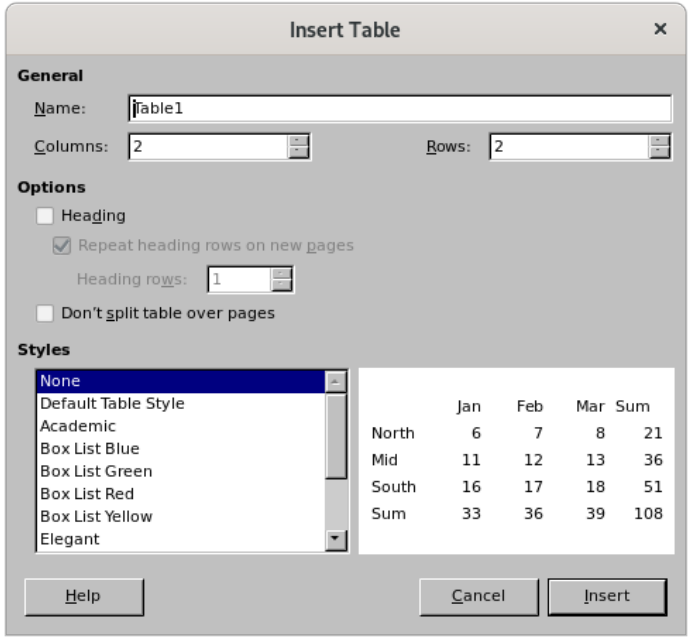
load document

```
from uitest.uihelper.common import get_url_for_data_file

with self.ui_test.load_file(get_url_for_data_file("tdf99334.odt")) as document:
```



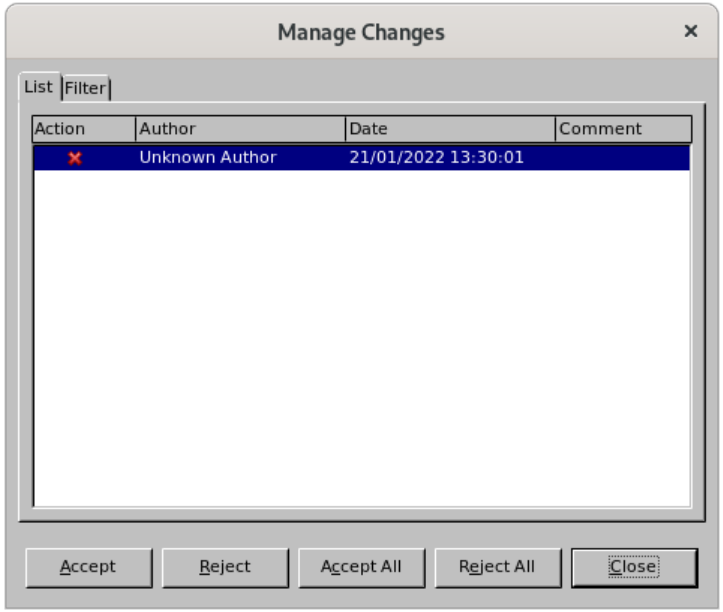
Execute dialog

```
with self.ui_test.execute_dialog_through_command(".uno:InsertTable") as xDialog:
```



