



pywinauto: кроссплатформенность и как залезть в душу к WPF и Qt

Василий Рябов

Ведущий программист Positive Technologies





pywinauto: что это?

- [pywinauto/pywinauto](https://pypi.org/project/pywinauto/): 6.2k+ звёзд
 - Автоматизация десктопного GUI: раньше только Windows
 - На чистом Python, лицензия: BSD 3-clause
- Можно пробовать **Linux** и **mac OS** (бета и альфа качество)
- **Кастомные** элементы **WPF** и **QML/Quick** (бета и альфа качество)
 - замена TestComplete и Squish (ушли из России)
- Есть подходы к Java AWT и Tkinter (нет в pywinauto)
- Кейсы, мемы, кровь и слёзы



pywinauto (предыдущие серии)

- Mark Mc Mahon и Win32 API (на Google Code): 2006 – 2010 (0.1.0)
- Переделка под 64-битность с 2010, на GitHub с 2015 (0.1.1 – 0.5.4)
- Поддержка MS UI Automation API: 2016-2019 (0.6.0 – 0.6.8)
- Большой рефакторинг
- Поддержка Linux, macOS
- Поддержка custom controls: WPF и Qt, QML/Quick (релиз 1.0.0b1)

Организационная история в open source



- <https://habr.com/ru/companies/pt/articles/903800/>



- Крупный контрибьютор: @airelil (Валентин, Окленд, Н. Зеландия)
- 10+ студентов (ННГУ им. Лобачевского, Polytech Nantes), 2015 – ...
- Небольшие патчи от одиночных разработчиков



pywinauto: как пользоваться

- `Application(backend="uia").start(...)` или `.connect(...)` по PID, title, ...
- `desk = Desktop(backend="uia")` без привязки к процессу

Спеки ("локаторы") окон и UI элементов (было в 0.6.9):

- `desk.window(title="...", control_type="...", ...).child_window(...)`
`.wait("exist enabled")` или `.wrapper_object()`

(стало в 1.0.0+)

- `desk.by(name="...", ...).by(...).find().wait_enabled()` или `.find_all()`



pywinauto: как пользоваться

- Wrappers: методы работы с окнами и UI элементами
- Автодетект типа элемента (возврат объекта нужного класса) через метакласс в Python
- Настоящий ввод: `.click_input()`, `.type_keys()`
 - нужен активный десктоп, см. [Remote Execution Guide](#)
- Дёрнуть методы-обработчики в самом приложении:
`.click()` / `.invoke()`, `.toggle()`, `.set_edit_text()`, ...



Часть 1

Базовые бэкенды (подсистемы)

Старый backend="win32"



Через модули `ctypes` и **`pywin32`**:

- Загрузка `kernel32.dll`, `user32.dll`
- Вызов C функций
 - все размеры C структур и типов зависят от 32/64-битности приложения
 - youtu.be/68h6w2KvTy4 (Парсинг бинарных данных с помощью `ctypes` ..., @PythoNN meetup 2023)
- OK: MFC, WinForms (частично)
- полная слепота: WPF, Qt, Chrome



Основной backend="uia"

Через сторонний модуль **comtypes**:

- Загрузка **UIAutomationCore.dll** и интерфейса **IUIAutomation**
- Метод **IUIAutomationElement.FindAll** (на C/C++) по умолчанию
- Объект **RawViewWalker** (на Python, медленный, 1.0.0+)
- OK: MFC, WinForms, WPF, Qt, Chrome (если вкл.), UWP, ...
- не OK: custom controls (Infragistics, часть Grid'ов), QML/Quick, Java AWT, GTK+ на Windows, встроенный в Python Tkinter



Linux (2015 – 2019, 2020 – 2022)

Зависимости:

- **python-xlib** (тоже maintainer)
- Встроенная в ОС **libatspi.so** (через ctypes)
- ОК: GTK+, стандартные Qt, Chrome (с флагом), Electron, ...
- Не ОК: Wayland дистрибутивы, недоисследовано 😊
 - Было подвисание потока у одной IDE (wait долгий не помогал)



mac OS (2018-2022)

Зависимости:

- ruobjс (несколько подпакетов)
- OK: GTK+, стандартные Qt, Chrome (с флагом), Electron, ...
- Не OK: недоисследовано 😊
- Есть тесты, нет CI:
 - образ macOS с отключенным SIP есть у Circle CI, но он ввёл санкции 😞
 - включение accessibility под обычным юзером возможно

