Doxygen to DoxyPress: A Journey from C++98 to C++11

Barbara Geller & Ansel Sermersheim

CPPCon - September 2015

Introduction

- Why documentation is Important
- Limitations of Doxygen
- Why DoxyPress
- Migrating code from C++98 to C++11
- Future plans for DoxyPress

Questions welcome anytime...

Why Documentation is Important

- Who needs documentation?
  - developers of your application
  - users of your library or application
  - your future self
- What should be documented
  - how to set up your environment
  - class and method documentation
  - overall system design
  - timeline or change log
  - error conditions
  - samples

Why Documentation is Important

- When to create documentation
  - day one of your project
  - yesterday
  - today
  - tomorrow

Overview of Doxygen

- Development started around 1995
- Open Source / GPL
- Written in C++

- Uses obsolete/unmaintained Qt 1.9 classes
- Core classes hand modified
- Non standard language translation functionality

Limitations of Doxygen

- Container classes store pointers (not values)
- Autodelete memory management
- Macros used to simulate variadic templates
- Riddled with raw pointers
- Code extremely difficult to read
    - very limited line breaks
    - prolific use of variable names like: bcli, bii, cli, cei, cni, di, dcli, ei, eli, evi, i, ii, iii, l, li, lti, mli, mnii, pli, mri, sl, sli, slii

Limitations of Doxygen

- extra <div> tags in HTML output
- blank lines can not be used in a table
- layout file is not fully customizable
- HTML 5 not fully supported
- HTML output is not W3C compliant
- project file is raw text, requires Lex to parse
- limited options sorting in a navigation tree
- <dl> can not contain multiple <dd>
- problems with auto brief
- unable to parse some macros

Now What?

- Unable to document our CopperSpice library
- Contacted the maintainer of Doxygen
- Not very receptive
- Initial direction was to help improve Doxygen
- Code was simply unmaintainable

What is DoxyPress

- DoxyPress is a fork of Doxygen 1.8.8
- Backported relevant changes through 1.8.10

- Full rewrite of DoxyWizard
- Name was changed to DoxyPressApp

- DoxyPress and DoxyPressApp link with the CopperSpice libraries

Doxygen - Example 1

- Existing current code
- String class returns ‘\0’ (a null char) if an invalid index is accessed
- Access off the end of a string is acceptable code

```
if (result.at(0) == ':' && result.at(1) == ':')  {
    . . .
}
```

DoxyPress - Example 1

- Using CopperSpice string class (QString)
- Accessing an invalid index is an error, which is similar to std::string

```
// A
int len = result.size();
if (len >= 2 && result.at(0) == ':' && result.at(1) == ':') {
    . . .
}

// B
if (result.startsWith(”::”))  {
    . . .
}
```

Doxygen - Example 2

- FORALL3() is a macro used to forward 3 parameters to a method

```
#define FORALL3(a1,a2,a3,p1,p2,p3) \
void OutputList::forall(void (OutputGenerator::*func)(a1,a2,a3), \
     a1,a2,a3) \
{ \
    QListIterator<OutputGenerator> it(m_outputs); \
    OutputGenerator *og; \
    for (it.toFirst(); og=it.current(); ++it) \
    { \
        if (og->isEnabled()) (og->*func)(p1,p2,p3); \
    } \
}
```

Doxygen - Example 2

- For 3 parameters there are 9 different forms

```
FORALL3(bool a1,HighlightedItem a2,const char *a3,a1,a2,a3)
FORALL3(bool a1,bool a2,bool a3,a1,a2,a3)
FORALL3(const ClassDiagram &a1,const char *a2,const char *a3,a1,a2,a3)
FORALL3(const char *a1,const char *a2,const char *a3,a1,a2,a3)
FORALL3(const char *a1,const char *a2,bool a3,a1,a2,a3)
FORALL3(const char *a1,int a2,const char *a3,a1,a2,a3)
FORALL3(const char *a1,const char *a2,SectionInfo::SectionType a3,a1,a2,a3)
FORALL3(uchar a1,uchar a2,uchar a3,a1,a2,a3)
FORALL3(Definition *a1,const char *a2,bool a3,a1,a2,a3)
```

- Same style of code exists for passing 6 (2 forms), 5 (2 forms), 4 (4 forms), 2 (9 forms), and 1 (12 forms)
- 200+ lines of code

DoxyPress - Example 2

- The entire FORALL macros were replaced with the following 9 lines of code

```
template<class BaseClass, class... Args, class... Ts>
void forall( void (BaseClass::*func)(Args...), Ts&&... vs)
{
    for (auto item : m_outputs ) {
        if (item->isEnabled()) {
            (item->*func)(vs...);
        }
    }
}
```

Overview of DoxyPress

- Removed all Qt 1.9 classes and containers
  - string classes auto convert to char *
  - containers were pointer based, not value based
- Code reformatted
- Enhanced source to use C++11
- Shared pointers instead of raw pointers
- Variadic templates instead of macro abuse
- Project file changed from text to JSON format
- Easy to convert a Doxygen project file to a DoxyPress project file

Overview of DoxyPress

- Extraneous <div>'s removed from the output
- Whitespace and blank lines allowed in a table
- Added \code{.mk} for documenting Makefiles
- Added \sortid X for sorting navigation tree
- Fixed <dl> so it can contain multiple <dd> tags
- Images no longer force a new paragraph
- Additional features and corrections: http://www.copperspice.com/docs/doxypress/timeline.html

Migrating from C++98 to C++11

- Ensure copy constructor is a deep copy

- Raw pointers ⇨ shared pointers
  - with raw pointers it is unclear who is responsible for object destruction
  - too easy to accidentally use a raw pointer after the object has been deleted
  - use QMakeShared in CopperSpice or std::make_shared instead of calling new

  - this type of pointer conversion can not be done gradually

Migrating from C++98 to C++11

- for loop
  - C++11 range based syntax
  - use auto for declaring iterators
- Container misuse
  - `QHash<QString, void *> files;`
  - `files.insert(“myFile”, (void *)0x08);`
  - a large amount of code used pointers
- Override
  - ensure methods which override a base class method are marked with “override”

Migrating from C++98 to C++11

- Character set encoding
  - use UTF-8 internally
  - program as if your application will be used internationally

- Strings
  - avoid using const char * (memory management issues)
  - use std::string class, or
  - use QString class in CopperSpice

- Use nullptr instead of 0
  - improves readability
  - zero can mean nullptr or an empty string

Future Plans for DoxyPress

- Switch to libClang for parsing
    - C, C++, Objective C, Objective C++

- Support for other languages like D and extended support for Python and C#

- Optimize internal structures for efficiencies

- User requests & Developer Contributions

Libraries & Applications

- CopperSpice
  - Libraries for developing GUI applications
- PepperMill
  - Converts old headers to CS standard C++ header files
- KitchenSink
  - Over 30 CopperSpice demos in one application
- Diamond
  - Programmers Editor which uses the CopperSpice libraries
- DoxyPress & DoxyPressApp
  - Documentation program, works with C++11

Where to find DoxyPress

- www.copperspice.com
- download.copperspice.com
- forum.copperspice.com

- ansel@copperspice.com
- barbara@copperspice.com

- Questions? Comments?