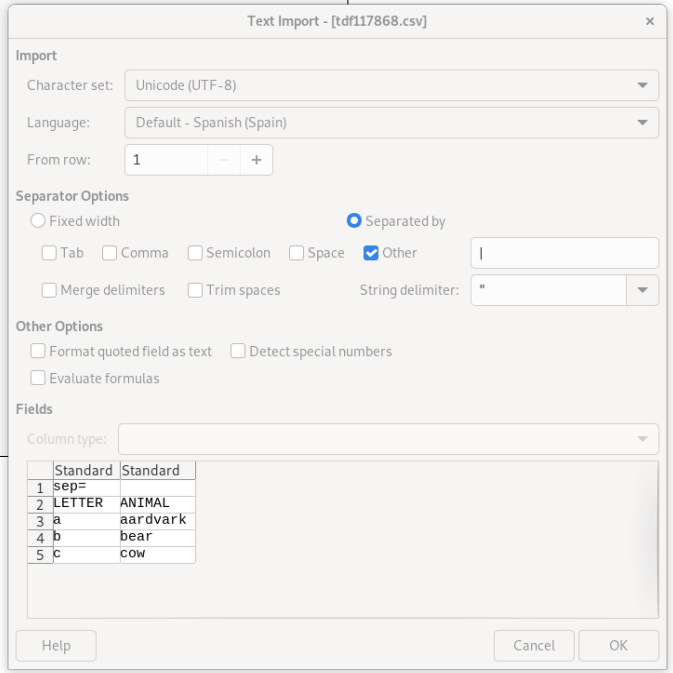
Execute modeless dialog

```
with self.ui_test.execute_modeless_dialog_through_command(".uno:AcceptTrackedChanges",
close_button="close"):
```



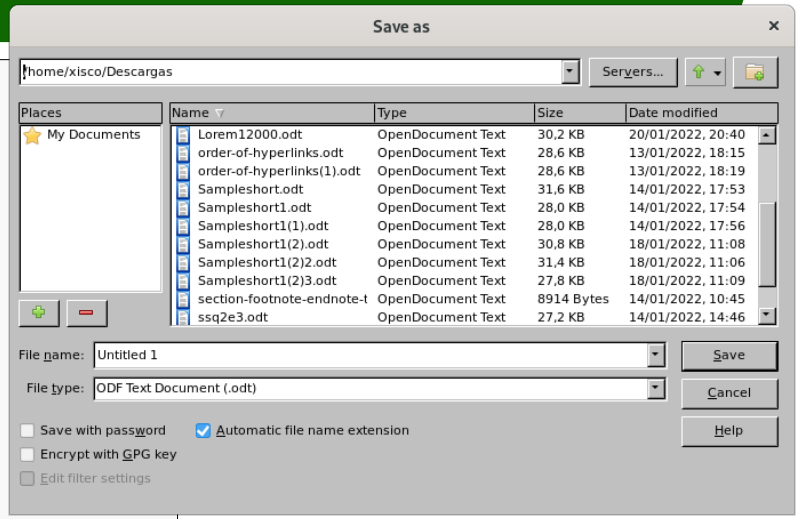
load document with dialog

```
from uitest.uihelper.common import get_url_for_data_file

with self.ui_test.execute_dialog_through_command(".uno:Open", close_button="") as
xOpenDialog:
    xFileName = xOpenDialog.getChild("file_name")
    xFileName.executeAction("TYPE", mkPropertyValues({"TEXT":
get_url_for_data_file("tdf145326.odt")}))

    xOpenBtn = xOpenDialog.getChild("open")
    with self.ui_test.wait_until_component_loaded():
        with self.ui_test.execute_blocking_action(xOpenBtn.executeAction,
                args=('CLICK', ()), close_button="yes"):
            pass
```



