

Librairies de développement Qt

Yannick Kergosien

Polytech Tours
Université François Rabelais de Tours

Janvier 2016

Sommaire

- 1 Le modèle MVC
- 2 Des modules utiles
- 3 Des modules très utiles
- 4 Déroulement des TPs

MVC

Depuis Qt 4 :

- Ensemble de classes qui utilisent une architecture modèle/vue (model/view)
- Gérer simplement la relation entre les données et la façon dont elles sont présentées à l'utilisateur
- Inspiré du modèle de conception (*design pattern*)
Modèle-Vue-Contrôleur
 - Un modèle de données
 - Une vue qui présente ces données
 - Un contrôleur ayant un rôle de logique de contrôle, gestion des événements et synchronisation

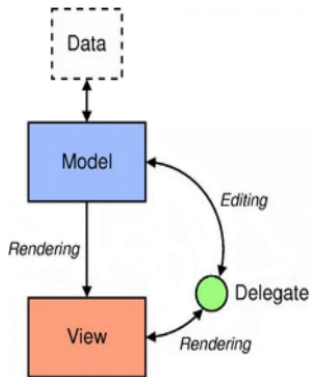
MVC

Le pattern MVC :

- Impose la séparation entre les données, la présentation et les traitements
 - Modèle : contient les données et s'occupe du traitement et des interactions des données avec les BdD, les fichiers, etc.
 - Vue : représentation graphique (ce que peut voir l'utilisateur)
 - Contrôleur : prend en charge l'interaction avec l'utilisateur, recevant tous les événements déclenchés par l'utilisateur (clic, sélection...) et mettant par la suite à jour les données

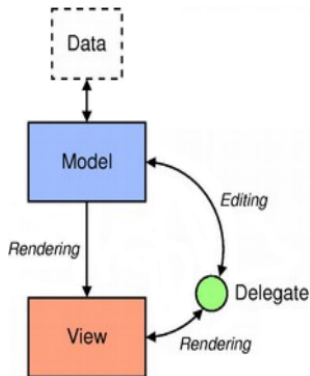
MVC

- Architecture flexible, claire et maintenable
- En pratique, le contrôleur est intégré à la vue à l'aide de délégué (delegate)



MVC

- Architecture flexible, claire et maintenable
- En pratique, le contrôleur est intégré à la vue à l'aide de délégué (delegate)



Les modèles

Les classes pour les modèles :

- QAbstractItemModel : classe la plus abstraite et la plus haute dans la hiérarchie (interface que tous les modèles doivent respecter)
- QStringListModel : stockage d'une liste de QString
- QFileSystemModel : un ensemble d'informations sur un fichier ou un répertoire du système de fichier local
- QStandardItemModel : gestion de tout type de structure, complexe ou non
- QSqlTableModel, QSqlRelationalTableModel : accès à une base de données

Les modèles

On peut créer ses propres modèles, il suffit de dériver d'un des classes suivantes :

- `QAbstractItemModel` : classe la plus abstraite, plus flexible mais plus complexe
- `QAbstractListModel` : classe abstraite pour un modèle de type liste
- `QAbstractTableModel` : classe abstraite pour un modèle de type tableau

Les vues

Il existe 3 types de vues pour les modèles :

- QListView : liste d'éléments
- QTableView : tableau d'éléments
- QTreeView : représentation d'éléments sous forme hiérarchique (arbre)

Les vues

Il existe 2 types de délégués (basés tous deux sur QAbstractItemDelegate) :

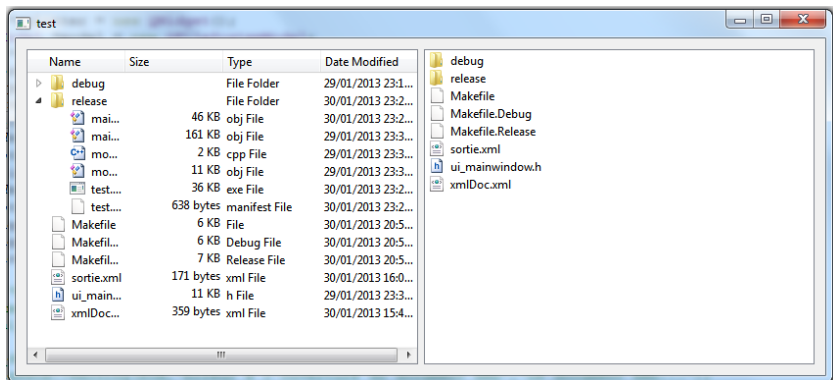
- QItemDelegate
- QStyledItemDelegate (les vues sont dotées par défaut de ce délégué)
- QSqlRelationalDelegate (pour les QSqlRelationalTableModel)

