هفت گام اولیه در یادگیری Qt

۱۳۸۸/۵/۲۰

نویسنده : فرید احمدیان

با تشکر از مهندس مهرداد مومنی

گام اول

خیلیا سراغ شبکه در لینوکس میرن و خیلیای دیگه دنبال برنامه نویسی و ...
من جزو دسته دومم و خیلی دوست دارم برنامه ای که مینویسم تو همه ی سیستم عامل ها اجرا شه چون موعتقدم در نهایت در بهترین حالت سیستم عامل های تجاری در کنار سیستم عامل های اپن سورس در کنار هم به خوبی و خوشی زندگی خواهند کرد و کسانی برنده اند که برنامه هاشون رو هر دو پلتفرم اجرا شه! (البته این نظر منه)
خوب در مقام تحقیق میشه گفت چند تا گزینه خوب برای این کار پیدا کردم :

java
c
c++
Python

تمام زبانهای بالا زبانهای خوبی هستن که بسته به شرایط باید استفاده شن اما به نظر خودم زبانی که تا حد ممکن قوی باشه و ساده و در تمام پلتفرم ها یکسان باشه و کتابهای فارسی زیادی داشته باشه و استادشم گیر بیاد java بوده.
خود من مقداری باهاش کار کردم و شاید تنها بدیشو این بدونم که کاربر قبل از استفاده از برنامه جاوای شما JVM رو باید نصب کنه و این برای کاربران عادی جامعه ما یه نمه سخته!
البته چیزای دیگه مثل سرعت کمتر برنامه های جاوا نسبت به دیگران و ... هست که برا من مهم نبوده!
و اما شایددر گزینه بعدی زبان نام اشنای c و c++ باشه اما خوب برای ایجاد برنامه های اپلیکیشن اگر فقط از اینا بخواهیم استفاده کنیم بابامون در میاد چون برا هر چیزی باید کد بنویسیم برای همین ،قالب های کاری (framework) برای این زبان ها ساخته شدن که کارها رو خیلی ساده تر کردن ، در تحقیقاتم به سه موردشون بر خوردم:

wxWidgets
GTK
QT

و باز هم باید بر اساس شرایط و نیازتون یکی رو انتخاب کنید که با یه خورده گوگل کردن نام های بالا اطاعات زیادی رو راجبشون پیدا کنید و انتخابتون رو اگاهانه انجام بدین.
به هزارو یه دلیل شخصی من qt رو انتخاب کردم .

برا اینکه بیشتر با کیوت اشنا شیم قسمتی از متن موجود در ویکی www.pylearn.com رو در اینجا میارم که برای پیدا کردن اطلاعات بیشتر میتونبد به منبع اصلی مراجعه کنید:

منبع: http://www.pylearn.com/fa/wiki/index.php/Qt

QT یک فریم ورک مولتی پلتفرم برای توسعه نرم افزار می باشد که اکثرا برای ایجاد برنامه هایی با رابط کاربری (GUI) مورد استفاده قرار می گیرد . اما پس از نسخه۴ امکان ایجاد برنامه های متنی نیز فراهم شده است . بیشترین استفاده از کیوتی در رابط گرافیکیKDE بوده است که یکی از مهمترین محیط های گرافیکی لینوکس می باشد . نرم افزار های بسیاری چون,Opera Google Earth, Skype, Qtopia و ... نیز توسط این ابزار ایجاد گردیده اند . این ابزار توسط یک شرکت نروژی به نام Trolltech ایجاد گردیده و با سرعت بسیاری در حال توسعه می باشد .

زبان برنامه نویسی در Qt بصورت پیش فرض C++ می باشد . تقریبا این ابزار را می توان با

محیط VC++ مقایسه نمود . اما امکان برنامه نویسی با زبانهای دیگر چون پایتون ، رابی ، PHP ، پرل ، پاسکال و حتی C# و جاوا نیز در Qt فراهم می باشد ! همانطور که گفته شد کیوتی تقریبا در اکثر سیستم عامل های موجود چون لینوکس ، ویندوز ، مک و سیستم های خاصی چون PDA ها و Smartphone ها قابل اجراست .

Qt از موتوری درونی و خاص خود برای ایجاد اشیا و پنجره ها استفاده می کند . بنابراین امکان اجرا بر روی چندین سیستم عامل و نیز استفاده از اشایی پیشرفته براحتی ممکن می باشد . در عین حال کیوتی در هر سیستم عامل برنامه هایی درست همانند ظاهر همان سیستم عامل یا اصطلاحا محلی (native) تولید می کند .

کیوتی اسمی کلی هست و شامل تمام ابزار و کتابخانه ها و طراح می شود . در حقیقت محیط و فریم ورک Qt شامل قسمت های مختلفی می باشد . هسته اصلی و داخلی آن شامل کتابخانه هایی بسیار گسترده در اکثرا زمینه های موجود چون پایگاه داده ، شبکه ، سیستم فایل ، اینترنت و ... می باشد . همچنین کیوتی شامل طراحی (Designer) گرافیکی و بسیار قدرتمند و ساده می باشد که برنامه نویسی و ایجاد پنجره ها را بسیار آسان و سریع قابل پیاده سازی می کند .

خوب بعد از انتخاب زبان و فریم ورک مورد نظرم دنبال یه محیط توسعه خوب (IDE) بودم که باز هم به چند گزینه بر خوردم:

Qt Creator
kdevelop
Edyuk
onkey Studio
Qt Visual Studio Integration

و فعلا باز هم به دلایل شخصی از Qt Creator استفاده میکنم . به این دلیل میگم دلایل شخصی تا نشون بدم شاید بهترین گزینه نباشه ولی بنا به قانون نسبیت هیچ چیزی مطلق نیست و بنا به شرایط شاید چیزی که در مواقع عادی بد باشه در شرایط شما خوب باشه! بحث فلسفی شد بماند!
اما بنا به توصیه بزرگان کیوت کار ایرانی بهتر اول ادم بدون محیط توسعه کد بزنه و کامپایل کنه تا روال دستش بیاد و خوب ما هم یه خورده حرفشون رو گوش کردیم و رفتیم در لینوکسما و کا ر را شروع کردیم و پس از مقداری اشتباه با کمک بزرگان (دستشون درد نکنه) روال دستمون اومد برای مثال بیایید یه برنامه ساده با کیوت بنویسیم:
تو یه ادیتور کدهای زیر رو تایپ کنید:

```
#include <QApplication>
#include <Qt/qwidget.h>
int main(int argc, char *argv[])
{
	QApplication app(argc, argv);
	QWidget window;
	window.resize(250, 150);
	window.setWindowTitle("Simple example");
	window.show();
```

```
    return app.exec();
}
```

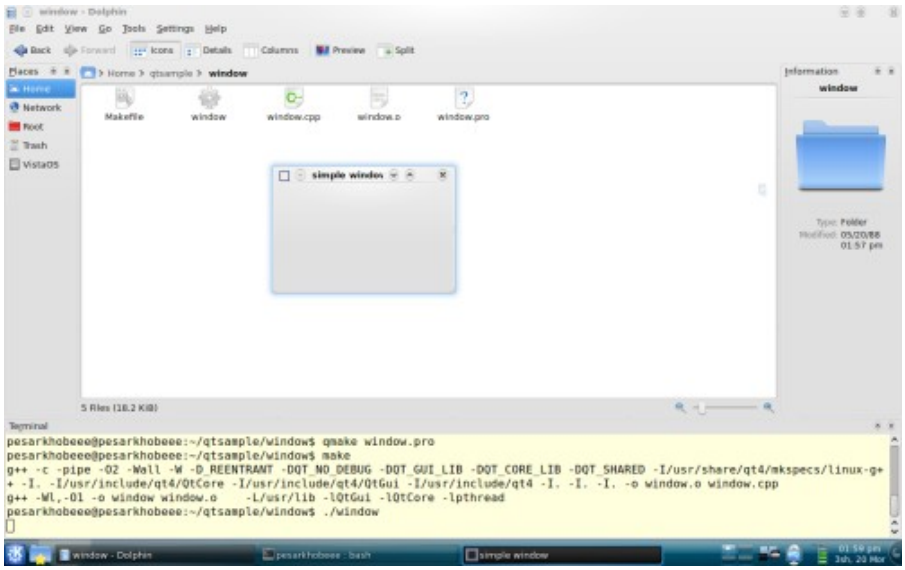
و مثلا با نام window.cpp ذخیره کنید و حالا برای اجرا باید مراحل زیر رو در کنسول لینوکستون در وشه ای که فایل بالا رو ذخیره کردین یکی یکی برین:

```
qmake -project
qmake window.pro
make
```

خوب نتیجه مراحل بالا اینه که اگه در کدهاتون مشکلی نباشه برنامتون کامپایل میشه و فایل اجراییش اماده میشه و حالا برای اجرا در همون جا دستور زیر روبزنید:

```
./window
```

و از اولین برنامتون لذت ببرید! من سعی خواهم کرد تمام تجربیاتمو به صورت مستند در بیارم تا نفرات بعد از من شاید راحتتر باشن