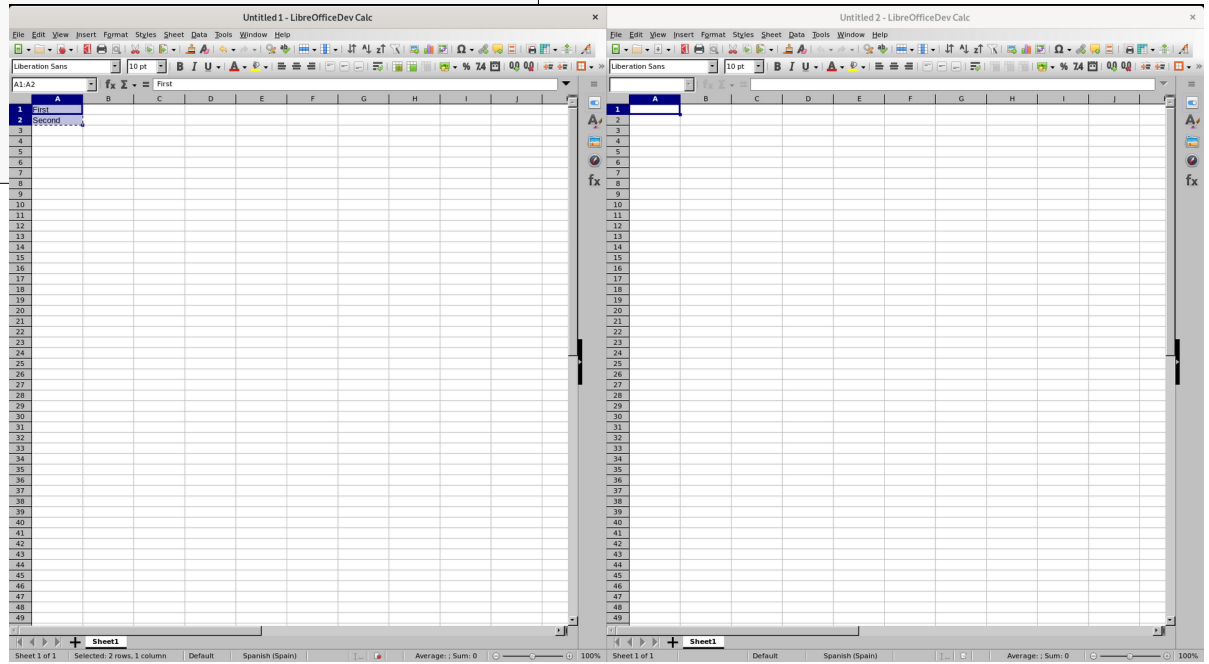
Save document and reload

```
from org.libreoffice.unotest import systemPathToFileUrl
from tempfile import TemporaryDirectory
import os.path

with TemporaryDirectory() as tempdir:
    xFilePath = os.path.join(tempdir, "tdf119661-tmp.odt")

    with self.ui_test.create_doc_in_start_center("writer"):

        with self.ui_test.execute_dialog_through_command(".uno:SaveAs",
close_button="open") as xDialog:

            xFileName = xDialog.getChild("file_name")
            xFileName.executeAction("TYPE", mkPropertyValues({"KEYCODE":"CTRL+A"}))
            xFileName.executeAction("TYPE", mkPropertyValues({"KEYCODE":"BACKSPACE"}))
            xFileName.executeAction("TYPE", mkPropertyValues({"TEXT": xFilePath}))

    with self.ui_test.load_file(systemPathToFileUrl(xFilePath)):
```



