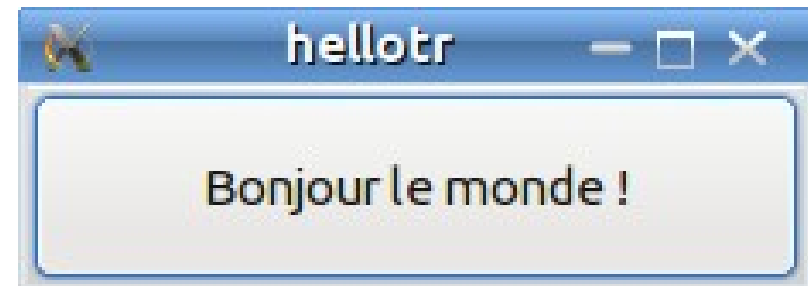
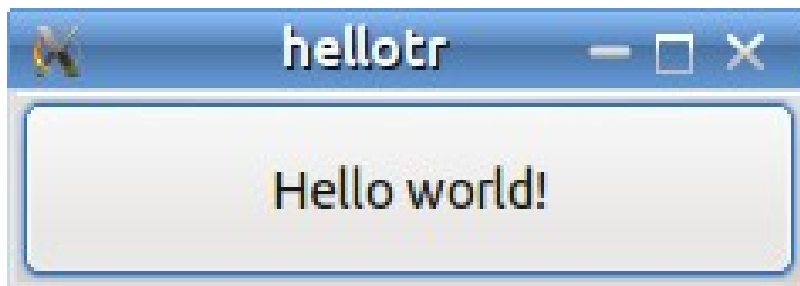


- Qt intègre son propre système de traduction. La prise en charge de plusieurs langues est extrêmement simple dans des applications Qt et ajoute peu de surcharge de travail pour le programmeur.
- Selon le manuel de **Qt Linguist**, l'internationalisation est assurée par la collaboration de trois types de personnes : les développeurs, le chef de projet et les traducteurs.

