PySide Documentation

Release 1.2.4

PySide Team

Mar 03, 2017

Contents

1 Introduction **3**

2 Compatibility **5**

3 Contents **7**
3.1 Installing PySide 7
3.2 Building PySide 8
3.3 Changes 16

4 Feedback and getting involved **27**

CHAPTER 1

Introduction

PySide is the Python Qt bindings project, providing access the complete Qt 4.8 framework as well as to generator tools for rapidly generating bindings for any C++ libraries.

The PySide project is developed in the open, with all facilities you'd expect from any modern OSS project such as all code in a git repository, an open Bugzilla for reporting bugs, and an open design process. We welcome any contribution without requiring a transfer of copyright.

The PySide reference documentation is hosted at http://pyside.github.io/docs/pyside/.

CHAPTER 2

Compatibility

PySide requires Python 2.6 or later and Qt 4.6 or better.

Note: Qt 5.x is currently under construction.

CHAPTER 3

Contents

Installing PySide

Installing PySide on a Windows System

Installing prerequisites

Install latest `pip` distribution: download get-pip.py and run it using the `python` interpreter.

Installing PySide

To install PySide on Windows you can choose from the following options:

1. Use pip to install the `wheel` binary packages:

```
pip install -U PySide
```

2. Use setuptools to install the `egg` binary packages (deprecated):

```
easy_install -U PySide
```

Note: Provided binaries are without any other external dependencies. All required Qt libraries, development tools and examples are included.

Installing PySide on a Mac OS X System

Installing prerequisites

Install latest `pip` distribution: download get-pip.py and run it using the `python` interpreter.

You need to install or build Qt 4.8 first, see the Qt Project Documentation.

Alternatively you can use Homebrew and install Qt with

```
$ brew install qt
```

Installing PySide

To install PySide on Mac OS X you can choose from the following options:

1. Use pip to install the `wheel` binary packages:

   ```
   $ pip install -U PySide
   ```

Installing PySide on a Linux System

We do not provide binaries for Linux. Please read the build instructions in section Building PySide on a Linux System.

Building PySide

Building PySide on a Windows System

Installing prerequisites

1. Install Python.
2. Install Qt 4.8 libraries for Windows VS 2008 edition when building against Python 2.6, 2.7 or 3.2.

   Install Qt 4.8 libraries for Windows VS 2010 edition when building against Python 3.3 or 3.4.
3. Install Cmake.
4. Install Windows SDK v7.0 when building against Python 2.6, 2.7 or 3.2.

   Install Windows SDK v7.1 when building against Python 3.3 or 3.4.
5. Install Git.
6. (Optional) Install OpenSSL.
7. Install latest `pip` distribution into the Python you installed in the first step: download get-pip.py and run it using the `python` interpreter of your Python 2.7 installation using a command prompt:

   ```
   c:\> c:\Python27\python get-pip.py
   ```
8. Install latest *wheel* distribution:

   ```
   c:\> c:\Python27\Scripts\pip install wheel
   ```

Note:

- If you are using the Windows SDK versions linked above, make sure that you have .NET Framework Version 4.0 installed on your system. Version 4.5 will not work.

- To avoid problems with environment variables, execute all commands below in a "Windows SDK Comamnd Prompt" that you will find in your Start Menu, instead of a standard cmd.exe command line.

Building PySide distribution

1. Download and extract PySide source distribution
2. Switch to the distribution directory:

```
c:\> cd PySide-1.2.4
```

3. Build the `wheel` binary distribution:

```
c:\> c:\Python27\python.exe setup.py bdist_wheel --qmake=c:\Qt\4.8.7\bin\qmake.
↪exe --openssl=c:\OpenSSL32bit\bin
```

Building PySide distribution from a Git repository

1. Clone `PySide` setup scripts from git repository:

```
c:\> git clone https://github.com/PySide/pyside-setup.git pyside-setup
```

