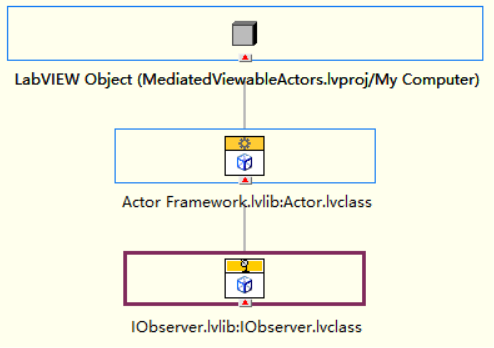
|  |
| --- |
| [键入公司名称] |
| MVA Framework |
| [键入文档副标题] |
|  |
| **周维** |
| **[选取日期]** |

|  |
| --- |
| [在此处键入文档摘要。摘要通常为文档内容的简短概括。在此处键入文档摘要。摘要通常为文档内容的简短概括。] |

# 继承至Actor类说明

* 1. IObserver：Actor

观察者定义了一个用于自动处理基于事件的消息的接口



### 重写AF

#### Pre Launch Init.vi

调用Obtain Unique ID.vi，获取 Actor 队列 DVR 引用唯一 ID，存入UID

### 1.1.2静态方法

#### Obtain Unique ID.vi

获取 Actor 队列 DVR 引用唯一 ID

#### Close Framework Event Log.vi

关闭框架事件日志

#### Configure Observer Logging.vi

配置观察者日志记录

#### Log Framework Event.vi

日志框架事件

#### Read Subscribe Msg.vi

读取订阅消息

#### Read Unique ID.vi

读取 Actor 队列 DVR 引用唯一 ID

#### Read Unsubscribe Msg.vi

读取取消订阅消息

#### Transfer Observer Resources.vi

转移观察员资源

#### Write Pre-Addressed Observer Msgs.vi

写入预先寻址的观察者消息

#### Request And Wait.vi

请求并等待

#### Request Data Persistence.vi

请求数据保持

#### Request Data Persistence.vim

请求数据保持

#### Request Data.vi

请求数据

#### Request Data.vim

请求数据

#### Request Message.vi

请求消息

#### Request Message.vi

请求消息

#### Request Multiple Data.vi

请求多个数据

#### Request Multiple Data.vim

请求多个数据

#### Request Synchronous Data.vim

请求同步数据

#### Request Value Reference Updates.vi

请求值引用更新

#### Request Value Reference Updates.vim

请求值引用更新

### 1.1.3动态方法

#### Observe Data.vi

观察数据

#### Subscribe.vi

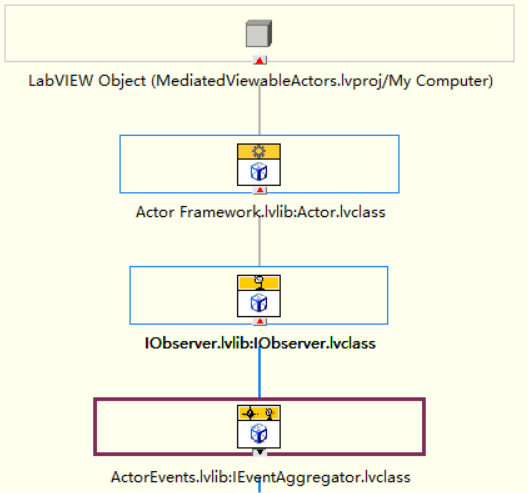
订阅

#### Unsubscribe.vi

取消订阅

* 1. IEventAggregator：IObserver

事件聚合器定义了一个用于自动处理基于事件的消息的接口



### 1.2.1重写AF

#### Pre Launch Init.vi

调用Send Launch Event Handlers.vi发送启动事件处理程序

### 1.2.2静态方法

#### Write Event Handler.vi

写入事件处理程序

#### Register For Control Events.vi

注册控件事件

#### Register For Listbox Events.vi

注册列表框事件

#### Register For MulticolumnListbox Events.vi

注册多列列表框事件

#### Register For Pane Events.vi

注册窗格事件

#### Register For Tree Events.vi

注册树形事件

#### Register For VI Events.vi

注册 VI 事件

### 1.2.3动态方法

#### Receive Control Drag Ended Event.vi

抽象方法

接收控件拖拖拽结束事件

#### Receive Control Drag Over Event.vi

抽象方法

接收控件拖拽经过事件

#### Receive Control Drag Source Update Event.vi

抽象方法

接收控件拖拽源更新事件

#### Receive Control Drag Starting Event.vi

抽象方法

接收控件拖拽开始事件

#### Receive Control Drop Event.vi

抽象方法

接收控件放置事件

#### Receive Control Key Down Event.vi

抽象方法

接收控件键按下事件

#### Receive Control Key Up Event.vi

抽象方法

接收控件键释放事件

#### Receive Control Mouse Down Event.vi

抽象方法

接收控件鼠标按下事件

#### Receive Control Mouse Enter Event.vi

抽象方法

接收控件鼠标进入事件

#### Receive Control Mouse Leave Event.vi

抽象方法

接收控件鼠标离开事件

#### Receive Control Mouse Up Event.vi

抽象方法

接收控件鼠标释放事件

#### Receive Listbox Double Click.vi

抽象方法

接收列表框双击事件

#### Receive Listbox Edit Cell.vi

抽象方法

接收列表框编辑单元格事件

#### Receive Menu Selection Event.vi

接收菜单选择事件

#### Receive Panel Close Event.vi

接收面板关闭事件

#### Receive Shortcut Menu Activation Event.vi

接收快捷菜单激活事件

#### Receive Shortcut Menu Selection Event.vi

接收快捷菜单选择事件

#### Receive Tree Double Click.vi

接收树形双击

#### Receive Tree Edit Cell Event.vi

接收树形编辑单元格事件

#### Receive Tree Item Close Event.vi

接收树形项目关闭事件

#### Receive Tree Item Open Event.vi

接收树形项目打开事件

#### Receive Value Change Event.vi

接收值改变事件

#### Receive VI Key Down Event.vi

抽象方法

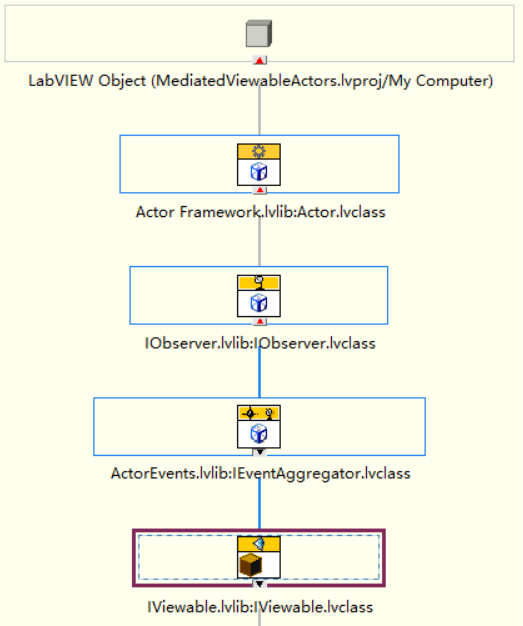
接收 VI 键按下事件

#### Launch Event Handlers.vi

启动事件处理程序

* 1. IViewable：IEventAggregator

可查看的父接口类和默认实现合二为一！可查看的Actor可以插入到子面板中。它们还具有其他几种与用户界面特别相关的能力。



### 1.3.1重写AF

##### Actor Core.vi

调用Store Actor Core Ref获取视图VI引用；

调用Initiate Updates初始化更新UI；

注册视图窗格大小改变事件，调用Send Resize Panel to Pane多个事件将引发多个事件前面板。只向发送一条消息任何UI大小变化的核心，让Viewable决定需要调整大小的内容。

