Qt Creator

Using Qt Creator Value-Add Features to Further Productivity

株式会社SRA
朝木卓見

Qt JAPAN SUMMIT 2015

SRA GROUP

自己紹介

- 名前: 朝木卓見
- 職業: Qtコンサルタント, Qt Certified Specialist
- Qt歴: Qt 1.0.1(1996)〜
- 活動場所:
  - 株式会社SRA
  - http://qt-labs.jp
  - 日本Qtユーザー会

アジェンダ

- Qt Creator紹介
- 基本的な使い方
  - プロジェクトの作成
  - エディタ、Qt Quick Designer
  - デバッグ、QML Profiler
- 商用版の追加機能
- Qt Creatorをもっと使いこなそう

Qt Creatorとは

- クロスプラットフォーム統合開発環境(IDE)
  - Qtで作られたQtのためのIDE
  - プラグインアーキテクチャによる高い拡張性、柔軟性
  - 対応言語: C++, QML, JavaScript
  - 非Qtプロジェクトにも対応
  - 対応プラットフォーム: Windows, Linux, OS X
  - ターゲット: デスクトップ, 組み込みLinux, iOS, Android, VxWroks, QNX, ベアメタル, etc.
    - ベアメタル: OSを持たない組み込み用コンピュータ
    - http://en.wikipedia.org/wiki/Bare_machine

http://doc.qt.io/qtcreator/index.html

対応プラットフォーム

Linux

OS X

Windows

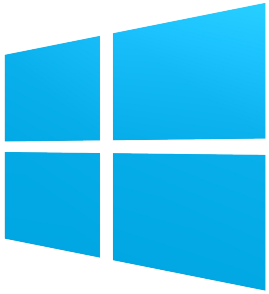
http://doc.qt.io/qtcreator/creator-os-supported-platforms.html

ターゲットプラットフォーム

iOS

VxWorks®

WIND RIVER



BlackBerry®

QNX™

http://doc.qt.io/qtcreator/creator-os-supported-platforms.html

Qt Creator


Qt Creator
ようこそ
編集
デザイン
Debug
プロジェクト
Analyze
ヘルプ
プロジェクト
サンプル
チュートリアル
Qt を始めよう!
アプリケーションの開発方法を学び、Qt Creator を探検する。
ここからスタート
Qt アカウント
Qt クラウドサービス
オンラインコミュニティ
ブログ
ユーザーガイド
新しいプロジェクト
プロジェクトを開く
セッション
最近使ったプロジェクト
default (現在のセッション)
main.c
1 問題
2 検索結果
3 アプリケーシ...
4 コンパイル出力
5 QML/JS コン...
6 To-Do エントリ
9 Test Results

主な機能

- エディタ: コード補完、ハイライト表示、分割表示、マルチウィンドウ、リファクタリング
- UIデザイン: GUIデザイナー(Qt Designer, Qt Quick Designer)
- デバッグ: GDB, LLDB, CDBに対応
- プロジェクト: プロジェクト固有設定(コードスタイルなど)、共有設定機能
- 解析: Valgrind, perf(Linux), QML Profiler
- ヘルプ: Qt Assistant

入手方法

- Qt Online Installer
  - Qt + Qt Creator
  - QtやQt Creatorをアップデート可能
- Qt Offline Installer
  - Qt + Qt Creator
  - バージョンを固定して利用する場合
- Qt Creator
  - Qt Creatorのみ
  - QtやQtのバイナリパッケージが不要な場合

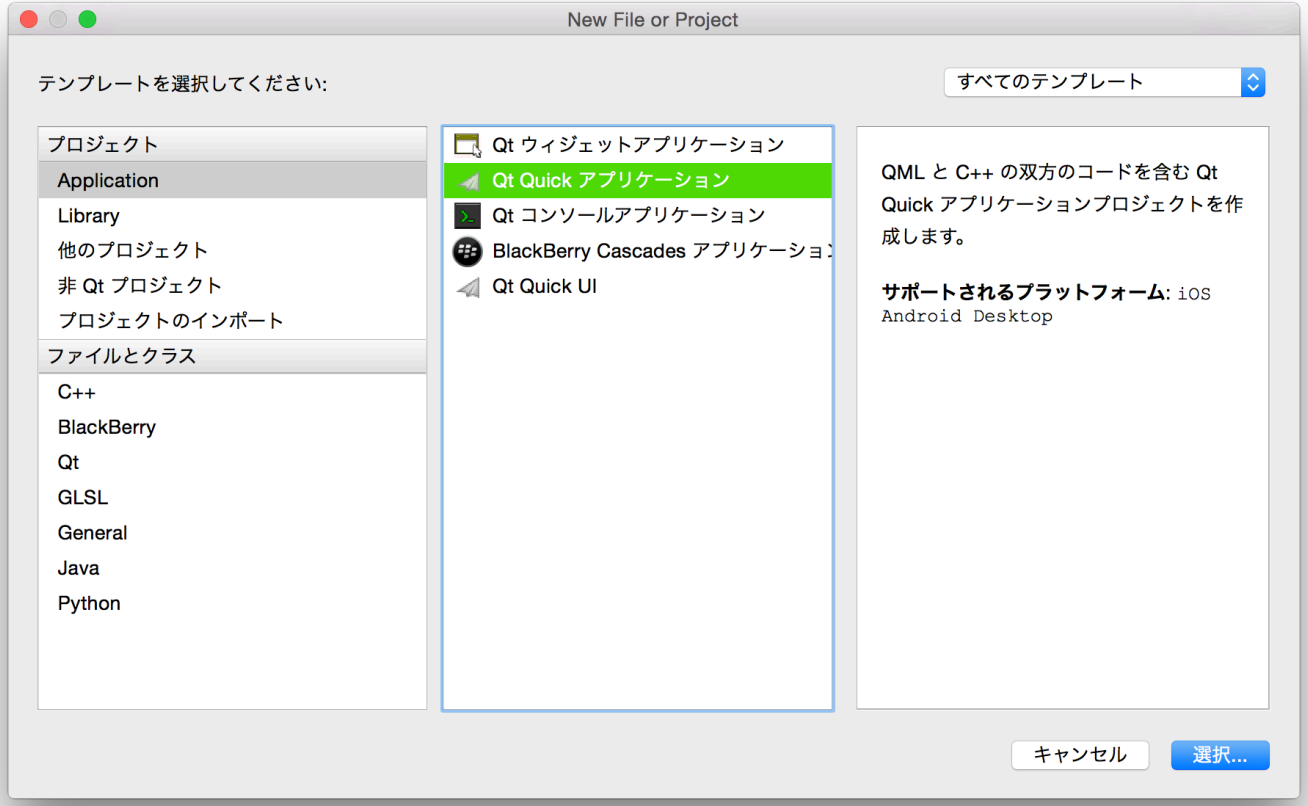
http://www.qt.io/download/

プロジェクトの作成

- 生成可能なプロジェクト
  - アプリケーション: Qt Widget, Qt Quick, Qt Console
  - ライブラリ: C++ライブラリ, Qt Quick プラグイン, Qt Creator プラグイン
  - 非Qt: cmake, qbs, qmakeプロジェクト(C, C++)
  - 外部プロジェクト: 各種バージョン管理システムのリポジトリクローン
  - その他: Qt ユニットテスト, subdirs qmake プロジェクト, 空のプロジェクト

http://doc.qt.io/qtcreator/creator-project-creating.html

プロジェクトの作成

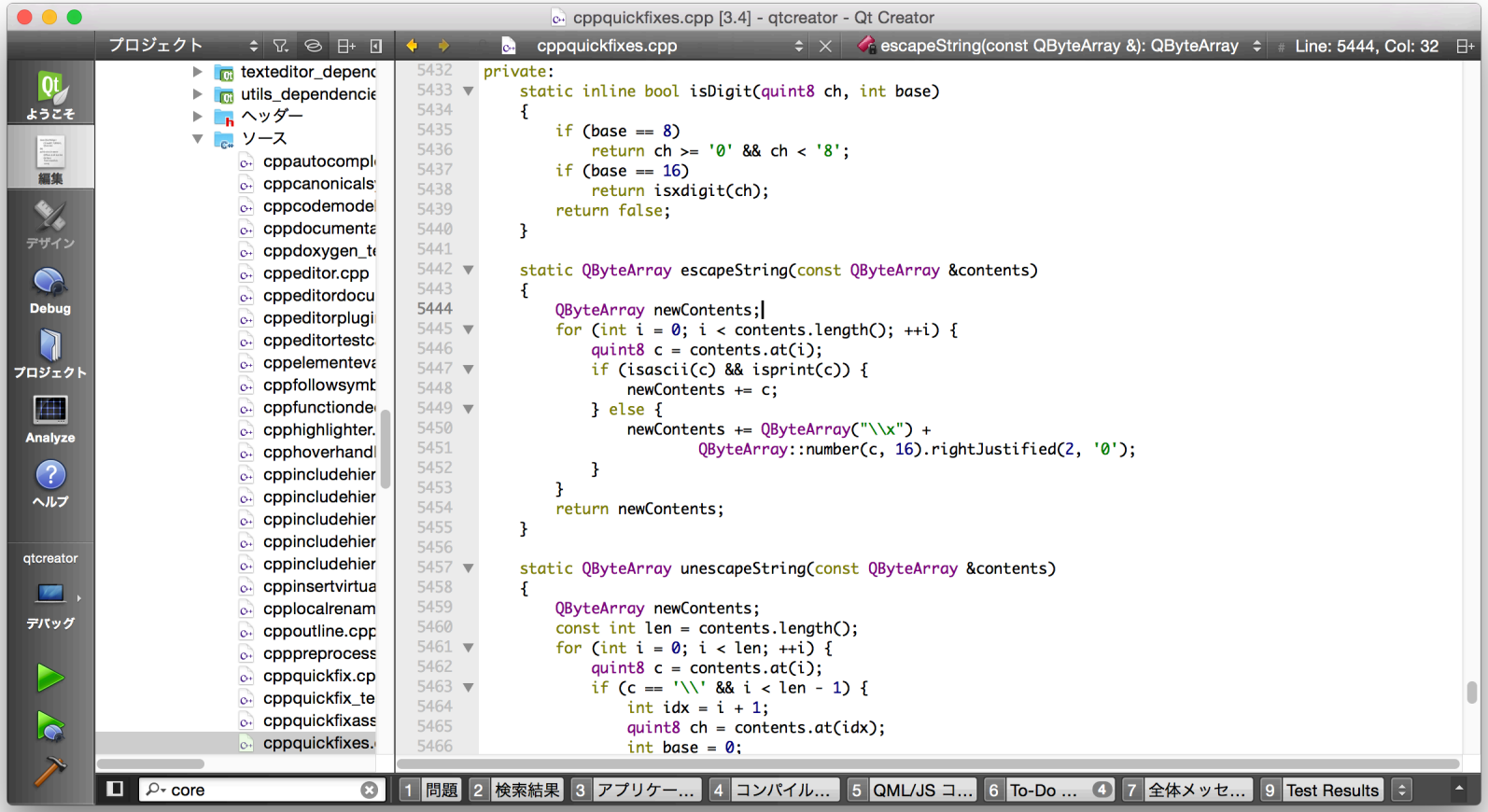

New File or Project
テンプレートを選択してください:
すべてのテンプレート
プロジェクト
Application
Library
他のプロジェクト
非 Qt プロジェクト
プロジェクトのインポート
ファイルとクラス
C++
BlackBerry
Qt
GLSL
General
Java
Python
Qt ウィジェットアプリケーション
Qt Quick アプリケーション
Qt コンソールアプリケーション
BlackBerry Cascades アプリケーション
Qt Quick UI
QML と C++ の双方のコードを含む Qt Quick アプリケーションプロジェクトを作成します。
サポートされるプラットフォーム: iOS Android Desktop
キャンセル
選択...

