# Shedlib

## Native and web-oriented GUIs in Python

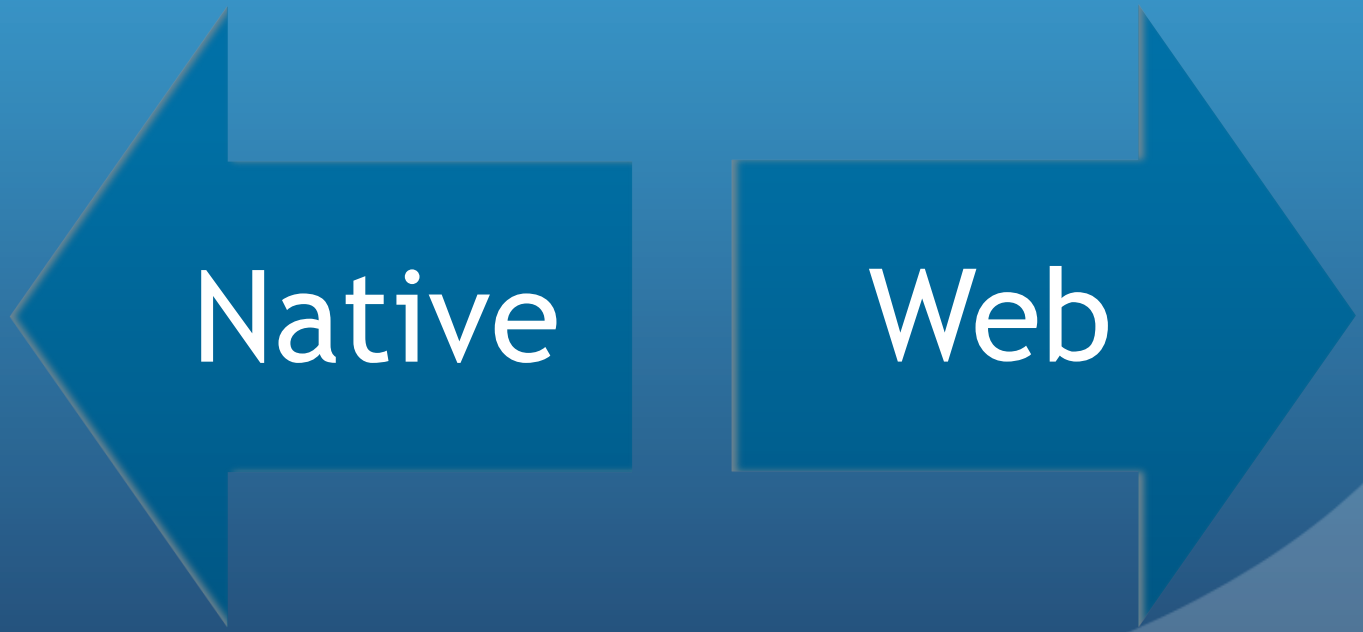
Angelo Mottola – a.mottola@gmail.com
Fabrizio Toso – fabrizio@digiturtle.com

# Applications today

| | Pros | Cons |
|---|---|---|
| **Native** | • Speed<br>• Rich interface<br>• OS access | • OS restricted<br>• Installation |
| **Web** | • Ubiquity | • Limited interface<br>• Browser restricted |
| **Hybrid (ex. Adobe AIR)** | • OS access<br>• Web/desktop | • Limited interface<br>• HTML/Javascript<br>• Flash/Actionscript |

# Native and web-based applications

- Different paradigms
- Different technologies
- Native application code ≠ web application code



