Showing Up for Python in GNOME

Dan Yeaw (@danyeaw:gnome.org)
About Me

Dan Yeaw (pronounced: Yaw)

Originally from California, now live in Michigan

Gaphor, Gvsbuild, PyGObject
Unleashing Interests with Python

```python
>>> import pypokedex
>>> pokemon = pypokedex.get(name="Decidueye")
>>> pokemon.name
'decidueye'
>>> pokemon.types
['grass', 'ghost']
>>> pokemon.base_stats
BaseStats(hp=78, attack=107, defense=75, sp_atk=100, sp_def=100)
```
PyGObject is the GTK and related library bindings for Python.
On PyGObject

The current state of the Python bindings for GObject-based libraries is making it really hard to recommend using Python as a language for developing GTK and GNOME applications.

Emmanuele Bassi (2022)
Commits Over Time

Figure 1: PyGObject Commits Over Time

- 💡 Major contributors left the project over time.
- 🐐 Christoph Reiter heroically held things together since 2017.
- 🐐 However, the number of changes started to fall off, especially after 2020.
Getting Involved in an Undermaintained Project

- Contributing to an undermaintained project can be difficult
- Each extra contribution is placing a burden on the developer
- Timely feedback to contributions is often not possible
- To outsiders, the GNOME project can feel hard to join, especially in these undermaintained areas
Community Building

- The GNOME Project Handbook greatly improves clarity on how to get involved
- The GNOME Foundation could also take a greater role
The State of Python in GNOME
Issue and Merge Request Triage

- Closed about 200 issues
- Total issue count went from over 300 to 175
- Open or draft merge requests went from 30 to 19
Fundamental Types

- Most objects inherit from GObject
- GtkExpression, GtkRenderNode, and GtkEvent do not
- These are defined as a GObject::TypeInstance
# Workbench Column View Example

**Column View**

Arrange a large and dynamic list of items in columns

[API Reference](#)  [Documentation](#)

<table>
<thead>
<tr>
<th>Title</th>
<th>Author</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitter</td>
<td>Akwaeke Emezi</td>
<td>2022</td>
</tr>
<tr>
<td>Itinerary of a Dramatist</td>
<td>Rodolfo Usigl</td>
<td>1940</td>
</tr>
<tr>
<td>Like Water for Chocolate</td>
<td>Laura Esquivel</td>
<td>1989</td>
</tr>
<tr>
<td>Saying Yes</td>
<td>Griselda Gambaro</td>
<td>1981</td>
</tr>
<tr>
<td>The Timeless Way of Building</td>
<td>Cristopher Alexander</td>
<td>1979</td>
</tr>
</tbody>
</table>
Sorting

- Gtk provides an easy way to sort columns
- Create a Sorter and then pass in a Gtk.PropertyExpression
- this -> item -> property
- Unfortunately, it isn't so easy without Fundamental Types
column_view = workbench.builder.get_object("column_view")
col1 = workbench.builder.get_object("col1")
col2 = workbench.builder.get_object("col2")
col3 = workbench.builder.get_object("col3")

model_func = lambda _item: None
tree_model = Gtk.TreeListModel.new(data_model, False, True, model_func)
tree_sorter = Gtk.TreeListRowSorter.new(column_view.get_sorter())
sorter_model = Gtk.SortListModel(model=tree_model, sorter=tree_sorter)
selection = Gtk.SingleSelection.new(model=sorter_model)
column_view.set_model(model=selection)
def str_sorter(object_a, object_b, column):
    a = getattr(object_a, column).lower()
    b = getattr(object_b, column).lower()
    return (a > b) - (a < b)

def int_sorter(object_a, object_b, column):
    a = getattr(object_a, column)
    b = getattr(object_b, column)
    return (a > b) - (a < b)

col1.set_sorter(Gtk.CustomSorter.new(str_sorter, "title"))
col2.set_sorter(Gtk.CustomSorter.new(str_sorter, "author"))
col3.set_sorter(Gtk.CustomSorter.new(int_sorter, "year"))
Sorting with Expressions

col1_exp = Gtk.PropertyExpression.new(Book, None, "title")
col2_exp = Gtk.PropertyExpression.new(Book, None, "author")
col3_exp = Gtk.PropertyExpressionExpression.new(Book, None, "year")

col1.sorter = Gtk.StringSorter.new(col1_exp)
col2.sorter = Gtk.StringSorter.new(col2_exp)
col3.sorter = Gtk.NumericSorter.new(col3_exp)
PyGObject is a Python package which provides bindings for GObject based libraries such as GTK, GStreamer, WebKitGTK, GLib, GIO and many more.

