Extend Python with ArcGIS Runtime for Qt

Mark Cederholm
UniSource Energy Services
Flagstaff, Arizona

2018 Esri Developer Summit
March 6-9 | Palm Springs Convention Center | Palm Springs, CA

Versions Used in Examples

- Python 3.6.4 [win: 64-bit]
- Qt 5.10.1
- ArcGIS Runtime for Qt 100.2.1
- PyQt 5.10.1
- SIP 4.19.8

Prerequisites

[I'm assuming you've already installed Python]

https://developers.arcgis.com/qt/latest/

mac:

- Install Xcode
- Install Qt 5.x
- Install ArcGIS Runtime for Qt 100.x

win:

- Install Visual Studio [2017 Community, Desktop Development with C++]
- Install Windows Driver Kit*
- Install Qt 5.x [kit: msvc2017 64-bit]
- Install ArcGIS Runtime for Qt 100.x
- Download 64-bit OpenSSL zip from https://indy.fulgan.com/SSL/
- Add libeay32.dll and ssleay32.dll to Qt bin folder

*Allows debugging in Qt Creator

5:01:08 PM

PyQt

- Library of Python wrappers for Qt classes
- Doc: http://pyqt.sourceforge.net/Docs/PyQt5/
- Source: https://www.riverbankcomputing.com/software/pyqt/download5

Installing PyQt

Make sure version matches your Qt build

- mac: pip3 install pyqt5
- win: pip install pyqt5

Behind an authenticating proxy?
[win]

```
set http_proxy=http://username:password@proxyAddress:port
set https_proxy=https://username:password@proxyAddress:port
```

Demo:

Launch a QML app
in Python

```
app = QtWidgets.QApplication(sys.argv)
engine = QtQml.QQmlApplicationEngine()
engine.load(QtCore.QUrl(qml))
rootlist = engine.rootObjects()
qw = rootlist[0]
qw.show()
```

“And now for something completely different…”

SIP

- Tool for generating Python wrapper code for C++ classes
- Results work seamlessly with PyQt
- Doc: http://pyqt.sourceforge.net/Docs/sip4/
- Source: https://www.riverbankcomputing.com/software/sip/download

Installing SIP Code Generator

[sip extension module is installed with pyqt5]

mac:

- Download and extract archive
- `python3 configure.py`
- `make`
- `make install`

Installing SIP Code Generator

win [in VS x64 command prompt]:

- Download and extract archive
- `python configure.py`
- `nmake /f Makefile`
- `nmake install`

qmake

- Generates makefiles from C++ code
- Is part of Qt, although it can work on non-Qt code
- Qt Creator is not required
- Well-documented

Building a Python Module

- Download PyQt source code (for SIP imports)
- Create SIP files: Core.sip, …
- Create qmake file: Core.pro

Building a Python Module

mac [in Terminal, with qmake in PATH]:

```
sip -c . -I ../sip
    -I ~/ArcGIS_SDKs/Qt100.2.1/sdk/include
    -t WS_MACX -t Qt_5_10_1 Core.sip
qmake -o Makefile Core.pro
make
make install
```

Building a Python Module

win [in VS x64 prompt, with qmake in PATH]:

```
sip -c . -I ../sip
    -I "C:\Program Files (x86)\ArcGIS SDKs\Qt100.2.1\sdk\include"
    -t WS_WIN -t Qt_5_10_1 Core.sip
qmake -o Makefile Core.pro
nmake /f Makefile
nmake install
```

SIP Tips

- One SIP file per header (.h) file
- Keep formatting similar to header file (aids upgrades)
- Full qualify all Esri names:

```
namespace Esri { namespace ArcGISRuntime {

class Map : public Esri::ArcGISRuntime::Object,
        public Esri::ArcGISRuntime::Loadable,
        public Esri::ArcGISRuntime::JsonSerializable
{
%TypeHeaderCode
#include <Map.h>
%End
```

SIP Tips

- Use /TransferThis/ and /Transfer/ annotation to transfer instance ownership to parent QObjects:

```
MapQuickView(QQuickItem* parent /TransferThis/ = nullptr);
static Basemap* streets(QObject* parent /Transfer/ = nullptr);
```

If no QObject parent is assigned, then the Python object calls the C++ destructor when it's garbage collected.