エディタ

- 主な機能
  - コード補完
    - シグナル・スロットも補完可能
  - シンタックスハイライト
  - 移動
  - ブックマーク
  - コードチェック
  - 自動インデント
  - コードテンプレート
  - キーボードマクロ

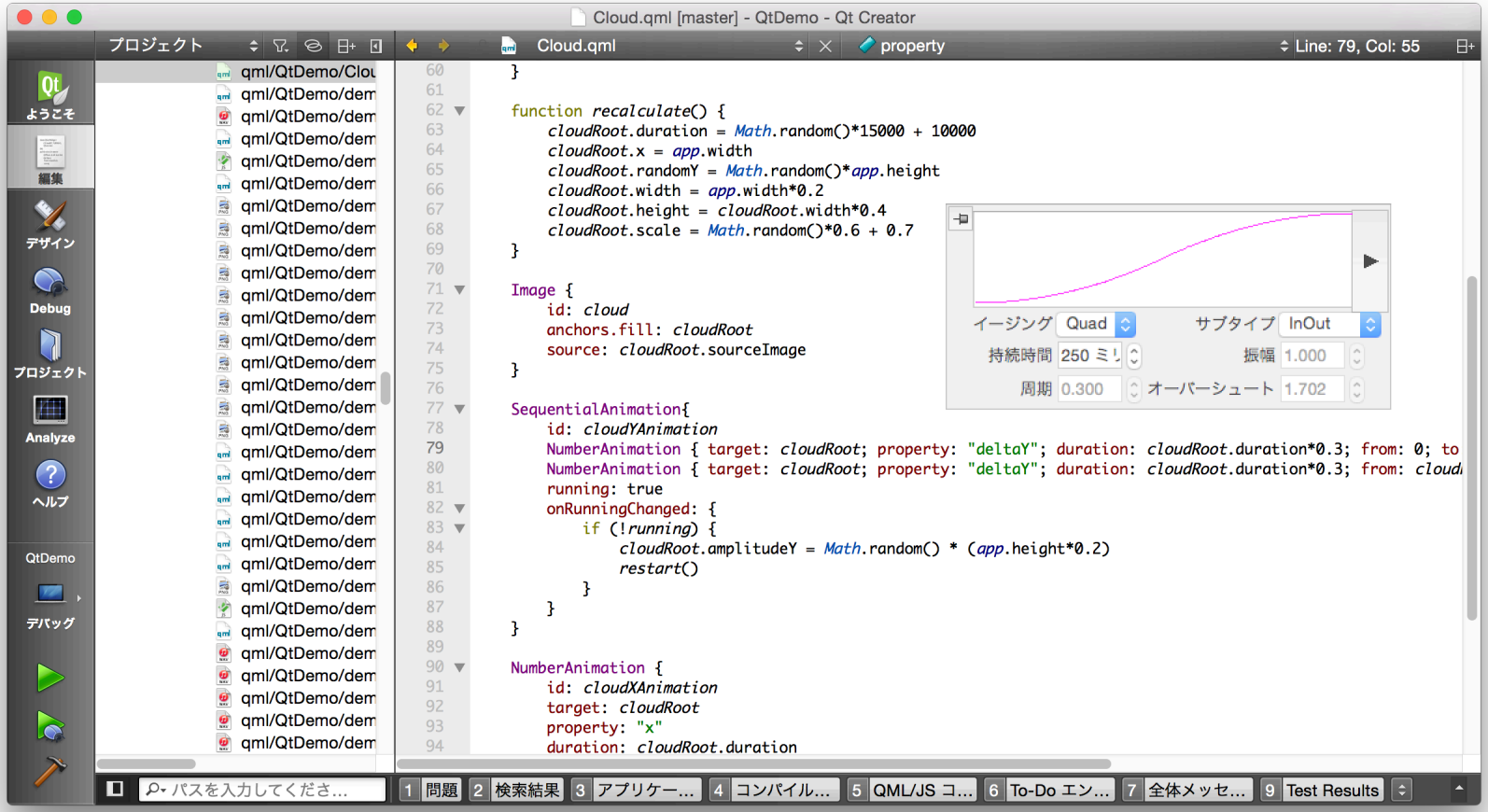
http://doc.qt.io/qtcreator/creator-editor-functions.html

エディタ: C++




```
5432 private:
5433     static inline bool isDigit(quint8 ch, int base)
5434     {
5435         if (base == 8)
5436             return ch >= '0' && ch < '8';
5437         if (base == 16)
5438             return isxdigit(ch);
5439         return false;
5440     }
5441
5442     static QByteArray escapeString(const QByteArray &contents)
5443     {
5444         QByteArray newContents;
5445         for (int i = 0; i < contents.length(); ++i) {
5446             quint8 c = contents.at(i);
5447             if (isascii(c) && isprint(c)) {
5448                 newContents += c;
5449             } else {
5450                 newContents += QByteArray("\\x") +
5451                         QByteArray::number(c, 16).rightJustified(2, '0');
5452             }
5453         }
5454         return newContents;
5455     }
5456
5457     static QByteArray unescapeString(const QByteArray &contents)
5458     {
5459         QByteArray newContents;
5460         const int len = contents.length();
5461         for (int i = 0; i < len; ++i) {
5462             quint8 c = contents.at(i);
5463             if (c == '\\' && i < len - 1) {
5464                 int idx = i + 1;
5465                 quint8 ch = contents.at(idx);
5466                 int base = 0;
```

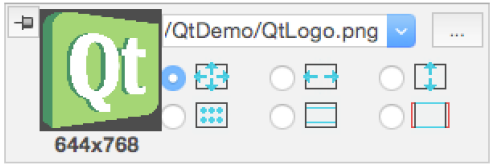
エディタ: QML

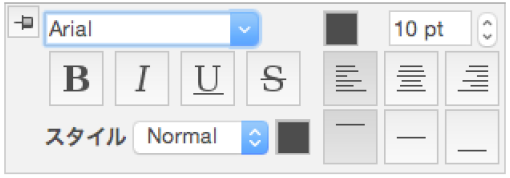


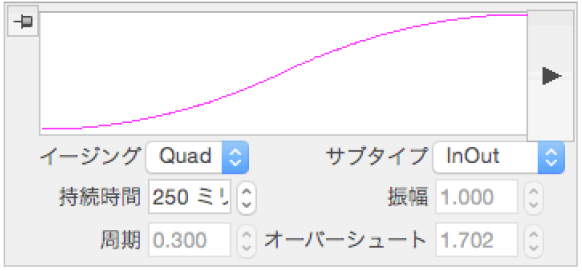

```
60      }
61
62      function recalculate() {
63          cloudRoot.duration = Math.random()*15000 + 10000
64          cloudRoot.x = app.width
65          cloudRoot.randomY = Math.random()*app.height
66          cloudRoot.width = app.width*0.2
67          cloudRoot.height = cloudRoot.width*0.4
68          cloudRoot.scale = Math.random()*0.6 + 0.7
69      }
70
71      Image {
72          id: cloud
73          anchors.fill: cloudRoot
74          source: cloudRoot.sourceImage
75      }
76
77      SequentialAnimation{
78          id: cloudYAnimation
79          NumberAnimation { target: cloudRoot; property: "deltaY"; duration: cloudRoot.duration*0.3; from: 0; to
80          NumberAnimation { target: cloudRoot; property: "deltaY"; duration: cloudRoot.duration*0.3; from: cloud
81          running: true
82          onRunningChanged: {
83              if (!running) {
84                  cloudRoot.amplitudeY = Math.random() * (app.height*0.2)
85                  restart()
86              }
87          }
88      }
89
90      NumberAnimation {
91          id: cloudXAnimation
92          target: cloudRoot
93          property: "x"
94          duration: cloudRoot.duration
```

エディタ: Qt Quick Toolbars

- QML編集中にプロパティの値をGUIで編集
  - Image, BorderImage: source, fillMode
  - Text: color, font, style, alignment
  - PropertyAnimation: duration, easing
  - Rectangle: color, gradient, border
- 右クリック or ツール→QML/JS から表示

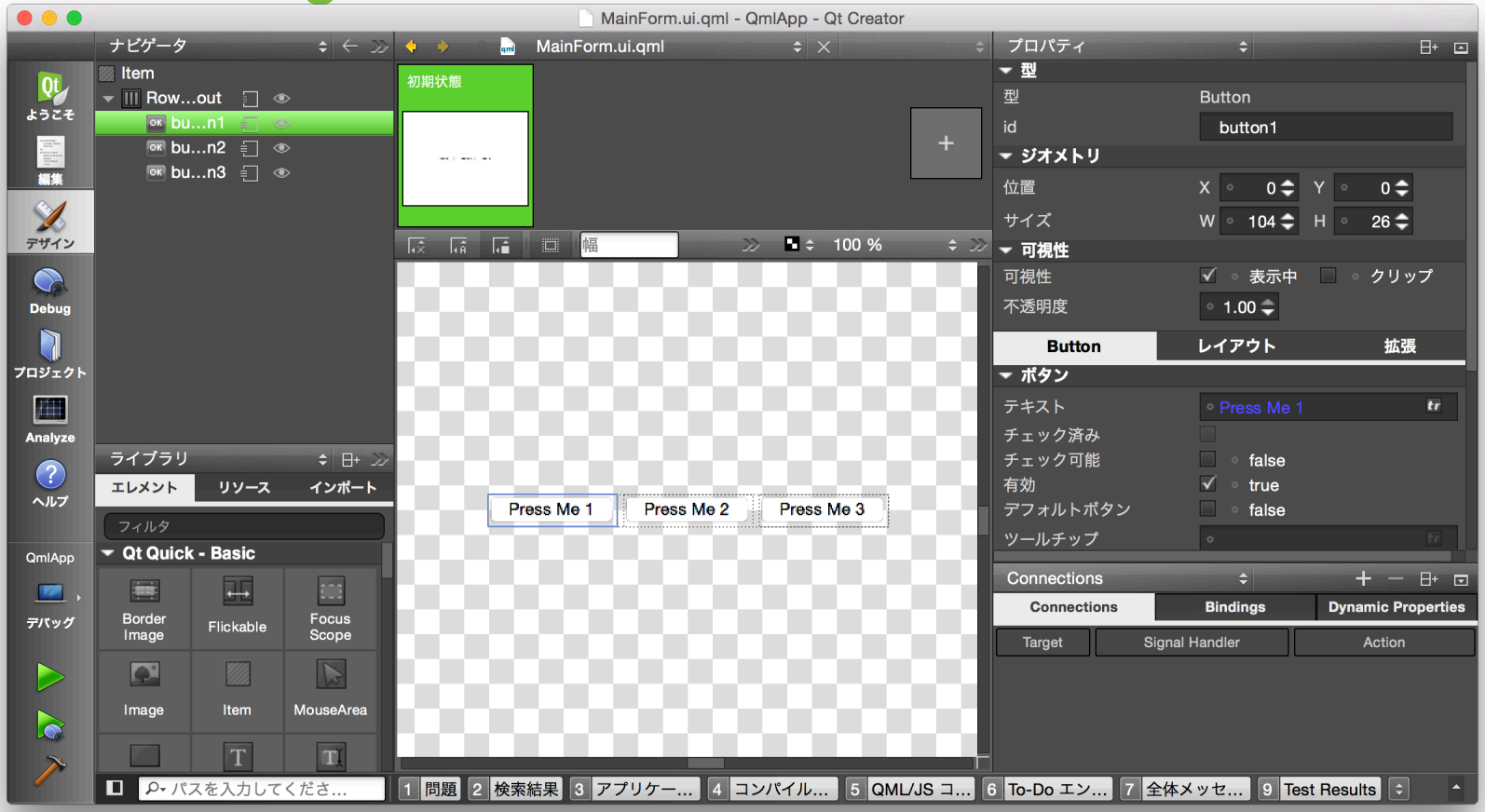






http://doc.qt.io/qtcreator/qt-quick-toolbars.html

Qt Quick Designer



http://doc.qt.io/qtcreator/creator-visual-editor.html

デバッグ

- GDB, LLDB, CDBに対応
- QML/JavaScriptのデバッグにも対応
  - JavaScriptへのブレークポイント設定
  - QMLのインスペクタ

