

# Image Beautify 模块报告之 View 层

报告人：黄文璨

报告日期：2018 年 7 月 23 日

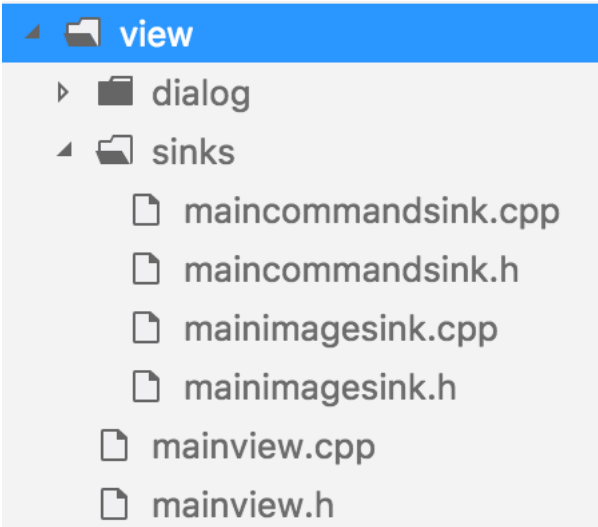
## 一、MVVM 模式中 View 层简介

MVVM 架构使得 View 层与 Model 层完全实现解耦合，View 层专门负责 UI 界面的显示和用户交互逻辑，可以是控件容器和集合。View 与 ViewModel 之间通过 Data Binding 进行数据绑定，通过 Commands 绑定操作的调用，当属性改变或命令执行时 ViewModel 向 View 发出通知 (Send Notifications)。由此可以发现，View 层的主要工作有以下三个部分。

- 1、设计用户界面并实现用户交互，界面显示所需的数据、用户交互发出的命令都与 ViewModel 层进行绑定。
- 2、实现属性通知的接收器，当 ViewModel 发出属性改变的通知时调用 View 层暴露的接口实时更新界面显示。
- 3、实现命令通知的接收器，当 ViewModel 发出命令执行情况的通知时调用 View 层暴露的接口实时进行反馈与更新界面显示。

## 二、本项目中的 View 层实现

### 2.1 View 层概览

View 层文件结构	描述
	在 ImageBeautify 项目中，View 层有一个主窗口类 mainview，另外还有许多子窗口类在 dialog 文件夹中，并且如上一章所述需要实现两个接收器(sink)，分别用于接收属性改变的通知 (mainimagesink) 和命令执行情况的通知 (maincommandsink)。

<div><div>view</div><div><div>dialog</div><div><div>algebraicdialog.cpp</div><div>algebraicdialog.h</div><div>binarymorphologydialog.cpp</div><div>binarymorphologydialog.h</div><div>classicspecialeffectdialog.cpp</div><div>classicspecialeffectdialog.h</div><div>clickableqlabel.cpp</div><div>clickableqlabel.h</div><div>clipboard.cpp</div><div>clipboard.h</div></div></div></div>	<p>左图为本项目中 View 层实现的部分子窗口类，都放在 dialog 文件夹中。</p>
---	---