گام دوم

خوب در این قسمت میخوام نمونه کدی رو که در قسمت قبل کامپایل و اجرا کردیم بیشتر تشریح کنم:

کدمون اینطوری بود:

```
#include <QApplication>
#include <Qt/qwidget.h>
int main(int argc, char *argv[])
{
	QApplication app(argc, argv);
	QWidget window;
	window.resize(250, 150);
	window.setWindowTitle("Simple example");
	window.show();
	return app.exec();
}
```

#include <QApplication>

#include <Qt/qwidget.h>

این دو قطعه کد کتابخونه های ضروری برای اجرای برناممون رو در کدمون درج میکنه

QApplication app(argc, argv);

این قطعه کد در تمام برنامه های QT البته بجز کنسولیاش باید باشه و شیع کلی برنامه رو مشخص میکنه

QWidget window;

این قطعه کد ابزارک(Widget) اصلی برناممون رو به اسم window تعریف میکنه.

window.resize(250, 150);

window.setWindowTitle(“Simple example”);

window.show();

کدهای بالا اول ابزارکمون رو ریساز میکنه و عرض و ارتفاشو تنظیم میکنه بعد عنوان ابزارکمون که در اینجا همون پنجره اصلیمون رو تنظیم میکنه و بعد اونو نمایش میده

return app.exec();

و در اخر توسط این کد حلقه اصلی برنامه شروع میشه.دوستانی که برنامه های c حساس به رویداد نوشتن میدونن که برای اینکه برنامه مدام رخ دادن رویدادها رو چک کنه باید در یک حلقه بی نهایت قرار بدیمش ، من فکر میکنم قطعه کد بالا یه همچین روالی رو شروع میکنه!

اینها چیزایی بود که من از این کدها پیدا کردم و شاید توضیحات خیلی بیشتری داشته باشن!

در اخرین مورد میخوام راجع به چیستیqmake که باهاش برناممون رو کامپایل کردیم بحث کنم:

qmake ابزاری که شرکتTrolltech برای اسان تر کردن روند تولید برنامه در همه پلتفرم ها ایجاد کرده!

qmake عمل ساختن makefile برای پروژتون رو اتوماتیک کرده انچنان که فقط با چندخط اطلاعات میتونه makefile بسازه!

برای کسانی که makefile روند کلی کامپایل رو نمیدونن متن زیر رو میارم:

کامپایلر

کامپایلرها وظیفه تبدیل زبانهای سطح بالا و قابل درک برای انسان را به سطح پایین ترین زبان، یعنی زبان ماشین به عهدا دارند.

معمولا (تقریبا همیشه!) پروژه های نرم افزاری از بیش از یک فایل تشکیل شده اند که هر کدام بخشی از متن برنامه یا Source Code را در بر دارد. شما باید ابتدا هر فایل نوشته شده را جداگانه به وسیله کامپایلر به زبان ماشین تبدیل کنید. اصطلاحا به فایلهایی که در این مرحله تولید می شود، فایل Object می گویند. محتوای لین فایلها به زبان ماشین است، اما هنوز قابل اجرا توسط کامپیوتر نیستند! چرا؟ چون هر کدام از آن فایلها یا وابستگیهایی در فایلهای دیگر دارند که در این وضعیت برای سیستم غیر قابل تشخیص است، یا اساسا به تنهایی مفهومی ندارند که قابل اجرا باشند.مثلا ممکن است یکی از آنها حاوی مجموعه ای از دستورات (اصطلاحا تابع) باشد که روی ورودی خود پردازش خاصی انجام می دهند و نتیجه را برمی گردانند. اما این توابع زمانی مفید هستند که از جایی فراخوانی شوند و احتمالا فراخوانی تابع مورد نظر ما هم از داخل یکی از فایلهای Object دیگر صورت می گیرد.

در اینجاست که شما باید با کمک لینکر آن فایلهای Object را به ترتیب مناسب به هم پیوند دهید تا در نهایت به یک فایل اجرایی دست پیدا کنید.

یک مثال عملی

فرض کنید برنامه ای به زبانC در یک فایل به نام main.c نوشته اید و حالا می خواهید آن را به فایل قابل اجرا توسط کامپیوتر تبدیل کنید. این کار به سادگی و در دو خط دستور (که می شود آنها را در یک خط هم خلاصه کرد) قابل اجرا است:

gcc -c main.c $

gcc -o main main.o $

دستور اول از فایل C به نام main.c یک فایل Object به نام main.o می سازد و دستور دوم نیز از فایل main.o یک فایل اجرایی به نام main می سازد که حالا دیگر می توانید آن را اجرا کنید.

اما حالا فرض کنید که فایل main.c وابستگیهایی هم به دو فایل دیگر به نامهای dep.c و dep.h داشته باشد:

$ gcc -c main.c

$ gcc -c dep.c

$ gcc -o main main.o dep.o

می بینید که یک مرحله به مراحل کامپایل برنامه اضافه شد

حالا فرض کنید که به جای یک فایل یا سه فایل، این پروژه از دهها و صدها فایل تشکیل شده باشد. چند ساعت باید وقت صرف کنید تا پروژه را کامپایل کنید، تازه اگر همه چیز به خوبی پیش برود؟

به همه اینها اضافه کنید گزینه های خط دستور کامپایلر را که گاهی کار کامپایل از خط دستور را به فرایندی بسیار پیچیده تبدیل می کنند.

حالا هربار که تغییراتی دربرنامه بدهید، باید همه کارها را از اول انجام دهید یا به اندازه کافی فصفر بسوزانید و به خاطر داشته باشد که هر بار در کدام فایلها تغییر داده اید تا فقط آنها را کامپایل کنید و کمی وقت کمتری صرف کنید! آیا راه بهتری نیست؟

Makefile

چقدر خوب می شد اگر می توانستید همه این دستورات کامپایل را فقط یک بار تایپ کنید. خبر خوب! می توانید همه این دستورات را در فایل که به Makefile معروف است قرار دهید و برنامه ای به نام make را اجرا کنید تا آن برنامه خودش از روی Makefile برایتان بقیه کارها را انجام دهد و خروجی نهایی قابل اجرا را بسازد

پس Makefile مجموعه دستورات لازم برای کامپایل یک پروژه نرم افزاری است که در قالبی خاص در یک فایل متنی ذخیره شده است.

به خصوص اگر شما برنامه نویس نباشید، برایتان قابل توجه خواهد بود که مجبور نیستید هربار اینهمه دستورات عجیب و غریب را تایپ کنید و تنها با تایپ دستور make زندگی برایتان آسان می شود.

make

همانطور که اشاره شد، make برنامه ای است که از روی Makefile کار کامپایل پروژه را انجام می دهد. در کل دو مزیت عمده می توان برای make در نظر گرفت:

۱/همانطور که قبلا هم گفتم با استفاده از Makefile کارهای سخت (نوشتن دستورات پیچیده کامپایل) فقط یکبار انجام می شود.

۲/make از روی برچسب زمانی فایلهای سورس وObject تشخیص می دهد که کدامیک از فایلها نیاز به دوباره کامپایل شدن دارند و به این ترتیب از کامپایل مجدد و غیر ضروری فایلهایی که از زمان آخرین کامپایل تغییری نکرده اند، جلوگیری می شود چراکه فرایندکامپایل به ویژه در کامپیوترهای قدیمی و با پروژه های بزرگ کاری بسیار وقت گیر است

make بعد از فراخوانی به ترتیب دنبال یک فایل با یکی از نامهای makefile، GNUmakefile یا Makefile می گردد تا دستورات کامپایل را از داخل آن خوانده و اجرا کند

برگرفته از سایت:shabaneh.ir

از qmake میتونیم در اکثر پروژهای نرم افزاری استفاده کنیم و لزوما نباید باQt باشن.

برای بدسا اوردن اطلاعات بیشتر راجع بهqmake به اینجا مراجعه کنید.