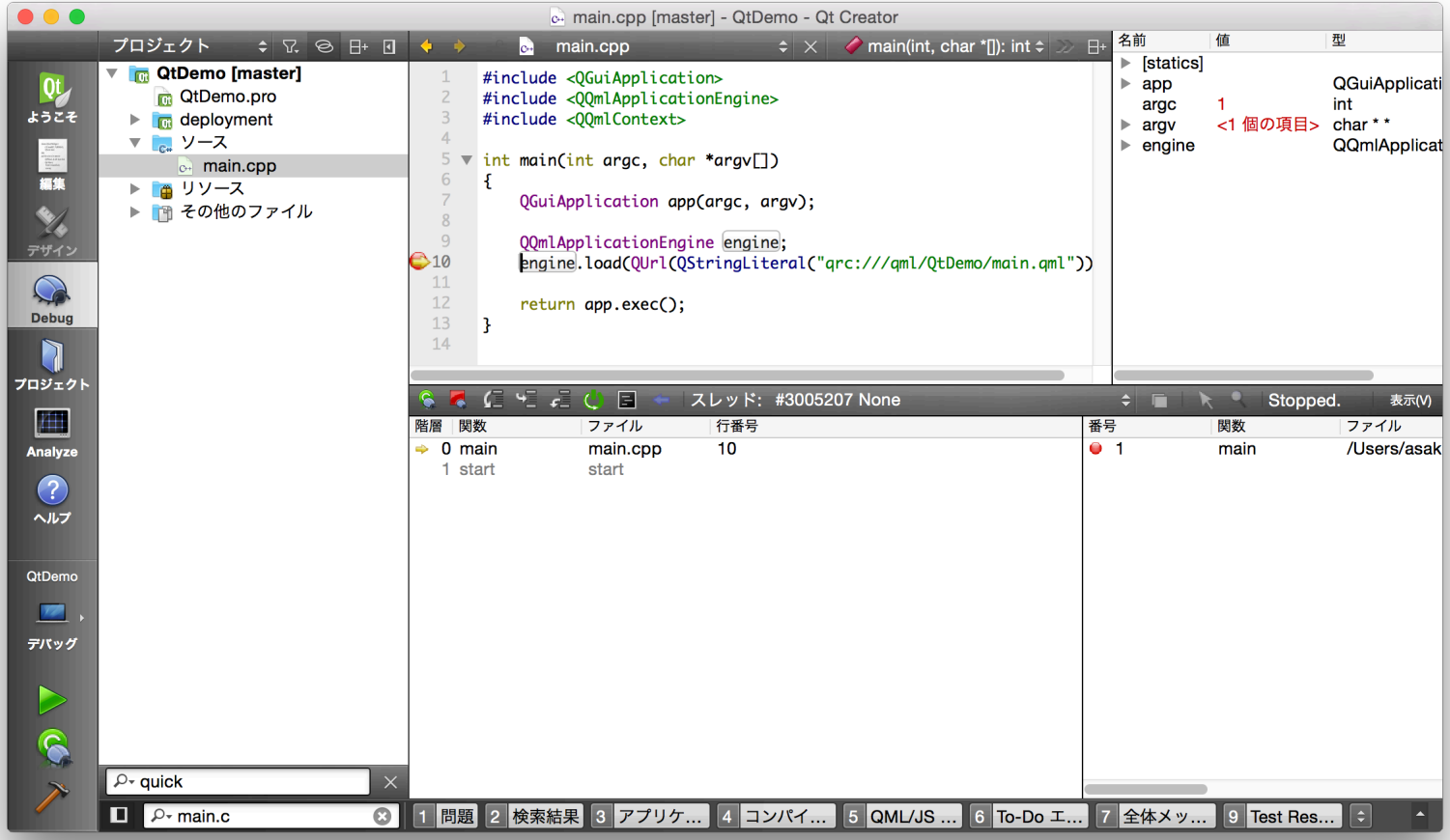
- GDB: Python scripting extensionへの対応が必要
  - 特に組み込みLinuxの場合に注意

Windowsでのデバッグ(Visual Studio)

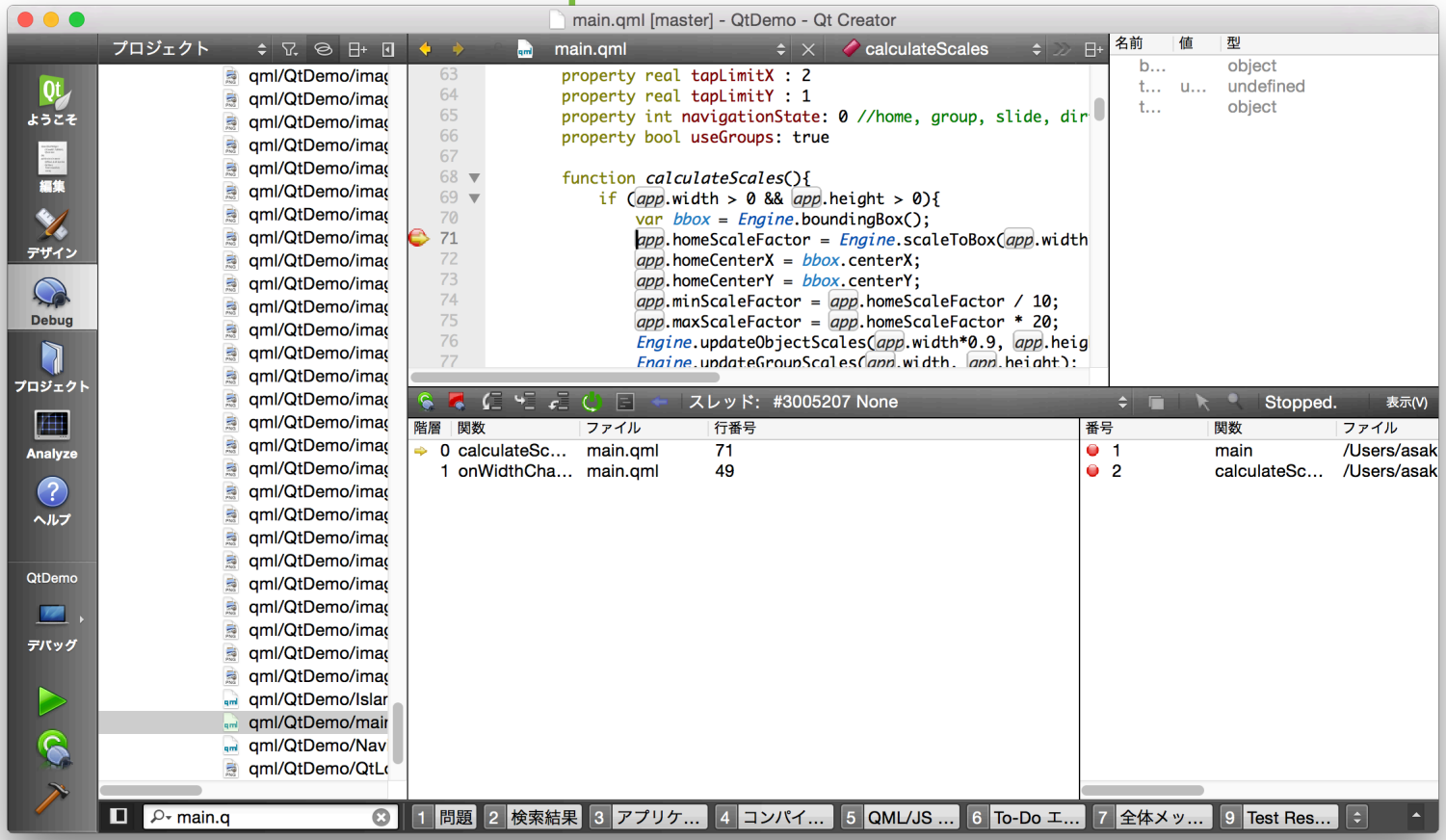
- Debugging tools for Windowsのインストールが必要
  - シンボルサーバーのセットアップも必要
- デバッグが遅い場合
  - リビルド
  - proファイルに以下を追加
    - QMAKE_LFLAGS_DEBUG += /INCREMENTAL:NO

http://wiki.qt.io/Qt_Creator_Windows_Debugging

デバッグ: C++


main.cpp [master] - QtDemo - Qt Creator
プロジェクト
QtDemo [master]
QtDemo.pro
deployment
ソース
main.cpp
リソース
その他のファイル
main.cpp
main(int, char *[]): int
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext>
int main(int argc, char *argv[])
{
QGuiApplication app(argc, argv);
QQmlApplicationEngine engine;
engine.load(QUrl(QStringLiteral("qrc:///qml/QtDemo/main.qml")))
return app.exec();
}
名前 値 型
[statics]
app QGuiApplicati
argc 1 int
argv <1 個の項目> char * *
engine QQmlApplicat
スレッド: #3005207 None
Stopped.
表示(V)
階層 関数 ファイル 行番号
0 main main.cpp 10
1 start start
番号 関数 ファイル
1 main /Users/asak
ようこそ
編集
デザイン
Debug
プロジェクト
Analyze
ヘルプ
QtDemo
デバッグ
quick
main.c
1 問題
2 検索結果
3 アプリケ...
4 コンパイ...
5 QML/JS ...
6 To-Do エ...
7 全体メッ...
9 Test Res...

デバッグ: QML/JavaScript



解析

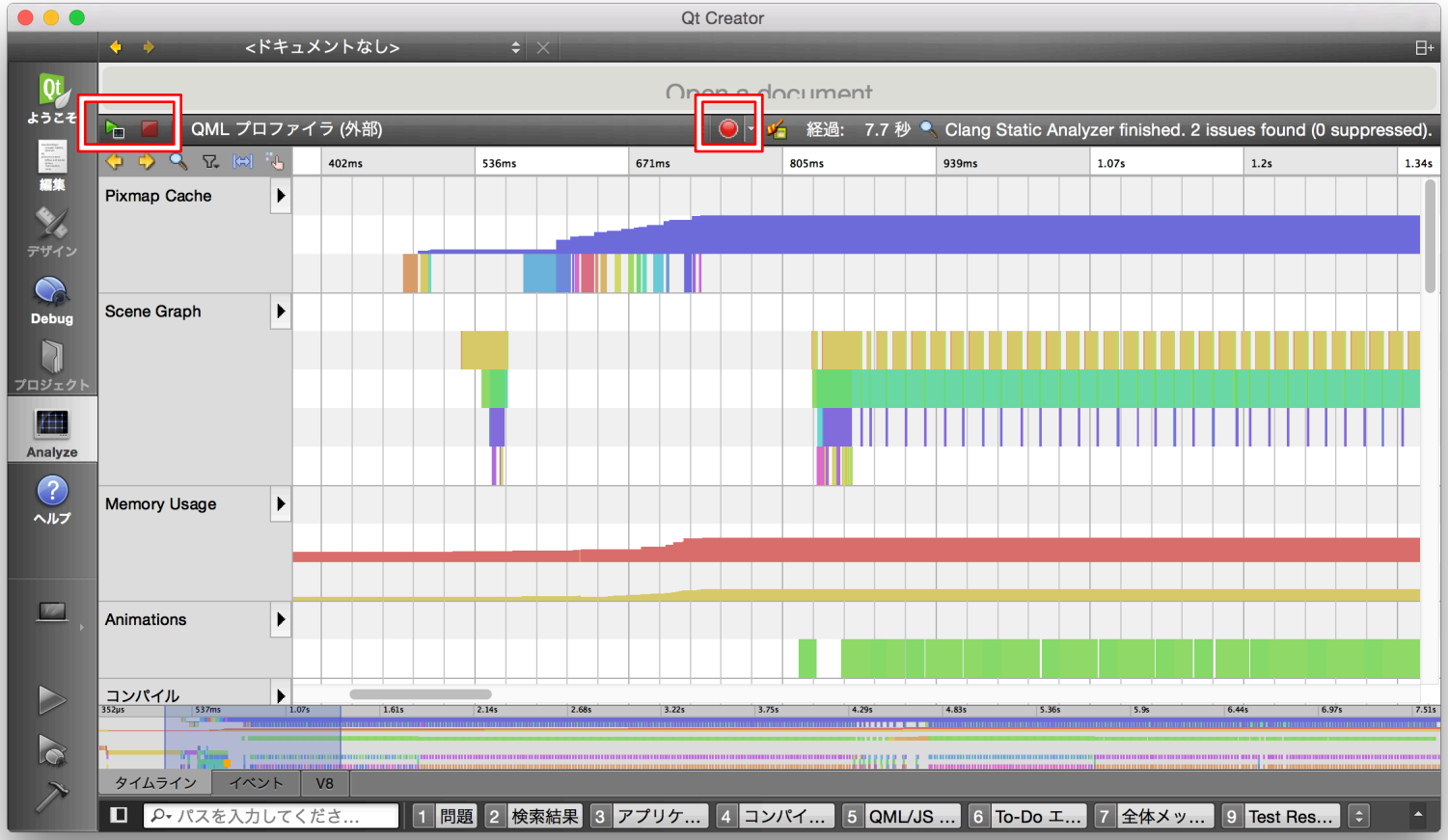
- QML Profiler
- Valgrind Code Analysis Tool
  - メモリアナライザ: http://doc.qt.io/qtcreator/creator-analyzer.html
  - 関数プロファイラ: http://doc.qt.io/qtcreator/creator-cache-profiler.html
- Clang Static Analyzer(商用)
- CPU Usage Analyzer(商用)