Часть 2

Кастомные WPF и Qt



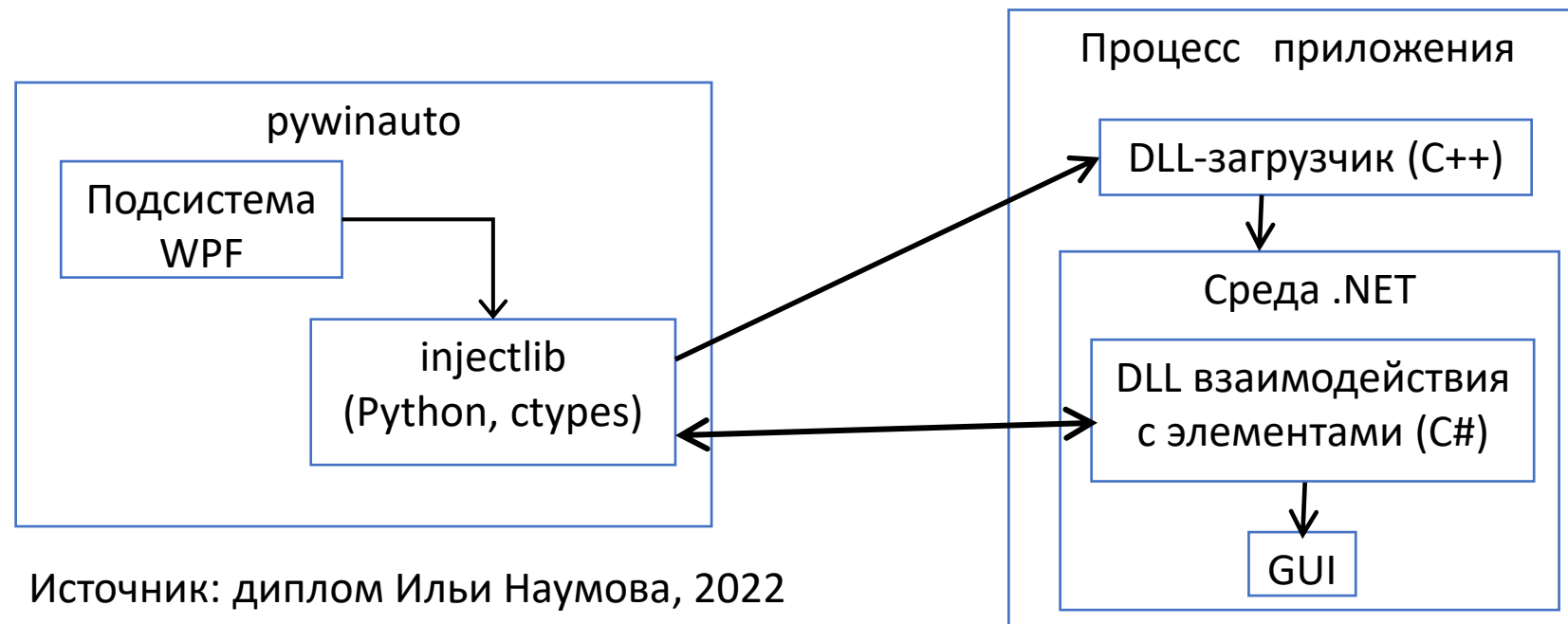
Кастомные WPF элементы

- Infragistics, EremEx Controls, MahApps, ... (Test Complete, Squish)
- Доклад Владимира Перевалова + microPoC, 2017
 - Внедрение DLL в процесс WPF приложения
 - Запуск .NET framework
 - .NET reflection для получения внутренних свойств
- Нам осталось:
 - организовать обмен данными между pywinauto и remote thread
 - и продуктивизировать 😊



Application(backend="wpf") (2020 – 2022)

- Новая библиотека: injectlib (C++, C# и Python)
- Системные требования: .NET Framework 4.0+ (вышел в 2010)





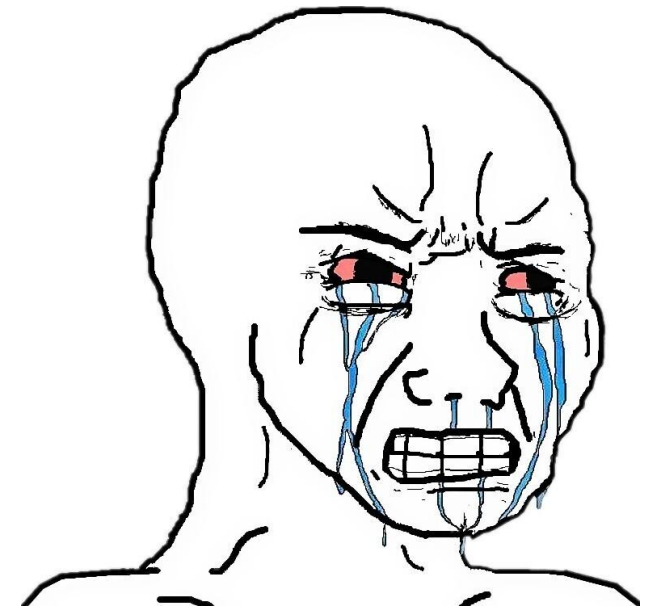
Qt: QML/Quick (2024 – 2026)

- Всё аналогично WPF
- Добавлен Linux x86_64
 - Хабрастатья 2019 года: [CreateRemoteThread для Linux](#)
 - Отладочный вызов ptrace()
 - Вместо named pipe – UDS (Unix Domain Socket)
- Два бэкенда: “qt5” и “qt6”
- Качество: скорее альфа
 - Нужно расширять покрытие типов элементов



Часть 3

Кейсы, мемы, кровь и слёзы



Источник: безутешный wojak nerd

Как быть с остальным? (Java AWT, Tkinter)



- Java AWT
 - [gaozhao1989/pyjab](https://github.com/gaozhao1989/pyjab) (Java Access Bridge через **ctypes**)
- Tkinter
 - Появилась поддержка accessibility в Tkinter 9.1
 - В Python 3.14 пока старый Tkinter 8.6
 - python/cpython/issues/145578 Prepare Tkinter 9.1 screenreader support
- GTK+ на Windows
 - GTK 4.18+ (AccessKit backend): GTK_A11Y=accesskit



Что ещё хочется сделать

- Обновить доки (deploy на ReadTheDocs сломан), разобрать баги
- Сравнение скриншотов с эталоном с помощью Computer Vision
 - Хабрастатья 2025: <https://habr.com/ru/companies/otus/articles/956492/>
- Автоматическая запись скрипта (record/replay)
 - Прототип на Python – медленный, нужен C/C++/Objective C/Rust
 - [pywinauto/pywinauto/pull/701](https://github.com/pywinauto/pywinauto/pull/701)
- Agents.md, MCP, поддержка агентской разработки



Контакты

- Tg: @vasily_v_ryabov
- vk.ru/varjag



@VASILY_V_RYABOV

- Организация на GitHub: github.com/pywinauto
- StackOverflow tag: stackoverflow.com/questions/tagged/pywinauto