گام سوم

خوب امروز میخوام برنامه قبلی رو کمی کاملتر کنم و مقداری از خصوصیاتشو تنظیم کنم!
برنامه قبلی رو به این صورت بازنویسی میکنیم

```
#include <QApplication>
#include <QDesktopWidget>
#include <QWidget>
#include <QIcon>
int main(int argc, char *argv[])
{
int WIDTH = 250;
int HEIGHT = 150;
int screenWidth;
int screenHeight;
int x, y;
QApplication app(argc, argv);
QWidget window;
QDesktopWidget *desktop =
QApplication::desktop();
screenWidth = desktop->width();
screenHeight = desktop->height();
x = (screenWidth - WIDTH) / 2;
y = (screenHeight - HEIGHT) / 2;
window.resize(WIDTH, HEIGHT);
window.move( x, y );
window.setWindowTitle("Center");
window.setToolTip("Center window");
window.setWindowIcon(QIcon("icon.jpg"));
window.show();
return app.exec();
}
```

#include <QdesktopWidget>

توسط این کتابخانه ما به مقادیر خصوصیات دسکتاپمان دسترسی پیدا میکنیم! در این مثال برای فهمیدم طول و عرض مانیتور!

```
#include<QIcon>
```

برای اینکه بتوانیم به برنامه خودمون یک icon بدیم باید این کتابخونه رو استفاده کنیم.

```
Int WIDTH = 250;
int HEIGHT = 150;
int screenWidth;
int screenHeight;
int x, y;
```

دو متغیر اول طول عرض پنجره برنامه مان را ذخیره میکنند و دو متغیر دوم برای اینست که طول و عرض دسکتاپمان را ذخیره کنند و دو متغیر x و y برای ذخیره محل قرار گیری پنجره است

```
QDesktopWidget *desktop =QApplication::desktop();
screenWidth = desktop->width();
screenHeight = desktop->height();
```

کار این کدها بدست اوردن طول و عرض دسکتاپ است ولی دقیقا نفهمیدم چه طوری ؟ اومده یه اشاره گر به نام desktop ساخته و بعد توسط شی ء گرایی فکر کنم ارجاعش داده به دسکتاپ واقعی (فکرکنم)

```
x = (screenWidth – WIDTH) / 2;
y = (screenHeight – HEIGHT)/ 2;
```

این خط ها مختصات نقطه ای رو که پنجره برنامه در انجا قرار گیرد حساب میکند

```
window.move( x, y );
```

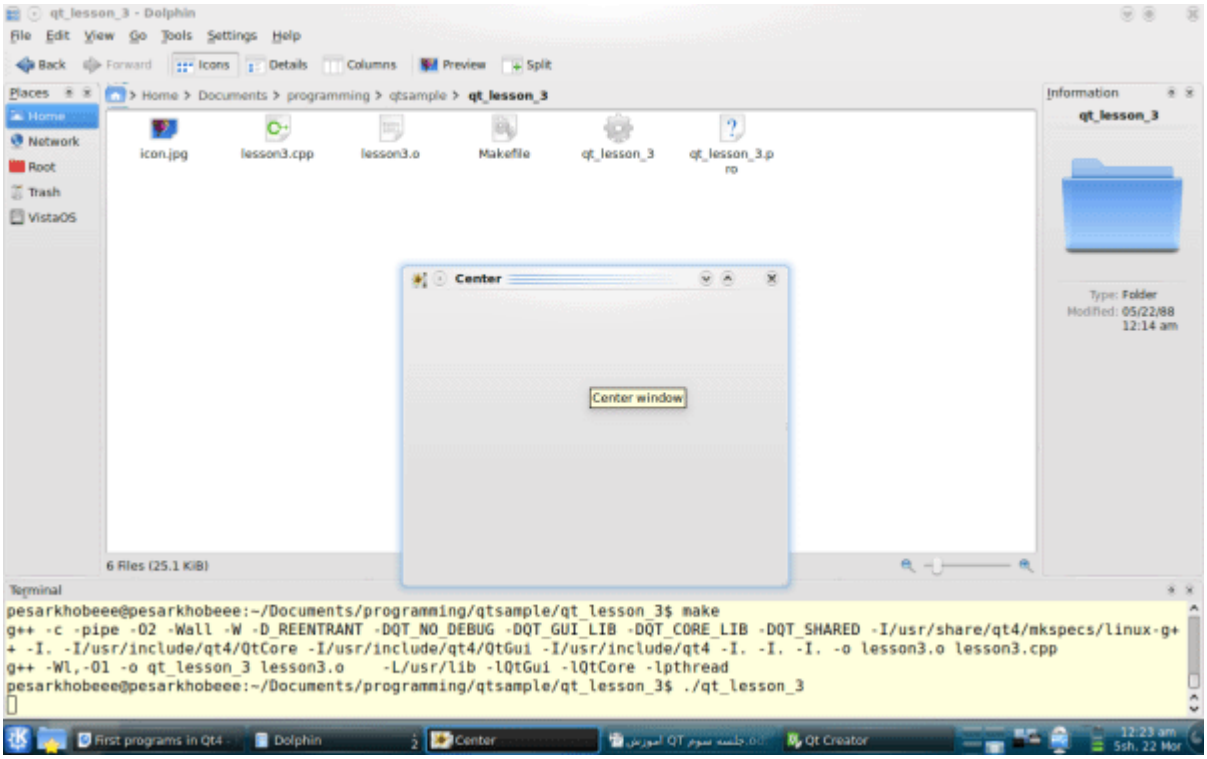
این کد پنجره برنامه را به مختصات مورد نظر هدایت میکند

```
window.setToolTip(“Centerwindow”);
```

عمل تنظیم tooltip پجره را انجام میدهند

```
window.setWindowIcon(QIcon(“icon.jpg”));
```

در این کد ما عکس icon.jpg را که در همان پوشه برنامه است به عنوان ایکون برنامه تعیین میکنیم.
خوب
در این قسمت ما تونستیم پنجره ای رو در قسمت قبل ساختیم بیشتر شخصی سازی کنیم!
علی یارتون


qt_lesson_3 - Dolphin
File Edit View Go Tools Settings Help
Back Forward Icons Details Columns Preview Split
Places
Home
Network
Root
Trash
VistaOS
Home > Documents > programming > qtsample > qt_lesson_3
icon.jpg
lesson3.cpp
lesson3.o
Makefile
qt_lesson_3
qt_lesson_3.pro
Information
qt_lesson_3
Type: Folder
Modified: 05/22/88 12:14 am
Center
Center window
6 Files (25.1 KiB)
Terminal
pesarkhobeee@pesarkhobeee:~/Documents/programming/qtsample/qt_lesson_3$ make
g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -DQT_SHARED -I/usr/share/qt4/mkspecs/linux-g++ -I. -I/usr/include/qt4/QtCore -I/usr/include/qt4/QtGui -I/usr/include/qt4 -I. -I. -I. -o lesson3.o lesson3.cpp
g++ -Wl,-O1 -o qt_lesson_3 lesson3.o -L/usr/lib -lQtGui -lQtCore -lpthread
pesarkhobeee@pesarkhobeee:~/Documents/programming/qtsample/qt_lesson_3$ ./qt_lesson_3
First programs in Qt4
Dolphin
Center
Qt Creator
12:23 am
Ssh. 22 Mor

گام چهارم

خوب به لطف خدا پس از برخورد با چند تا مشکل وحل اونها قسمت چهارم اموزش هم اماده شد.
توجه : قبل از خوندن مقاله زیر به این نکته اشاره میکنم که این نوشته ها برداشت های شخصی خودم هست و ممکن هست از نظر فنی توضیحاتم درست نباشه!
در این قسمت ما برناممون رو که تا قسمت قبل پنجرشو ساختیم و تنظیمش کردیم کاملتر میکنیم و بهش یه سری کنترل اضافه میکنیم و بعد اون کنترل ها رو در لایه میزایم تا نظمشون حفظ شه!

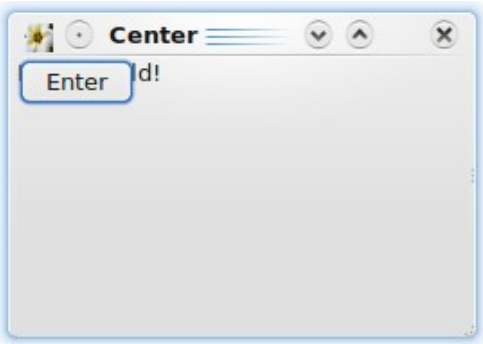
اما سوال اول اینه که کنترل چیه؟
کنترل ها اجزای قابل تعریف برای استفاده در برنامه های گرافیکی هستند به تعبیر ساده تر اجزایی که ما در برنامه هامون میبینیم مثل دکمه ها و منو ها و ... کنترل محسوب میشه!
یکی از خوبی های کیوت هم داشتن تعداد وسیعی از کنترل هاست که به راحتی با ترکیب این کنترل ها میتونیم برناممون رو بسازیم .
نکته : در دنیای کیوت به کنترل ، widget (ابزارک) میگن!
ما در این برنامه از دو نوع کنترل یکی برچسب یا Label و دومی دکمه یا Button استفاده میکنیم.
خوب مثل هر چیز دیگه ای برای استفاده از اینها باید اول کتابخانه مربوطشون رو در برنامه درج کنیم.

```
#include <QLabel>
#include <QPushButton>
```

و اما برای تعریفشون در برنامه از خط های زیر استفاده میکنیم:

```
QLabel *label = new QLabel (“Hello World!”,&window);
QPushButton *button = new QPushButton(“Enter”, &window);
```

خوب در خط اول ما کنتلری به نام label را از روی کلاس Qlabel میسازیم و مقداری رو که قراره نمایش بده رو Hello World تعریف میکنیم.
در خط دوم هم یک دکمه از روی کلاس QpushButton میسازیم و مقداری که نمایش میده رو تنظیم میکنیم.
در تعریف هر دوی این کنترل ها ما دو ارگومان دادیم که در دومین ارگومان &window ما مشخص میکنیم والد این کنترل کدام پنجرس؟
اگر این را مشخص نکنیم کنترل های ما در پنجرمان نشان داده نمیشوند.
خوب حالا بیایید برنامه قبلی رو با اضافه کردن دستورات فوق تکمیل کنیم.