QML Profiler

- QMLのプロファイリング
  - qmlのコンパイル、アイテムの生成、バインディング、シグナルの処理時間
  - 描画、シーングラフ(商用版)
- リモート実行時も取得可能
  - ネットワーク経由 or qmlprofilerコマンド経由
- Qt Quick 2(Qt 5)で対応イベントを大幅に拡大
  - オーバーヘッド増
- パフォーマンス解析には必須

http://doc.qt.io/qtcreator/creator-qml-performance-monitor.html

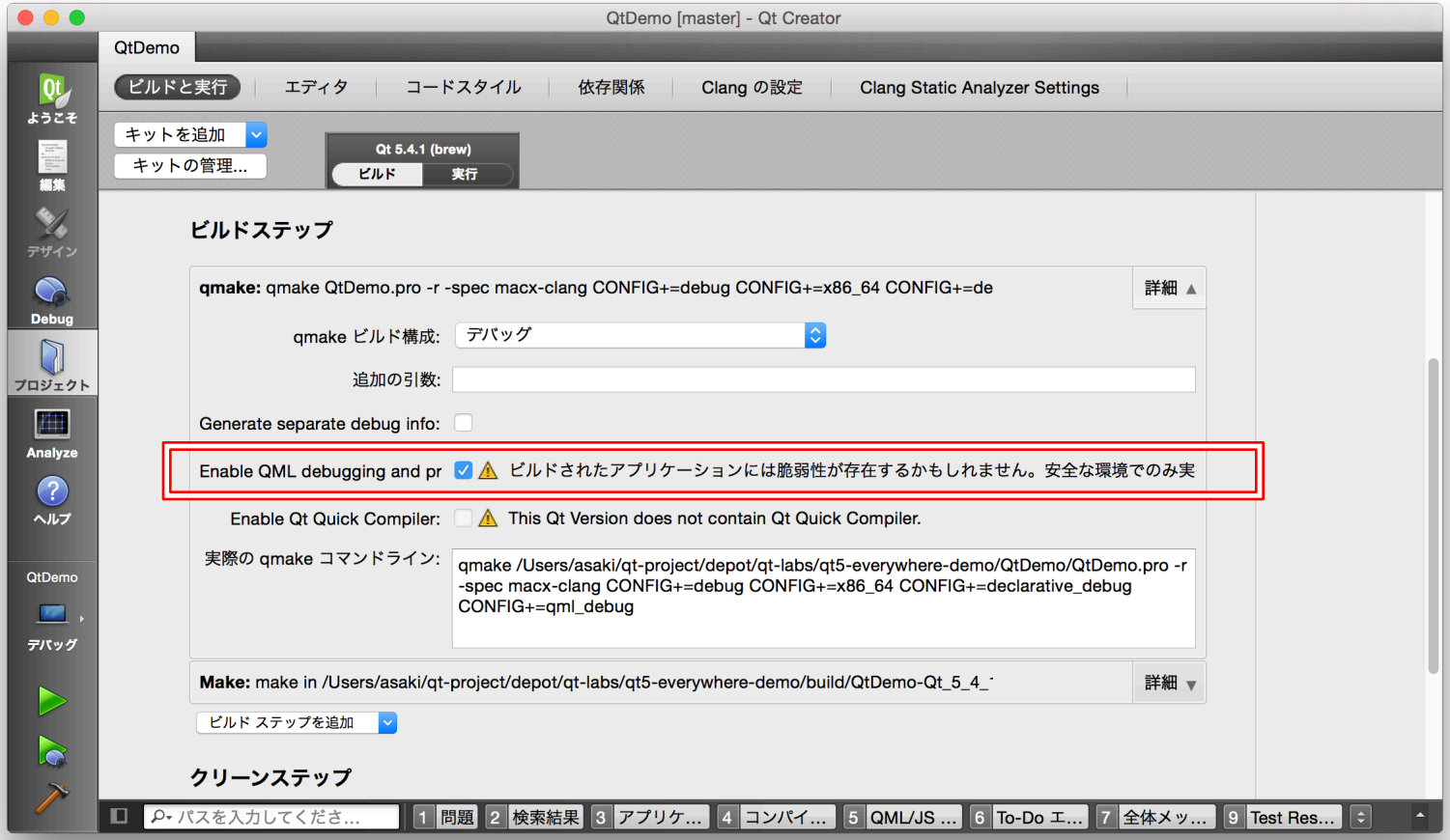
QML Profiler



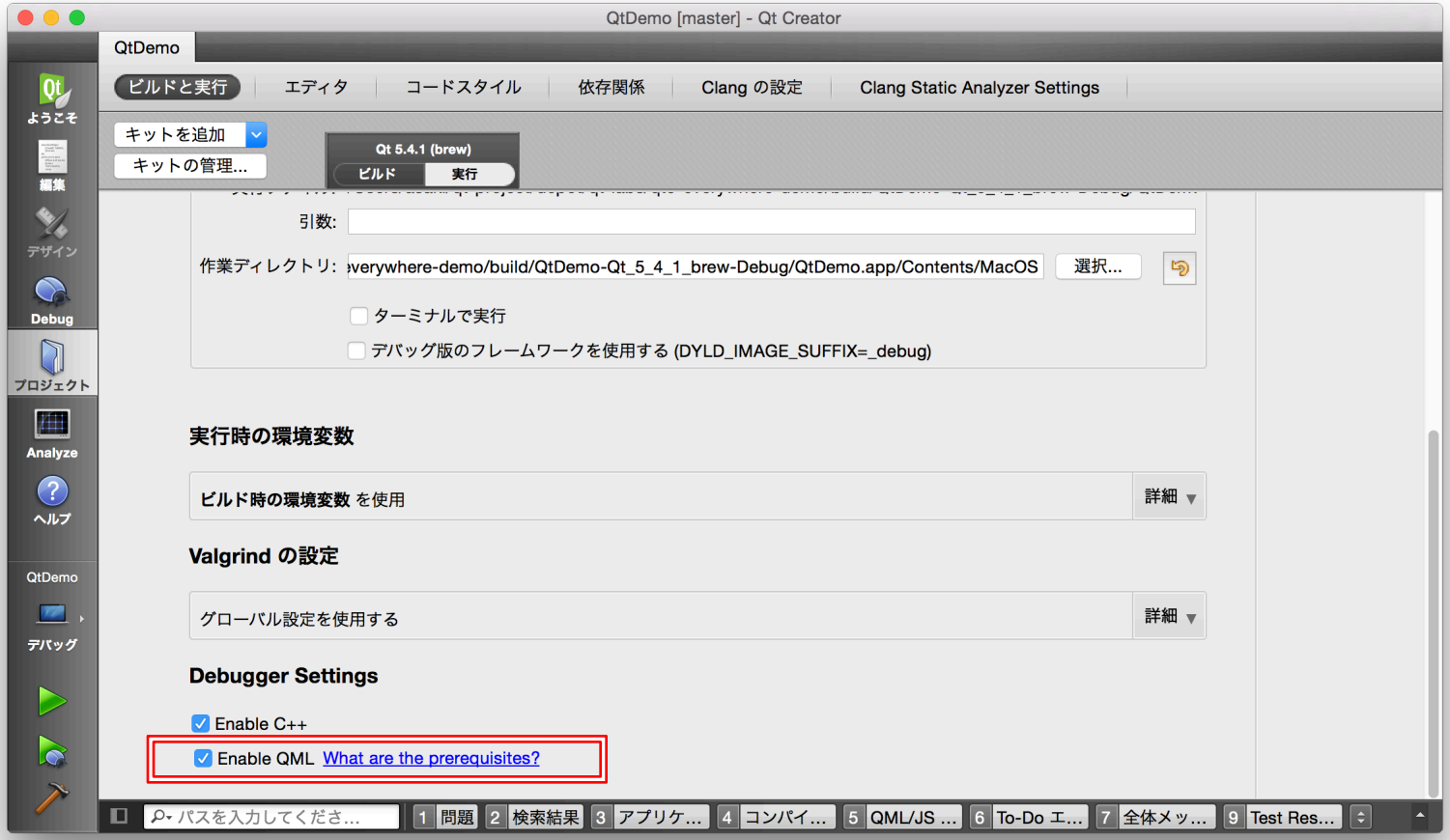
QML Profilerの使い方

- プロジェクトの設定(リリースビルド時に注意)
  - ビルド: ビルドステップ → qmake → 詳細
    - “Enable QML debugging and profiling:” に✓
  - 実行: Debugger Setting
    - “Enable QML” に✓
- Analyze → QML プロファイラ
  - アプリケーションがQt Creatorから起動可能な場合
  - リモート実行時も可
- Analyze → QML プロファイラ (外部)
  - アプリケーションの起動を手動で行う場合