# Goals

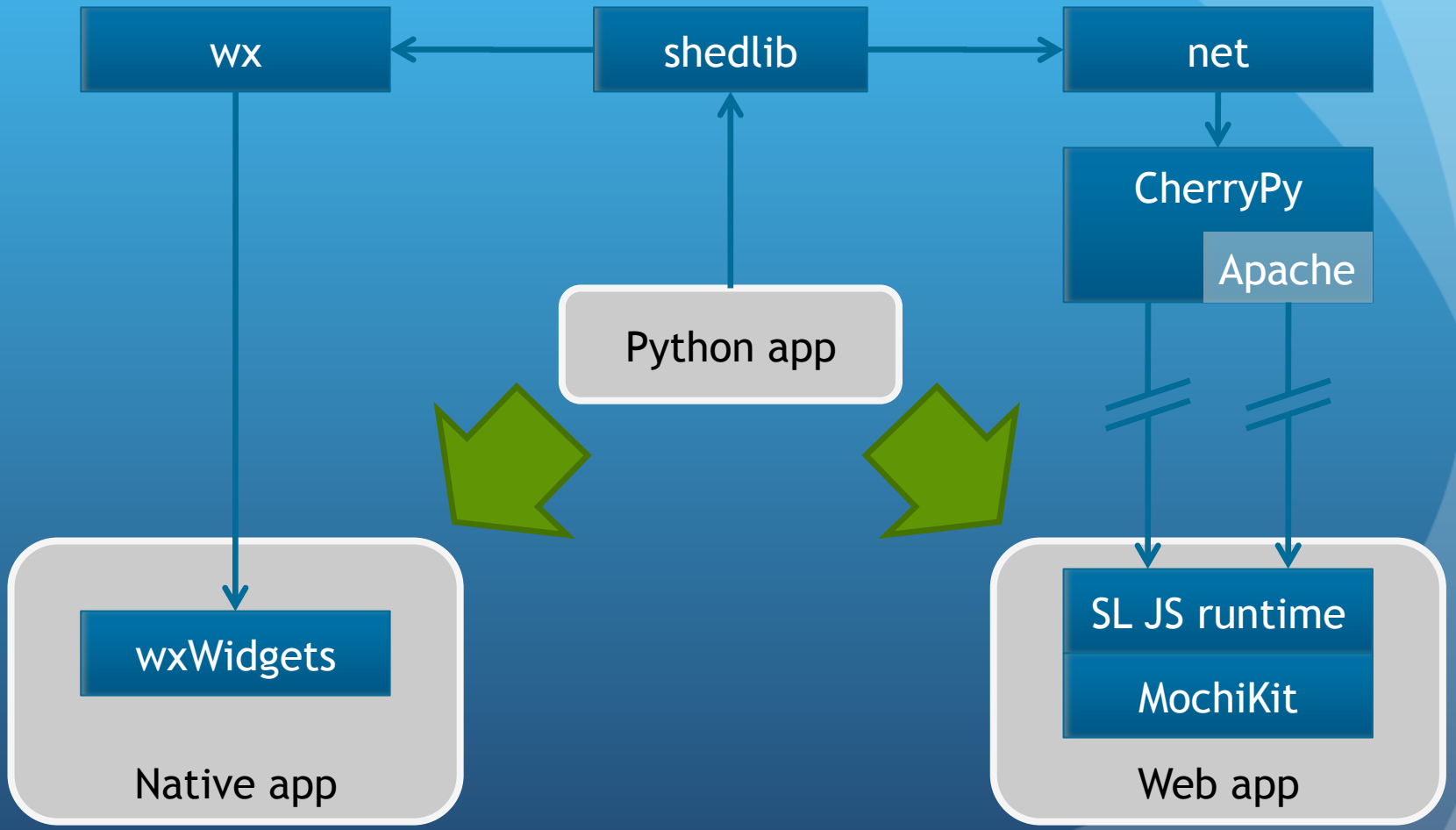
- Rich interface
- Rapid deployment of both native and desktop applications
- No need for browser plugins
- Same source code
- Python!

Shedlib

# What is Shedlib

- Python extension written in C++
- GUI class hierarchy
- Interfaces definition
  - By code
  - Via XML files
- Visual driver architecture
  - wx driver
  - net driver

# System architecture



# Hello World

# Application structure

- Application = archive
- Resource index

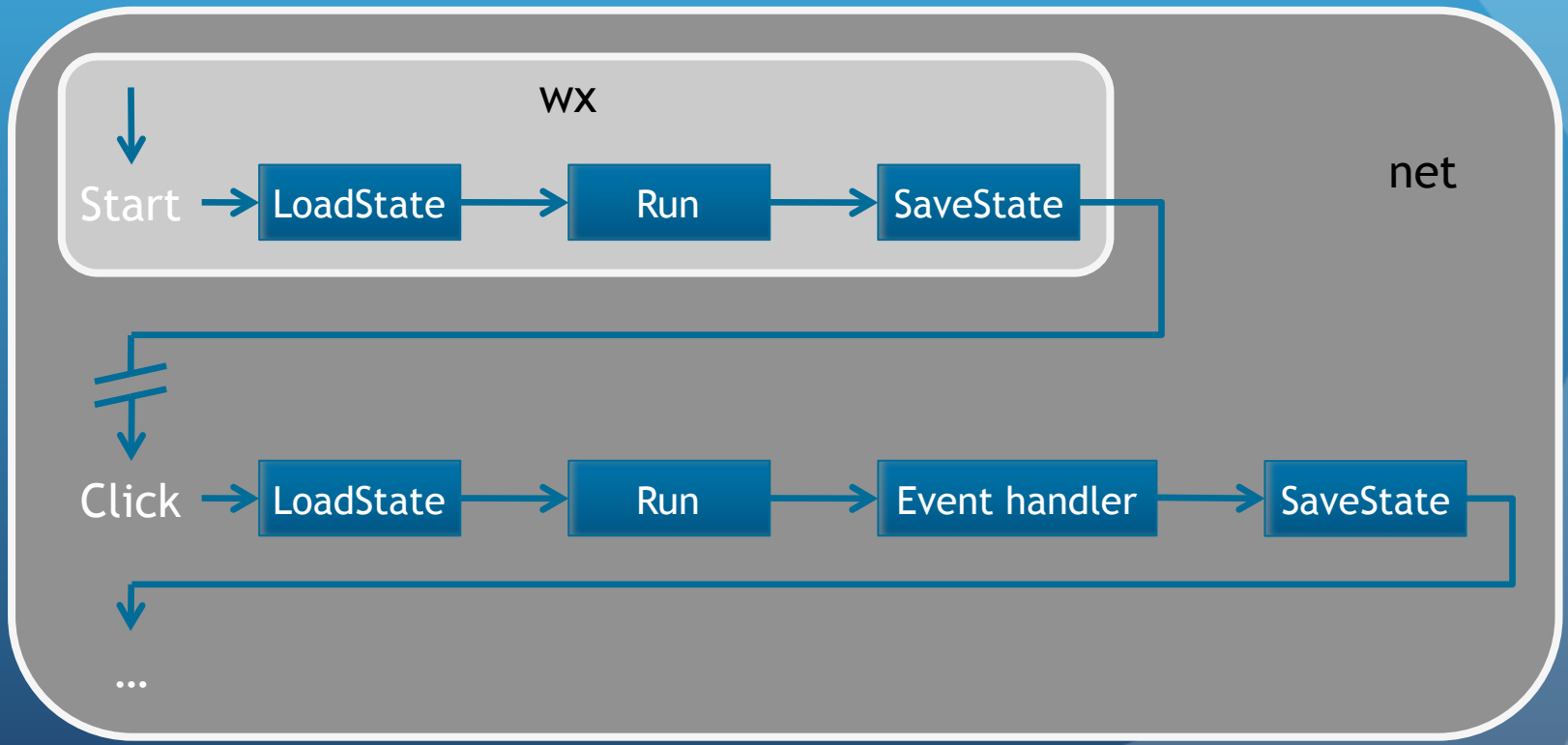
```
<index>
     <resource name = "logo"> images/logo.png </resource>
     <resource name = "init"> hello.py </resource>
</index>
```