2. Switch to the `pyside-setup` directory:

```
c:\> cd pyside-setup
```

3. Build the *wheel* binary distribution:

```
c:\> c:\Python27\python.exe setup.py bdist_wheel --version=1.2.4 --qmake=c:\Qt\4.
↪8.7\bin\qmake.exe --openssl=c:\OpenSSL32bit\bin
```

4. To build the development version of `PySide` distribution, ignore the –version parameter:

```
c:\> c:\Python27\python.exe setup.py bdist_wheel --qmake=c:\Qt\4.8.7\bin\qmake.
↪exe --openssl=c:\OpenSSL32bit\bin
```

Installing PySide distribution

1. After the successful build, install the distribution with `pip`:

```
c:\> c:\Python27\Scripts\pip install dist\PySide-1.2.4-cp27-none-win32.whl
```

Installing PySide distribution into `virtual` Python environment

1. Install latest `virtualenv` distribution:

```
c:\> c:\Python27\Scripts\pip install virtualenv
```

2. Use `virtualenv` to make a workspace:

```
c:\> c:\Python27\Scripts\virtualenv env
```

3. Switch to the `env` directory:

```
c:\> cd env
```

4. Install the distribution with `pip`:

```
c:\> Scripts\pip install ..\dist\PySide-1.2.4-cp27-none-win32.whl
```

Building PySide on a Mac OS X System

Mac OS X is a Unix flavor, partially based upon BSD Unix.

The supported Mac OS X versions created by Apple are

- OS X 10.6 *Snow Leopard*
- OS X 10.7 *Lion*
- OS X 10.8 *Mountain Lion*
- OS X 10.9 *Mavericks*
- OS X 10.10 *Yosemite*

Mac OS X is a proprietary UNIX flavor of BSD Unix and only partially similar to Linux. Therefore, the usual packages from Linux distributions cannot be used without modifications.

There are several known package managers which provide support for Mac OS X, namely

- MacPorts
- Fink
- Homebrew

The main purpose of all of these projects is to provide the missing Linux packages for Mac OS X.

Throughout this tutorial, we are only using Homebrew, because it appears to be the most light-weight package manager available. All installations are made to /usr/local/(bin|lib|include|shared) by simple symlinks.

But it should be easy to translate these instructions for the other, heavier package managers.

Installing prerequisites

1. Install Package Manager:

```
$ ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/install)"
```

   Follow the on-screen instructions to make adjustions, especially run

```
$ brew doctor
```

   Also see the homebrew homepage for further information

2. Install Xcode (optional):

   Follow the on-screen instructions. If you selected any extensions to be installed, wait for their completion before you proceed.

Note: If you are using Mavericks, you can also use the Xcode Command Line Tools without actually installing Xcode (not tested, see this article: How to Install Command Line Tools in OS X Mavericks (Without Xcode)).

3. Install the Xcode command Line Tools:

   After Xcode installation has finished, you can open a command shell and issue

   ```
   $ xcode-select --install
   ```

   This will open a dialog window with further instructions. After the command line tools are installed, you will not need to use Xcode again in order to set up PySide.

4. Install build dependencies:

   ```
   $ brew install python cmake qt
   ```

   Remark: This installs `Homebrew` Python, which is fine for you as a single user. If you are considering to build installers for external users, see the section `About PySide Distributions`.

5. Install latest `pip` distribution into the Python you installed in the first step: download get-pip.py and run it using the `python` interpreter of your Python 2.7 installation using a command prompt:

   ```
   $ wget https://bootstrap.pypa.io/get-pip.py
   $ sudo python2.7 get-pip.py
   ```

   Note: There are situations with older Python versions, where the above procedure does not work. You can then use this last-resort work-around (tested):

   ```
   $ wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
   $ sudo python2.7 ez_setup.py
   $ sudo easy_install pip
   $ sudo pip install setuptools -U
   $ sudo pip install wheel -U
   ```

6. Install latest `wheel` distribution:

   ```
   $ sudo pip2.7 install wheel
   ```

About PySide Distribution

If you want to build PySide for your own use, the above instructions are ok.