### 1.3.2静态方法

#### Bind Boolean Text.vi

将按钮的布尔文本绑定到已发布的值。

您必须提供对绑定数据有效的格式字符串。

#### Bind Control Caption.vi

将控制终端（控件或指示器）的标题绑定到已发布的字符串。

#### Bind Formatted String.vi

将字符串控制终端（控件或指示器）绑定到已发布的值。

您必须提供对指定数据类型有效的格式字符串。

#### Bind Listbox Item Names.vi

将单列列表框的项目名称绑定到已发布的字符串数组。

#### Bind MCListbox Item Names.vi

将多列列表框的项目名称绑定到已发布的二维字符串数组。

#### Bind Multiple Terminal Values.vi

将多个控制终端值（控件或指示器）绑定到已发布的数据，其名称和类型与有线终端引用的名称和类型匹配。

#### Bind Named Numeric Strings.vi

将名称数字控制终端的字符串绑定到已发布的字符串数组。

#### Bind Terminal Enabled State.vi

将控制终端（控件或指示器）的启用状态绑定到已发布的布尔或数字数据。

通过连接可选的输入端子，指定基准值和端子状态之间的逻辑关系。

布尔值的默认值：

已发布TRUE=已启用

发布FALSE=禁用并变灰

数字的默认值：

公开值大于0=已启用

发布值不大于0=禁用并变灰

#### Bind Terminal Value.vi

将控制终端（控件或指示器）的值绑定到发布。

#### Bind Terminal Visibility.vi

将控制终端（控件或指示器）的可见状态绑定到已发布的布尔或数字数据。

通过连接可选输入端子，指定基准值和端子可见性之间的逻辑关系。

布尔值的默认值：

已发布TRUE=可见

已发布FALSE=隐藏

数字的默认值：

公开值大于0=可见

已发布值不大于0=隐藏

#### Write Application Style Reference.vi

写入应用程序样式引用

#### Read Application Style Reference.vi

读取应用程序样式引用

#### Set Custom Window Size.vi

设置自定义窗口大小

#### Transfer Viewable Resources.vi

转移可查看资源

#### Evaluate Destination Subpanel.vi

评估目标子面板

#### Read Insertion Info.vi

读取插入信息

### 1.3.3动态方法

#### Catch Nested Value Update.vi

捕获嵌套值更新

#### Close Front Panel.vi

关闭前面板

#### Defer Panel Updates.vi

延迟面板更新

#### Initialize Front Panel.vi

初始化前面板

#### Insert Front Panel.vi

插入前面板

#### Open Front Panel.vi

打开前面板

#### Removed From Subpanel.vi

从子面板中删除

#### Resize Panel to Pane.vi

抽象方法

将面板调整为窗格

#### Resume Panel Updates.vi

恢复面板更新

#### Set Cursor Busy.vi

设置光标繁忙

#### Set Cursor Unbusy.vi

设置光标无效

#### Set Minimum Panel Size.vi

设置最小面板尺寸

#### Set Panes to Origins.vi

将窗格设置为原点

#### Receive Panel Close Event.vi

接收面板关闭事件

#### Receive Value Change Event.vi

接收值更改事件