Des exemples

Code

```
1   QWidget *maWid = new QWidget();
2   QHBoxLayout * malay= new QHBoxLayout(maWid);
3   QFileSystemModel *model = new QFileSystemModel;
4   model->setRootPath(QDir::currentPath());
5   QModelIndex parentIndex = model->index(QDir::
        currentPath());
6   QTreeView *tree = new QTreeView(maWid);
7   tree->setModel(model);
8   tree->setRootIndex(parentIndex);
9   QListView *list = new QListView(maWid);
10  list->setModel(model);
11  list->setRootIndex(parentIndex);
12  malay->addWidget(tree);
13  malay->addWidget(list);
14  maWid->show();
```

Des exemples



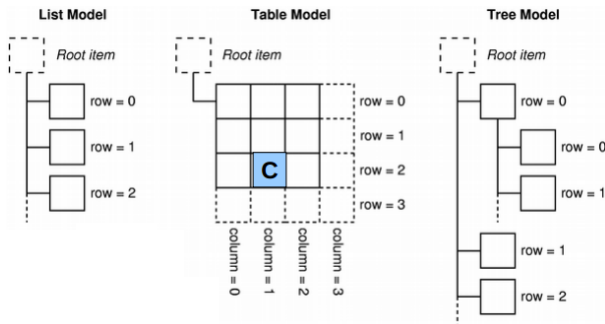
L'Index

Index de modèle :

- Objectif : séparer la représentation des données et la façon dont elles sont consultées
- Utilisé par les vues et les délégués
- Idée : gère/pointe/représente un élément dans un modèle (ex : gérer les sélections par l'utilisateur)

Concept de modèle

Toutes sous-classes de `QAbstractItemModel` représentent les données comme une structure hiérarchique contenant des tableaux d'objets



```
QModelIndex indexC = model->index(2, 1);
```

Concept de modèle

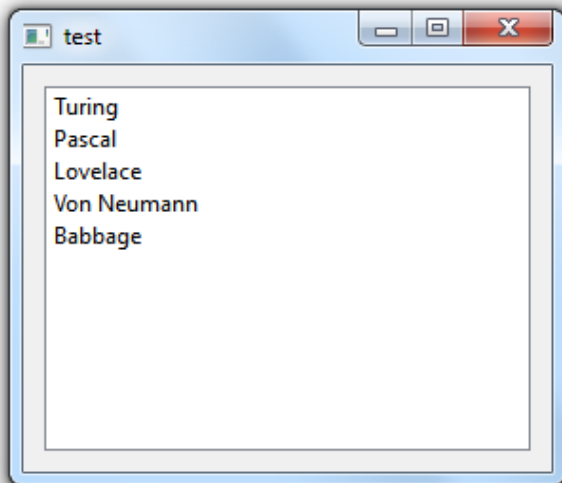
- Mécanisme signaux/slots pour que les modèles notifient à toutes les vues attachées tout changement dans les données
- Lorsqu'on utilise l'architecture MV(C), il faut :
 - Définir et créer le modèle
 - Définir et créer la vue
 - Associer la vue et le modèle
 - Paramétrer les délégués

Autre exemple

Code

```
1  QStringList listeNom;  
2  listeNom << "Turing" << "Pascal" << "Lovelace" << "  
    VonNeumann" << "Babbage";  
3  
4  QStringListModel *modele = new QStringListModel(  
    listeNom);  
5  
6  QListView *vueListe = new QListView(box);  
7  
8  vueListe->setModel(modele);
```


Concept de modèle



Manipuler un modèle

Quelques méthodes :

- index, hasIndex
- insertColumn, insertRow, insertColumns, insertRows
- removeColumn, removeRow, removeColumns, removeRows
- columnCount , rowCount
- setData, data