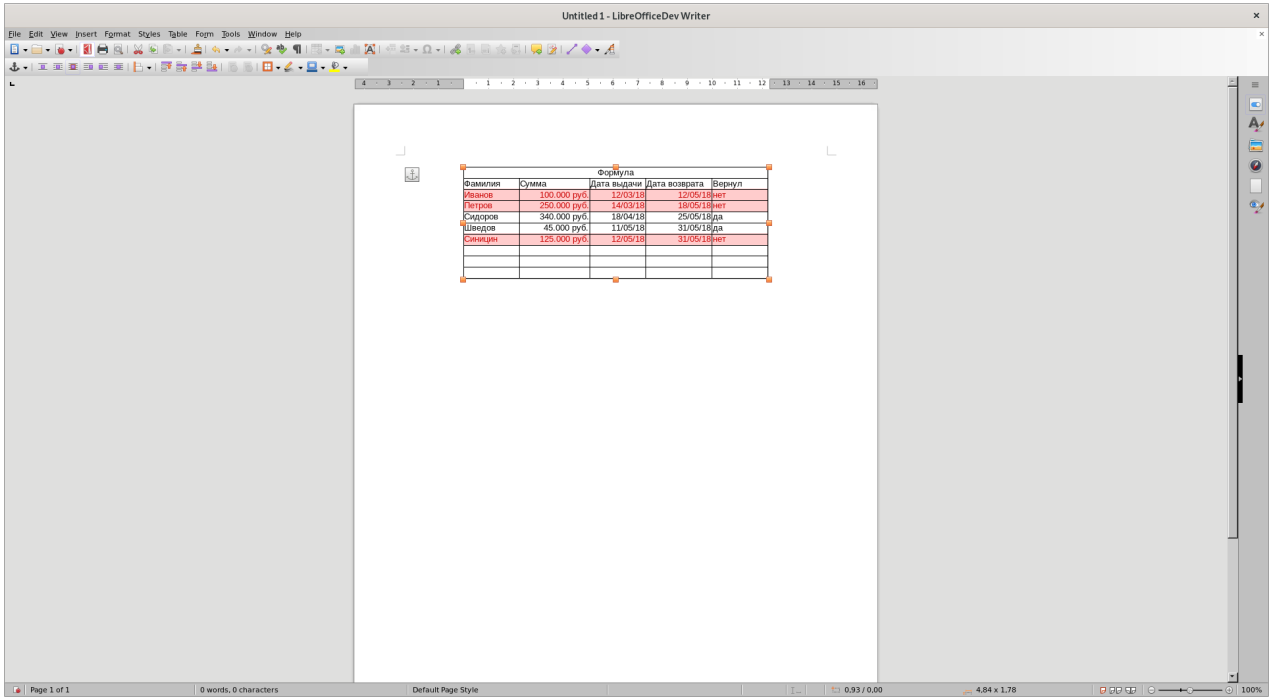
Different documents

```
with self.ui_test.create_doc_in_start_center("calc") as document:

    with self.ui_test.load_empty_file("calc") as document2:

        frames = self.ui_test.get_frames()
        # switch view to the first document
        frames[0].activate()

        # switch view back to the second document
        frames[1].activate()
```



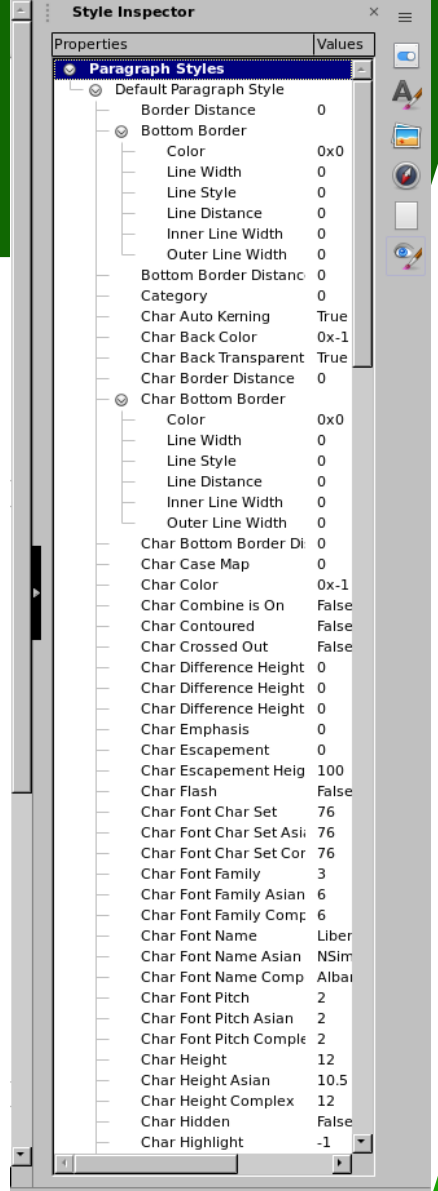
Interaction between components

```
with self.ui_test.create_doc_in_start_center("calc") as document:
    ...
with self.ui_test.load_empty_file("writer") as writer_document:
    ...
```



Sidebar

```
xWriterDoc = self.xUITest.getTopFocusWindow()
xWriterEdit = xWriterDoc.getChild("writer_edit")

self.xUITest.executeCommand(".uno:Sidebar")
xWriterEdit.executeAction("SIDEBAR", mkPropertyValues(
        {"PANEL": "InspectorTextPanel"}))

# Close sidebar after using it
self.xUITest.executeCommand(".uno:Sidebar")
```

