QSqlTableModel Interactions Guide

Core Classes and Basic Operations

Introduction

This document explains QSqlTableModel's interactions with other essential Qt classes for basic CRUD operations (Create, Read, Update, Delete).

1 Core Interaction Classes

Class	Purpose
QTableView	Displays model data in tabular form
QSqlDatabase	Manages database connections
QSqlQuery	Executes raw SQL commands
QSqlRecord	Represents a single database record
QDataWidgetMapper	Links widgets to specific fields
QItemSelectionModel	Handles row/column selection

2 Basic Operations

2.1 Setup (Read)

```
// 1. Create model
QSqlTableModel *model = new QSqlTableModel(this);
model->setTable("players");  // Table name
model->setEditStrategy(QSqlTableModel::OnManualSubmit);

// 2. Connect to view
QTableView *view = new QTableView;
view->setModel(model);

// 3. Load data
model->select();
```

2.2 Adding Records (Create)

```
// Method 1: Using insertRow()
int newRow = model->rowCount();
model->insertRow(newRow);
model->setData(model->index(newRow, 1), "Messi"); // Name
model->setData(model->index(newRow, 2), "FW");    // Position
model->submitAll();  // Save to database

// Method 2: Using QSqlRecord
QSqlRecord record = model->record();
record.setValue("name", "Ronaldo");
record.setValue("position", "FW");
model->insertRecord(-1, record);
```

2.3 Editing Records (Update)

```
// Get selected row
QModelIndex idx = view->currentIndex();

// Update data
model->setData(model->index(idx.row(), 2), "CAM"); // Change
    position
model->submitAll();

// Alternative: Direct SQL
QSqlQuery query;
query.exec("UPDATE players SET position='GK' WHERE id=1");
model->select();  // Refresh model
```

2.4 Deleting Records

```
// Delete selected row
model->removeRow(view->currentIndex().row());
model->submitAll();

// With confirmation
if (QMessageBox::Yes == QMessageBox::question(this, "Confirm"
    ,
     "Delete player?")) {
     model->removeRow(...);
}
```

3 Key Interaction Patterns

- Model-View:
  - `view->setModel(model)` creates binding
  - Changes in model auto-update view (if `select()` called)
- Database Sync:
  - `submitAll()` writes changes to database
  - `revertAll()` discards pending changes
- Selection Handling:
  - `view->selectionModel()->selectedRows()` gets selected items
  - Connect to `clicked()` signals for action triggers

Best Practices

- Always check `lastError()` after operations
- Use `OnManualSubmit` strategy for better control
- For complex relations, consider `QSqlRelationalTableModel`
- Remember to `model->select()` after external changes