Using Precompiled Headers

Precompiled headers (PCH) are a performance feature supported by some compilers to compile a stable body of code, and store the compiled state of the code in a binary file. During subsequent compilations, the compiler will load the stored state, and continue compiling the specified file. Each subsequent compilation is faster because the stable code does not need to be recompiled.

qmake supports the use of precompiled headers on some platforms and build environments, including:

- Windows
  - nmake
  - Visual Studio projects (VS 2008 and later)
- macOS, iOS, tvOS, and watchOS
  - Makefile
  - Xcode
- Unix
  - GCC 3.4 and above
  - clang

Adding Precompiled Headers to Your Project

The precompiled header must contain code which is *stable* and *static* throughout your project. A typical precompiled header might look like this:

```
// Add C includes here

#if defined __cplusplus
// Add C++ includes here
#include <stdlib>
#include <iostream>
#include <vector>
#include <QApplication> // Qt includes
#include <QPushButton>
#include <QLabel>
#include "thirdparty/include/libmain.h"
#include "my_stable_class.h"
...
```

Qt DOCUMENTATION

Note: A precompiled header file needs to separate C includes from C++ includes, since the precompiled header file for C files may not contain C++ code.

Project Options

To make your project use precompiled headers, you only need to define the PRECOMPILED_HEADER variable in your project file:

```
PRECOMPILED_HEADER = stable.h
```

qmake will handle the rest, to ensure the creation and use of the precompiled header file. You do not need to include the precompiled header file in HEADERS, as qmake will do this if the configuration supports precompiled headers.

The MSVC and g++ specs targeting Windows enable `precompile_header` by default.

Using this option, you may trigger conditional blocks in your project file to add settings when using precompiled headers. For example:

```
precompile_header:!isEmpty(PRECOMPILED_HEADER) {
DEFINES += USING_PCH
}
```

To use the precompiled header also for C files on MSVC nmake target, add `precompile_header_c` to the CONFIG variable. If the header is used also for C++ and it contains C++ keywords/includes, enclose them with `#ifdef __cplusplus`).

Notes on Possible Issues

On some platforms, the file name suffix for precompiled header files is the same as that for other object files. For example, the following declarations may cause two different object files with the same name to be generated:

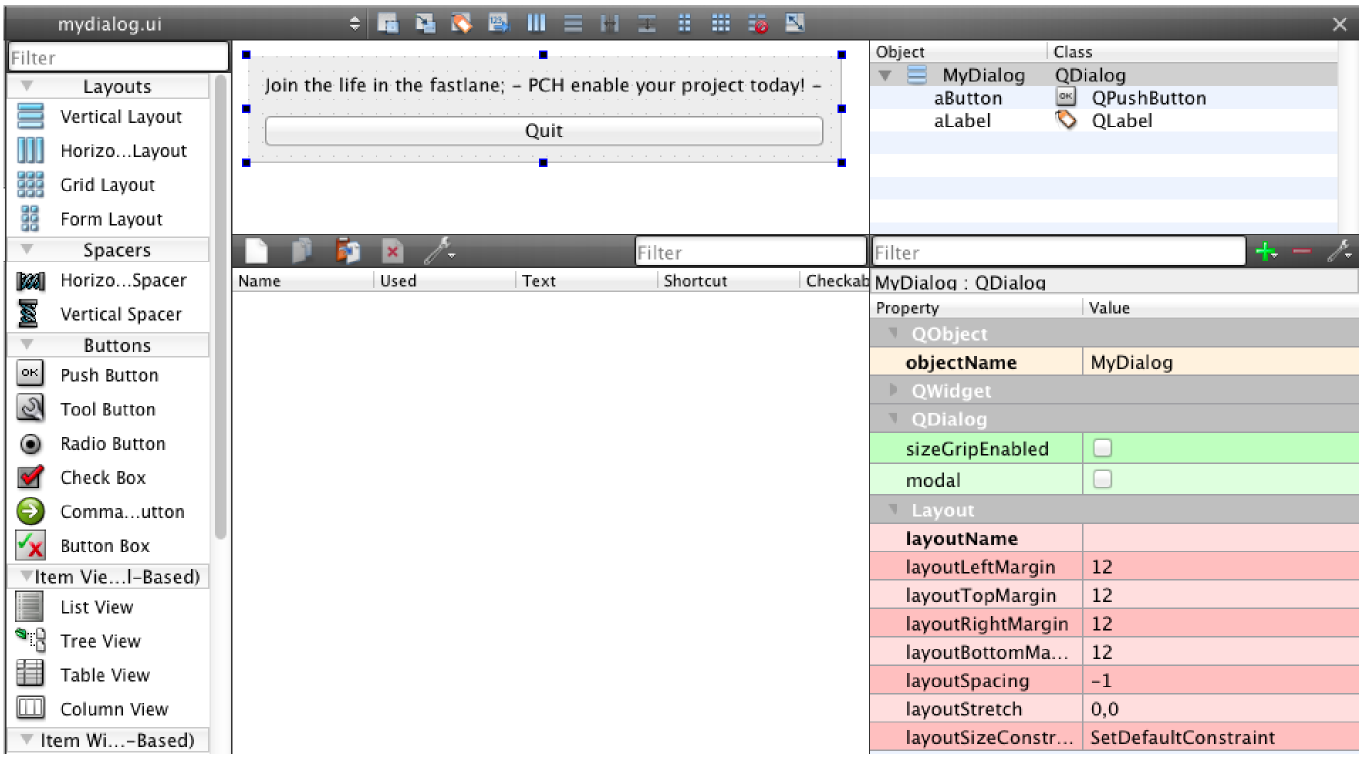
```
PRECOMPILED_HEADER = window.h
SOURCES            = window.cpp
```