- Init script

```
import shedlib

class Application ( shedlib.Application ):

     def Run ( self ):
         self.frame = shedlib.Frame ( title = 'Hello world!' )
         self.frame.Show ()

shedlib.Run( Application () )
```

# Application state / 1

- Necessity of LoadState and SaveState methods
- shedlib.State class: session data and history data



# Application state / 2

```
import shedlib

class Application ( shedlib.Application ):

      def __init__ ( self ):
            self.count = 0
            self.frame = None

      def LoadState ( self, state ):
            if state.session is not None:
                  self.count, self.frame = state.session

      def SaveState ( self ):
            return shedlib.State ( session = ( self.count, self.frame ) )

      def SayHello ( self ):
            self.count += 1
            print 'Hello %d times' % self.count

      def Run ( self ):
            if self.frame is None:
                  self.frame = shedlib.Frame ( title = 'Hello world' )
                  self.frame[0] = shedlib.Button (
                        pos = ( 50, 50 ),
                        label = 'Say hello',
                        onClick = 'shedlib.GetApplication().SayHello()'
                  )
            self.frame.Show ()

shedlib.Run( Application () )
```

# Hello World (reprise)

# What's next?

- Support for all widgets in net
- More widgets!
  - Calendar control
  - Generic combo control
  - …
- Visual driver based on QT
- Release under LGPL
- Feedback

# Grazie!

Mailing list: http://tinyurl.com/shedlib

http://download.easybyte.it/docs/shedlib_pycon3.pdf

# Interface elements / 1

- Python classes

| | | | | | |
|---|---|---|---|---|---|
| Frame ☑ | ToolBarItem ☑ | SplitView | ListBox ☑ | Line ☑ | ScrollBar |
| MenuBar ☑ | Sizer ☑ | TabView ☑ | ComboBox ☑ | Hyperlink ☑ | Progress |
| Menu ☑ | SizerItem ☑ | TabViewPage ☑ | GroupBox ☑ | Image ☑ | SearchField ☑ |
| MenuItem ☑ | Panel ☑ | Button ☑ | CheckBox ☑ | Slider ☑ | Report ☑ |
| StatusBar ☑ | FoldPanel | TextField ☑ | Radio ☑ | Spin | TreeView |
| ToolBar ☑ | ScrollView | TextView ☑ | Label ☑ | SpinField | Window ☑ |

- Handling widget hierarchy

```
frame = shedlib.Frame ( title = 'Test' )

frame.Attach ( shedlib.Button ( label = 'Click me' ) )
button = frame[0]
frame[1] = shedlib.TextField ( name = 'myTextfield' )
del frame['myTextField']
```

# Interface elements / 2

- Properties

```
myTextField.color = ( 255, 0, 0 )
myTextField.align = shedlib.TextField.ALIGN_RIGHT

content = myTextField.value
```

- XML definition

```
<interface>
    <frame name="test_frame" title="Test" style="caption|close|resize">
        <button label="Click me" pos="20,20" size="100,20" />
        <textfield pos="20,50" size="100,20" />
    </frame>
</interface>
```

```
widgets = shedlib.LoadInterface( 'test_frame_def' )[1]
frame = widgets[0].Create ()
frame.Show ()
```