#### Launch Event Handlers.vi

启动事件处理程序

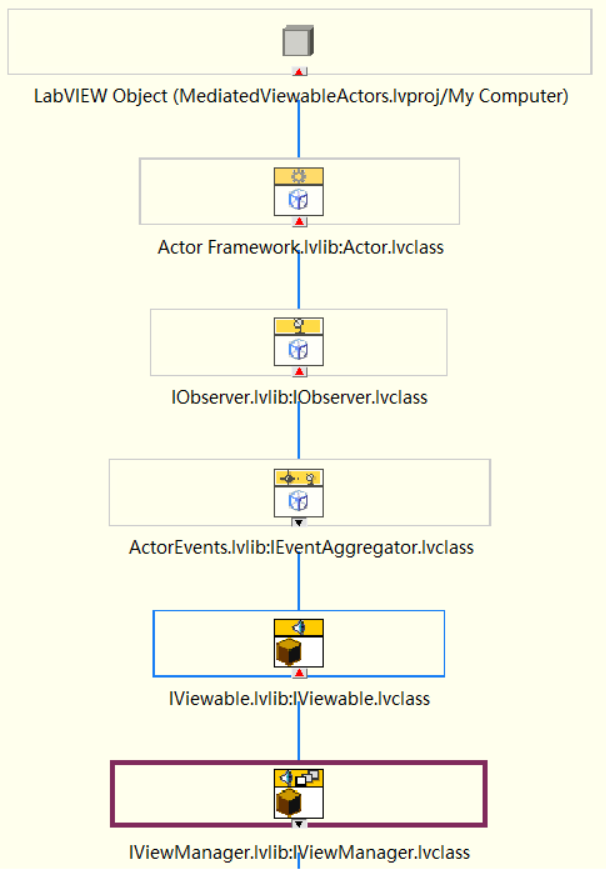
#### Send Value Update.vi

发送值更新

#### Initiate Updates.vi

初始化更新

## IViewManager：IViewable



### 1.4.1重写AF

#### Pre Launch Init.vi

调用父方法

调用Send Launch Nested Views发送启动嵌套视图消息，启动所有视图

### 1.4.2静态方法

#### Read Element of Views.vi

读取视图元素

#### Read Nested Viewable Enqueuers.vi

读取嵌套视图消息队列

#### Write Startup Launch Nested Views Msg.vi

写入启动启动嵌套视图消息

#### Write Startup Views.vi

写入启动视图

### 1.4.3动态方法

#### Write Nested Viewable Enqueuers.vi

写入嵌套视图消息队列

#### Defer Panel Updates.vi

延迟面板更新

#### Resize Panel to Pane.vi

将面板调整为窗格

#### Resume Panel Updates.vi

恢复面板更新

#### Set Cursor Busy.vi

设置光标繁忙

#### Set Cursor Unbusy.vi

设置光标无效

#### Set Panes to Origins.vi

将窗格设置为原点

#### Launch Nested Views.vi

启动嵌套视图

## IViewModel：IViewManager

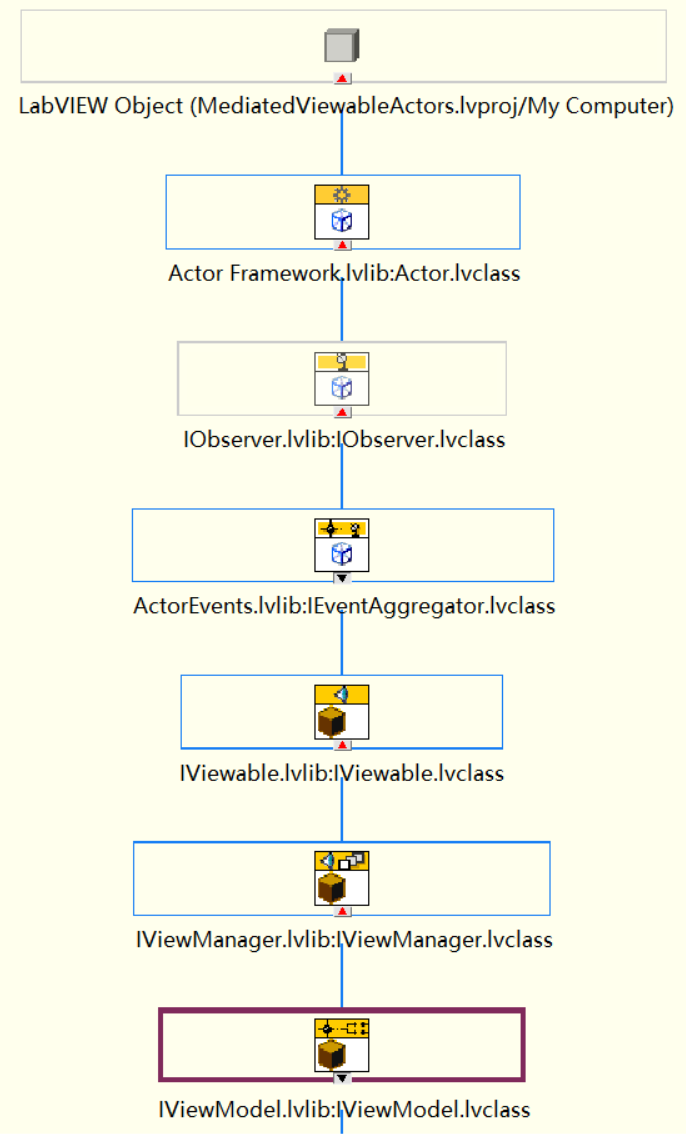
视图模型处理应用程序业务逻辑（模型）和用户界面（视图）之间的关系。请注意，在旧的关注点分离框架中，视图模型被称为“控制器”。在此实现中，其具体职责包括：

-在应用程序启动时启动模型、视图配置和中介

-在“视图”中点击按钮，并将这些事件作为Actor消息传递给模型。

-捕获运行时菜单选择并采取适当的操作，包括修改菜单状态。

-根据模型状态将用户可读状态或指令字符串数据发布到视图（使模型与所有视图关注点保持解耦）



### 1.5.1重写AF

#### Pre Launch Init.vi

调用Send Launch Root Model发送启动根模型消息，启动中介和根模型

调用父方法

#### Actor Core.vi

前面板放置了一个子面板，视图会插入这个子面板

调用New Protected Subpanel新建受保护子面板，把子面板引用存入ProtectedSubpanel类，再把ProtectedSubpanel类存入DVR，DVR自带互斥锁，避免多线程竞争插入子面板

调用Register For VI Events注册VI前面板关闭事件，注册的事件由IEventAggregator的Launch Event Handlers方法启动IEventHandler的子Actor

### 1.5.2静态方法

#### Read Root Model Enqueuer.vi

读取根模型消息队列

#### Read Mediator Enqueuer.vi

阅读中介消息队列

#### Write Subpanel Refnum.vi

写入子面板引用

#### Publish Data.vi

发布数据

#### Write Startup Model-View Configuration.vi

此构造函数仅在公共调用时被弃用。它应该被可扩展的“Construct MVA Application.vim”所取代，它通过向一个或多个接收器添加可选的框架级事件日志来扩展。