But when you are considering to build PySide for other versions or other users, you need to be aware of the following caveat:

- Mac OS X has the concept of a `MACOSX_DEPLOYMENT_TARGET`
- The current deployment targets which work with PySide are 10.6 to 10.9 .
- All binary installers from https://www.python.org are built with the setting

```
$ export MACOSX_DEPLOYMENT_TARGET=10.6  # Snow Leopard
```

- The default setting for the deployment target of an extension (like PySide) is always inherited from the Python used for building. You can set the deployment target higher than that, but not below the OS X version that was set during building your Python installation.
- Current distributions like Homebrew set the deployment target to the same value as the OS version they are built with. (I.E. 10.9 for Mavericks).
- Example: A PySide, built on Mavericks, will therefore not run on a Python that was built for Mountain Lion.

Recommendation:

- Use Homebrew's simplicity for your own machine. Do not use it for distributing.
- Use one of the Python.org Distributions or
- Build your own Python, either from a tar archive (Python 2.7 or Python 3.4), or from a Mercurial repository with an explicit setting of `MACOSX_DEPLOYMENT_TARGET`.

Building PySide distribution

1. Download `PySide` source distribution:

   ```
   $ wget https://pypi.python.org/packages/source/P/PySide/PySide-1.2.4.tar.gz
   ```

2. Extract the source distribution:

   ```
   $ tar -xvzf PySide-1.2.4.tar.gz
   ```

3. Switch to the distribution directory:

   ```
   $ cd PySide-1.2.4
   ```

4. Build the `wheel` binary distribution:

   ```
   $ python2.7 setup.py bdist_wheel
   ```

Building PySide distribution from a Git repository

1. Clone `PySide` setup scripts from git repository:

   ```
   $ git clone https://github.com/PySide/pyside-setup.git pyside-setup
   ```

2. Switch to the `pyside-setup` directory:

   ```
   $ cd pyside-setup
   ```

3. Build `PySide` distribution:

   ```
   $ python2.7 setup.py bdist_wheel --version=1.2.4
   ```

4. To build the development version of `PySide` distribution, ignore the –version parameter:

   ```
   $ python2.7 setup.py bdist_wheel
   ```

Installing PySide distribution

1. After the successful build, install the distribution with `pip`:

```
$ sudo pip2.7 install dist/PySide-1.2.4-cp27-none-linux-x86_64.whl
```

Installing PySide distribution into `virtual` Python environment

1. Install latest `virtualenv` distribution:

```
$ sudo pip2.7 virtualenv
```

2. Use `virtualenv` to make a workspace:

```
$ virtualenv-2.7 env
```

3. Activate the virtual Python in the `env` directory:

```
$ source env/bin/activate
```

4. Install the distribution with `pip`:

```
(env) $ pip install ../dist/PySide-1.2.4-cp27-none-linux-x86_64.whl
```

5. Leave the virtual environment (optional):

```
(env) $ deactivate
$
```

Building PySide on a Linux System (Ubuntu 12.04 - 14.04)

Installing prerequisites

1. Install build dependencies:

```
$ sudo apt-get install build-essential git cmake libqt4-dev libphonon-dev python2.
↪7-dev libxml2-dev libxslt1-dev qtmobility-dev libqtwebkit-dev
```

2. Install latest `pip` distribution into the Python you installed in the first step: download get-pip.py and run it using the `python` interpreter of your Python 2.7 installation using a command prompt:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python2.7 get-pip.py
```

3. Install latest `wheel` distribution:

```
$ sudo pip2.7 install wheel
```

Building PySide distribution

1. Download `PySide` source distribution:

```
$ wget https://pypi.python.org/packages/source/P/PySide/PySide-1.2.4.tar.gz
```

2. Extract the source distribution:

```
$ tar -xvzf PySide-1.2.4.tar.gz
```

3. Switch to the distribution directory:

```
$ cd PySide-1.2.4
```

4. Build the `wheel` binary distribution:

```
$ python2.7 setup.py bdist_wheel --qmake=/usr/bin/qmake-qt4
```

5. Optionally you can build standalone version of distribution with embedded Qt libs:

```
$ python2.7 setup.py bdist_wheel --qmake=/usr/bin/qmake-qt4 --standalone
```

Building PySide distribution from a Git repository

1. Clone `PySide` setup scripts from git repository:

```
$ git clone https://github.com/PySide/pyside-setup.git pyside-setup
```

2. Switch to the `pyside-setup` directory:

```
$ cd pyside-setup
```

3. Build `PySide` distribution:

```
$ python2.7 setup.py bdist_wheel --qmake=/usr/bin/qmake-qt4 --version=1.2.4
```

4. Optionally you can build standalone version of distribution with embedded Qt libs:

```
$ python2.7 setup.py bdist_wheel --qmake=/usr/bin/qmake-qt4 --version=1.2.4 --
↪standalone
```

5. To build the development version of `PySide` distribution, ignore the –version parameter:

```
$ python2.7 setup.py bdist_wheel --qmake=/usr/bin/qmake-qt4
```

Installing PySide distribution

1. After the successful build, install the distribution with `pip`. The file name may vary depending on your platform so look into the `dist` directory for the correct name:

```
$ ls dist
PySide-1.2.4-cp27-none-linux-x86_64.whl
$ sudo pip2.7 install dist/PySide-1.2.4-cp27-none-linux-x86_64.whl
```

Installing PySide distribution into `virtual` Python environment

1. Install latest `virtualenv` distribution:

   ```
   $ sudo pip2.7 virtualenv
   ```

2. Use `virtualenv` to make a workspace:

   ```
   $ virtualenv-2.7 env
   ```

3. Switch to the `env` directory:

   ```
   $ cd env
   ```

4. Install the distribution with `pip` (wheel binary file name may vary by platform):

   ```
   $ bin/pip2.7 install ../dist/PySide-1.2.4-cp27-none-linux-x86_64.whl
   ```

PySide Setup Script command line options

Usage on a Windows System

```
c:\> c:\Python27\python.exe setup.py [distribution_type] [options]
```

Usage on a Linux/Mac OS X System

