Python Scripts made simple w/ IDE_utils
Hi

- Alain Romedenne
- alain.romedenne@libreoffice.org
- LibreOfficiant
- LOfficiant
Agenda

- Intro
  - APSO, MRI/xRay extensions
  - IDE’s Benefits
  - Steps: Start, ..., Stop
  - IDE_utils features

- IDEs
  - IDE’s docstrings
  - Geany, PyZo, PyCharm

- Demos
  - Lorem Ipsum
  - Writer
  - @retry decorator
  - logging

- Links
but wait ...

- must-have APSO extension
- MRI / xRay object browsers

I am your guide to demos
Demo – APSO extension
IDE Benefits

- Objects explorer/colorizer
- Code completion
- Debugging
- PEP compliance
- Test Driven Dev (TDD)
- Version Control (Git …)
and many more …
IDE_utils steps

- Start libO
- Connect to libO service
- XSCRIPTCONTEXT Adaptor
- Run macro
- Stop libO

+ No compulsory step

Runner context mgr
connect()
surrogate object
...
Runner context mgr
IDE_utils features

- Optional steps
- x-platforms, x-products
- On-demand startup options
- pipe/socket
- Service / context pooling
- KISS
IDE_utils docstrings

This module provides:

1. a substitute to `XSCRIPTCONTEXT` (LibreOffice) built-in, to be used within Python IDE such as Anacoda, Geany, KDevelop, PyCharm, etc.
2. a (LibreOffice) Runner context manager with `start`, `stop` paradigms to launch `Office` instances and to facilitate `setup`, `teardown` unit testing stops

Instructions:

1. Copy this module into your `<OFFICE>/program/directory`
2. OR include it into your IDE project directory
3. Include one of the below examples into your Python macro
4. Run your (LibreOffice) macro from your preferred IDE

Examples:

```python
import uno
def my_1st_macro(): pass # Your code goes here
def my_2nd_macro(): pass # Your code goes here
def my_own_macro(): pass # Your code goes here
```

# Runners.jsen argument file /ORU (LibreOffice)Office pipe
Demo 1 – Lorem Ipsum API ‘MsgBox’

The IDE starts a named pipe - as a LibreOffice background service -
- connects to it,
- sets XSCRIPTCONTEXT substitute,
- executes the macro and
- stops LibreOffice service
Demo 2 – ‘Hello Writer’

- LibreOffice starts first as a ‘socket’ service
- The IDE connects to port #2017, sets a XSCRIPTCONTEXT substitute and run the script:
  - which creates a Writer doc and updates it
- LibreOffice remains active
Demo 3 – @retry decorator

- The IDE waits for ‘LinusTorvalds’ named pipe, attempts to connect are reported in console ...
- LibreOffice starts as ‘LinusT...’ piped service
- The IDE connects, the script executes
Demo 4 – Failed session

- The IDE looks for LibreOffice as a service
- The IDE attempts to connect twice
- The IDE throws an exception
Demo 5 – Logging execution

- The IDE starts LibreOffice as a service
- The IDE connects to it
- The IDE sets XSCRIPTCONTEXT
- The IDE executes the macro
- The IDE stops LibreOffice as a service
e-References

- LibO Python Wiki, LibO start options
- LibO API Documentation
- Geany, Pyzo, PyCharm
- IDE_utils
<table>
<thead>
<tr>
<th>Feature</th>
<th>IDE 1</th>
<th>IDE 2</th>
<th>IDE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>docstrings</td>
<td>🟢</td>
<td>🟢</td>
<td></td>
</tr>
<tr>
<td>Lorem ipsum</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
</tr>
<tr>
<td>Hello Writer</td>
<td></td>
<td></td>
<td>🟢</td>
</tr>
<tr>
<td>Report connections</td>
<td></td>
<td></td>
<td>🟢</td>
</tr>
<tr>
<td>Throwing exception</td>
<td>🟢</td>
<td></td>
<td>🟢</td>
</tr>
<tr>
<td>Logging execution</td>
<td>🟢</td>
<td></td>
<td>🟢</td>
</tr>
</tbody>
</table>