Construct MVA Application.vim

构建 MVA 应用程序

### 1.5.3动态方法

#### Receive Panel Close Event.vi

接收面板关闭事件

#### Write Nested Viewable Enqueuers.vi

写入嵌套视图消息队列

#### Catch Nested Value Update.vi

捕获嵌套值更新

#### Initiate Updates.vi

初始化更新

#### Initialize Front Panel.vi

初始化前面板

#### Send Value Update.vi

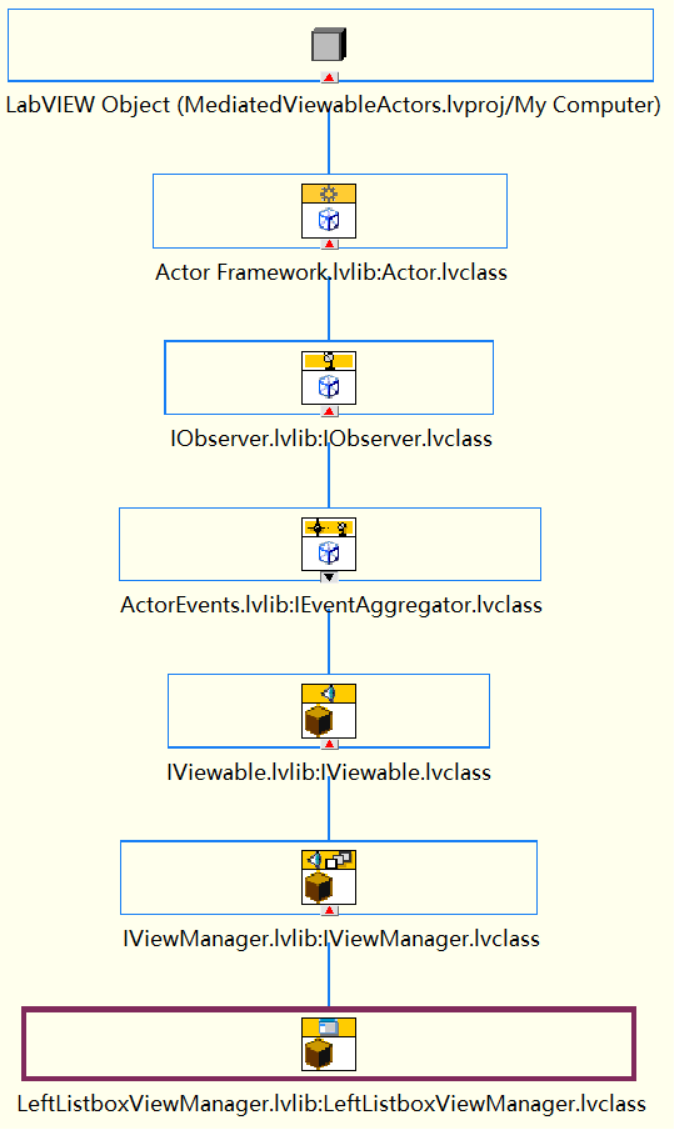
发送值更新

#### Launch Root Model.vi

启动根模型

## LeftListboxViewManager：IViewManager

类似于NI典型配置窗口的视图布局（左侧为列表框，右侧为子面板）



### 1.6.1重写AF

Actor Core.vi

调用New Protected Subpanel新建受保护子面板，把子面板引用存入ProtectedSubpanel类，再把ProtectedSubpanel类存入DVR，DVR自带互斥锁，避免多线程竞争插入子面板

注册了列表框值改变事件，调用Send Set View by Listbox Index发送按列表框索引设置视图，动态把视图插入子面板

### 1.6.2静态方法

#### Assemble LeftListboxViewManager.vi

组装左侧列表框视图管理器

#### Set View by Listbox Index.vi

按列表框索引设置视图

### 1.6.3动态方法

#### Initialize Front Panel.vi

初始化前面板

## QuadViewManager：IViewManager

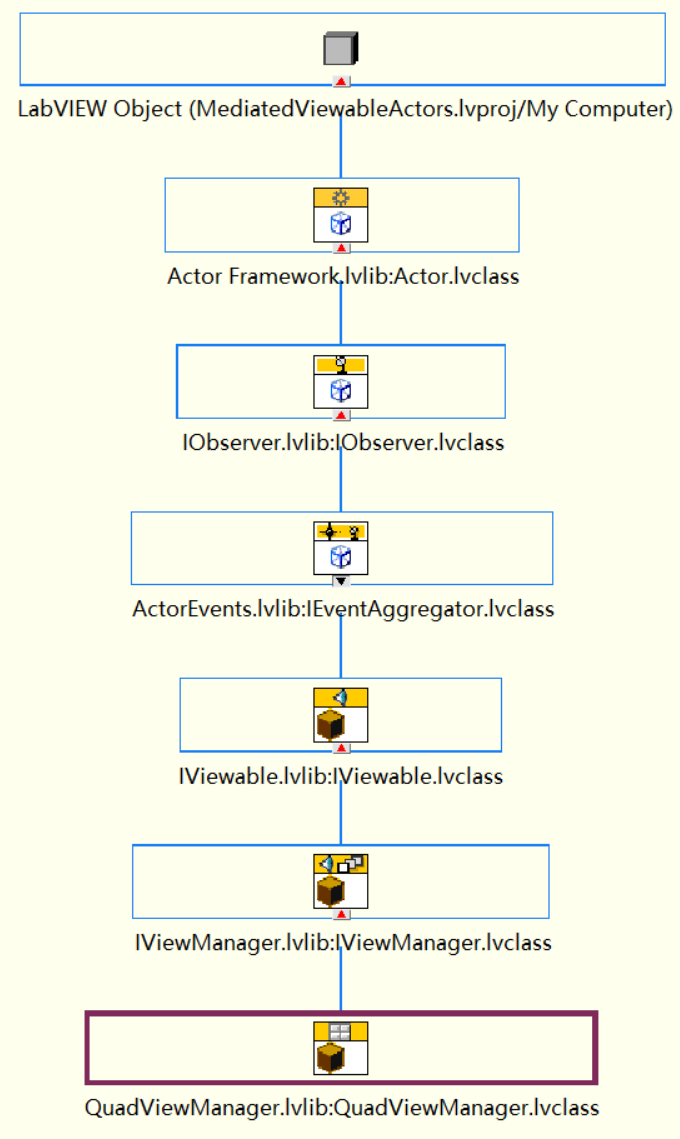
四边形布局包含四个大小相等的自动缩放子面板，顺序如下：

面板1：左上角

面板2：右上角

面板3：左下角

面板4：右下角



### 1.7.1重写AF

Actor Core.vi

前面板放置了四个子面板

调用New Protected Subpanel新建受保护子面板，把子面板引用存入ProtectedSubpanel类，再把ProtectedSubpanel类存入DVR，DVR自带互斥锁，避免多线程竞争插入子面板