```
python2.7 setup.py [distribution_type] [options]
```

Distribution types

`bdist_wheel` Create a wheel binary distribution. This distribution type can be installed with `pip`.

`bdist_egg` Create an egg binary distribution. This distribution type can be installed with `easy_install`.

`bdist_wininst` Create a standalone windows installer with embedded Qt libs and development tools. This distribution type can be installed with `easy_install`.

`install` Install package to site packages folder.

`develop` Install package in `development mode`, such that it's available on `sys.path`, yet can still be edited directly from its source folder.

`sdist` Create a full source distribution with included sources of PySide Setup Scripts, PySide, Shiboken, PySide Tools and PySide Examples. Can be used to build binary distribution in offline mode.

Options

`--qmake` Specify the path to qmake. Useful when the qmake is not in path or more than one Qt versions are installed.

`--openssl` Specify the path to OpenSSL libs.

`--only-package` Skip rebuilding everything and create distribution from prebuilt binaries. Before using this option first time, the full distribution build is required.

--cmake Specify the path to cmake. Useful when the cmake is not in path.

--standalone When enabled, all required Qt libs will be included in PySide distribution. This option is allways enabled on Windows. On Linux it's disabled by default.

> **Note:** This option does not work on Mac OS X, yet.

--version Specify what version of PySide distribution to build. This option is available only when the setup scripts are cloned from git repository.

--list-versions List available versions of PySide distributions.

--ignore-git Don't pull sources from git repository.

--make-spec Specify the cmake makefile generator type. Available values are `msvc` on Windows and `make` on Linux/Mac OS X.

--no-examples Don't include PySide examples in PySide distribution

--jobs Specify the number of parallel build jobs

--jom Use jom instead of nmake with msvc

--build-tests Enable building the tests

Changes

1.2.4 (2015-10-14)

Complete list of changes and bug fixes

PySide-setup

- Make sure that setup.py is run with an allowed python version

1.2.3 (2015-10-12)

Complete list of changes and bug fixes

PySide

- Fix PYSIDE-164: Fix possible deadlock on signal connect/emit

Shiboken

- Don't ignore classes in topology
- Process global enums in declaration order
- Return enums in declaration order (order added)

PySide-setup

- On Linux and MacOS systems there is no more need to call the post-install script

1.2.2 (2014-04-24)

Complete list of changes and bug fixes

PySide

- Fix PYSIDE-190: QCoreApplication would deadlock on exit if the global QThreadPool.globalInstance() is running a QRunnable with python code
- Change GlobalReceiver to explicitly 'use' [dis]connectNotify of the base class in order to avoid hiding these with its own overloads.
- Add explicit casts when initializing an int[] using {}'s, as required by C++11 to be "well formed"
- Fix PYSIDE-172: multiple rules for file
- Use file system encoding instead of assumed 'ascii' when registering qt.conf in Qt resource system

Shiboken

- Remove rejection lines that cause the sample_list test to fail
- Remove protected from samblebinding test
- Add parsing of 'noexcept' keyword
- Fix function rejections (i.e. support overloads)
- Fix building with python 3.3 and 3.4
- Doc: Stop requiring sphinx.ext.refcounting with Sphinx 1.2+
- Fix for containers with 'const' values
- Fix compilation issue on OS X 10.9
- Only use fields in PyTypeObject when defining types
- Fix buffer overrun processing macro definitions
- Fix 'special' include handling
- Fix finding container base classes
- Refactor and improve added function resolving
- Work around MSVC's deficient <cmath> in libsample/transform.cpp
- Fix description of sample/transform unit test
- Change wrapping and indent of some code in Handler::startElement to improve consistency
- Fix '%#' substitution for # > 9
- Improve dependencies for tests

1.2.1 (2013-08-16)

Major changes

PySide

- In memory qt.conf generation and registration

Shiboken

- Better support for more than 9 arguments to methods
- Avoiding a segfault when getting the .name attribute on an enum value with no name

PySide-setup

- Switched to the new setuptools (v0.9.8) which has been merged with Distribute again and works for Python 2 and 3 with one codebase