2.2 View 与 ViewModel 间的数据绑定与命令绑定

属性绑定 (data binding)	
View	ViewModel
View 中的数据指针 shared_ptr	ViewModel 中对应的数据指针
<pre>std::shared_ptr&lt;QImage&gt; image; std::shared_ptr&lt;QImage&gt; subimage; // view log property std::shared_ptr&lt;bool&gt; undoEnabled; std::shared_ptr&lt;bool&gt; redoEnabled; std::shared_ptr&lt;QString&gt; undoMsg; std::shared_ptr&lt;QString&gt; redoMsg; // image property std::shared_ptr&lt;bool&gt; isBinary; std::shared_ptr&lt;bool&gt; isGray;</pre>	<pre>private:     std::shared_ptr&lt;QImage&gt; image;     std::shared_ptr&lt;QImage&gt; subimage;     std::shared_ptr&lt;Model&gt; model;      // view log property     std::shared_ptr&lt;bool&gt; undoEnabled;     std::shared_ptr&lt;bool&gt; redoEnabled;     std::shared_ptr&lt;QString&gt; undoMsg;     std::shared_ptr&lt;QString&gt; redoMsg;      // image property     std::shared_ptr&lt;bool&gt; isBinary;     std::shared_ptr&lt;bool&gt; isGray;</pre>
View 设置数据绑定入口函数	ViewModel 将属性暴露出来
<pre>void setImage(std::shared_ptr&lt;QImage&gt;); void setSubImage(std::shared_ptr&lt;QImage&gt;); void setUndoEnabled(std::shared_ptr&lt;bool&gt;); void setRedoEnabled(std::shared_ptr&lt;bool&gt;); void setUndoMsg(std::shared_ptr&lt;QString&gt;); void setRedoMsg(std::shared_ptr&lt;QString&gt;); void setIsBinary(std::shared_ptr&lt;bool&gt;); void setIsGray(std::shared_ptr&lt;bool&gt;);</pre>	<pre>std::shared_ptr&lt;QImage&gt; getImage(); std::shared_ptr&lt;QImage&gt; getSubImage(); std::shared_ptr&lt;bool&gt; getUndoEnabled(); std::shared_ptr&lt;bool&gt; getRedoEnabled(); std::shared_ptr&lt;QString&gt; getUndoMsg(); std::shared_ptr&lt;QString&gt; getRedoMsg(); std::shared_ptr&lt;bool&gt; getIsBinary(); std::shared_ptr&lt;bool&gt; getIsGray();</pre>

命令绑定(commands binding)	
View	ViewModel
View 中的命令指针（部分），类型为命令基类	ViewModel 中对应的数据指针（部分）
<pre>std::shared_ptr&lt;ICommandBase&gt; openFileCommand; std::shared_ptr&lt;ICommandBase&gt; saveFileCommand; std::shared_ptr&lt;ICommandBase&gt; openSubDialogCommand; std::shared_ptr&lt;ICommandBase&gt; openSubFileCommand; std::shared_ptr&lt;ICommandBase&gt; undoCommand; std::shared_ptr&lt;ICommandBase&gt; redoCommand;</pre>	<pre>std::shared_ptr&lt;OpenFileCommand&gt; openfilecommand; std::shared_ptr&lt;SaveFileCommand&gt; savefilecommand; std::shared_ptr&lt;OpenSubFileCommand&gt; opensubfilecommand; std::shared_ptr&lt;OpenSubDialogCommand&gt; opensubdialogcommand; std::shared_ptr&lt;UndoCommand&gt; undocommand; std::shared_ptr&lt;RedoCommand&gt; redocommand;</pre>
View 设置命令绑定入口函数（部分）	ViewModel 将属性（部分）暴露出来
<pre>void setOpenFileCommand(std::shared_ptr&lt;ICommandBase&gt;); void setSaveFileCommand(std::shared_ptr&lt;ICommandBase&gt;); void setOpenSubDialogCommand(std::shared_ptr&lt;ICommandBase&gt;); void setOpenSubFileCommand(std::shared_ptr&lt;ICommandBase&gt;); void setUndoCommand(std::shared_ptr&lt;ICommandBase&gt;); void setRedoCommand(std::shared_ptr&lt;ICommandBase&gt;);</pre>	<pre>std::shared_ptr&lt;ICommandBase&gt; getOpenFileCommand(); std::shared_ptr&lt;ICommandBase&gt; getSaveFileCommand(); std::shared_ptr&lt;ICommandBase&gt; getOpenSubFileCommand(); std::shared_ptr&lt;ICommandBase&gt; getOpenSubDialogCommand(); std::shared_ptr&lt;ICommandBase&gt; getUndoCommand(); std::shared_ptr&lt;ICommandBase&gt; getRedoCommand();</pre>

