UE MOBJ [4L103]

Jean-Paul Chaput
Jean-Paul.Chaput@lip6.fr

SESI

2018-2019

VIII.2

```
#include <QAbstractTableModel>

class InstancesModel : public QAbstractTableModel {
    Q_OBJECT;
  public:
                 InstancesModel ( QObject* parent=NULL );
                ~InstancesModel ();
    void         setCell         ( Cell* );
    Cell*        getModel        ( int row );
    int          rowCount        ( const QModelIndex& parent=QModelIndex()
    int          columnCount     ( const QModelIndex& parent=QModelIndex()
    QVariant     data            ( const QModelIndex& index, int role=Qt::D
    QVariant     headerData      ( int section
                                 , Qt::Orientation orientation
                                 , int role=Qt::DisplayRole ) const;
  private:
    Cell* cell_;
};
```

VIII.2

```
InstancesModel::InstancesModel ( QObject* parent )
  : QAbstractTableModel(parent)
  , cell_(NULL)
{ }

InstancesModel::~InstancesModel ()
{ }

void  InstancesModel::setCell ( Cell* cell ) {
  emit layoutAboutToBeChanged();
  cell_ = cell;
  emit layoutChanged();
}
```

VIII.2

```
int  InstancesModel::rowCount(const QModelIndex& parent) const
{ return (cell_) ? cell_->getInstances().size() : 0; }

int  InstancesModel::columnCount(const QModelIndex& parent) const
{ return 2; }

QVariant  InstancesModel::data( const QModelIndex& index
                              , int role ) const
{
  if (not cell_ or not index.isValid()) return QVariant();
  if (role == Qt::DisplayRole) {
    int row = index.row();
    switch ( index.column() ) {
      case 0: return cell_->getInstances()[row]->getName().c_str();
      case 1: return cell_->getInstances()[row]
                          ->getMasterCell()->getName().c_str();
    }
  }
  return QVariant();
}
```

VIII.2

```
QVariant  InstancesModel::headerData( int  section
                                    , Qt::Orientation orientation
                                    , int  role ) const
{
  if (orientation == Qt::Vertical) return QVariant();
  if (role != Qt::DisplayRole) return QVariant();

  switch ( section ) {
    case 0: return "Instance";
    case 1: return "MasterCell";
  }
  return QVariant();
}

Cell* InstancesModel::getModel ( int row )
{
  if (not cell_) return NULL;
  if (row >= (int)cell_->getInstances().size()) return NULL;
  return cell_->getInstances()[ row ]->getMasterCell();
}
```

VIII.3

```
class InstancesWidget : public QWidget {
    Q_OBJECT;
  public:
                    InstancesWidget ( QWidget* parent=NULL );
              void  setCellViewer   ( CellViewer* );
              int   getSelectedRow  () const;
    inline    void  setCell         ( Cell* );
  public slots:
              void  load            ();
  private:
              CellViewer*     cellViewer_;
              InstancesModel* baseModel_;
              QTableView*     view_;
              QPushButton*    load_;
};
```

VIII.3

```
InstancesWidget::InstancesWidget ( QWidget* parent )
  : QWidget         (parent)
  , cellViewer_     (NULL)
  , baseModel_      (new InstancesModel(this))
  , view_           (new QTableView(this))
  , load_           (new QPushButton(this))
{
  setAttribute( Qt::WA_QuitOnClose   , false );
  setAttribute( Qt::WA_DeleteOnClose, false );
  setContextMenuPolicy( Qt::ActionsContextMenu );

  view_->setShowGrid                ( false );
  view_->setAlternatingRowColors( true );
  view_->setSelectionBehavior    ( QAbstractItemView::SelectRows );
  view_->setSelectionMode        ( QAbstractItemView::SingleSelection );
  view_->setSortingEnabled       ( true );
  view_->setModel                ( baseModel_ ); // On associe le modele.

  // ...
}
```

VIII.3

```
InstancesWidget::InstancesWidget ( QWidget* parent )
{
  QHeaderView* horizontalHeader = view_->horizontalHeader();
  horizontalHeader->setDefaultAlignment  ( Qt::AlignHCenter );
  horizontalHeader->setMinimumSectionSize( 300 );
  horizontalHeader->setStretchLastSection( true );

  QHeaderView* verticalHeader = view_->verticalHeader();
  verticalHeader->setVisible( false );

  load_->setText( "Load" );
  connect( load_, SIGNAL(clicked()), this, SLOT(load()) );
}
```

VIII.3

```
int  InstancesWidget::getSelectedRow () const
{
  QModelIndexList selecteds = view_->selectionModel()
                                   ->selection().indexes();
  if (selecteds.empty()) return -1;
  return selecteds.first().row();
}


void  InstancesWidget::load ()
{
  int selectedRow = getSelectedRow();
  if (selectedRow < 0) return;
  cellViewer_->setCell( baseModel_->getModel(selectedRow) );
}
```

VII.2

```
#include <exception>

class Error : public std::exception {
  private:
    std::string  message_;
  public:
    Error( string message ) throw() { message_=message; };
  public:
    ~Error() throw() {};
  public:
    const char* what() const throw()
                       { return message_.c_str(); };
};
```

VII.2

```
while(true) {
  switch(state) {
    // Reading node contents.
  } // End switch(state).

  throw Error("[ERROR]␣Cell::fromXml():␣Unknown␣or␣misplaced␣tag.");
} // End while(true).
```

VII.2

```
int main(int argc, char* argv[]) {
try {
   Cell* cell = Cell::load("halfadder");
}
catch ( int& e ) {
  cerr << "[ERROR]␣code:␣" << e << endl;
  exit(1);
}
catch ( Error& e ) {
  cerr << "[ERROR]␣" << e.what() << endl;
  exit(1);
}
catch ( ... ) {
  cerr << "[ERROR]␣Dans␣quel␣etat␣j'erre." << endl;
  exit(1);
}
```