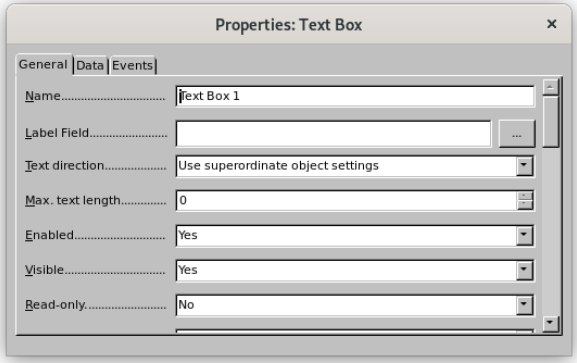


Navigator

```
xWriterDoc = self.xUITest.getTopFocusWindow()
xWriterEdit = xWriterDoc.getChild("writer_edit")

self.xUITest.executeCommand(".uno:Sidebar")

xWriterEdit.executeAction("SIDEBAR",
mkPropertyValues({"PANEL": "SwNavigatorPanel"}))

# wait until the navigator panel is available
xNavigatorPanel =
self.ui_test.wait_until_child_is_available('NavigatorPanel')

xContentTree = xNavigatorPanel.getChild("contenttree")

# Close sidebar after using it
self.xUITest.executeCommand(".uno:Sidebar")
```

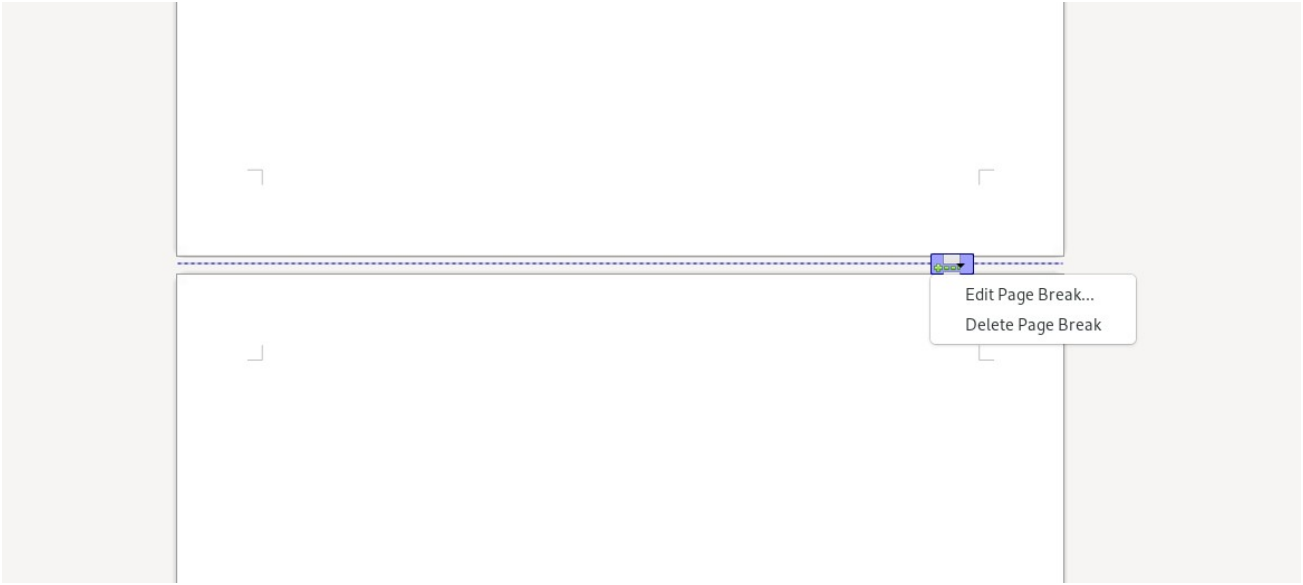
Forms

```
self.xUITest.executeCommand(".uno:JumpToNextFrame")

with self.ui_test.execute_modeless_dialog_through_command(".uno:ControlProperties",
close_button=""):
    xChild = self.ui_test.wait_until_child_is_available('listbox-Text type'))
```