To avoid potential conflicts like these, give distinctive names to header files that will be precompiled.

Example Project

You can find the following source code in the `examples/qmake/precompile` directory in the Qt distribution:

mode.



Join the life in the fastlane; - PCH enable your project today! -

Quit

Object	Class
MyDialog	QDialog
aButton	QPushButton
aLabel	QLabel

MyDialog : QDialog

Property	Value
QObject	
objectName	MyDialog
QWidget	
QDialog	
sizeGripEnabled	☐
modal	☐
Layout	
layoutName	
layoutLeftMargin	12
layoutTopMargin	12
layoutRightMargin	12
layoutBottomMa...	12
layoutSpacing	-1
layoutStretch	0,0
layoutSizeConstr...	SetDefaultConstraint

stable.h

```
/* Add C includes here */

#if defined __cplusplus
/* Add C++ includes here */

# include <iostream>
# include <QApplication>
# include <QPushButton>
# include <QLabel>
#endif
```

myobject.h

```
#include <QObject>

class MyObject : public QObject
{
public:
    MyObject();
    ~MyObject();
};
```

```
#include <iostream>
#include <QDebug>
#include <QObject>
#include "myobject.h"

MyObject::MyObject()
    : QObject()
{
    std::cout << "MyObject::MyObject()\n";
}
```

util.cpp

```
void util_function_does_nothing()
{
    // Nothing here...
    int x = 0;
    ++x;
}
```

main.cpp

```
#include <QApplication>
#include <QPushButton>
#include <QLabel>
#include "myobject.h"
#include "mydialog.h"

int main(int argc, char **argv)
{
    QApplication app(argc, argv);

    MyObject obj;
    MyDialog dialog;

    dialog.connect(dialog.aButton, SIGNAL(clicked()), SLOT(close()));
    dialog.show();

    return app.exec();
}
```

precompile.pro

```
TEMPLATE  = app
LANGUAGE  = C++
CONFIG   += cmdline precompile_header
```

```
PRECOMPILED_HEADER  = stable.h

HEADERS     = stable.h \
              mydialog.h \
              myobject.h
SOURCES     = main.cpp \
              mydialog.cpp \
              myobject.cpp \
              util.cpp
FORMS       = mydialog.ui
```

< Advanced Usage Configuring qmake >

Qt DOCUMENTATION

Wiki
Downloads
Marketplace

Feedback Sign In