Créer son propre modèle :

- Dériver d'une des classes suivantes : QAbstractItemModel, QAbstractListModel, QAbstractTableModel
- Redéfinir certaines méthodes virtuelles

Autre exemple

Code 1/3

```
1 class NomTableModel : public QAbstractTableModel {
2     private:
3         QStringList listeNom;
4     public:
5         NomTableModel(const QStringList &p_listeNom,
6             QObject *parent=0): QAbstractTableModel(parent)
7         {
8             if(p_listeNom.count() > 0)
9                 this->listeNom = p_listeNom;
10        }
11        int rowCount(const QModelIndex &parent =
12            QModelIndex()) const
13        {
14            return this->listeNom.count();
15        }
16    }
```

Autre exemple

Code 2/3

```
1      int columnCount(const QModelIndex &parent =
           QModelIndex()) const
2      {
3          return 2; // pour insérer les (futures) prénoms
4      }
5      QVariant data(const QModelIndex &index, int role)
           const
6      {
7          if(! index.isValid())
8              return QVariant();
9
10         if(((unsigned int)index.row())>=this->listeNom.
               count())||(index.column())>= 2))
11             return QVariant();
```

Autre exemple

Code 3/3

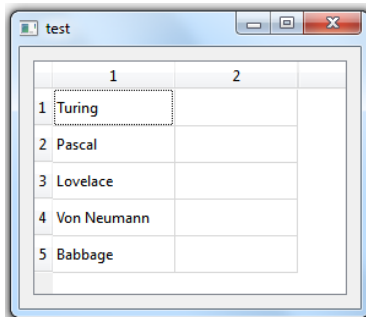
```
1      if ((role == Qt::DisplayRole))
2      {
3          if(index.column() == 0) // la première colonne
4              return this->listeNom.at(index.row());
5              // retourne le nom demandé
6      }
7      else
8          return QVariant();
9  }
10};
```

Autre exemple

Code 3/3

```
1  QStringList listeNom;  
2  listeNom << "Turing" << "Pascal" << "Lovelace" << "  
    Von_ Neumann" << "Babbage";  
3  
4  NomTableModel *modele = new NomTableModel(listeNom);  
5  QTableView *vueTable = new QTableView;  
6  vueTable->setModel(modele);
```

Concept de modèle



Concept de modèle

- On veut modifier directement le modèle (ajouter des prénoms)
 - Il faut redéfinir setData
`QAbstractItemModel::setData(const QModelIndex & index,
const QVariant & value, int role = Qt::EditRole)`
- On veut interdire/autoriser l'édition de certains éléments
 - Il faut redéfinir flags
`Qt::ItemFlags flags(const QModelIndex & index) const`

Autre exemple

Code

```
1 private:
2   QStringList listeNom; QStringList listePrenom;
3   ...
4   QVariant data(const QModelIndex &index, int role){
5       if(!index.isValid()) return QVariant();
6       if((index.row()>=this->listeNom.count())|| (index.
           column()>=listePrenom.count()))
7           return QVariant();
8       if((role==Qt::DisplayRole || role==Qt::EditRole)){
9           if(index.column() == 0)
10              return this->listeNom.at(index.row());
11           else if(index.column() == 1)
12              return this->listePrenom.at(index.row());
13       }
14       return QVariant();
15 }
```

Autre exemple

Code

```
1  ...
2  bool setData(const QModelIndex &index, const QVariant
    &value, int role)
3  {
4      if(index.isValid() && role == Qt::EditRole)
5      {
6          this->listePrenom.replace(index.row(), value.
            toString());
7          emit(dataChanged(index, index));
8          return true;
9      }
10     return false;
11 }
```

Autre exemple

Code

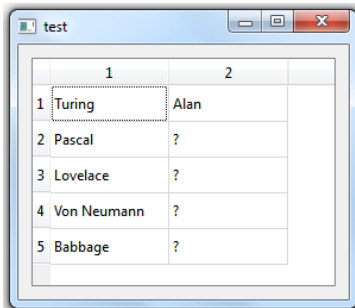
```
1  ...
2  Qt::ItemFlags flags(const QModelIndex &index) const
3  {
4      if(!index.isValid())
5          return Qt::ItemIsEnabled;
6      if(index.column() == 0) {
7          return Qt::ItemIsEnabled | Qt::ItemIsSelectable;
8      }
9      else if (index.column() == 1) {
10         return Qt::ItemIsEnabled | Qt::ItemIsSelectable |
11             Qt::ItemIsEditable;
12     }
13     return QAbstractTableModel::flags(index);
14 }
```