- Support for building windows binaries with only Windows SDK installed (Visual Studio is no more required)
- Removed –msvc-version option. Required msvc compiler version is now resolved from python interpreter version

1.2.0 (2013-07-02)

Major changes

PySide

- Fix multiple segfaults and better track the life time of Qt objects
- Fix multiple memory leaks

Shiboken

- Install the shiboken module to site-packages
- Fix multiple segfaults

PySide-setup

- On Windows system, when installing PySide binary distribution via easy_install, there is no more need to call the post-install script
- Support for building windows binaries outside of Visual Studio command prompt
- Build and package the shiboken docs when sphinx is installed

Complete list of changes and bug fixes

PySide

- Set up PYTHONPATH for tests correctly
- Fix potential segfault at shutdown
- Fix PYSIDE-61
- Tell Qt to look for qml imports in the PySide package
- fix build in C++11 mode
- Fix QByteArray memory leak
- Ignore QtCore import errors when initializing plugins folder
- Preload OpenSSL DLLs on Windows.
- Look first in the PySide package for Qt's plugins folder, instead of just in Qt's install or build folder
- Add explicit type conversion to fix mingw compile error
- Use QObject property to invalidate wrapper before deletion
- Invalidate metaObject wrapper before deletion
- Fix reference leak on convertion from a C++ map type to Python dict
- Change the order of pysitetest and signals directories because signals/disconnect_test.py depends on pysidetest module

Shiboken

- Removed old logos from html docs
- Add missing return on module init error
- Don't break -Werror=non-virtual-dtor
- Fixing shiboken test for minimal binding test
- Decref reference to type object
- Fix segfault when using shiboken.delete
- Use non-static method def for instance methods
- Fix bug introduced when recursive_invalidate was added
- fix build in C++11 mode
- Prevent infinite recursion in invalidate
- Fix possible conflict with garbage collector
- Fix possible crash at exit
- Fix handling of unsigned long long and provide unittests
- Add test to illustrate issue on typedef enum
- Use getWrapperForQObject to convert if generating for PySide
- Allow compilation without a python shared library

- Use parent class's metaObject if wrapper is NULL
- Optionally assert on free'd pointer with a valid wrapper
- Find python3 libraries when built with pydebug enabled
- Fix PYSIDE-108 bug and add example
- PYSIDE-83 Fix segfault calling shiboken.dump
- Fix and test case for bug PYSIDE-72
- Override all functions with the same name, not just one
- Update vector conversion
- Add typedef examples to minimal
- Add test files back to cmake
- Don't use it->second after erasing it
- Find function modifications defined in the 2nd+ base class. Fixes bug PYSIDE-54.
- Set a default hash function for all ObjectTypes. Fix bug PYSIDE-42.
- Fix compilation when there is no libxslt installed on the system.
- Fixed resolving of SOABI. SOABI is implemented on Linux, but not on Windows
- Don't use inline methods in dllexported classes to keep VC++ happy
- Use SpooledTemporaryFile in 2.6+ os.tmpfile() fails on win32 if process doesn't have admin permissions

PySide-setup

- Support for building windows binaries outside of Visual Studio command prompt
- Build and package the shiboken docs when sphinx is installed
- Support Ubuntu 13.04 and Fedora 18
- Fixed "develop" setuptools command
- Documentation updates
- Add –build-tests option to enable building the tests
- Add –jom and –jobs options
- Add –no-examples option to exclude the examples
- Add –relwithdebinfo option to enable a release-with-debug-info build mode
- Add –ignore-git option
- Add –make-spec option to specify make generator

1.1.2 (2012-08-28)

Bug fixes

- During signal emission don't get return type after callback
- Invalidate QStandardModel::invisibleRootItem in clear() method

- QAbstractItemModel has wrong ownership policy for selectionModel()
- Improved QVector to python conversion
- Disable docstring generation if tools aren't found.
- Fixed some issues compiling PySide using VC++
- Install the shiboken module to site-packages
- Fix compilation when there is no libxslt installed on the system.
- Set a default hash function for all ObjectTypes.
- Fix segfault calling shiboken.dump

1.1.1 (2012-04-19)

Major changes

- Unified toolchain! No more GeneratorRunner and ApiExtractor, now you just need Shiboken to compile PySide.

Bug fixes

- 1105 Spyder fails with HEAD