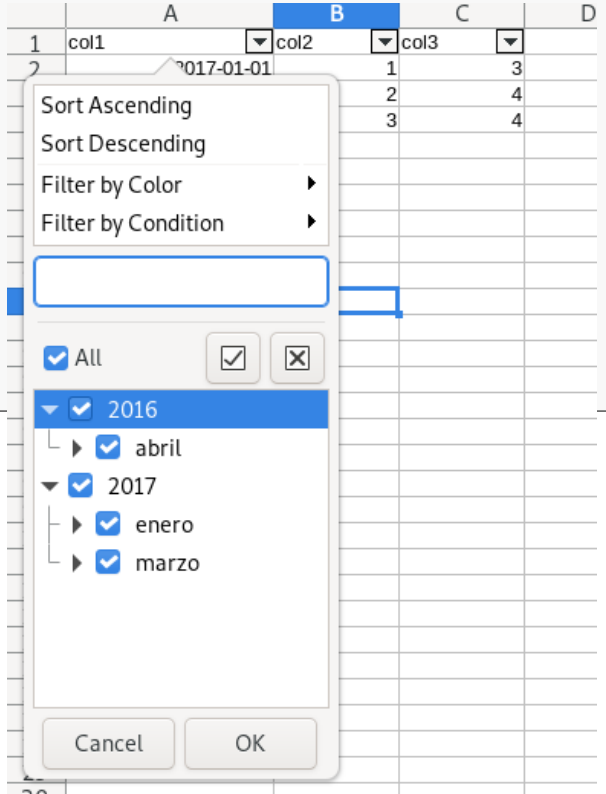
Pagebreak

```
self.xUITest.executeCommand(".uno:InsertPagebreak")

xPageBreak = self.ui_test.wait_until_child_is_available('PageBreak')

with self.ui_test.execute_dialog_through_action(xPageBreak, "EDIT") as xDialog:
```