Autre exemple

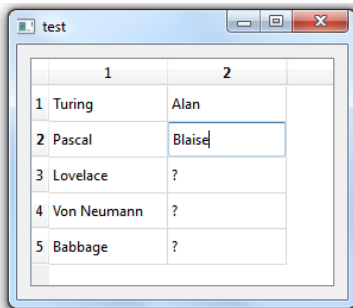
Code

```
1  ...
2  QStringList listeNom ;
3  listeNom << "Turing" << "Pascal" << "Lovelace" << "
    Von_ Neumann" << "Babbage";
4  QStringList listePrenom ;
5  listePrenom << "Alan" << "?" << "?" << "?" << "?";
6  NomTableModel *modele = new NomTableModel(listeNom ,
    listePrenom);
7  QTableView *vueTable = new QTableView ;
8  vueTable->setModel(modele);
```

Concept de modèle



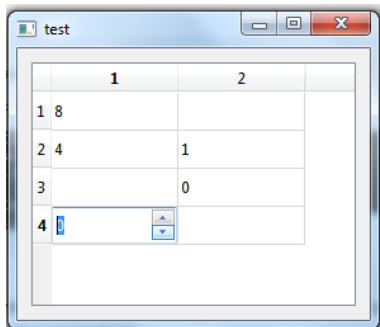
	1	2
1	Turing	Alan
2	Pascal	?
3	Lovelace	?
4	Von Neumann	?
5	Babbage	?



	1	2
1	Turing	Alan
2	Pascal	Blaise
3	Lovelace	?
4	Von Neumann	?
5	Babbage	?

Quelques mots sur les Delegates

- Concept : Permettre l'édition ou la modification de données par éléments depuis la vue.
- Exemple :



Autre exemple

Code

```
1 class SpinBoxDelegate : public QItemDelegate
2 {
3     Q_OBJECT
4 public:
5     SpinBoxDelegate(QObject *parent = 0);
6     QWidget *createEditor(...) const;
7     void setEditorData(...) const;
8     void setModelData(...) const;
9     void updateEditorGeometry(...) const;
10 };
```

Autre exemple

Code

```
1 QWidget *SpinBoxDelegate::createEditor(QWidget *parent
    , const QStyleOptionViewItem & /* option */, const
    QModelIndex & /* index */) const
2 {
3     QSpinBox *editor = new QSpinBox(parent);
4     editor->setMinimum(0);
5     editor->setMaximum(100);
6
7     return editor;
8 }
```


Autre exemple

Code

```
1 void SpinBoxDelegate::setEditorData(QWidget *editor ,  
   const QModelIndex &index) const  
2 {  
3     int value = index.model()->data(index , Qt::  
        EditRole).toInt();  
4  
5     QSpinBox *spinBox = static_cast<QSpinBox*>(editor)  
        ;  
6     spinBox->setValue(value);  
7 }
```

Autre exemple

Code

```
1 void SpinBoxDelegate::setModelData(QWidget *editor ,
   QAbstractItemModel *model, const QModelIndex &
   index) const
2 {
3     QSpinBox *spinBox = static_cast<QSpinBox*>(editor)
   ;
4     int value = spinBox->value();
5
6     model->setData(index, value, Qt::EditRole);
7 }
```

Autre exemple

Code

```
1 void SpinBoxDelegate::updateEditorGeometry(QWidget *  
    editor ,  
2     const QStyleOptionViewItem &option , const  
        QModelIndex & /* index */) const  
3 {  
4     editor->setGeometry(option.rect);  
5 }
```

Autre exemple

Code

```
1  QStandardItemModel model(4, 2);
2  QTableView tableView;
3  tableView.setModel(&model);
4
5  SpinBoxDelegate delegate;
6  tableView.setItemDelegate(&delegate);
```

Autre exemple

Encore d'autres exemples...