- 1126 Segfault when exception is raised in signalInstanceDisconnect
- 1135 SIGSEGV when loading custom widget using QUiLoader when overriding createWidget()
- 1041 QAbstractItemModel has wrong ownership policy for selectionModel()
- 1086 generatorrunner segfault processing #include
- 1110 Concurrency error causes GC heap corruption
- 1113 Instantiating QObject in user-defined QML element's constructor crashes if instantiated from QML
- 1129 Segmentation fault on close by QStandardItem/QStandardItemModel
- 1104 QSettings has problems with long integers
- 1108 tests/QtGui/pyside_reload_test.py fails when bytecode writing is disabled
- 1138 Subclassing of QUiLoader leads to "Internal C++ object already deleted" exception (again)
- 1124 QPainter.drawPixmapFragments should take a list as first argument
- 1065 Invalid example in QFileDialog documentation
- 1092 shiboken names itself a 'generator'
- 1094 shiboken doesn't complain about invalid options
- 1044 Incorrect call to parent constructor in example
- 1139 Crash at exit due to thread state (tstate) being NULL
- PYSIDE-41 QModelIndex unhashable

1.1.0 (2012-01-02)

Major changes

- New type converter scheme

Bug fixes

- 1010 Shiboken Cygwin patch
- 1034 Error compiling PySide with Python 3.2.2 32bit on Windows
- 1040 pyside-uic overwriting attributes before they are being used
- 1053 pyside-lupdate used with .pro files can't handle Windows paths that contain spaces
- 1060 Subclassing of QUiLoader leads to "Internal C++ object already deleted" exception
- 1063 Bug writing to files using "QTextStream + QFile + QTextEdit" on Linux
- 1069 QtCore.QDataStream silently fails on writing Python string
- 1077 Application exit crash when call QSyntaxHighlighter.document()
- 1082 OSX binary links are broken
- 1083 winId returns a PyCObject making it impossible to compare two winIds
- 1084 Crash (segfault) when writing unicode string on socket
- 1091 PixmapFragment and drawPixmapFragments are not bound
- 1095 No examples for shiboken tutorial
- 1097 QtGui.QShortcut.setKey requires QKeySequence
- 1101 Report invalid function signatures in typesystem
- 902 Expose Shiboken functionality through a Python module
- 969 viewOptions of QAbstractItemView error

1.0.9 (2011-11-29)

Bug fixes

- 1058 Strange code in PySide/QtUiTools/glue/plugins.h
- 1057 valgrind detected "Conditional jump or move depends on uninitialised value"
- 1052 PySideConfig.cmake contains an infinite loop due to missing default for SHIBOKEN_PYTHON_SUFFIX
- 1048 QGridLayout.itemAtPosition() crashes when it should return None
- 1037 shiboken fails to build against python 3.2 (both normal and -dbg) on i386 (and others)
- 1036 Qt.KeyboardModifiers always evaluates to zero
- 1033 QDialog.DialogCode instances and return value from QDialog.exec_ hash to different values
- 1031 QState.parentState() or QState.machine() causes python crash at exit
- 1029 qmlRegisterType Fails to Increase the Ref Count

- 1028 QWidget winId missing
- 1016 Calling of Q_INVOKABLE method returning not QVariant is impossible...
- 1013 connect to QSqlTableModel.primeInsert() causes crash
- 1012 FTBFS with hardening flags enabled
- 1011 PySide Cygwin patch
- 1010 Shiboken Cygwin patch
- 1009 GeneratorRunner Cygwin patch
- 1008 ApiExtractor Cygwin patch
- 891 ApiExtractor doesn't support doxygen as backend to doc generation.

1.0.8 (2011-10-21)

Major changes

- Experimental Python3.2 support
- Qt4.8 beta support

Bug fixes

- 1022 RuntimeError: maximum recursion depth exceeded while getting the str of an object
- 1019 Overriding QWidget.show or QWidget.hide do not work
- 944 Segfault on QIcon(None).pixmap()

1.0.7 (2011-09-21)

Bug fixes

- 996 Missing dependencies for QtWebKit in buildscripts for Fedora
- 986 Documentation links
- 985 Provide versioned pyside-docs zip file to help packagers
- 981 QSettings docs should empathize the behavior changes of value() on different platforms
- 902 Expose Shiboken functionality through a Python module
- 997 QDeclarativePropertyMap doesn't work.
- 994 QIODevice.readData must use qmemcpy instead of qstrncpy
- 989 Pickling QColor fails
- 987 Disconnecting a signal that has not been connected
- 973 shouldInterruptJavaScript slot override is never called
- 966 QX11Info.display() missing
- 959 can't pass QVariant to the QtWebkit bridge
- 1006 Segfault in QLabel init

- 1002 Segmentation fault on PySide/Spyder exit
- 998 Segfault with Spyder after switching to another app
- 995 QDeclarativeView.itemAt returns faulty reference. (leading to SEGFAULT)