خوب همونطور که میبینید کنترل های ما با موفقیت ایجاد شدن ولی روی هم افتادن!
برای مدیریت کنترل ها در صفحه پنجرمان کیوت کنترلی در اختیار ما گزاشته به نام Qlayout که میتونید اطلاعات کاملی از این کنترل در این صفحه بدست بیاورید.

اما اگه بخوام توضیح بدم Qlayout یک راه ساده و قوی را برای مرتب کردن خودکار کنترل ها فراهم میکنه!
Qlayout به صورت پیش فرض به سه صورت میتونه کنترل ها رو کنار هم مرتب کنه به صورت عمودی و یا افقی و یا ترکیبی از هر دو که در اصلاح شبکه ای میگن در زیر شکلهایی از هر سه روش میارم:

لایه افقی که توسط کلاس QHBoxLayout تغریف میشود:

لایه عمودی که توسط کلاس QVBoxLayout تغریف میشود:

لایه شبکه ای که توسط کلاس QGridLayout تغریف میشود:

خوب حالا ما این قصد رو داریم تا برای مدیریت کنترلها از لایه افقی استفاده کنیم برای این کار اول باید کتابخونه مورد نیازو در برنامه درج کنیم:

```
#include <QLayout>
```

و بعد برای تعریف لایه افقی و ستون های لایه افقی و قرار دادن کنترلها در این ستون ها از دستورات زیر استفاده میکنیم:

```
QHBoxLayout *layout = new QHBoxLayout;
```

```
layout->addWidget(label);
layout->addWidget(button);
window.setLayout(layout);
```

خوب ما در خط اول یک اشاره گر از نوع QHBoxLayout تعریف کردیم و بعد در خط دوم گفتیم در ستون اول اون کنترل label مون رو قرار بده و در خط بعدی گفتیم که در ستون دوم button رو قرار بده و سپس این لایه ساخته شده رو در خط چهارم به پنجرمون اعمال کردیم که نتیجش :

رو مشاهده میکنیم !
خوب اینم از کل کدهای برنامه ما کی طی این چهار جلسه تکامل پیدا کرده:

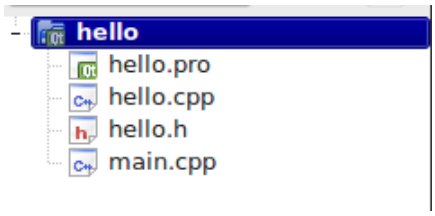
```
#include <QApplication>
#include <QDesktopWidget>
#include <QWidget>
#include <QIcon>
#include <QLabel>
#include <QPushButton>
#include <QLayout>
int main(int argc, char *argv[])
{
int WIDTH = 250;
int HEIGHT = 150;
int screenWidth;
int screenHeight;
int x, y;
QApplication app(argc, argv);
QWidget window;
QDesktopWidget *desktop = QApplication::desktop();
```

```
screenWidth = desktop->width();
screenHeight = desktop->height();
x=(screenWidth-WIDTH) / 2;
y=(screenHeight-HEIGHT) / 2;
window.resize(WIDTH, HEIGHT);
window.move( x, y );
window.setWindowTitle("Center");
window.setToolTip("Center window");
window.setWindowIcon(QIcon("icon.jpg"));
QLabel *label = new QLabel ("Hello World!",&window);
QPushButton *button = new QPushButton("Enter", &window);
QHBoxLayout *layout = new QHBoxLayout;
layout->addWidget(label);
layout->addWidget(button);
window.setLayout(layout);
window.show();
return app.exec();
}
```

و اما برای تمرین پشنهاد میکنم با کمک این سایت:
/http://zetcode.com/tutorials/qt4tutorial/widgets
کنترل های مختلف کیوت رو به برنامتون اضافه کنیدو لذت ببرید و در ادامه روشهای مختلف layout رو هم امتحان کنید.

گام پنجم

برای اینکه بتونیم برنامومون رو که در مراحل پیش نوشتیم کاملتر کنیم باید اون رو به استانداردها نزدیک تر کنیم به معنای دقیق تر واقعا از قدرت زبانمون استفاده کنیم و برنامومون رو به صورت کلاس بندی شده بنویسیم و برای مدیریت بهتر اون رو در چند فایل تقسیم کنیم .
اونایی که با Qt Creator کار کردن دیدن اون به صورت خود کار پروژمون رو در چند فایل تقسیم میکنه و به صورت شی گرایی کد میزنه!



ما هم به صورت دستی میخواهیم همون کارو کنیم تا با اصول اشنا شیم (از بدی های با IDE کار کردن همینه تا مدتها نمیدونید تو چیا ضعف دارین و ساختار کلی یه برنامه واقعا چه طوریه) خوب فایلهای برنامومون به این شرح هستند:

۱ . **main.cpp**
فایل اصلی ما محسوب میشه که نقطه اغازین برنامه ما محسوب میشه و فقط تعاریف اصلی برنامه درش هست!

۲. **hello.cpp**
کلاس اصلی ما درون این فایل نوشته میشه و در اصل کار اصلی رو انجام میده!

۳. **hello.h**
فایل header یا در فارسی سر آیند برنامومون که توابع و کلاسهای ما در اون تعریف میشه!

۴. **hello.pro**
فایلی که به کامپایرمون میگه تحت چه ترتیبی باید برنامومون روmake کنه در قسمتهای قبلی یه توضیح کامل در موردش نوشتم!

خوب حالا که با ساختار اولیه برنامومون اشنا شدیم ببینیم هر کدوم از این فایلها چه کدهایی شامل میشوند.

فایل **main.cpp**

```
#include <QtGui>
#include “hello.h”

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

hello *Hello = new hello;
Hello->setInterface();
Hello->show();

return app.exec();
}
```

خوب اولین کد :

#include <QtGui>

به جای تمام include هایی که در قسمت های قبل داشتیم و باعث جلوگیری از سردرگمی در میان انبوه include میشه.

#include “hello.h”

فایل سرایند ما رو include میکنه در اینجا باید به تفاوت <> و ”” در دستور include اشاره کنم مورد اول میگه کامپایلر برو از مکان پیشفرض کتابخونه های خودت کتابخونه مورد نظرمو پیدا کن اما دومی میگه فایل مورد نظرم تو پوشه پروژس!

hello *Hello = new hello;

در اینجا اشاره گری (اشاره گر متغیری است که در ان ادرس یک خانه حافظه مینشیند) به نام Hello از روی کلاس hello مان که در فایل سر ایندمان تعریف و در فایل hello.cpp تشریح کردیم ! میسازیم.

Hello->setInterface();
Hello->show();

وقتی به جای حالت معمولی ما توسط اشاره گر یک نمونه از کلاسمون میسازیم برای دسترسی به توابع و متغیر ها به جای نقطه . باید از <- استفاده کنیم!
در خط اول تابعی که خودم در کلاس hello نوشتم رو صدا کردم که بعدا توضیح میدم چیه! و اما در خط دوم تابع show کلاس hello خودمو صدام کردم اما اگه کلاسمو نگاه کنید چنین تابعی رو پیدا نخواهید کرد اما چون کلاس hello از Qwidget تمام ویژگی هاشو ارث برده میتونیم از توابع اون هم استفاده کنیم.

خوب بریم سراغ فایل بعدی:

hello.h

```
#ifndef hello_h
#define hello_h
#include <Qt/qwidget.h>


class QLabel;
class QPushButton;
class hello : public QWidget
{
public:
hello(QWidget *parent = 0);
void setInterface();
private:
QLabel *label;
QPushButton *button;
};
#endif
```

خوب در فایل سر ایندمان مهمترین کاری که کردیم تعریف کلاس hello با کلمه کلیدی class و سپس تعیین ارث بری ان از کلاس Qwidget توسط : بود.
بعد از تعریف توابع و اشیاء عمومی و خصوصی اون رو مشخص کردیم!تنها چیزی در این تعاریف لازم میدونم توضیح بدم ارگومان تابع زیر هست:

hello(QWidget *parent = 0);

در اینجا به عنوان ارگومان کلاسمون اشاره گرparent هست که به صورت پیش فرض مقدارش صفر هست و این ارگومان تعیین میکنه که پنجره ای که کلاسمون میسازه پنجره اصلیه که با صفر مشخص میشه یا فرزند یه پنجره دیگس (اعداد دیگه به عنوان ورودی)!