Sommaire

- 1 Le modèle MVC
- 2 Des modules utiles**
- 3 Des modules très utiles
- 4 Déroulement des TPs

QtSQL

- Ensemble de classes de haut niveau pour manipuler des bases de données
- S'utilise de façon simple et transparente

Exemple: table test

id	num	nom	prenom	ville	cp
1	1000	Billaut	Jean-Charles	Tours	37200
2	1000	Proust	Christian	Joué-lès-Tours	37300
3	1234	Néron	Emanuel	Chambray les Tours	37170

Création d'une nouvelle connexion

Code

```
1  QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL", "mabdd");
2  db.setHostName("host");
3  db.setDatabaseName("test");
4  db.setUserName("lang4");
5  db.setPassword("hcv8,BJWL4LLWQNh");
6  bool ok = db.open();
```

Code

```
1  QSqlDatabase maBDD = QSqlDatabase::database("mabdd");
```


Exécuter une requête

Code

```
1 QSqlQuery query(db);  
2 query.exec("SELECT nom, prenom, num FROM test WHERE  
   num > 1000");
```

Code

```
1 while(query.next()) {  
2     QString nom = query.value(0).toString();  
3     int num = query.value(2).toInt();  
4     std::cout << num << " :: " << qPrintable(nom) <<  
       std::endl;  
5 }
```

Exécuter une requête

Code

```
1 QSqlQuery query(db);  
2 query.exec("INSERT INTO test (num,nom,prenom,ville  
    ,cp) VALUES (2000,'Tkindt','Vincent','Tours  
    ','37000')");  
3
```

- Pas terrible pour l'insertion, si les valeurs à insérer sont contenues dans des variables
- Construire une chaîne de caractère en faisant attention à gérer les caractères spéciaux
- Et il y a de fortes chances que ce soit le cas dans une application réelle

Exécuter une requête

Code

```
1  QSqlQuery query(db);
2  query.prepare("INSERT INTO test (num,nom,prenom,
    ville,cp) VALUES (:num,:nom,:prenom,:ville,:cp)");
3  query.bindValue(":num",2000);
4  query.bindValue(":nom","Tkindt");
5  query.bindValue(":prenom","Vincent");
6  ...
7  query.exec();
8
9  query.bindValue(":num",3000);
10 query.bindValue(":nom","Slimane");
11 ...
12 query.exec();
```

Autre exemple

Encore d'autres exemples...

QtXml

XML : méta-langage (“langage de balisage extensible”) :

- Permet de définir des formats de fichiers
- Utilisé par les flux RSS, OpenDocument, SVG, ...
- Deux approches pour la manipulation :
 - DOM (*Simple API for XML*) : “Le tout en mémoire”
 - SAX (*Document Object Model*) : “capturer des événements ”
(pour les documents plus lourd)
- Ne pas oublier : QT += xml dans le .pro et les
#include<QtXml>

QtXml : DOM

Code 1/2

```
1 QDomDocument *dom = new QDomDocument("mon_xml");
2 QFile xml_doc("xmlDoc.xml");//Sélection du file XML.
3
4 if (!xml_doc.open(QIODevice::ReadOnly)) {
5     QMessageBox::warning(this, "Erreur à l'ouverture du
        document XML", "Le document XML...");
6     return;
7 }
8
9 if (!dom->setContent(&xml_doc)){//Associer le fichier à
    l'objet DOM.
10    xml_doc.close();
11    QMessageBox::warning(this, "Erreur à l'association...");
12    return;
13 }
```

QtXml : DOM

Code 2/2

```
1 xml_doc.close(); // Plus besoin du fichier, tout est  
   compris dans l'objet DOM  
2 ....  
3 // /\ Il manque les delete !
```

Idee de parcours d'un fichier Xml :

- ❶ On récupère un élément (QDomElement)
- ❷ On prend le premier noeud (QDomNode) (.firstChild())
- ❸ On parcourt de noeud en noeud (.nextSibling())
- ❹ Un noeud peut être convertit en élément (.toElement()) pour le parcourir (retour en 1)