- 990 Segfault when trying to disconnect a signal that is not connected
- 975 Possible memory leak
- 991 The __repr__ of various types is broken
- 988 The type supplied with currentChanged signal in QTabWidget has changed in 1.0.6

1.0.6 (2011-08-22)

Major changes

- New documentation layout;
- Fixed some regressions from the last release (1.0.5);
- Optimizations during anonymous connection;

Bug fixes

- 972 anchorlayout.py of graphicsview example raised a unwriteable memory exception when exits
- 953 Segfault when QObject is garbage collected after QTimer.singeShot
- 951 ComponentComplete not called on QDeclarativeItem subclass
- 965 Segfault in QtUiTools.QUiLoader.load
- 958 Segmentation fault with resource files
- 944 Segfault on QIcon(None).pixmap()
- 941 Signals with QtCore.Qt types as arguments has invalid signatures
- 964 QAbstractItemView.moveCursor() method is missing
- 963 What's This not displaying QTableWidget column header information as in Qt Designer
- 961 QColor.__repr__/__str__ should be more pythonic
- 960 QColor.__reduce__ is incorrect for HSL colors
- 950 implement Q_INVOKABLE
- 940 setAttributeArray/setUniformValueArray do not take arrays
- 931 isinstance() fails with Signal instances
- 928 100's of QGraphicItems with signal connections causes slowdown
- 930 Documentation mixes signals and functions.
- 923 Make QScriptValue (or QScriptValueIterator) implement the Python iterator protocol
- 922 QScriptValue's repr() should give some information about its data
- 900 QtCore.Property as decorator
- 895 jQuery version is outdated, distribution code de-duplication breaks documentation search

- 731 Can't specify more than a single 'since' argument
- 983 copy.deepcopy raises SystemError with QColor
- 947 NETWORK_ERR during interaction QtWebKit window with server
- 873 Deprecated methods could emit DeprecationWarning
- 831 PySide docs would have a "Inherited by" list for each class

1.0.5 (2011-07-22)

Major changes

- Widgets present on "ui" files are exported in the root widget, check PySide ML thread for more information[1];
- pyside-uic generate menubars without parent on MacOS plataform;
- Signal connection optimizations;

Bug fixes

- 892 Segfault when destructing QWidget and QApplication has event filter installed
- 407 Crash while multiple inheriting with QObject and native python class
- 939 Shiboken::importModule must verify if PyImport_ImportModule succeeds
- 937 missing pid method in QProcess
- 927 Segfault on QThread code.
- 925 Segfault when passing a QScriptValue as QObject or when using .toVariant() on a QScriptValue
- 905 QtGui.QHBoxLayout.setMargin function call is created by pyside-uic, but this is not available in the pyside bindings
- 904 Repeatedly opening a QDialog with Qt.WA_DeleteOnClose set crashes PySide
- 899 Segfault with 'QVariantList' Property.
- 893 Shiboken leak reference in the parent control
- 878 Shiboken may generate incompatible modules if a new class is added.
- 938 QTemporaryFile JPEG problem
- 934 A __getitem__ of QByteArray behaves strange
- 929 pkg-config files do not know about Python version tags
- 926 qmlRegisterType does not work with QObject
- 924 Allow QScriptValue to be accessed via []
- 921 Signals not automatically disconnected on object destruction
- 920 Cannot use same slot for two signals
- 919 Default arguments on QStyle methods not working
- 915 QDeclarativeView.scene().addItem(x) make the x object invalid
- 913 Widgets inside QTabWidget are not exported as members of the containing widget
- 910 installEventFilter() increments reference count on target object

- 907 pyside-uic adds MainWindow.setMenuBar(self.menubar) to the generated code under OS X
- 903 eventFilter in ItemDelegate
- 897 QObject.property() and QObject.setProperty() methods fails for user-defined properties
- 896 QObject.staticMetaObject() is missing
- 916 Missing info about when is possible to use keyword arguments in docs [was: QListWidgetItem's constructor ignores text parameter]
- 890 Add signal connection example for valueChanged(int) on QSpinBox to the docs
- 821 Mapping interface for QPixmapCache
- 909 Deletion of QMainWindow/QApplication leads to segmentation fault

CHAPTER 4

Feedback and getting involved

- Mailing list: http://lists.qt-project.org/mailman/listinfo/pyside
- Issue tracker: https://bugreports.qt-project.org/browse/PYSIDE
- Code Repository: http://qt.gitorious.org/pyside
- Examples Code Repository: https://github.com/PySide/Examples