QT Custom Dialog
This walks through creating a working temperature conversion dialog:

1) Make a new Chemeketa Basic Project. Open the .pro file and add this line to it:

   QT += widgets

   And delete the line:

   CONFIG -= qt //REMOVE THIS!!!

2) Add a new C++ class - we will have this be a subclass of QWidget:
3) Then modify main so it takes in command line arguments and passes them to a QAppication and executes it. The rest of your code will go between those two lines.

```cpp
#include <QApplication>
#include "TempConversionDialog.h"

int main(int argc, char* argv[])
{
    QApplication app(argc, argv);
    TempConversionDialog dlg;
    dlg.show();
    return app.exec();
}
```

You should be able to build and run at this point and get an empty dialog window.

4) Open TempConversionDialog.h... it should already have the basic class definition. Here is what you need to add:

```cpp
private:
    QLineEdit* editFahrenheit;
    QLineEdit* editCelsius;
```

You will also have to include `<QLineEdit>`

The QLineEdit will be the two text boxes for entering/displaying temperatures. These are the only two widgets we need to refer to once we are done constructing the dialog, hence they are the only members.

5) Open the TempConversionDialog.cpp. The constructor is already there, we just have to add code to put in widgets and wire them up. The two rows of labels and line edits we will lay out in a QFormLayout. It automatically lines up columns of labels and controls so they look like the picture to the right.

Add the following to the constructor. You will have to include `<QLabel>` and `<QFormLayout>`:

```cpp
//Setup textboxes and labels in a formlayout
QLabel* binaryLabel = new QLabel("Fahrenheit:");  
QLabel* decimalLabel = new QLabel("Celsius:");  
editFahrenheit = new QLineEdit();  
editCelsius = new QLineEdit();

QFormLayout* topPane = new QFormLayout;  
topPane->addRow(binaryLabel, editFahrenheit);  
topPane->addRow(decimalLabel, editCelsius);
```
6) The buttons will go in a horizontal box. The form layout and horizontal box will both go into a vertical box which we will set as the layout for our dialog. You will need to include `<QPushButton>`:

```cpp
//Setup buttons in a HBoxLayout
QPushButton* btnFToC = new QPushButton("Convert to F to C");
QPushButton* btnCToF = new QPushButton("Convert to C to F");

QHBoxLayout* bottomPane = new QHBoxLayout;
bottomPane->addWidget(btnFToC);
bottomPane->addWidget(btnCToF);

//Put two panes into main layout
QVBoxLayout* mainLayout = new QVBoxLayout;
mainLayout->addLayout(topPane);
mainLayout->addLayout(bottomPane);
setLayout(mainLayout);
```

Build and test the program – it should look pretty but do nothing.

7) Open TempConversionDialog.h. Declare two slot functions we will make be called when someone presses a button.

```cpp
public slots:
    void convertFtoC();
    void convertCtoF();
```

8) Time to implement the slot functions. Start with the FtoC conversion function. It needs to get the text from the Fahrenheit line edit, turn it into a number, do the math with that, turn it back into a string and then put the result in the Celsius line edit.

Here is code to implement its logic doing most of the work with QStrings. Notice there is a built in toDouble function – if takes a bool* that it sets to true or false to let you know if the conversion worked. You will have to include `<QMessageBox>`.

```cpp
//Mostly using QString
void TempConversionDialog::convertFtoC() {
    //Attempt to convert text of Line Edit to double
    bool ok;
    QString FString = editFahrenheit->text();
    double fTemp = FString.toDouble(&ok);

    if(ok) {
        double cTemp = 5.0 / 9.0 * (fTemp - 32);
        //use static function to make string from double
        QString CString = QString::number(cTemp);
        //display answer
        editCelsius->setText( CString );
    } else {
        //Make and show an error message
        QMessageBox errMsg;
        errMsg.setText( "That is not a number" );
        errMsg.exec();
    }
}
```
Now for CtoF. Here is code to implement its logic doing most of the work with std:: functions and strings. If our standard library stod function fails to parse the string into a double, it will throw an exception. You need to include <string> and use namespace std.

```cpp
//Mostly using std::string
void TempConversionDialog::convertCtoF() {
    //Get QString and convert to std::string
    string CString = editCelsius->text().toString();
    try {
        //use string to double function from <string>
        double cTemp = stod(CString);
        double fTemp = 9.0 / 5.0 * cTemp + 32;
        //and back to a string... but need a QString for putting in LineEdit
        QString FString = QString::number(fTemp);
        editFahrenheit->setText(FString);
    } catch (invalid_argument ex) {
        //if stod fails it will throw invalid_argument
        //Make and show an error message
        QMessageBox errMsg;
        errMsg.setText("That is not a number");
        errMsg.exec();
    }
}
```

Go back to the constructor, after the layout is all done and add the connections:

```cpp
//Connect signals
connect(btnFToC, SIGNAL(clicked()),
         this, SLOT(convertFtoC()));
connect(btnCToF, SIGNAL(clicked()),
         this, SLOT(convertCtoF()));
```

The buttons should now work — try typing in some temperatures and converting them.