[**(一) Qt Model/View 的简单说明**](http://qimo601.iteye.com/blog/1534309)

**博客分类：**

* [Qt](http://qimo601.iteye.com/category/189878)

[Qt](http://www.iteye.com/blogs/tag/Qt)[表格](http://www.iteye.com/blogs/tag/%E8%A1%A8%E6%A0%BC)[Model/View](http://www.iteye.com/blogs/tag/Model%2FView)[模型/视图](http://www.iteye.com/blogs/tag/%E6%A8%A1%E5%9E%8B%2F%E8%A7%86%E5%9B%BE)

**目录：**

[(一) Qt Model/View 的简单说明](http://qimo601.iteye.com/blog/1534309) 、[预定义模型](http://qimo601.iteye.com/blog/1534309)

[(二)使用预定义模型 QstringListModel例子](http://qimo601.iteye.com/blog/1534322)

[(三)使用预定义模型QDirModel的例子](http://qimo601.iteye.com/blog/1534325)

[(四)Qt实现自定义模型基于QAbstractTableModel](http://qimo601.iteye.com/blog/1534331)

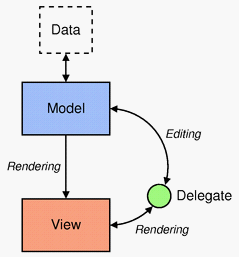
([五)Qt实现自定义模型基于QAbstractItemModel](http://qimo601.iteye.com/blog/1535292)

[(六) Qt实现自定义委托 QDelegate](http://qimo601.iteye.com/blog/1536444)

**（一）使用Qt Model/View的好处**

    1、利用 Qt 的模型 / 视图架构，我们可以只从模型中获取实际在视图中显示所需要的数据。这样在处理非常大的数据集的时候，可以更加快速，而不至于降低性能。通过把一个模型注册到两个或多个视图，就可以让用户使用不同的方式查看数据以及和数据交互。 Qt 对于多个视图会自动保持同步，从而使对一个视图的改变会影响到全部视图。

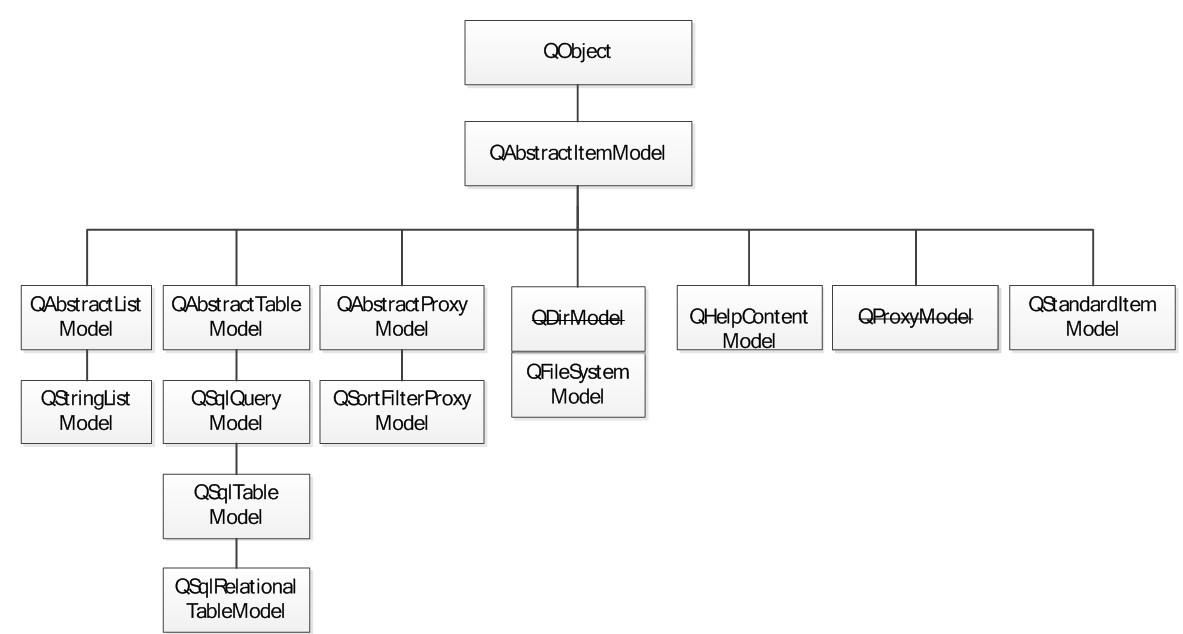
    2、模型 / 视图架构的另外一个好处是：如果决定改变底层数据集的存储方式，只需要修改模型，而视图仍能够继续正常工作。



**（二）预定义模型**

     为了充分使用Qt 模型 / 视图的好处，我们不想再采取 Qt 以前版本中的**项视图简便类** QListWidget、 QTableWidget、 QTreeWidget。它们需要读取要被设置到一个项视图窗口部件中的所有数据、使用项对象呈现数据元素并且回写数据源。

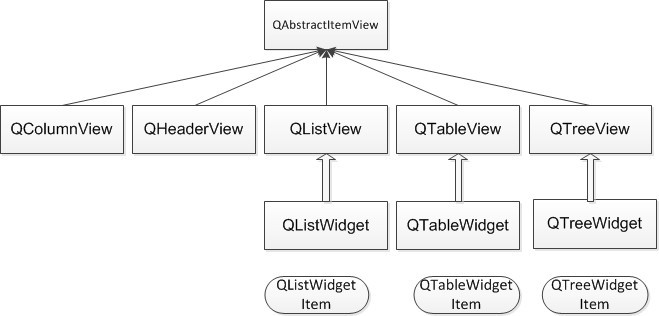
    Qt4使用预定义模型 划线的是Qt4.7 中已经不再推介的 Model 类



|  |  |
| --- | --- |
| QStringListModel | 用于存储简单的字符串列表。 |
| QStandardItemModel | 存储任意的分层次的数据，管理复杂的树型结构数据项，每项都可以包含任意数据。 |
| QDirModel(最新版已经用QfileSystemModel代替这个类了) | 封装本地文件系统 |
| QSqlQueryModel | 封装一个SQL数据集 |
| QSqlTableModel | 封装一个SQL表 |
| QSqlRelationTableModel | 利用外键封装一个SQL表 |
| QSortFileterProxyModel | 排序和/或筛选另一个模型 |
| 假如这些标准Model不满足你的需要，你应该子类化QAbstractItemModel,QAbstractListModel或是QAbstractTableModel来定制。 |  |

**（三）视图**

Qt中视图类的继承关系如下图



**（三）委托Delegate**

Qt中委托的继承关系见下图