## 2.3 View 层属性通知接收器

View 层的属性通知接收器负责响应 ViewModel 层发出的属性改变的通知并及时更新视图。接收器类与 View 通过对象指针的聚合实现相互调用。

```
class MainImageSink : public IPropertyNotification
{
public:
    MainImageSink(MainView *mainview);
    virtual void OnPropertyChanged(const propertyType pt);

private:
    MainView * mainview;
};
```

MainImageSink 需要实现基类中的 OnPropertyChanged 函数以响应属性的改变，具体实现时需要按属性类型分类调用 View 层的接口。

```
void MainImageSink::OnPropertyChanged(const propertyType pt)
{
    if(pt == MAIN_IMAGE){
        mainview->update();
    } else if(pt == SUB_IMAGE){
        mainview->updateSubImage();
    } else if(pt == LOG){
        mainview->updateLogManager();
    }
}
```

View 层通过将消息接收器暴露出去，并挂载到 ViewModel 的通知代理函数中，实现完整的通知与消息接收信号链。

View	ViewModel
View 中的私有信号接收器	ViewModel 继承了通知代理基类
<pre>std::shared_ptr&lt;MainImageSink&gt; mainViewSink; std::shared_ptr&lt;MainCommandSink&gt; mainCommandSink;</pre>	<pre>class ViewModel {     : public Proxy_PropertyNotification&lt;ViewModel&gt;     , public Proxy_CommandNotification&lt;ViewModel&gt; };</pre>
View 暴露接收器的出口函数	APP 层中将接收器挂载到 ViewModel 的通知代理中
<pre>std::shared_ptr&lt;IPropertyNotification&gt; getMainViewSink(); std::shared_ptr&lt;ICommandNotification&gt; getMainCommandSink();</pre>	<pre>viewModel-&gt;AddPropertyNotification(view-&gt;getMainViewSink()); viewModel-&gt;AddCommandNotification(view-&gt;getMainCommandSink());</pre>

## 2.4 View 层命令通知接收器

与属性通知接收器类似，View 层的命令通知接收器负责响应 ViewModel 层发出的命令完成结果的通知并及时更新视图状态。接收器类与 View 通过对象指针的聚合实现相互调用。

```
class MainCommandSink : public ICommandNotification  
{  
public:  
    MainCommandSink(MainView *mainview);  
    virtual void OnCommandComplete(const commandsType cmd, bool  
OK);  
  
private:  
    MainView * mainview;  
};
```

MainCommandSink 需要实现基类中的 OnCommandComplete 函数以响应命令执行完成的通知，具体实现时需要按命令类型分类调用 View 层的接口。

```
void MainCommandSink::OnCommandComplete(const commandsType cmd, bool  
OK){  
    switch (cmd) {  
    case OPEN_FILE:  
        if(OK);  
        else mainview->HandleOpenFileException();  
        break;  
  
    (此处省略若干条命令类型)  
  
    default:  
        break;  
    }  
}
```

View 层通过将命令接收器暴露出去，并挂载到 ViewModel 的通知代理函数中，实现完整的通知与消息接收信号链。

View	ViewModel
View 中的私有信号接收器	ViewModel 继承了通知代理基类
<pre>std::shared_ptr&lt;MainImageSink&gt; mainViewSink; std::shared_ptr&lt;MainCommandSink&gt; mainCommandSink;</pre>	<pre>class ViewModel {     : public Proxy_PropertyNotification&lt;ViewModel&gt;     , public Proxy_CommandNotification&lt;ViewModel&gt; }</pre>
View 暴露接收器的出口函数	APP 层中将接收器挂载到 ViewModel 的通知代理中
<pre>std::shared_ptr&lt;IPropertyNotification&gt; getMainViewSink(); std::shared_ptr&lt;ICommandNotification&gt; getMainCommandSink();</pre>	<pre>viewModel-&gt;AddPropertyNotification(view-&gt;getMainViewSink()); viewModel-&gt;AddCommandNotification(view-&gt;getMainCommandSink());</pre>