Talk to D-Bus from a Python application

Vendula Poncova
Brussels, 2023
D-Bus

protocol - bus daemon - inter-process
dasbus library

Python - GLib - D-Bus
Part 1: Client
Create a D-Bus proxy

Specify the D-Bus name of the service and D-Bus path of the object.

```python
bus = SessionMessageBus()
proxy = bus.get_proxy(
    "org.example.Chat",
    "/org/example/Chat/Rooms/3"
)
```
Get a D-Bus property

Get a property of the D-Bus proxy.

```python
print(proxy.Name)
```
Call a D-Bus method

Call a method of the D-Bus proxy.

```python
proxy.SendMessage("Hello World!")
```
```python
def callback(message):
    print(message)

proxy.MessageReceived.connect(callback)
```
Run the event loop

Process asynchronous communication.

```python
loop = EventLoop()
loop.run()
```
Part 2: Service
Register a D-Bus service

Specify a D-Bus name of the service.

```python
bus = SessionMessageBus()
bus.register_service("org.example.Chat")
```
bus.publish_object(
    "/org/example/Chat/Rooms/3",
    Room()
)
Start the event loop

Start to handle incoming D-Bus calls.

```python
loop = EventLoop()
loop.run()
```
Part 3: Object
<?xml version="1.0" ?>

<node>
  <interface name="org.example.Chat.Room">
    <method name="SendMessage">
      <arg direction="in" type="s"/>
    </method>
  </interface>
</node>
Use a decorator to generate the XML.

```python
@dbus_interface("org.example.Chat.Room")
class Room(object):
```
Define a D-Bus method

Use type hints to define types.

```python
def SendMessage(self, msg: str):
    print(msg)
```
Define a D-Bus property

Use type hints to define types.

```python
@property
def Name(self) -> Str:
    return "Room 3"
```
Define a D-Bus signal

Specify the arguments and types.

```python
@dbus_signal
def MessageReceived(self, msg: Str):
    pass
```
Part 4: Features
Manage D-Bus names

Pre-define D-Bus names and paths of services, objects and interfaces.

```python
proxy = CHAT.get_proxy('ROOM_3')
```
Generate D-Bus paths

Manage a group of publishable objects.

path = ROOM_CONTAINER.to_object_path(room)
Handle D-Bus errors

Map names of DBus errors to exceptions.

```python
try:
    proxy.SendMessage(...)
except ChatError:
    ...
```
Handle D-Bus structures

Represent dictionaries of variants by Python objects.

data = UserData.from_structure(structure)
print(data.name)
Use Unix file descriptors

Send and receive Unix file descriptors.

```python
proxy = bus.get_proxy(
    ...,
    client=GLibClientUnix
)
```
I don't need X

Features are optional.
I need Z

Classes are replaceable.
https://github.com/rhinstaller/dasbus