اما شاید سوال براتون پیش امده باشه این Qwidget که اینهمه ازش استفاده کردیم اصلا چی هست؟
کلاس Qwidget کلاس پایه برای تمام اشیای رابط کاربریمونه!برای اشنایی بیشتر با این کلاس و همچنین توابع و سایر خصوصیاتش به لینک زیر مراجعه بفرمایید:

http://doc.trolltech.com/4.5/qwidget.html

خوب مغز اصلی سراینمونو فهمیدیم ولی اول فایل امده:

#ifndef hello_h

تا اونجایی که من با سواد ناقصم فهمیدم این شرط چک میکنه که ایا برای بار اوله که سرایندمون در برنامه استفاده میشه یا نه اگه برای بار اول باشه هر چیزی بین این

ifndef = if not define

بود اجرا میشه وگر نه که از تکرار جلوگیری میکنه برای اشنایی بیشتر به لینک زیر مراجعه کنید:

http://www.cprogramming.com/reference/preprocessor/ifndef.html

حالا موقشه بریم سراغ فایل بعدیمون:

hello.cpp

```
#include <QtGui>
#include "hello.h"

hello::hello(QWidget *parent)
:QWidget(parent)
{
label = new QLabel ("Hello World!");
button = new QPushButton("Enter");
QHBoxLayout *layout = new QHBoxLayout;
layout->addWidget(label);
layout->addWidget(button);
this->setLayout(layout);
}
void hello::setInterface()
{
int WIDTH = 250;
int HEIGHT = 150;
int screenWidth;
int screenHeight;
int x, y;
QDesktopWidget *desktop = QApplication::desktop();
screenWidth = desktop->width();
screenHeight = desktop->height();
x = (screenWidth - WIDTH) / 2;
y = (screenHeight - HEIGHT) / 2;
this->resize(WIDTH, HEIGHT);
this->move( x, y );
this->setWindowTitle("Center");
```

```
this->setToolTip("Center window");
this->setWindowIcon(QIcon("icon.jpg"));
}
```

اکثر کدهایی که در اینجا میبینم رو در قسمت های قبل نوشتیم ولی در اینجا اونها رو در یک کلاس مشخص قرار دادیم!
در اینجا برای کلاس hello دو تابع رو تشریح میکنیم یکی تابع hello است که چون هم نام با کلاسمون هست تابع سازنده مان محسوب میشه یعنی اینکه وقتی یک شی ء از روی کلاس میسازیم به صورت خود کار اجرا میشه و احتیاجی به صدا زدنش نیس!
اما تابع دوممون setInterface نام داره و اگه یادتون باشه در فایل main.cpp صداش کردم و اونجا گفتم در موردش بیشتر توضیح میدم ! خوب توضیحش خیلی سادس من تمام کدهایی که پنجره برناممون رو تنظیم میگرد در این تابع کلاسمون قرار دادم که با فر اخوانیش تمام این تغییرات بر روی پنجره مون اعمال میشه تنها تفاوت این کدها با قبلیا اینه که بجایwindow اینجا از this استفاده کردم اونم بخاطر اینه که کلاسمون خودش از کلاس Qwidget مشتق شده!
خوب اخرین چیزی که در مورد این فایل میخوام توضیح بدم :: چی هست ؟
برای تعریف واقعی تابع عضو باید اعلام کرد که این تابع متعلق به کدام کلاس است . برای این کار نام کلاس مربوطه باید قبل از نام تابع به همراه اپراتور :: (scope resolution operator) عملگر تعیین میدان دید) اورده شود!

: و اخرین فایل پروژمون
hello.pro

```
SOURCES = hello.cpp \
main.cpp
HEADERS = hello.h
```

خوب با توجه به اینکه قبلا هم در مورد این فایل اشاره کرده بودم و همونطور که میبینید ساختار بسیار ساده و تابلویی داره احتیاج نمیبینم توضیح بدم.
برای کامپایل پروژه هم از همون دستورات قبلی استفاده میکنیم با این تفاوت که حالا که دستی فایل پروژمون (hello.pro) را ساختیم احتیاج نیست دستور qmake -project رو بزنیم و فقط کافیه از دو دستور زیر استفاده کنیم:

```
qmake hello.pro
make
```

و برای اجراش هم :

```
./hello
```

گام ششم

خوب بلاخره به قسمت پایانی گام های اولیه رسیدیم!
در این گام از این سری اموزش ها ما برنامه ای رو که در پنج گام قبل درست کردیم رو تکمیل میکنیم و کاری میکنیم تا وقتی کاربر روی دکمه برناممون کلیک کرد متن برچسبمون عوض شه و با پایان این قسمت ما در مجموع این شش گام با کلیت یک برنامه ساده کیوت آشنا شدیم .
و از این به بعد نمونه های اموزشی کیوت رو خیلی راحتتر میفهمیم و اگه کمی پایه باشید! میتونید با کمی تلاش برنامه های ساده ای بنویسید.
خوب یکی از ویژگی هایی که کیوت رو نسبت به بقیه متفاوت کرده راهکار اون برای تعامل با رخدادهایی که در برنامه رخ میده!
در کیوت مکانیزمی تعبیه شده به اسم Signal & Slot که فلسفه این مکانیزم اینطوری که در برنامه ما که طبیعتا رویدادگراست وقتی که برای یک کنترل یا همونWidget یه شرایط خاص مثلا کلیک شدن پیش میاد اون کنترل از خودش یک نشونه مخصوص این رخداد رو آزاد میکنه که اصطلاحا Signal نامیده میشه ! خوب ما اگه بخواهیم برای این حالت از کنترلمون کدهایی رو بنویسیم باید این کدها رو در یک متد از کلاسمون قرار بدیم که اصطلاحا بهشSlot میگن و تنها چیزی که میمونه اینه که برنامه چطوری باید تشخیص بده وقتی در یک کنترل خاص رویداد خاصی رخ داد اون رو به کدوم Slot ارجا بده که کیوت این کار رو توسط تابع connect اجام میده!

شاید نتونسته باشم مطلبو خوب برسونم اما وقتی مثالو ببینید حتما متوجه میشید!
خوب اول از همه میریم سراغ فایل سرایندمون که همه چیز کلاسمون در اون تعریف میشه:

hello.h

به این فایل دو خط زیرو اضافه میکنیم:

```
private slots:
void showHello();
```

خوب همون طور که میبینید در کیوت نسبت به سی پلاس پلاس معمولی چیزهایی اضافه داریم که در اینجا public slots و private slots هست !
کاملا واضحه که داریم اول مشخص میکنیم که اسلات مورد نظرمونpublic یا private و بعد اونو تعریف اولیه میکنیم.
و اما در جایی که میخواهیم از مکانیزم سیگنال اسلات و یه سری چیز دیگه استفاده کنیم باید یه کد خاص رو به اون کلاس که فکر کنم شبیه یه ماکرو باشه رو اضافه کنیم!
پس کد Q_OBJECT رو به تعریف کلاسمون به این صورت اضافه میکنیم:

```
class hello : public QWidget
{
Q_OBJECT
```

خوب حالا که تعریف اولیه کردیم میریم سراغ فایل اصلی کلاسمون یعنیhello.cpp :
یک تابع جدید به کلاسمون اضافه میکنیم که همون اسلات ماست:

```
void hello::showHello()
{
QString str = QString::fromUtf8("افتتاح سایت زنجان لاگ مبارک باد<br>
www.zanjanlug.org");
```

```
label->setText(str);
}
```

خوب در اینجا ما متغیری به نام str از نوع Qstring ساختیم و گفتیم تحت یونیکد utf8 باشه تا با فارسی مشکل نداشته باشیم و سپس متن برچسبمون رو این متغیر قرار دادیم.
تنها کاری که میمونه اینه که توسط تابع connect مبدا و مقصد و نوع سیگنال رو مشخص کنیم پس در تابع hello کلاسمون این تکه کد رو اضافه میکنیم:

```
connect(button, SIGNAL(clicked()),this,SLOT(showHello()));
```

خوب همون طور که میبینید مثل باقلوا داره منظور ما رو مشخص میکنه!!! در ارگومان اول این تابع کنترل مبدا ما مشخص میشه و در ارگومان دوم ما نوع سیگنال مورد نظر که در اینجا کلیک شدن هست مشخص میشه و در ارگومان سوم مقصد ما و در ارگومان چهارم ما نام اسلات ما مشخص میشه.