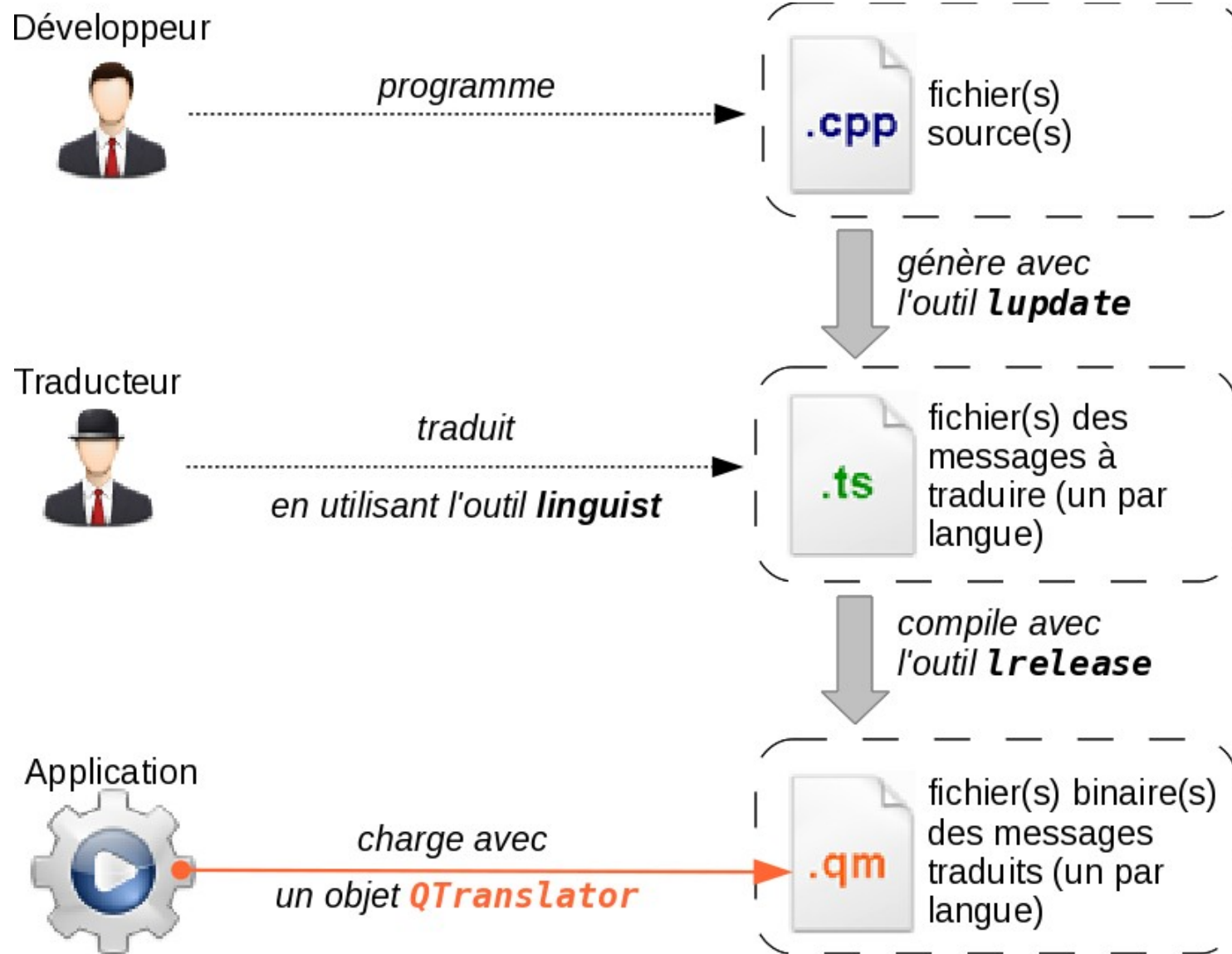


Qt Linguist (<http://qt-project.org/doc/qt-4.8/linguist-manual.html>)



- Dans leur code source, les **développeurs** entrent des chaînes de caractères dans leur propre langue. Ils doivent permettre la traduction de ces chaînes grâce à la méthode **tr()**. En cas d'ambiguïté sur le sens d'une expression, ils peuvent également indiquer des commentaires destinés à aider les traducteurs.
- Le **chef de projet** déclare les fichiers de traduction (un pour chaque langue) dans le fichier de projet (.pro). L'utilitaire **lupdate** parcourt les sources à la recherche de chaînes à traduire et synchronise les fichiers de traduction avec les sources. Les fichiers de traductions sont des fichiers XML portant l'extension **.ts**.
- Les **traducteurs** utilisent **Qt Linguist** pour renseigner les fichiers de traduction. Quand les traductions sont finies, le chef de projet peut compiler les fichiers .ts à l'aide de l'utilitaire **lrelease** qui génère des fichiers binaires portant l'extension **.qm**, exploitables par le programme. Ces fichiers sont lus à l'exécution et les chaînes de caractères qui y sont trouvées remplacent celles qui ont été écrites par les développeurs.

Internationalisation (3/6)





- Le développeur

```
hellotr.cpp  
QPushButton hello(QPushButton::tr("Hello world!"));
```

- Le chef de projet

```
SOURCES      = hellotr.cpp      hellotr.pro  
TRANSLATIONS = hellotr_fr.ts
```

```
$ lupdate -verbose hellotr.pro
```

- Le traducteur

```
$ linguist hellotr_fr.ts
```

- Le chef de projet

```
$ lrelease hellotr_fr.ts
```

Internationalisation (5/6)



- Le traducteur utilise **Qt Linguist** pour renseigner les fichiers de traduction.

The screenshot displays the Qt Linguist application interface. The main window is titled "Qt Linguist" and features a menu bar with "File", "Edit", "Translation", "Validation", "Phrases", "View", and "Help". Below the menu bar is a toolbar with various icons. The interface is divided into several panels:

- Context:** A table with columns "Context" and "Items". It shows a checked entry for "QPushButton" with "1/1" items.
- Strings:** A list of source text entries, including "Hello world!".
- Sources and Forms:** A code editor showing C++ source code with Qt widget includes and a main function that calls `hello.show()` and `return app.exec();`.
- Settings for 'hellotr_la' - Qt Linguist:** A dialog box with two sections: "Source language" (Language: POSIX, Country/Region: Any Country) and "Target language" (Language: French, Country/Region: France). It includes "OK" and "Cancel" buttons.
- Source text:** A text field containing "Hello-world!".
- French translation:** A text field containing "Bonjour le monde!".
- French translator comments:** An empty text field.
- Phrases and guesses:** A table with columns "Source phrase", "Translation", and "Definition". It shows "Hello world!" translated to "Bonjour le ..." with a "Guess (Ctrl+1)" definition.
- Warnings:** An empty panel.

The status bar at the bottom right indicates "1/1".

- Exemple :

```
#include <QApplication>
#include <QPushButton>
#include <QTranslator>
```

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

```
    QTranslator translator;
    translator.load("hello_fr");
    app.installTranslator(&translator);
```

```
    QPushButton hello(QPushButton::tr("Hello world!"));
```

```
    hello.resize(100, 30);
    hello.show();
    return app.exec();
}
```

/ Qt indexe chaque chaîne traduisible dans le contexte de traduction qui est généralement le nom de la sous-classe QObject utilisé. Un contexte de traduction est défini pour les nouvelles classes héritant de QObject et en utilisant la macro Q_OBJECT. */*