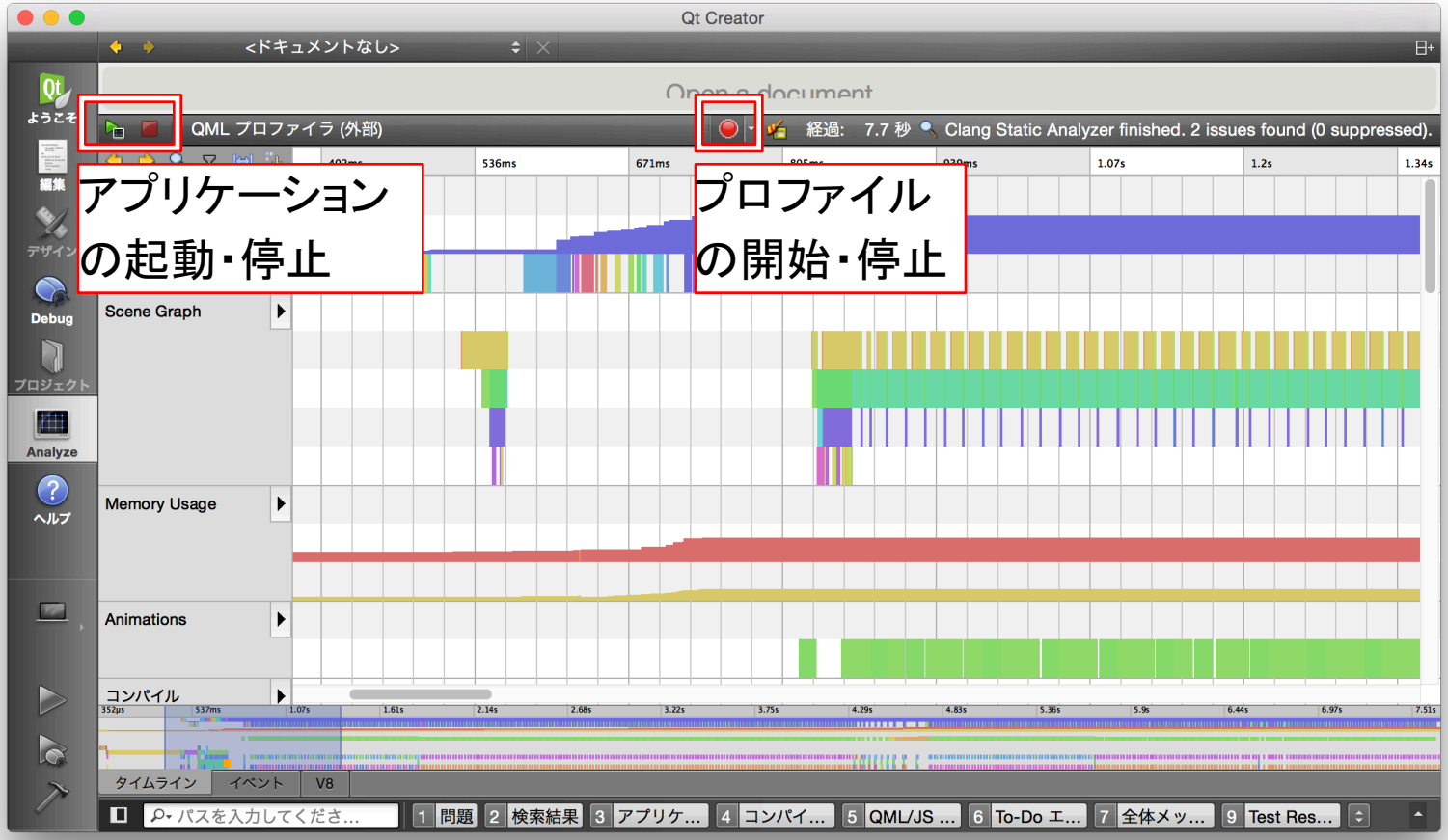
QML Profiler: ビルド設定



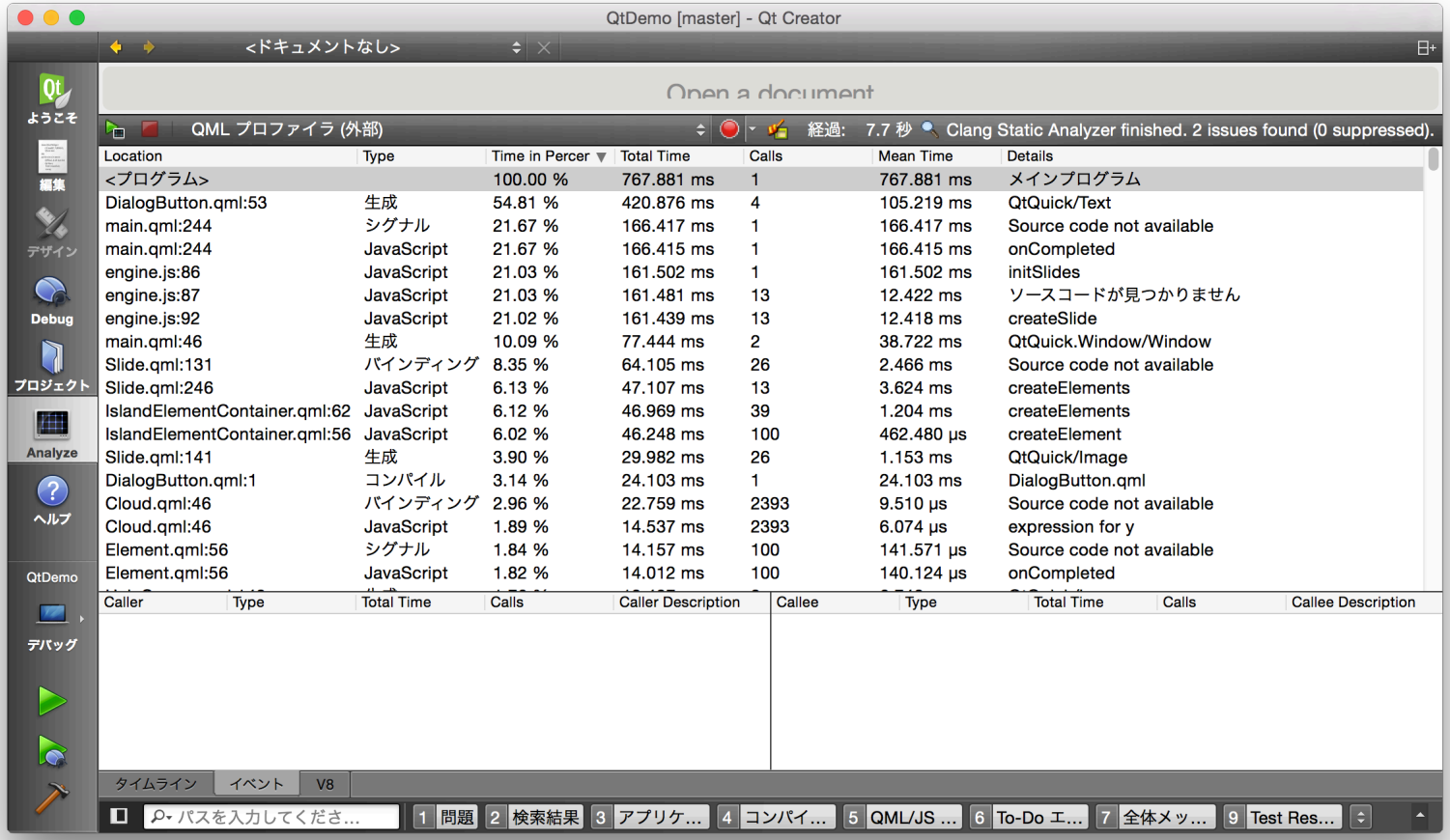
QML Profiler: 実行設定



QML Profiler: タイムライン



QML Profiler: イベント



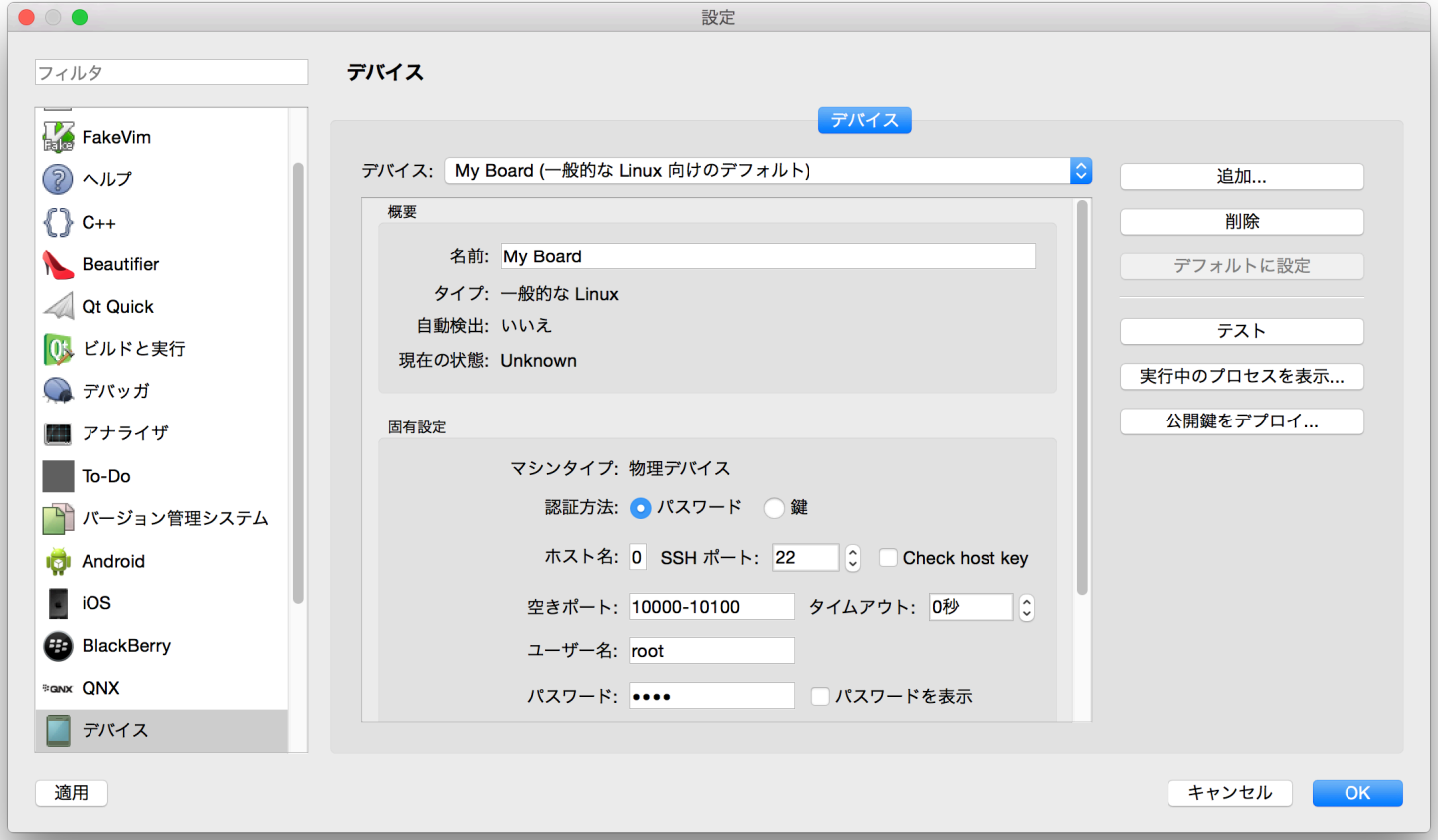
組み込みLinux対応

- Qt for Device Creation (Boot 2 Qt)
  - 設定不要で利用可能
  - ボードを自動的に認識
  - adbを用いたボード制御
- その他のQt (BSP付属、独自ビルド)
  - キットの作成が必要
    - キット: Qtやツールチェイン、デバイスの設定をまとめたもの

組み込みLinux対応設定

- ホスト側
  - Qt:設定→ビルドと実行→Qtバージョン
    - http://doc.qt.io/qtcreator/creator-project-qmake.html
  - コンパイラ: 設定→ビルドと実行→コンパイラ
    - http://doc.qt.io/qtcreator/creator-tool-chains.html
  - デバッガ: 設定→ビルドと実行→Debuggers
    - Python scripting extensionに対応したGDBが必要
    - http://doc.qt.io/qtcreator/creator-debugger-engines.html
  - デバイス: 設定→デバイス
    - 実行するデバイスのIPアドレスなど
    - http://doc.qt.io/qtcreator/creator-developing-generic-linux.html
  - キット: 設定→ビルドと実行→キット
    - Qt, コンパイラ, デバッガ, デバイスの組み合わせ
    - http://doc.qt.io/qtcreator/creator-targets.html
- デバイス側
  - ssh, sftpサーバ
  - ホストとのネットワークアクセス
  - ホストのデバッガに対応するgdbserver