خوب اگر تغییراتو به درستی اعمال کنید و برنامه رو یک بار کامپایل و اجرا کنید و روی دکمه enter بزنید باید با صحنه زیر روبرو بشید:

در اخر کدهایی که در مجموع این شش قسمت تکمیل شدن رو ضمینه میکنم و خیلی راحت میتونید با زدن دستور:

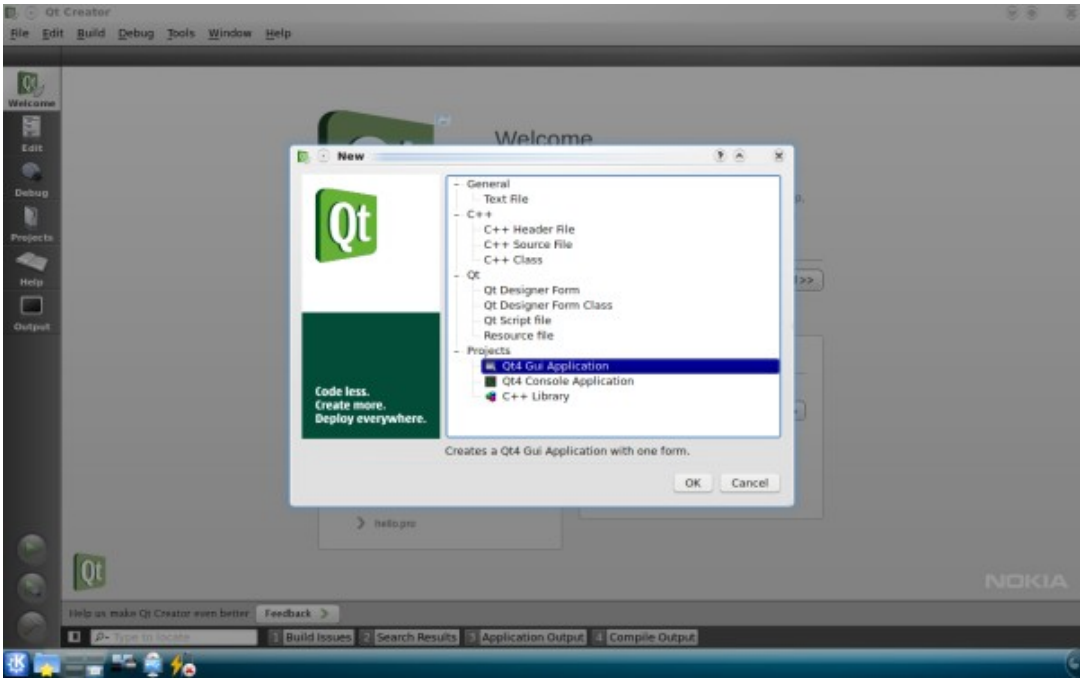
```
./hello
```

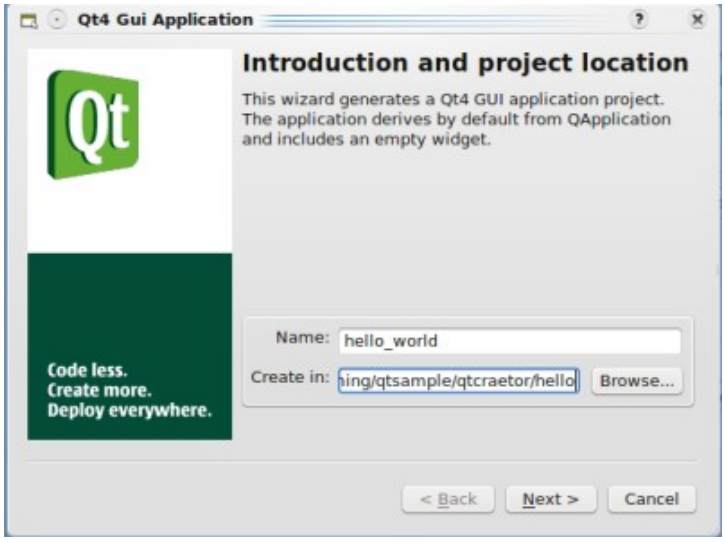
این کدهارو اجرا کنید و برنامه رو ببینید .
برای دریافت برنامه کافیه نسخه zip برنامه رو از سایت زنجان لاگ دانلود کنید.
به امید اینکه این سری از اموزش ها نظرتونو جلب کرده باشه امیدوارم به بزرگی خودتون خطاهای من رو در نوشتن تجربیات خودم در یادگیری این قالب قدرتمند برنامه نویسی ببخشید.

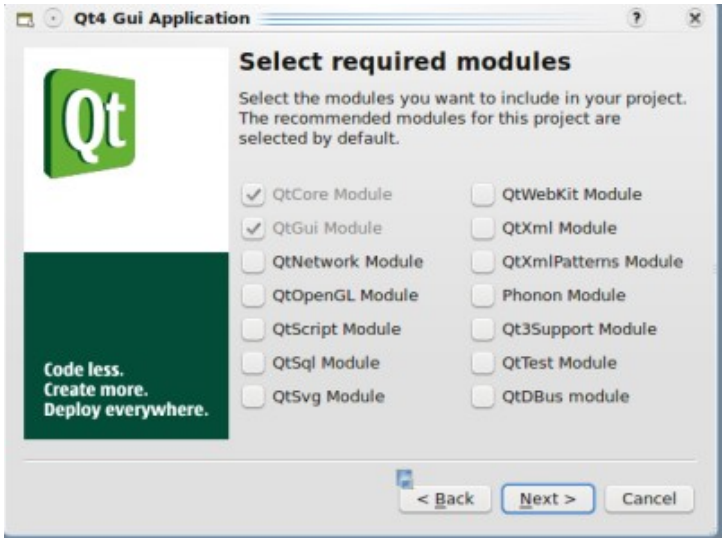
گام هفتم : کار با Qt Creator

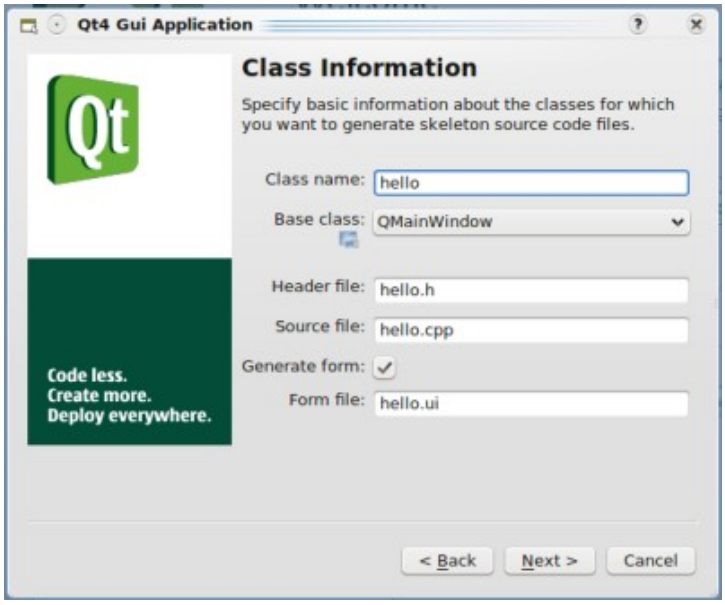
یکی از دوستانم این اموزش ها رو خونده بود و وقتی ازش نظرشو پرسیدم گفت : فرید خیلی خوبو جالبه اما اگه ادم بخواد یه برنامه بزرگ بنویسه اینطوری دخلش میاد!
منم گفتم بابا اینطوری نیس که ، این کارو کردیم تا یه بار با روال کلی اشنا شیم و در مرحله بعد با کلاسای کیوت باید اشنا شیم تا بدونیم چه ابزارهایی رو در اختیار داریم و اما در مورد برنامه بزرگ درست کردن هم باید بگم خیلی راحت باIDE های موجود که در گام اول معرفی کردم میشه برنامه رو ساخت.
برای همین معادل همون برنامه ای رو که در شیش قسمت اول ساختم با صرف یک ربع وقت با IDE قدرتمند Qt Creator میسازیم تا ببینیم چقدر سادس!
خوب این اموزش بیشتر تصویریه تا نوشتاری و فقط جاهایی که احساس کردم احتیاج داره توضیح دادم.

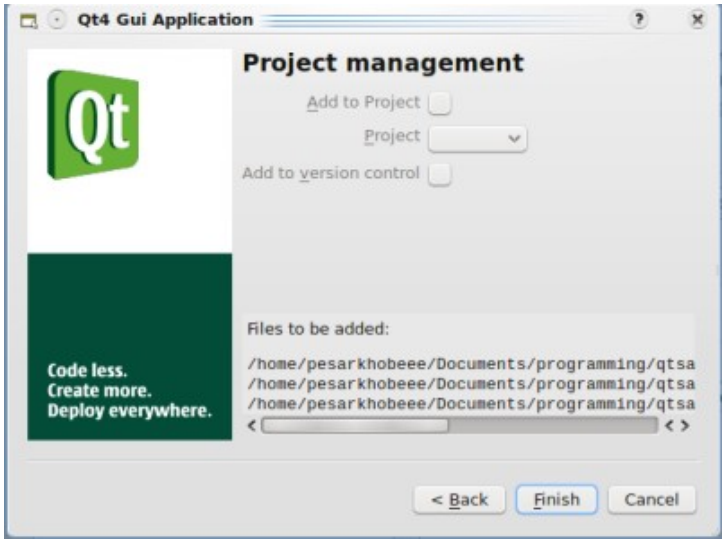
از منوی file گزینه new :