### 1.7.2静态方法

Assemble QuadViewManager.vi

组装四视图管理器

### 1.7.3动态方法

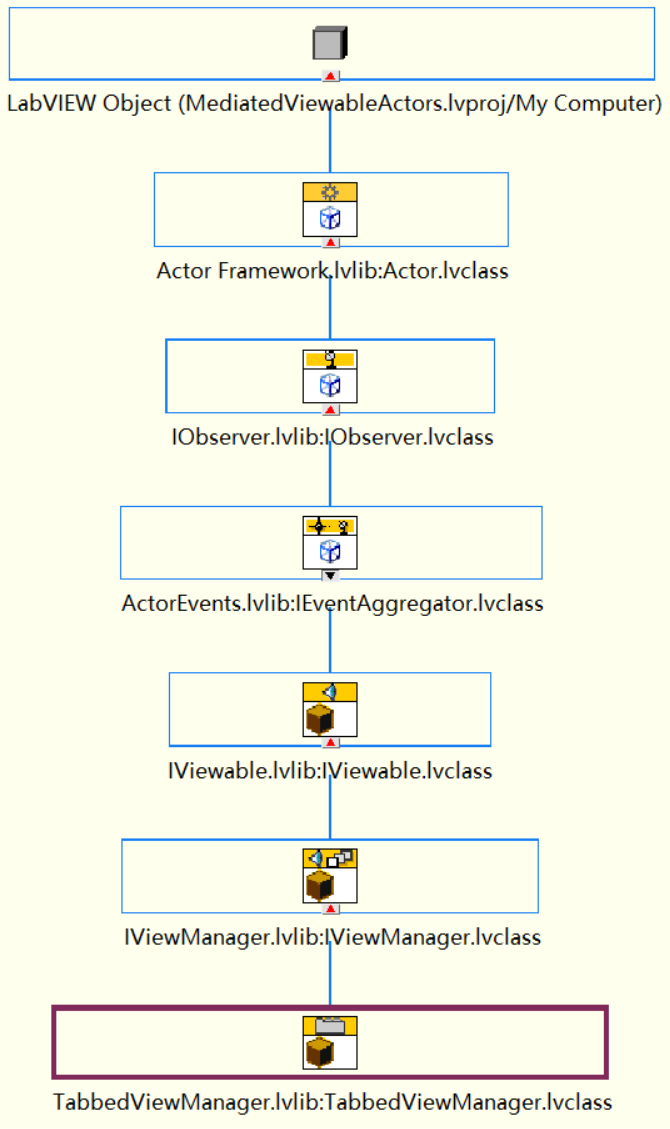
Initialize Front Panel.vi

初始化前面板

## TabbedViewManager：IViewManager

选项卡视图管理器

共有10个选项卡页面



### 1.8.1重写AF

#### Actor Core.vi

调用New Protected Subpanel新建受保护子面板，把子面板引用存入ProtectedSubpanel类，再把ProtectedSubpanel类存入DVR，DVR自带互斥锁，避免多线程竞争插入子面板