設定: デバイス



http://doc.qt.io/qtcreator/creator-developing-generic-linux.html

商用版の機能

機能	Windows	Linux	Mac	備考
Auto Test	○	○	○	
Clang Static Analyzer	○	○	○	要clang
QML Profiler拡張	○	○	○	
Qt Quick Designer拡張	○	○	○	
CPU Usage Analyzer	−	△	−	要perf、Boot2Qt 5.x用
Boot2Qt対応	−	○	−	
VxWorks対応	○	○	−	

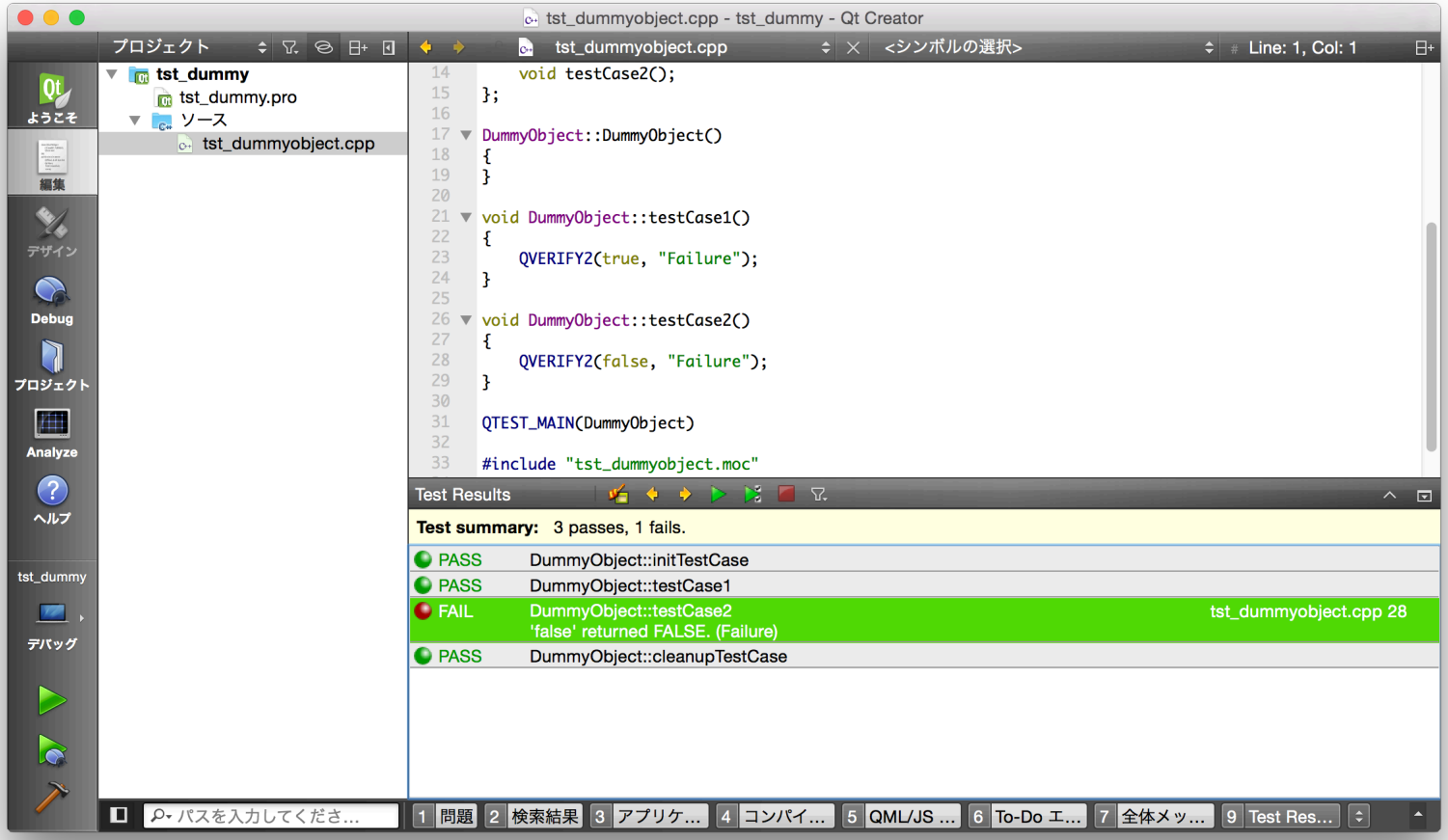
http://doc.qt.io/qtcreator/creator-commercial-overview.html

Auto Test

- Qt Testモジュールを用いたユニットテスト用
  - Auto Testプロジェクトテンプレート
    - OS X 版にはバグで同梱されていない: https://bugreports.qt.io/browse/QTCREATORBUG-14490
  - Qt Testの出力を整形
  - 実行するテストの選択

http://doc.qt.io/qtcreator/creator-autotest.html

Auto Test


tst_dummyobject.cpp - tst_dummy - Qt Creator
プロジェクト
tst_dummy
tst_dummy.pro
ソース
tst_dummyobject.cpp
tst_dummyobject.cpp
<シンボルの選択>
Line: 1, Col: 1
14 void testCase2();
15 };
16
17 DummyObject::DummyObject()
18 {
19 }
20
21 void DummyObject::testCase1()
22 {
23 QVERIFY2(true, "Failure");
24 }
25
26 void DummyObject::testCase2()
27 {
28 QVERIFY2(false, "Failure");
29 }
30
31 QTEST_MAIN(DummyObject)
32
33 #include "tst_dummyobject.moc"
Test Results
Test summary: 3 passes, 1 fails.
PASS DummyObject::initTestCase
PASS DummyObject::testCase1
FAIL DummyObject::testCase2 tst_dummyobject.cpp 28
'false' returned FALSE. (Failure)
PASS DummyObject::cleanupTestCase
ようこそ
編集
デザイン
Debug
プロジェクト
Analyze
ヘルプ
tst_dummy
デバッグ
パスを入力してくださ...
1 問題
2 検索結果
3 アプリケ...
4 コンパイ...
5 QML/JS ...
6 To-Do エ...
7 全体メッ...
9 Test Res...

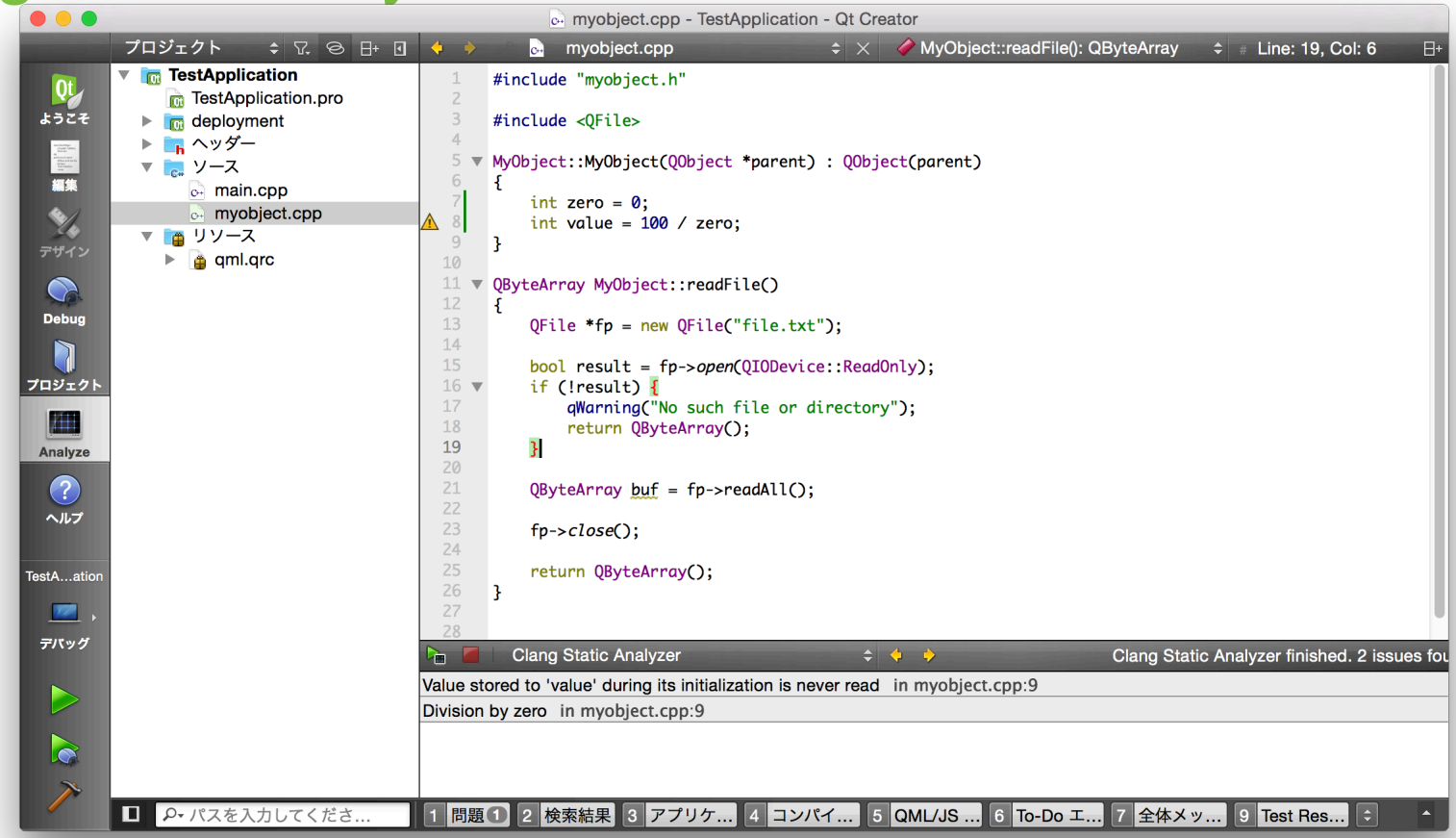
Clang Static Analyzer

- Clangを用いて静的解析を行う
- 対応する項目はClangに依存
  - http://clang-analyzer.llvm.org/available_checks.html
- clangコマンドの別途インストールが必要