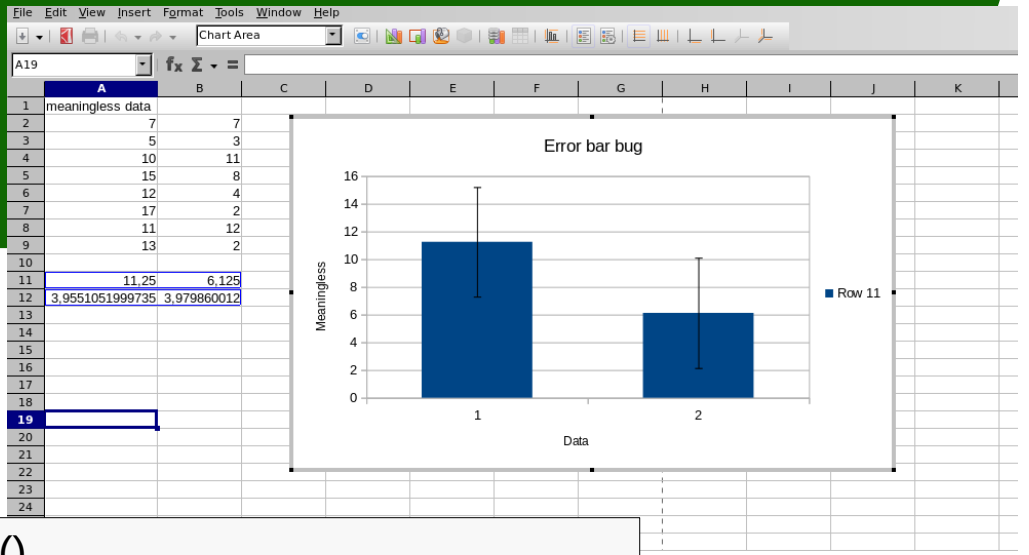


Autofilter

```
xGridWin = self.xUITest.getTopFocusWindow().getChild("grid_window")
xGridWin.executeAction("LAUNCH", mkPropertyValues({"AUTOFILTER": "", "COL": "0", "ROW": "0"}))

xFloatWindow = self.xUITest.getFloatWindow()
xCheckListMenu = xFloatWindow.getChild("FilterDropDown")
xTreeList = xCheckListMenu.getChild("check_tree_box")
xFirstEntry = xTreeList.getChild("0")
xFirstEntry.executeAction("CLICK", tuple())

xOkBtn = xFloatWindow.getChild("ok")
xOkBtn.executeAction("CLICK", tuple())
```



Charts




```
xCalcDoc = self.xUITest.getTopFocusWindow()
gridwin = xCalcDoc.getChild("grid_window")

gridwin.executeAction("SELECT", mkPropertyValues({"OBJECT": "Object 1"}))
gridwin.executeAction("ACTIVATE", tuple())

xChartMainTop = self.xUITest.getTopFocusWindow()
xChartMain = xChartMainTop.getChild("chart_window")

xSeriesObj =  xChartMain.getChild("CID/D=0:CS=0:CT=0:Series=0")
with self.ui_test.execute_dialog_through_action(xSeriesObj, "COMMAND",
    mkPropertyValues({"COMMAND": "FormatYErrorBars"})) as xDialog:
```

New Wrapper

- sc/source/ui/view/gridwin.cxx

```
FactoryFunction ScGridWindow::GetUITestFactory() const
{
    return ScGridWinUIObject::create;
}
```

New Wrapper

- sc/source/ui/inc/uiobject.hxx

```
class ScGridWinUIObject : public WindowUIObject
{
    VclPtr<ScGridWindow> mxGridWindow;

public:
    ScGridWinUIObject(const VclPtr<ScGridWindow>& xGridWin);

    virtual StringMap get_state() override;

    virtual void execute(const OUString& rAction, const StringMap& rParameters) override;

    virtual std::unique_ptr<UIObject> get_child(const OUString& rID) override;

    static std::unique_ptr<UIObject> create(vcl::Window* pWindow);

    virtual std::set<OUString> get_children() const override;

protected:
    virtual OUString get_name() const override;
}
```

New Wrapper

- sc/source/ui/uitest/uiobject.cxx

```
StringMap ScGridWinUIObject::get_state()
{
    StringMap aMap = WindowUIObject::get_state();

    aMap[u"SelectedTable"_ustr] = OUString::number(mxGridWindow->getViewData().GetTabNo());
    aMap[u"CurrentColumn"_ustr] = OUString::number(mxGridWindow->getViewData().GetCurX());
    aMap[u"CurrentRow"_ustr] = OUString::number(mxGridWindow→getViewData().GetCurY());
    ….
    return aMap;
}
```

New Wrapper

- sc/source/ui/uitest/uiobject.cxx

```
void ScGridWinUIObject::execute(const OUString& rAction,
        const StringMap& rParameters)
{
    if (rAction == "SELECT")
    {
        bool bExtend = false;
        if (rParameters.find(u"EXTEND"_ustr) != rParameters.end())
        {
            auto itr = rParameters.find(u"EXTEND"_ustr);
            if (itr->second.equalsIgnoreAsciiCase("true") || itr->second == "1")
                bExtend = true;
        }
        ...
}
```

Links

- https://wiki.documentfoundation.org/Development/UITests
- https://mmohrhard.wordpress.com/2016/09/07/ui-testing-in-libreoffice/
- https://mmohrhard.wordpress.com/2016/09/10/writing-a-libreoffice-calc-ui-test/
- https://mmohrhard.wordpress.com/2016/12/02/libreoffice-ui-test-tutorial-part-2-improve-the-introspection-library/

Thanks

Xisco Fauli
The Document Foundation
xiscofauli@libreoffice.org