It supports Linux, Windows, and macOS and works with Python 3.9+ and PyPy3. PyGObject, including this documentation, is licensed under the LGPLv2.1+.

Homepage

If you want to write a Python application for GNOME or a Python GUI application using GTK, then PyGObject is the way to go. To get started, check out the “GNOME Developer Documentation”. For more information on specific libraries, check out the GNOME Python API documentation.

```python
import sys
import gi
```
Packaging and Development Environment Improvements
Legacy Packaging

- setup.py requires arbitrary code execution
- pyproject.toml is a more explicit way to declare dependencies

The steps to build a Python project then can be separated:

1. Checkout the project
2. Install the build system
3. Execute the build
meson-python

meson-python is a build backend for Python leveraging Meson

**pyproject.toml**

```
[tool.meson-python.args]
setup = ['-Dtests=false', '-Dwheel=true', '--wrap-mode=nofallback']

[build-system]
build-backend = "mesonpy"
requires = ['meson-python>=0.12.1', 'pycairo>=1.16']
```

**Build and Test**

```
$ meson setup _build
$ meson test -C _build
```
Figure 2: Packaging Categorization by Anna-Lena Popkes
Modernize API Docs

Modernize building docs using GI-DocGen and Sphinx

Template

class Template(**kwargs)

Methods

classmethod from_file(filename)

Parameters: filename

classmethod from_resource(resource_path)

Parameters: resource_path
Main Branch

- Small change to rename the primary branch to main
- Improves exclusivity and standardization with other GNOME projects
Overview of Async IO

- Cooperative multitasking
- Scheduled concurrently, but not actually run at the same time
- Can provide large speedups if waiting on slower tasks
Async IO with my Kids
Async IO in Python

```
import asyncio

async def hello():
    print('Hello ...')
    await asyncio.sleep(1)
    print('... World!')

async def main():
    await asyncio.gather(hello(), hello())

asyncio.run(main())

Hello ...
Hello ...
... World!
... World!
```
Python Async with Gbulb

Gbulb uses the full GLib EventLoop

```python
import asyncio, gbulb

gbulb.install(gtk=True)

loop = asyncio.get_event_loop()
loop.run_forever(application=my_gapplication_object)
```
async def idle_test():
    bus = await Gio.bus_get(Gio.BusType.SYSTEM)
    print(
        await bus.call(
            "org.freedesktop.DBus",
            "/org/freedesktop/DBus",
            "org.freedesktop.DBus",
            "ListNames",
            None, None, 0, -1,
        )
    )
Experimental: Async IO Integration

```python
policy = GLibEventLoopPolicy()
asyncio.set_event_loop_policy(policy)
loop = policy.get_event_loop()
loop.run_until_complete(idle_test())

```
The Future
Wheels for Windows

- Python 3.8+ no longer loads DLLs on the path
- Building GTK using MSVC with `pip install pygobject` doesn't work for getting started
- Solution: build wheels of PyGObject with the DLLs included
Port to libgirepository-2.0

libgirepository is now part of GLib
The main enhancement is it now uses GObject.TypeInstance instead of C struct aliasing
Move API Docs

- Combine and merge the API docs to https://pygobject.gnome.org
- This would finish centralizing all docs
PyGObject is a great choice for building apps in GNOME.
Call to Action

https://gitlab.gnome.org/GNOME/pygobject

- Contributions of any kind will help continue to help the community thrive
- Submit and help triage issues
- Continue to help us improve the docs
- Help us fix bugs and implement features
- Add examples to Workbench
- Build projects with PyGObject
- Chat with us at #python:gnome.org
Wrap Up

Thanks!

Thank you so much to everyone who has contributed to PyGObject, and special thanks to Christoph Reiter and Arjan Molenaar who help maintain it.

License

Creative Commons Attribution-Noncommercial (CC BY-NC)

Slides

https://github.com/danyeaw/presentations/tree/main/showing-up-for-python-in-gnome
Questions?