- 簡易チェックとしては有用

http://doc.qt.io/qtcreator/creator-clang-static-analyzer.html

Clang Static Analyzer

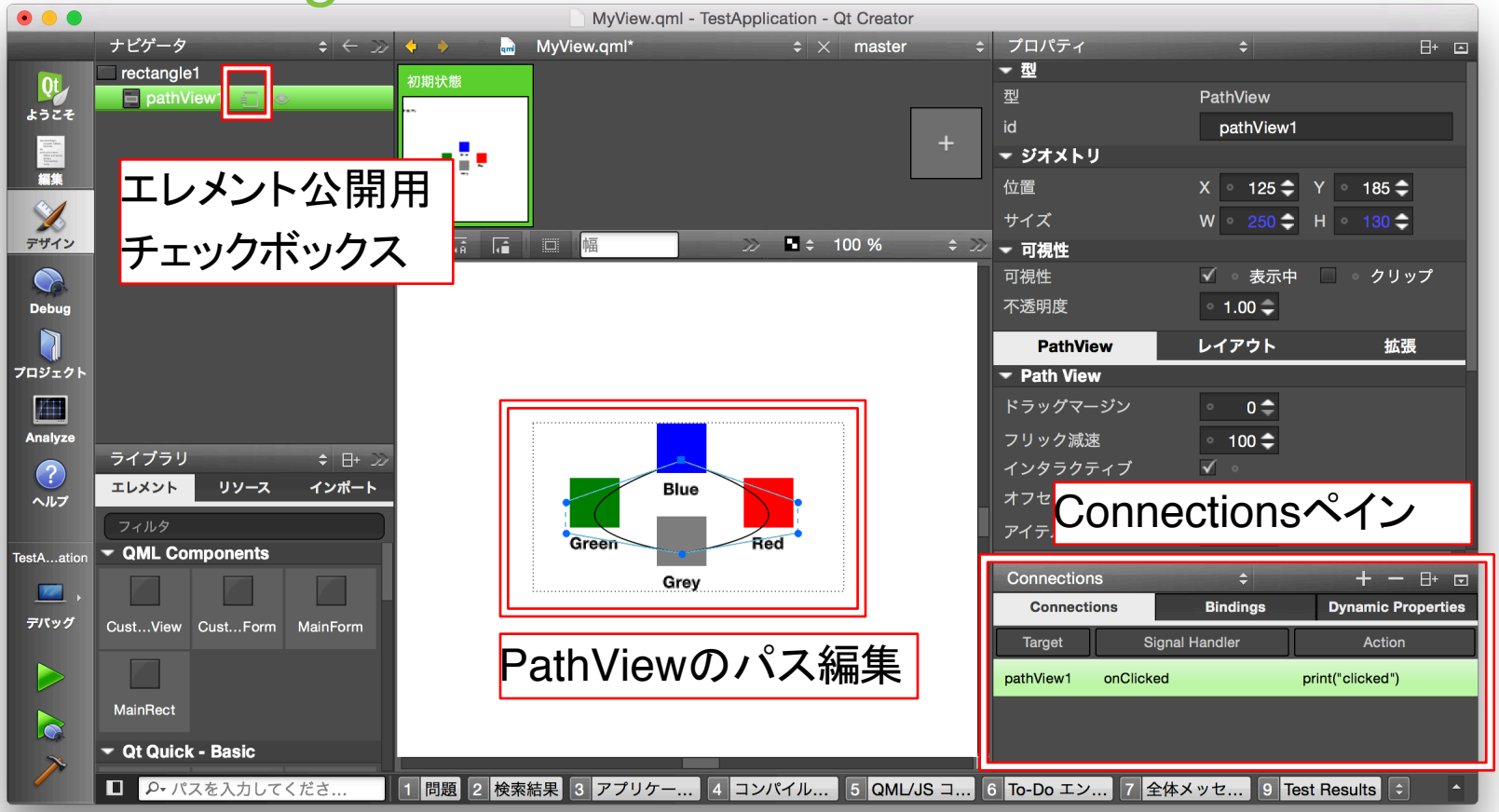


Qt Quick Designer拡張

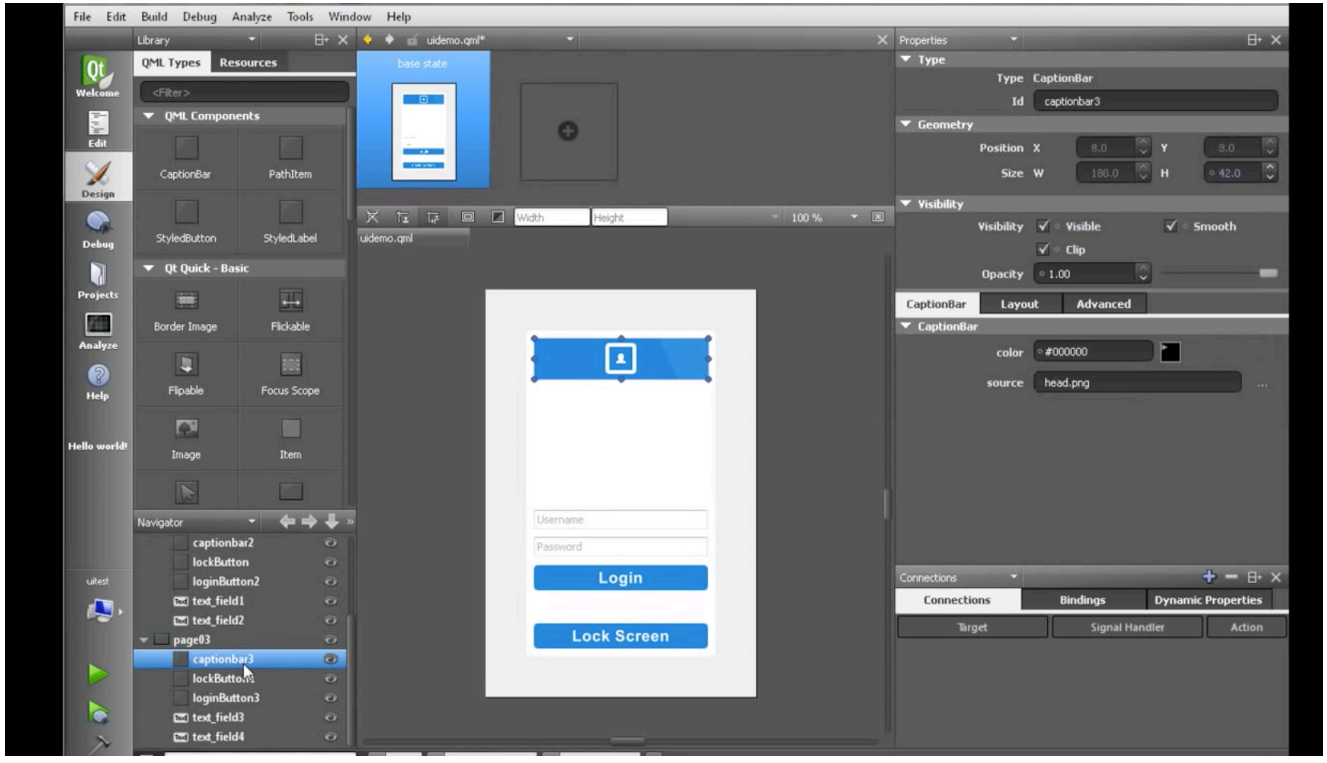
- PathViewのパスの編集機能
  - Qt Quick UI FormではPathViewが使えないため工夫が必要
- エレメント公開用チェックボックス
  - Qt Quick UI Formでほぼ必須
- Connectionsペイン
  - Connection エレメントを作成
  - Qt DesignerのようにGUIでの編集は不可能
  - エレメント内の onFoo ハンドラは対象外

http://doc.qt.io/qtcreator/creator-qtquick-designer-extensions.html

Qt Quick Designer拡張



Qt Quick Designer(商用版)の使用例



https://www.youtube.com/watch?v=cOViDcYWNCI

QML Profiler拡張機能

イベント	Qt Quick 1	Qt Quick 2	商用版のみ	備考
Pixmap Cache		✓	✓	
Scene Graph		✓	✓	Qt Quick 2の描画イベント
Memory Usage		✓	✓	JavaScriptのヒープサイズ
Input Events	✓	✓	✓	
Painting	✓			
Animations		✓		
Compiling	✓	✓		
Creating	✓	✓		
Binding	✓	✓		
Handling Signal	✓	✓		
JavaScript		✓		JavaScriptの関数実行

http://doc.qt.io/qtcreator/creator-qml-performance-monitor.html

QML Profilerの拡張イベント

ツール	保存	表示	備考
Qt Creator(OSS)	×	×	
Qt Creator(商用)	○	○	
qmlprofiler	○	ー	コマンドラインツール

http://doc.qt.io/qtcreator/creator-qml-performance-monitor.html

CPU Usage Analyzer

- Linuxカーネルのperf機能を用いてCPUの利用状況を解析
  - サンプリングベースのプロファイル
  - perfコマンドの別途インストールが必要
  - Boot2Qt 5.x以降用
    - 現時点では利用不可能(Boot2Qt 4.x)
    - perfparserコマンドがB2Qt 5.xに付属予定
- “Generate separate debug info”を推奨
  - Project→ビルド→qmake→詳細

http://doc.qt.io/qtcreator/creator-cpu-usage-analyzer.html

CPU Usage Analyzer

Qt Quick Compiler(商用版)

- プロジェクトの設定で有効化
  - ビルド→ビルドステップ→qmake→詳細→“Enable Qt Quick Compiler:” に✓
  - プロジェクトのリビルド
- Qt Quick Compiler
  - qmlファイルをC++にコンパイル
    - qmlファイルの実行時コンパイル時間を削減
  - qmlファイルがqrcで埋め込まれていること
    - システム側のqml(Qt Quick Controls, Qt Graphical Effects)は対象外
  - QMLデバッグとの併用は不可能(プロファイルは可能)
  - アプリのコンパイル時間、バイナリサイズ増大

http://doc.qt.io/QtQuickCompiler/

Qt Quick Compiler

設定

- 便利な設定
  - C++→ファイル命名規則
    - ファイル名をキャメルケースにする場合
    - ライセンステンプレートの設定

プラグインの有効・無効化

- Qt Creatorの機能はプラグインで追加・削除可能
- ヘルプ → プラグインについて
- デフォルトで無効なプラグイン
  - Build Systems: AutotoolsProjectManager
  - C++: Beautifier, ClangCodeModel
  - Device Support: BareMetal, VxWorks
  - Qt Quick: QmlProjectManager
  - Utilities: AutoTest, EmacsKeys, HelloWorld, Todo

http://doc.qt.io/qtcreator/creator-configuring.html#enabling-and-disabling-plugins

コードモデル

- ソースコードの情報をモデル化したもの
  - 補完、ハイライト、高度な検索などに利用
- 二つのコードモデル
  - Qt Creator内製モデル: デフォルト
  - Clangコードモデル: Clangベースのコードモデル
    - 遅いがより正確、C++11などへの対応
    - デフォルトでは無効
    - 補完とハイライトにのみ対応
- ヘルプ → プラグインについて → C++ → ClangCodeModel に✓
  - 変更後、Qt Creatorの再起動が必要
- 設定 → C++ → コードモデル → 各言語 → Clang

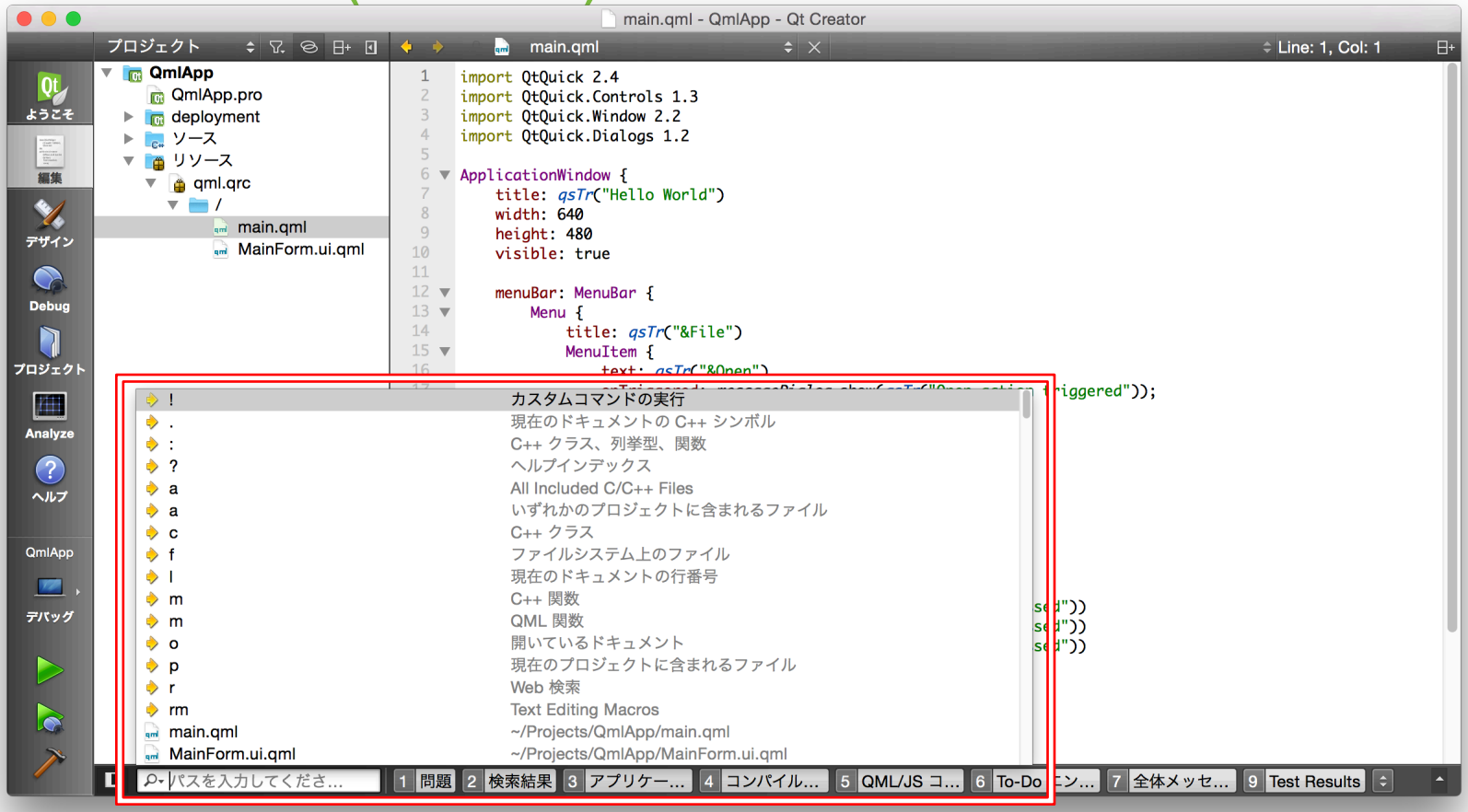
http://doc.qt.io/qtcreator/creator-clang-codemodel.html