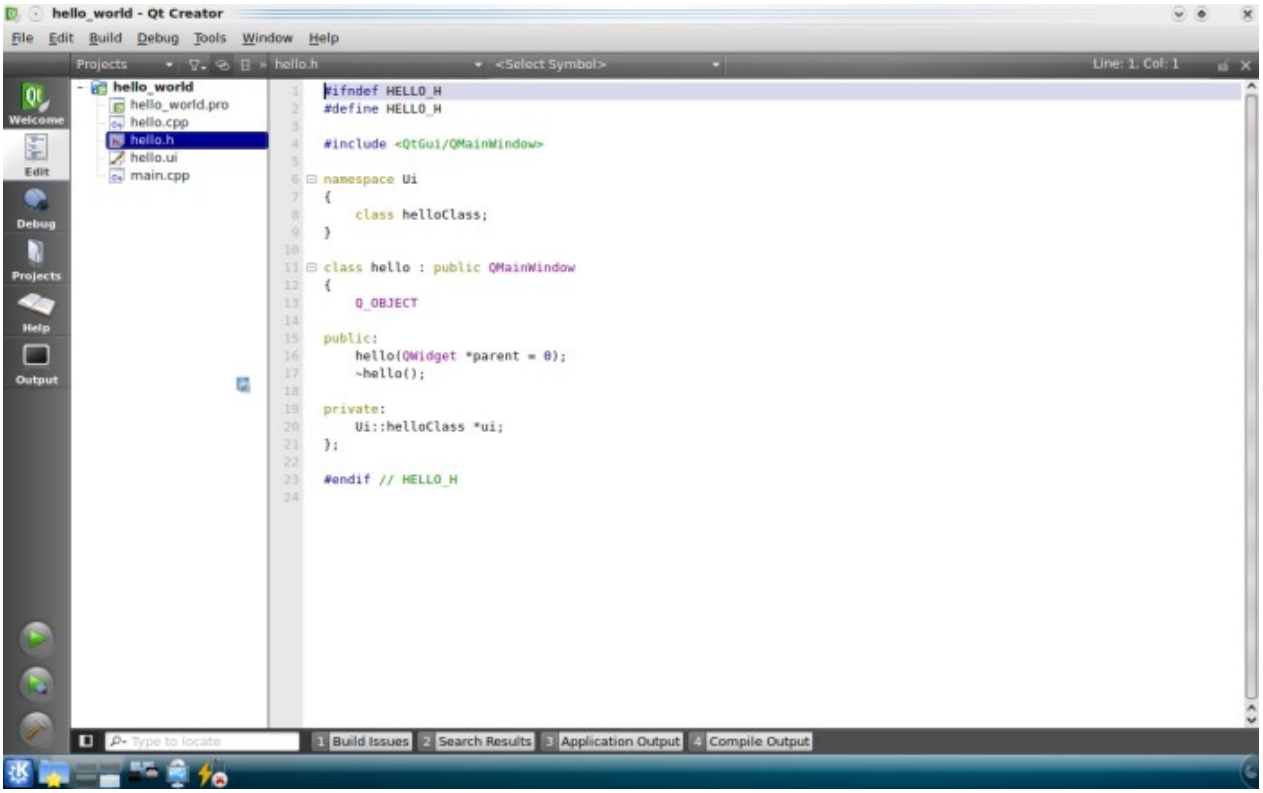

Qt4 Gui Application
Introduction and project location
This wizard generates a Qt4 GUI application project. The application derives by default from QApplication and includes an empty widget.
Code less.
Create more.
Deploy everywhere.
Name:
hello_world
Create in:
ning/qtsample/qtcraetor/hello
Browse...
< Back
Next >
Cancel


Qt4 Gui Application
Select required modules
Select the modules you want to include in your project. The recommended modules for this project are selected by default.
Code less.
Create more.
Deploy everywhere.
QtCore Module
QtGui Module
QtNetwork Module
QtOpenGL Module
QtScript Module
QtSql Module
QtSvg Module
QtWebKit Module
QtXml Module
QtXmlPatterns Module
Phonon Module
Qt3Support Module
QtTest Module
QtDBus module
< Back
Next >
Cancel

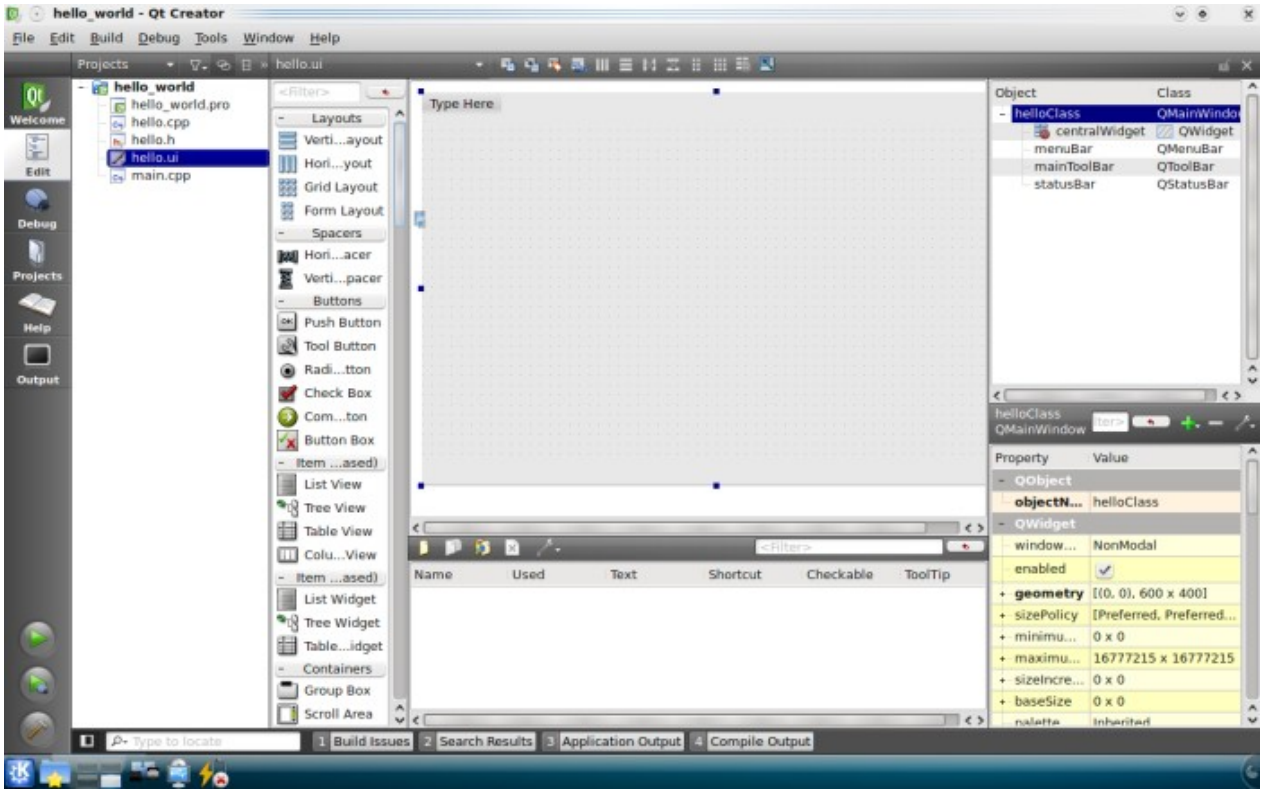




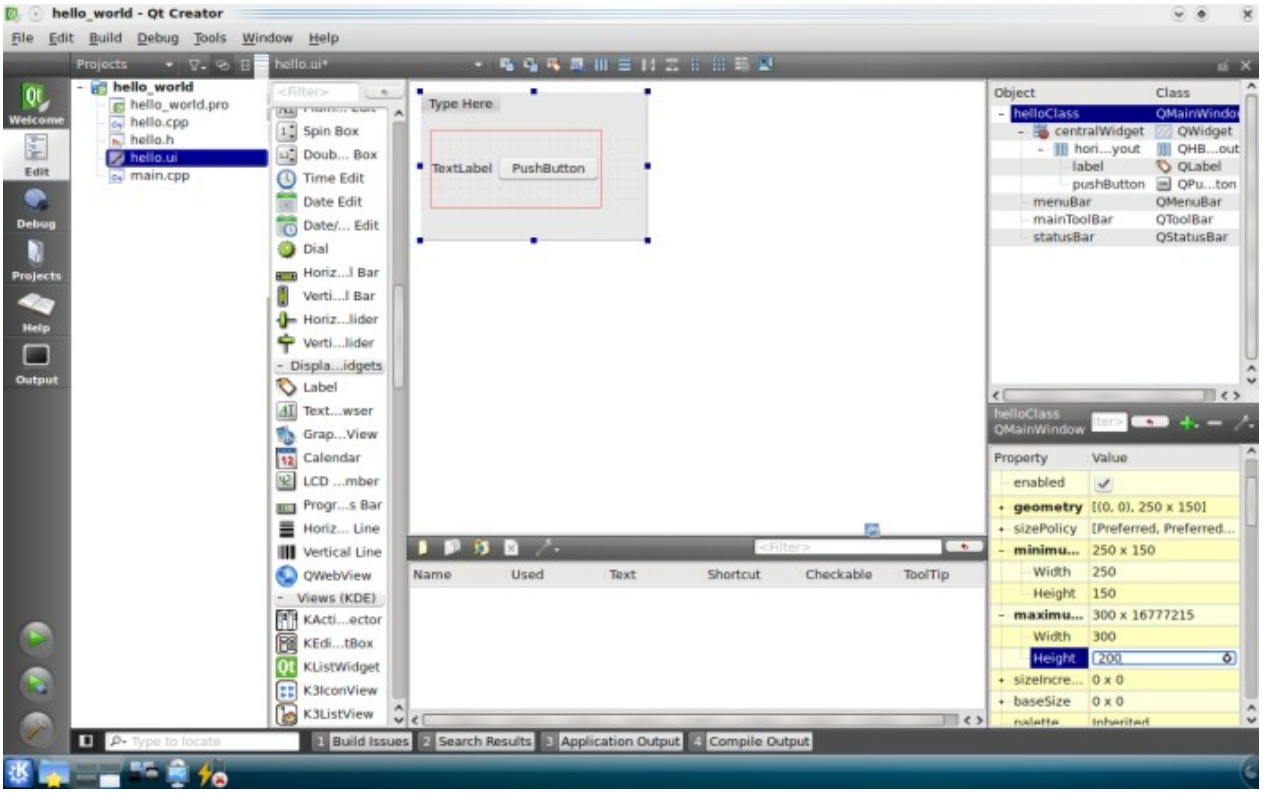
خوب برنامه مون رو ایجاد کردیم و الان در محیط qt creator میبینیم که تمام فایلهای مورد نیازو خودش ساخته