注册选项卡值改变事件，调用Send Set View by Tab Index发送按选项卡索引设置视图，动态把视图插入子面板

### 1.8.2静态方法

#### Assemble TabbedViewManager.vi

组装选项卡视图管理器

#### Set View by Tab Index.vi

按选项卡索引设置视图

### 1.8.3动态方法

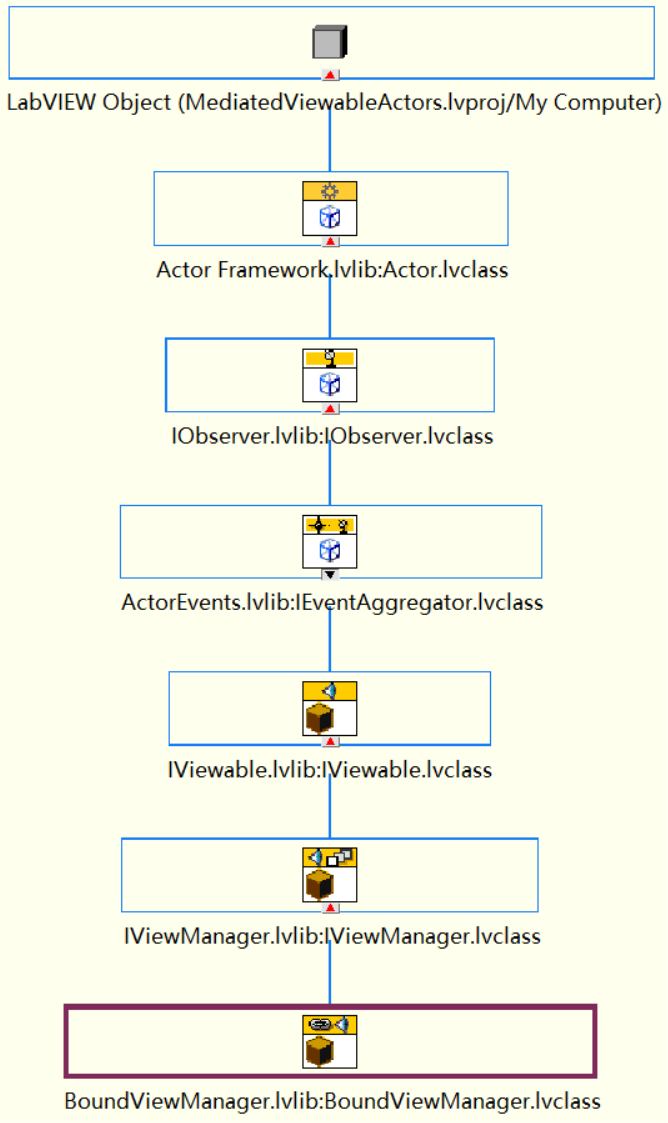
#### Initialize Front Panel.vi

初始化前面板

## 1.9 BoundViewManager：IViewManager

当发布的字符串选择器值与指定给其中一个绑定（嵌套）视图的名称匹配时，BoundViewManager会自动插入一个视图。

绑定关系通常在启动时通过注入Viewable类型及其绑定字符串别名来建立。未知的选择器值将被忽略，使之前的Viewable保持不变。



### 1.9.1重写AF

#### Pre Launch Init.vi

调用父方法

调用Create Lookup Table为动态创建查找表，可查看的队列引用存储

调用New Basic Subscription新建基本订阅或New Value Change Subscription新建值改变订阅

调用Request Message请求发布特定消息，选择字符串的值改变

#### Actor Core.vi

调用New Protected Subpanel新建受保护子面板，把子面板引用存入ProtectedSubpanel类，再把ProtectedSubpanel类存入DVR，DVR自带互斥锁，避免多线程竞争插入子面板