QtXml : DOM

Xml

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <actu_des_enseignants type="info">
3      <information>Il collectionne des photos de
        lampadaire.</information>
4      <information>Il a posé pour la revue Femme
        Actuelle (avril 2008).</information>
5      <information>Il joue toujours à HOTS.</
        information>
6      <information>Il a 6 enfants.</information>
7  </actu_des_enseignants>
```

- On veut parcourir ce fichier Xml.

QtXml : DOM

Xml : exemple lecture

```
1  QDomElement dom_element = dom.documentElement();
2  cout << dom_element.tagName() << endl; //Affiche
    actu_des_enseignants
3  cout << dom_element.attribute("type") << endl; //
    Affiche info
4  QDomNode noeud = dom_element.firstChild();
5  while(!noeud.isNull())
6  {
7      dom_element=noeud.toElement();
8      if(!dom_element.isNull())
9          QMessageBox::information(this, "Une info",
              dom_element.text());
10     noeud = noeud.nextSibling();
11 }
```

QtXml : DOM

Xml : exemple écriture 1/2

```
1 QDomDocument doc;  
2 QDomNode xmlNode = doc.createProcessingInstruction("xml",  
    "version=\"1.0\" encoding=\"UTF-8\"");  
3 doc.insertBefore(xmlNode, doc.firstChild());  
4  
5 QDomElement root = doc.createElement("actu_des_enseignants");  
6 root.setAttribute("type", "Intox");  
7 doc.appendChild(root);  
8  
9 QDomElement info = doc.createElement("intox");  
10 root.appendChild(info);  
11 QDomText nomText = doc.createTextNode("blablabla");  
12 info.appendChild(nomText);
```

QtXml : DOM

Xml : exemple écriture 2/2

```
1 QFile file( "sortie.xml" );  
2 file.open(QIODevice::WriteOnly);  
3  
4 QTextStream flux(&file);  
5 flux << doc.toString();  
6  
7 file.close();
```

QtXml : SAX

Xml : exemple lecture

```
1 QDomStreamReader reader;  
2 QFile file("xmlDoc.xml");  
3 file.open(QFile::ReadOnly | QFile::Text);  
4 reader.setDevice(&file); //Initialisation  
5 reader.readNext(); //Renvoie balise ouvrante/fermante  
6 while (!reader.atEnd()){  
7     if (reader.isStartElement()){ //si balise ouvrante  
8         if (reader.name() == "actu_des_enseignants"){  
9             //reader.attributes().value("type").toString()  
10            reader.readNext();  
11            //reader.readElementText()  
12            ....  
13        }  
14    }  
15    reader.readNext();  
16 }
```

QtXml : SAX

Xml : exemple écriture

```
1 QFile file("sortie.xml");
2 file.open(QFile::WriteOnly | QFile::Text);
3 QDomStreamWriter writer(&file);
4 writer.setAutoFormatting(true); // Pour indentation
5 writer.writeStartDocument(); // Écrit l'en-tête
6 writer.writeStartElement("actu_des_enseignants");
7 writer.writeAttribute("type", "Intox");
8 writer.writeTextElement("intox", "blablabla");
9 writer.writeEndElement();
10 writer.writeEndDocument();
11 file.close();
```

QtWebkit

- Rendu de page web (\approx navigateur très simple)
- Prend en compte : HTML 5, HTML/XHTML, CSS, Javascript, SVG, XPath, AJAX
- Supporte les cookies, un historique, ...
- Inclus d'autres classes :
 - QNetworkAccessManager, QNetworkRequest
 - QNetworkCookieJar, QNetworkCookie
 - QTcpSocket, QSslSocket, QAuthenticator
 - ...

QtWebkit

Afficher une page web