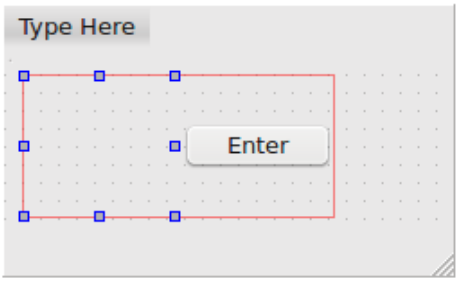
تنها تفاوت فایل های ساخته شده با فایل های خودمون اضافه شدن یک فایل به نام hello.ui هست که پسوند ui به معنای user interface هست !
یعنی برای ساختن رابط کاربری خیلی راحت رو این فایل کلیک میکنیم و بعد با چند تا کلیک تمام کنترل های مورد نیازمونو به برنامه اضافه میکنیمو ویژگیهاشونو تنظیم میکنیم.



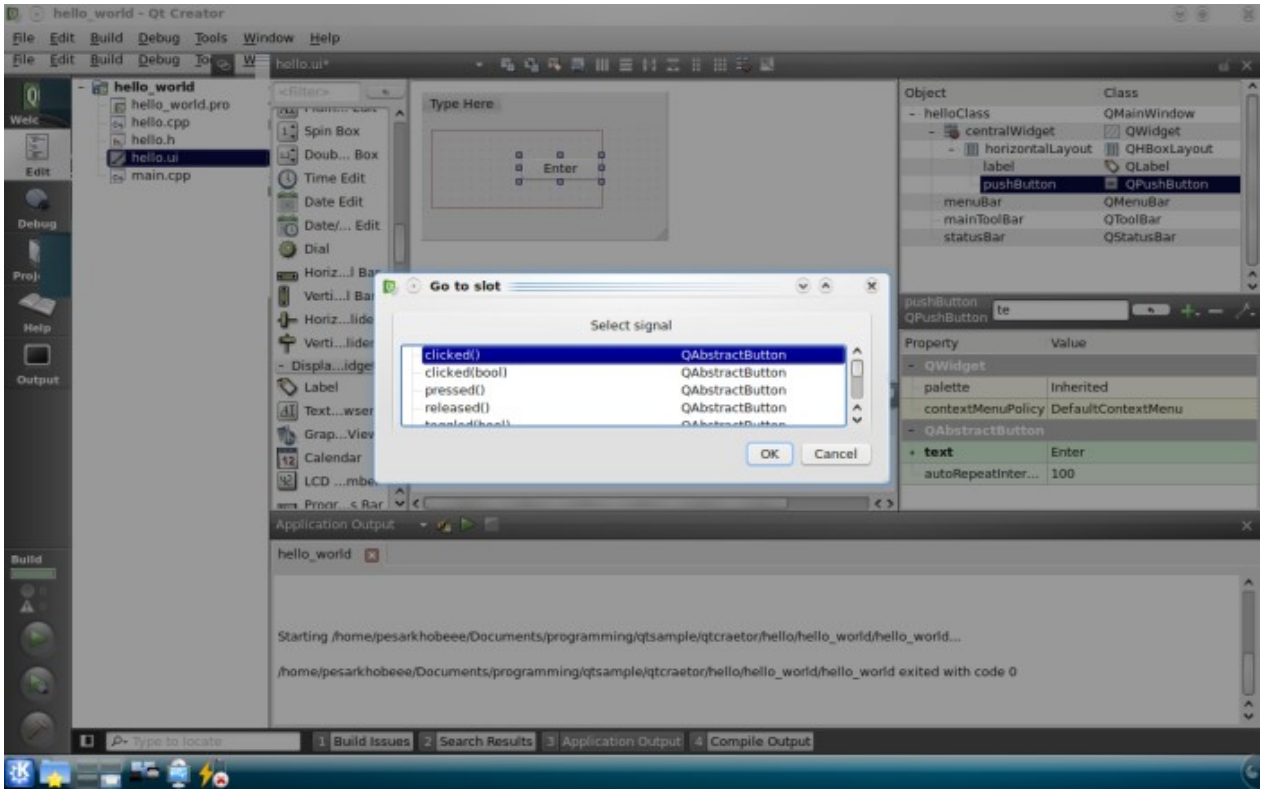
خوب کسانی که قبلا با محیط های توسعه مثل visual studio کار کردن این نما خیلی اشنا میاد ! همونطور که در عکس هم مشخصه خیلی راحت از پنل سمت چپ کافیه کنترل مورد نظرتو بندازید تو پنجره تون و بعد به صورت گرافیکی از پنل سمت راست ویژگیهاشو تنظیم کنید.
این قسمتو توضیح نمیدم چون خودتون با کمی دستکاری میتونید با این محیط اشنا شید و با کمی استفاده از انگلیسی دوران دبیرستانتون میتونید با خصوصیات کنترل هاتون بازی کنیدو یاد بگیرید.
خوب حالا موقشه یادمون بیاریم که تو برنامه ای که دستی نوشته بودیم چیا داشتیم ؟
یه دکمه (button) و یه برچسب (label) که بوسیله یه لایه (layout) مرتب شده بودن پس از پنل سمت چپمون دنبال این کنترل ها میگردیمو به پنجرمون اضافه میکنمیم .



خوب حالا که کنترل ها رو اضافه کردیم سعی میکنیم که خصیه های کنترل هامونو طوری تنظیم کنیم که انتظار داریم!
ولی خوب اگه فقط برای دیدنه اگه همینطوری هم اضافه کنیم کار خواهد کرد.



الان ما چهره برنامممون رو درست کردیم بدون حتی یه خط کد نوشتن و تنها کافیه بگیم هر وقت روی کلیدمون کلیک شد متن بر چسبمون عوض شه! سیگنال و اسلات که یادتون هست؟
خوب برای اینکار کافیه رو کلید کلیک راست کنید و گزینهGo To slot رو بزنید با اینکار میتونید تمام سیگنالای ممکن رو ببینید و سیگنال مورد نظرتونو انتخاب کنید.



خوب از سیگنال ها سیگنال کلیک شدنو انتخاب میکنم و خود برنامه من رو به قسمت ادیتور کدها و به تابعی که اسلات اون سیگنال محسوب میشه هدایت میکنه و کافیه تو اون تابع کد زیرو بزنم:

```
QString str = QString::fromUtf8("افتتاح سایت زنجان لاگ مبارک باد <br> www.zanjanlug.org");
ui->label->setText(str);
```

اگه بخام کدهای بالا رو تشریح کنم باید بگم در خط اول یک رشته به فرمت utf8 درست کردم به نام str و در خط دوم گفتم که در ui یک کنترل دارم به نام label که خصیصه text رو میخام ست کنم و مساوی str قرار بدم.

نکته:
اینجا یه نکته وجود داره و اونم اینه اگه میخواهید این خط رو توسط قابلیت تکمیل خودکار ادیتور راحت بنویسد یک بار باید روی دکمه کامپایل برنامتون زده باشید تا این قابلیت رو داشته باشین اونم بخاطر اینه که فایل ui ما باید یک بار توسط uic (user interface compiler) کامپایل بشه تا ما بتونیم در اتو کامپلت تمام کنترل های موجود و خصیصه هاشونو ببینیم!

خوب با همین دو تاکد برنامه ما به سر انجام میرسه و برای کامپایل کردن هم کافیه روی اون علامت دکه پلی سبز کردن سمت چپ بزنیم تا برناممون به سادگی هر چه تمام تر اجرا بشه!

تبریک میگم.
اینم از اولین برنامه ما که توسط Qt Creator با سادگی هر چه تمام تر ساختیم . خیلی راحت میتونیم با بازی کردن تو این محیط هر روز چیزای جذاب ترو بهتری بسازیم.
لذت ببرید