### 1.9.2静态方法

#### Assemble BoundViewManager.vi

装配绑定视图管理器

### 1.9.3动态方法

#### Initialize Front Panel.vi

初始化前面板

#### Insert Bound View.vi

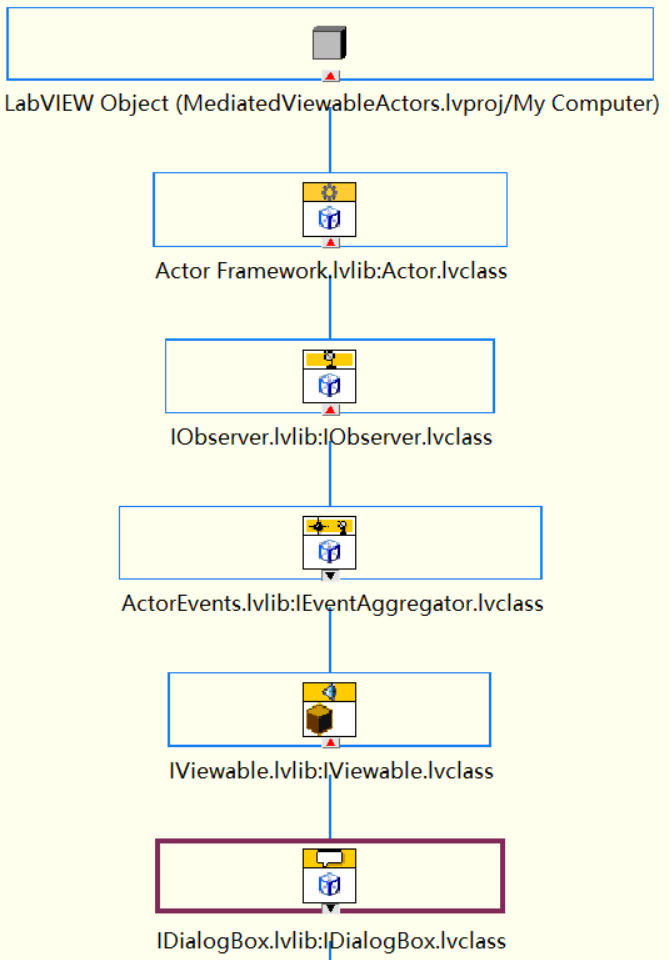
插入绑定视图

#### Write Nested Viewable Enqueuers.vi

写入嵌套视图消息队列

## 1.10 IDialogBox：IViewable

对话框父类



### 1.10.1重写AF

### 1.10.2静态方法

Launch Dialog Box.vi

启动对话框

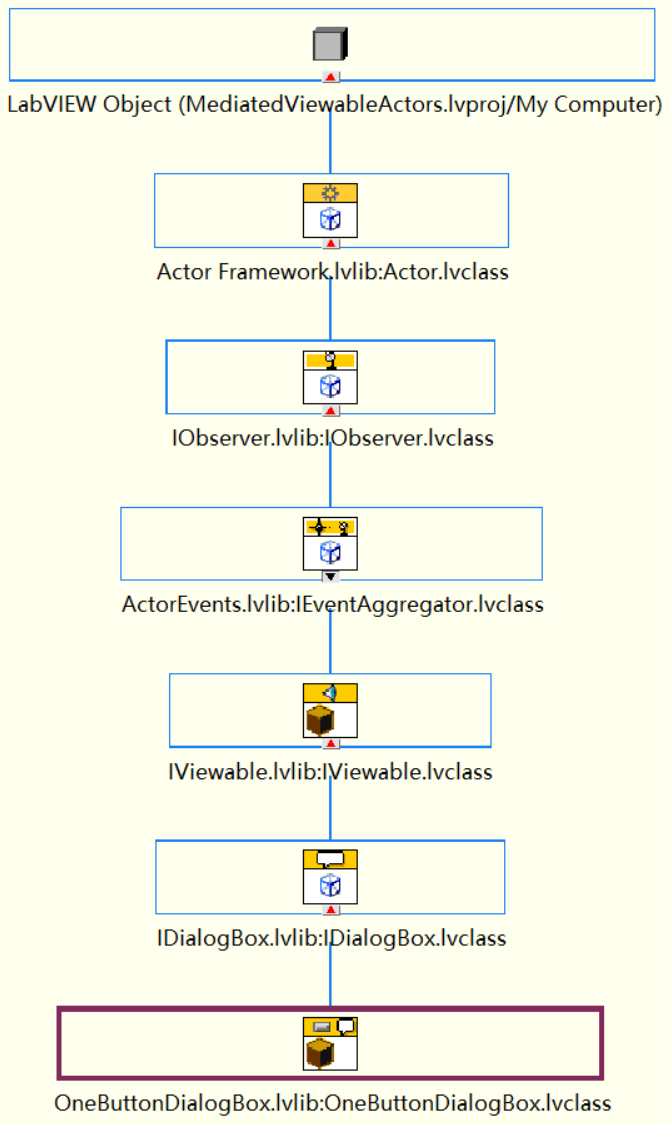
### 1.10.3动态方法

Initiate Updates.vi

初始化更新

## 1.11 OneButtonDialogBox：IViewable

单按钮对话框



### 1.11.1重写AF

#### Actor Core.vi

设置消息文本字符串控件值，确定按钮布尔文本键选中

调用Register For Control Events注册控件值改变事件，注册的事件由IEventAggregator的Launch Event Handlers方法启动IEventHandler的子Actor

