ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA134

PyDM - STATUS UPDATE

H. H. Slepicka*, M. Gibbs, SLAC National Accelerator Laboratory, Menlo Park, USA

Abstract

PyDM (**Py**thon **D**isplay **M**anager) is a Python and Qt-based framework for building user interfaces for control systems providing a no-code, drag-and-drop system to make simple screens, as well as a straightforward Python framework to build complex applications. Here we report the state of PyDM as well as the new functionality that has been added in the last year of development, including full support for EPICS PVAccess and other structured data sources, and also the features targeted for release in 2020.

PROJECT STATE

Over the past year PyDM had a couple of feature and bug fix releases which included infrastructure work along with new widgets and enhancements for the user experience while creating screens on the Qt Designer.

INFRASTRUCTURE

PyDM supports Linux, macOS and Windows and Python 2.7, 3.6 and 3.7. In order to ensure that the codebase is healthy and new code added does not break the existing features and platform compatibility, project relies on continuous integration (CI). PyDM initially relied on the cloud-based tools TravisCI (for builds and tests on Linux and macOS), and AppVeyor (for Windows builds and tests). An effort was made during the past year to migrate away from two different systems to the new Azure Pipelines provided by Microsoft.

Azure Pipelines

Azure Pipelines is free for open-source projects and offers 10 parallel builds with support for the three platforms in which PyDM works. This allowed us to reduce the build and test time and also to enhance the operations performed when new code is added to the master branch and new tags are released.

When new code is added to the master branch a new version of the package is uploaded to the ”pydm-dev” channel at Anaconda and is immediately available for users for early tests and development. If a new release tag is created, the pipeline creates a new version of the package and uploads it to the ”pydm-tag” channel at Anaconda, deploys the documentation to the GitHub Pages website (https://slaclab.github.io/pydm) and also uploads a source package to the Python Package Index (PyPI).

OBTAINING THE PACKAGE

PyDM is available via PyPI and Anaconda channels, as well as source via GitHub.

* slepicka@slac.stanford.edu

PyPI

Since PyDM is now available at PyPI, it can be easily added as a dependency to other python packages as well as installed at computers via ”pip install pydm”.

Anaconda

The PyDM package is available to Anaconda users at the following channels:

- pydm-dev: latest development build;
- pydm-tag: official release versions;
- conda-forge: official release versions;

NEW WIDGET - TEMPLATE REPEATER

The *PyDMTemplateRepeater* widget was added in version 1.7 of PyDM.

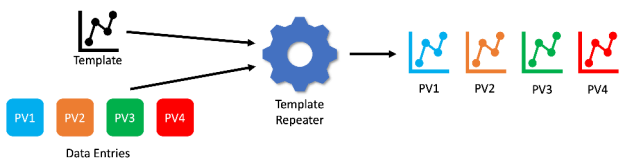

Figure 1: PyDM Template Repeater Widget design concept

The *PyDMTemplateRepeater* lets users specify a display file as a ”template”, and then specify a JSON file as a ”data source” that will fill in multiple instances of that template with values. The widget can lay out the instances vertically, horizontally, or in a ’flow’ layout, which is similar to a grid layout that wraps to new rows as the row fills. If used in a python-based display, users can directly supply a list of dictionaries to act as a data source - expand the widget capabilities and allowing it to be hooked up to databases, web APIs, etc. This widget was built to create huge displays with lists of controls for dozens of devices, in a dynamic way, without having to write any code.

LIVE PROTOTYPING WITH QT DESIGNER

In order to provide higher fidelity for users developing displays with the Qt Designer, PyDM now offers an option to connect widgets to live data and render embedded displays while inside Qt Designer. Previously, widgets would only connect to the data plugins when running the screen.

The *PYDM_DESIGNER_ONLINE* environment variable controls this feature and when this variable is defined the live prototyping will be enabled.

It proved to be useful and reduce the amount of work that users have due to resizing and rearrangement of widgets at the screen due to the data being displayed when connected to the data plugin.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA134

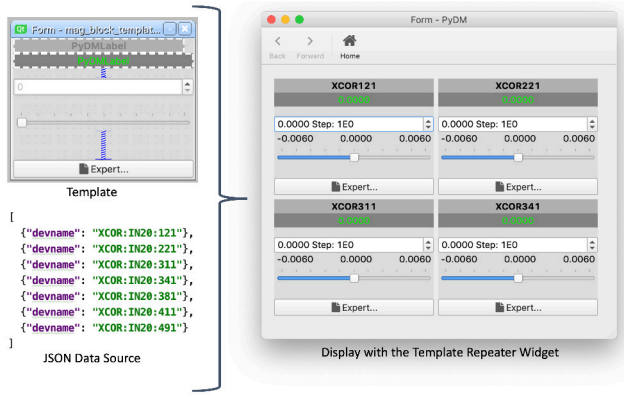

Figure 2: PyDM Template Repeater widget rendering Magnet control templates based on a list from a JSON file.