```
1   QWebView *view = new QWebView(parent);  
2   view->load(QUrl("http://qt.nokia.com/"));  
3   view->show();
```

- QWebView → QWebPage → n QWebFrame

QtWebkit

Derrière la QWebView :

- Envoie et réception des données grâce à QNetworkAccessManager
 - get() et put() : envoie et réception des “données réseaux”
 - head() et post() : envoie des requêtes HTTP
- Les demandes encapsulés grâce à QNetworkRequest
- Les réponses encapsulés grâce à QNetworkReply

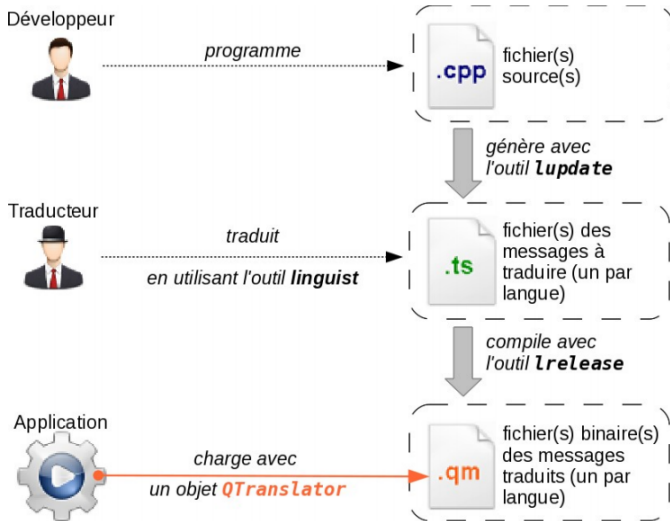
Sommaire

- 1 Le modèle MVC
- 2 Des modules utiles
- 3 Des modules très utiles**
- 4 Déroulement des TP

QtLinguist

- Objectif : améliorer la collaboration entre les développeurs et les traducteurs
- Idée : Pouvoir faire fonctionner son application sous plusieurs langues
- Protocole :
 - Utilisation de la macro “tr()” pour toutes les chaînes dans le code source
 - Ajout dans le fichier .pro : TRANSLATIONS = apptr_fr.ts \ apptr_en.ts
 - L'utilitaire lupdate parcourt les sources à la recherche des *tr* et synchronise les fichiers de traduction avec les sources (fichiers XML portant l'extension .ts).
 - Les traducteurs utilisent QTlinguist (application graphique) pour traduire ces fichiers qui seront générés en fichiers binaires (.qm) à l'aide de l'utilitaire lrelease

QtLinguist



QtLinguist

Instancier la traduction

```
1QTranslator translator;  
2translator.load("apptr_fr");//Utilisera les fichiers .  
   qm  
3app.installTranslator(&translator);
```

Et encore d'autres modules

- QPainter :
 - Pour le dessin
 - QPoint, QLine, QRect, QPolygon, QPainterPath, QRegion, ...
- QtOpenGL :
 - Héritage de QGLWidget
 - 3 méthodes à redéfinir : initializeGL(), paintGL() et resizeGL(int w, int h)
- QTest :
 - Vos tests sont des slots et le framework les déclenche
 - Bien adapté aux tests GUI (simulation de clicks et tests de benchmark)

Sommaire

- 1 Le modèle MVC
- 2 Des modules utiles
- 3 Des modules très utiles
- 4 Déroulement des TPs**

Déroulement des TPs

- 10 séances au total
 - 2,5 premières sur la STL
 - 6,5 autres sur Qt
 - 1 séance d'évaluation (Mercredi 23 Mars)
- Objectif : revoir et approfondir les notions vues en cours par le biais d'un projet

Déroulement des TPs

- En binôme
- Projet guidé jusqu'à la dernière séance
- Projet qui tiendra dans les créneaux (normalement)
- Evaluation :
 - Le projet rendu
 - Suite à toutes les séances
 - Et suite à la liste des modifications à faire en 2h

Déroulement des TPs

- Bien s'en sortir lors de la liste des modifications à faire :
 - Connaître son projet
 - Commentaire
 - Codé de manière modulable et flexible
 - **MVC !**
- Source unique !

Déroulement des TPs

- Idée du projet :
 - Des clients qui souhaitent des rendez-vous avec différents services (banques, assurances, etc.)
 - Des ressources à gérer (banquiers, courtiers, ...)
 - Un algorithme de planification (simple!)
- Source unique !

Déroulement des TPs

- Compétences travaillées et évaluées:
 - L'utilisation (intelligente et efficace) de la STL
 - Créer une interface graphique (connaître les principaux composants et leurs fonctionnalités, et l'ergonomie)
 - QtSQL et QtXML
 - et bien sûr le modèle MVC (view, model, delegate)
- Source unique !