SIP Tips

- Comment out unwanted, internal, unsupported, or meaningless items:

```
//Polygon(Polygon&& other);
//Polygon& operator= (const Polygon& other);
//Polygon& operator=(Polygon&& other);
…
//Polygon(const std::shared_ptr<QRTImpl::PolygonImpl>& impl);
//std::shared_ptr<QRTImpl::PolygonImpl> getImpl() const;
…
Esri::ArcGISRuntime::Error loadError() const; //override;
…
private:
FeatureTableListModel(); //= delete;
//QHash<int, QByteArray> m_roles;
```

SIP Tips

- Expand Qt macros where it makes sense:

```
private:
//Q_DISABLE_COPY(Map)
Map(const Map &);
Map &operator=(const Map &);
```

SIP Tips

- Replace enum expressions with numeric values:

```
enum FeatureTableRoles
    {
      NameRole =                257,    //Qt::UserRole + 1,
      GeometryTypeRole =        258,    //Qt::UserRole + 2,
      HasGeometryRole =         259,    //Qt::UserRole + 3,
      EditableRole =            260,    //Qt::UserRole + 4,
      NumberOfFeaturesRole =    261,    //Qt::UserRole + 5,
      LoadErrorRole =           262,    //Qt::UserRole + 6,
      LoadStatusRole =          263,    //Qt::UserRole + 7
    };
```

SIP Problems

- QList<qint64> [affects QueryParameters, MapServiceLayerIdInfo, MobileBasemapLayer]

  workaround = Mapped Type

  https://github.com/bholland
- QFlags<enum class> [affects LayerViewStatusFlags]

  workaround = ???

SIP Problems

- Building cross-referenced modules: import recursion error in Python

  workaround = ???

[Building a single giant module works, but that is not the desired approach]

qmake Tips

- Pay attention to Qt classes referenced
- Add the appropriate modules to the Qt variable
- If a SIP file defines multiple classes, include all CPP files generated
- Check generated CPP files for missing namespaces

Demo:

A Python QtWidgets App

```
class RuntimeWidgets(QtWidgets.QMainWindow):
…
        env = RtCore.ArcGISRuntimeEnvironment.instance()
        result = env.setLicense(key)
…
        mv = RtUI.MapGraphicsView(self)
        mv.mapScaleChanged.connect(self.ScaleChanged)
        mv.setWrapAroundMode(RtUI.WrapAroundMode.Disabled)
        bm = RtMapping.Basemap.topographic(self)
        mp = RtMapping.Map(bm, self)
        mv.setMap(mp)
        self.setCentralWidget(mv)
```

The MapQuickView Conundrum

- Grab MapView instance from QML and cast to Python object? **No**
- Register Python-wrapped MapQuickView as QML type? **No**
- Create MapQuickView in Python and add to Quick UI? **Yes**

The MapQuickView Conundrum

- You *can* create a MapQuickView in Python and add it to a QML UI
- However, certain QML properties (e.g. anchors, width, height) cannot be set using setProperty strings
- Thus, resizing depends on connecting to signals from the container item

Demo

A Python QtQuick App

```
class RuntimeQuick(QtCore.QObject):
    def __init__(self):
        QtCore.QObject.__init__(self)
…
 def Setup(self):
…
        mc = qw.findChild(QtQuick.QQuickItem, "mapViewContainer")
…
        mv = RtUI.MapQuickView(mc)
…
        mc.widthChanged.connect(self.SizeChanged)
        mc.heightChanged.connect(self.SizeChanged)
```

5:01:08 PM

Improving Performance

- Writing a coarse-grained object in C++ can dramatically improve performance over writing the same logic in Python
- Create and debug the object in Qt Creator
- Add SIP wrappers in the release build to create a Python module
- See example code (download)

ArcGIS Runtime for Python?

- A work in progress (68% complete, not including LocalServer)
- Current source code available for download
- To Do: put it up on GitHub?
- There are cleaner approaches, but leveraging PyQt and SIP is the simplest

Questions?

- Mark Cederholm
  mcederholm@uesaz.com
- This presentation and examples may be downloaded at:

http://www.pierssen.com/arcgis/runtime.htm