クイックアクセス(Locator)

- ウィンドウ左下の入力欄
  - Ctrl+K(OS X: Command+K)で開始
  - ファイルやヘルプなどへ素早くアクセス
    - プレフィックス無し: 全プロジェクトのファイル
    - ? : ヘルプ
    - l : 行番号指定
    - o : 編集中のファイル
    - r : Web検索

http://doc.qt.io/qtcreator/creator-editor-locator.html

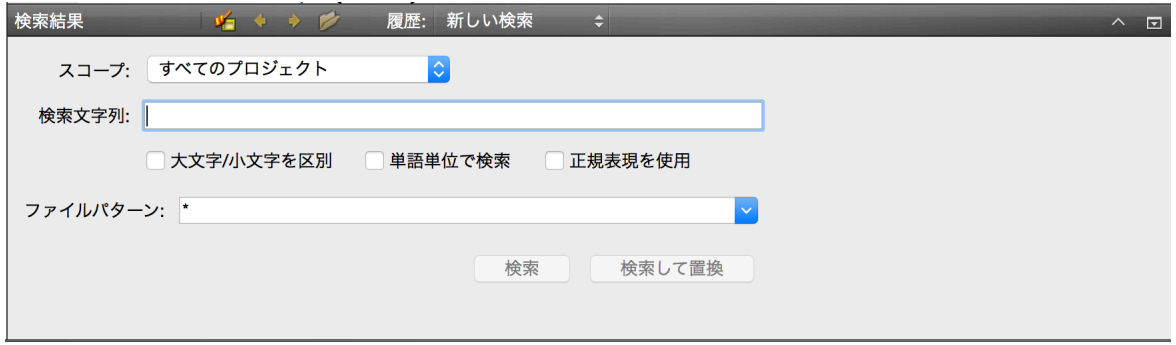
クイックアクセス(Locator)



検索

- ファイル内検索: Ctrl + F

  - オプションはサーチアイコンをクリックして変更
- 複数のファイルを検索: Ctrl + Shift + F



http://doc.qt.io/qtcreator/creator-editor-finding.html

QMLのチェック

- ツール → QML/JS → チェックを実行
- アプリケーションのQMLファイルの簡易静的解析を実行
  - QML のプロパティの型
  - パフォーマンス確保用のヒント
  - etc.
- QML開発時の参考に

http://doc.qt.io/qtcreator/creator-checking-code-syntax.html

バージョン管理システム(VCS)

- 様々なバージョン管理システムに対応
  - Bazaar
  - ClearCase
  - CVS
  - Git
  - Mercurial
  - Perforce
  - Subversion

http://doc.qt.io/qtcreator/creator-version-control.html

プロジェクトの設定共有

- プロジェクトの設定
  - 使用するキット, ビルドオプション, コードスタイル, etc
    - <project>.pro.user: 各ユーザー用(自動的に生成)
    - <project>.pro.shared: プロジェクト内で共有用
- .pro.sharedの作成方法
  - .pro.userをコピー
  - 不要な項目を削除
    - ProjectExplorer.Project.Updater.FileVersion は必要なので残す
  - 作成した .pro.shared をリポジトリに追加して共有

http://doc.qt.io/qtcreator/creator-sharing-project-settings.html

リファクタリングアクション

- コードの記述を補助
  - シンボル出現箇所の検索
  - シンボルの変更
  - メソッド定義の生成
  - 選択範囲から関数・エレメントを作成
  - Q_PROPERTY用メンバの作成
  - 仮想関数のオーバーライドの生成, etc.
- 右クリック→アクションの選択

http://doc.qt.io/qtcreator/creator-editor-refactoring.html

外部ツール

- Qt Creatorから外部のツールを実行
  - Linguist: lrelease, lupdate
  - Qt Quick: プレビュー(qmlviewer, qmlscene)
  - その他: vi, ソート
- ツールの追加などのカスタマイズも可能

http://doc.qt.io/qtcreator/creator-editor-external.html

機能拡張

- Qt Creatorはプラグインで拡張可能なアーキテクチャ
- コードモデルでリファクタリング系の実装も比較的容易
  - http://qt-labs.jp/2014/09/extending-qt-creator.html
- Qt Creator Pluginテンプレートも活用

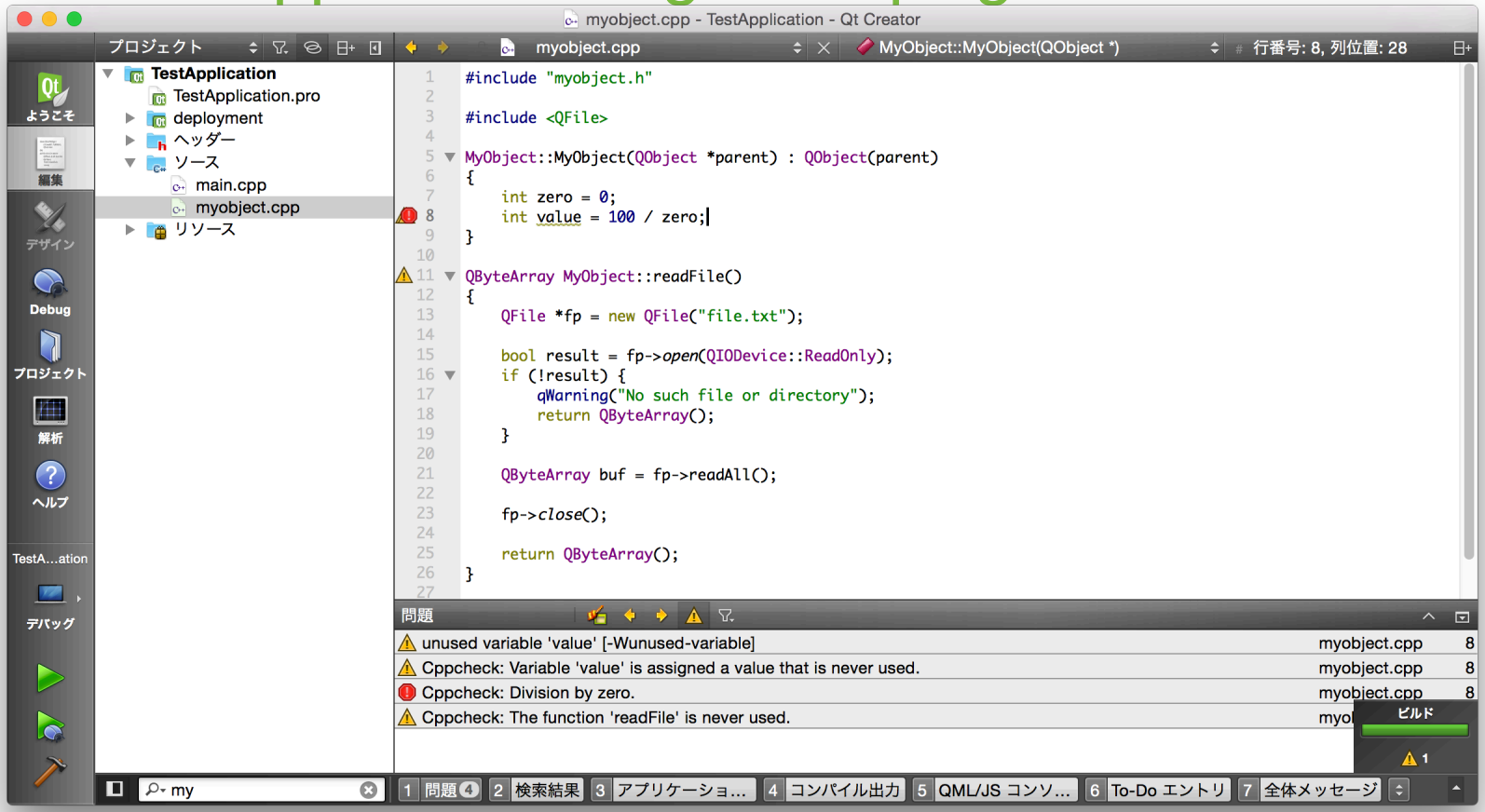
http://doc-snapshots.qt.io/qtcreator-extending/extending-index.html

Qt Creator CppCheck integration plugin

- Qt Creator 用 CppCheck プラグイン
  - サードパーティ製プラグイン
  - 別途インストールが必要
- CppCheck
  - オープンソース(GPL)のC/C++用静的解析ツール
  - http://cppcheck.sourceforge.net

http://sourceforge.net/projects/qtc-cppcheck/

Qt Creator CppCheck integration plugin



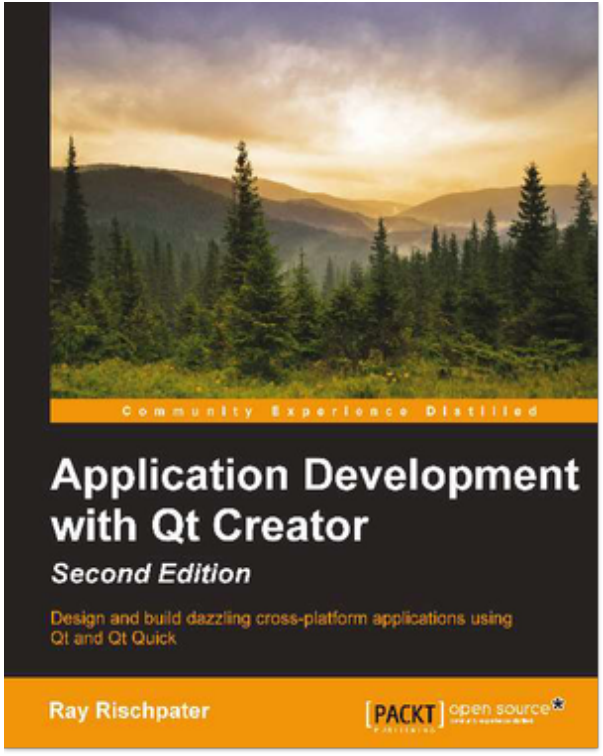
翻訳プロジェクト

- Qt Creatorの翻訳
  - コミュニティによるボランティア

https://www.transifex.com/projects/p/qt-creator/

Application Development with Qt Creator - Second Edition

- Qt入門書
  - Qt 5.3 & Qt Creator 3.1



https://www.packtpub.com/application-development/application-development-qt-creator-2nd-edition

Thank you

http://www.qt.io/qtjapansummit2015/