PYDM 2.0

PyDM 2.0 will bring changes into how data plugins work and how data is provided to widgets. Despite being a major change to the PyDM architecture, existing widgets and displays will continue to work without changes. The breaking changes were confined to the data plugins which are independent pieces of code that can be easily modified.

Over the next sections we will highlight what will change on PyDM with the upcoming version 2.0.

Out With The Old, In With The New

Python 2.7 will no longer be supported due to its end-of-life [1]. Most (if not all) of the packages that PyDM requires will drop support for Python 2.7 with new releases and PyDM must also move ahead so it can leverage the new Python features and provide better performance, scalability and extensibility.

As soon as Python 3.8 is officially released, late October 2019, it will be added to the test matrix to ensure that PyDM is compatible with this new release. Code will be kept compatible with at least Python 3.6.

Structured Data

Up until PyDM 1.x series, only scalar and scalar arrays were allowed to be transferred as values to Channels via the Qt signals, which imposed a limitation on the type and amount of data that could be handled. Moreover, this design also imposed a copy of the data for each connection tied to a data plugin, since each piece of data was transferred using an individual Qt Signal which was sent to the proper slot at each of the widgets sharing the same connection. With PyDM 2.x series, the data transferring mechanism was refactored in such a way that now data plugins store a data payload dictionary at a central storage area called the *DataStore*.

The DataStore class contains two dictionaries, data and introspection. Both are keyed on the channel address, and the values are the data payload and a introspection lookup table, respectively. This new design allows PyDM to store structured and complex data to be consumed by widgets and the introspection lookup maps fields in the data structures to properties needed by PyDM widgets, like value and alarm state.

Widgets designed with a specific data plugin in mind can take advantage of the additional metadata stored in the DataStore to enhance the user experience and provide more capabilities which were not possible with the previous design of PyDM 1.x series.

A PyDMChannel is a class that connects a widget to a specific data plugin through a protocol identifier along with a connection string according to the format defined by the data plugin.

When new data is available to the PyDMConnection, it stores the data in the DataStore and emits a signal to all channels connected to it notifying them that new data is available. The PyDMChannel instances fetch data from the DataStore for the specific channel address and execute the subscribed callbacks for the widget with the newer data and introspection information.

A detailed migration guide will be released with the doc-umentation for the new version.

Code Organization

PyDM previously shipped with a couple of data plugins in it. In order to reduce the amount of dependencies and allow for a faster and more modular development of PyDM 2.0, we decided that all data plugins will become external packages to be installed on a needed basis by users depending on their needs.

Splitting the data plugins will provide greater code coverage and data source specific tests to be executed on each data plugin package.

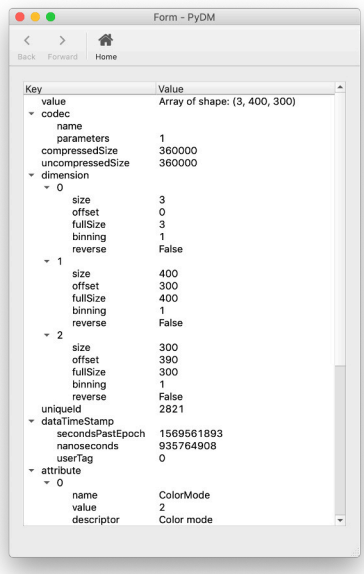

Figure 3: PyDM TreeView widget rendering structured data from an EPICS PVAccess NTNDArray.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA134

New Widget - TreeView

The *PyDMTreeView* is a useful widget to visualize structured and hierarchical data.

Its initial goal was to help in the troubleshooting process for new PyDM data plugins and it proved to be very valuable as a flexible, general-purpose data inspection widget.

CONCLUSION

This past year, many features were added and bugs were fixed in the PyDM codebase. The deprecation of Python 2.7 will open new frontiers to be explored and enhancements to be made. Support for structured data, along with the many other improvements to become available with PyDM 2.0, will open new opportunities for widgets, data plugins and user experience enhancements.

ACKNOWLEDGMENTS

The authors would like to thank all the contributors to PyDM as well as users that provided valuable feedback, bug reports and feature requests.

REFERENCES

[1] B. Peterson. (2008). PEP 373 – Python 2.7 Release Schedule, https://www.python.org/dev/peps/pep-0373/