### 1.11.2静态方法

#### Launch One-Button Dialog Box.vi

启动单按钮对话框

### 1.11.3动态方法

#### Receive Panel Close Event.vi

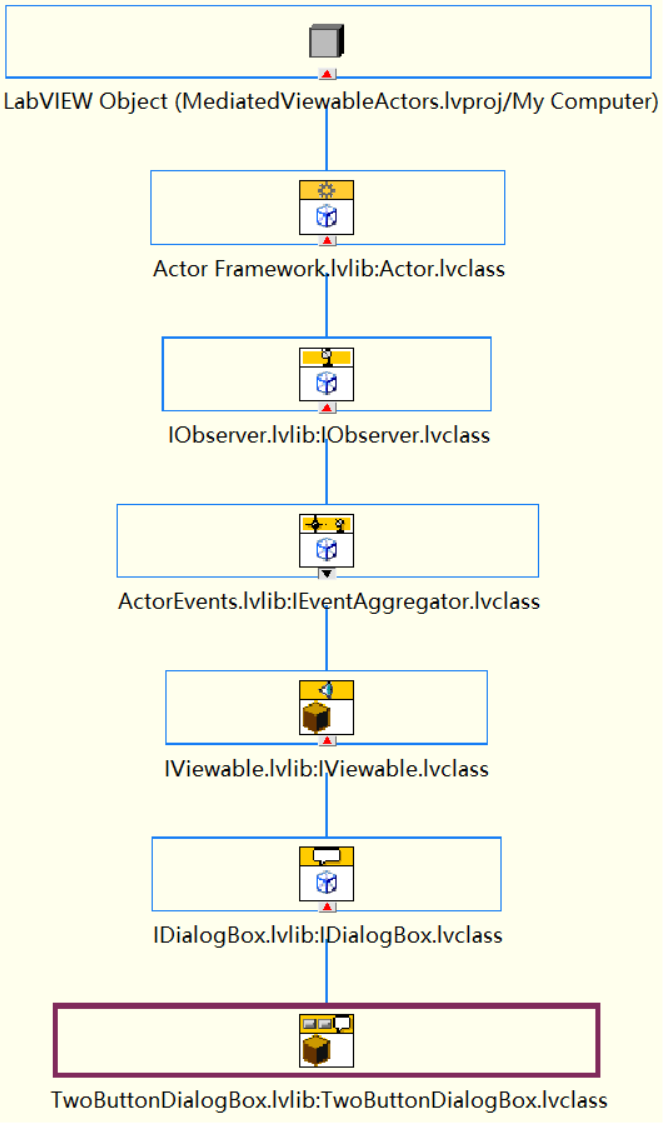
接收面板关闭事件

#### Receive Value Change Event.vi

接收值改变事件

## 1.12 TwoButtonDialogBox：IViewable

双按钮对话框



### 1.12.1重写AF

#### Actor Core.vi

设置消息文本字符串控件值，确定按钮布尔文本键选中

调用Register For Control Events注册确定取消控件值改变事件，注册的事件由IEventAggregator的Launch Event Handlers方法启动IEventHandler的子Actor

### 1.12.2静态方法

#### Launch Two-Button Dialog Box.vi

启动双按钮对话框

### 1.12.3动态方法

#### Receive Panel Close Event.vi

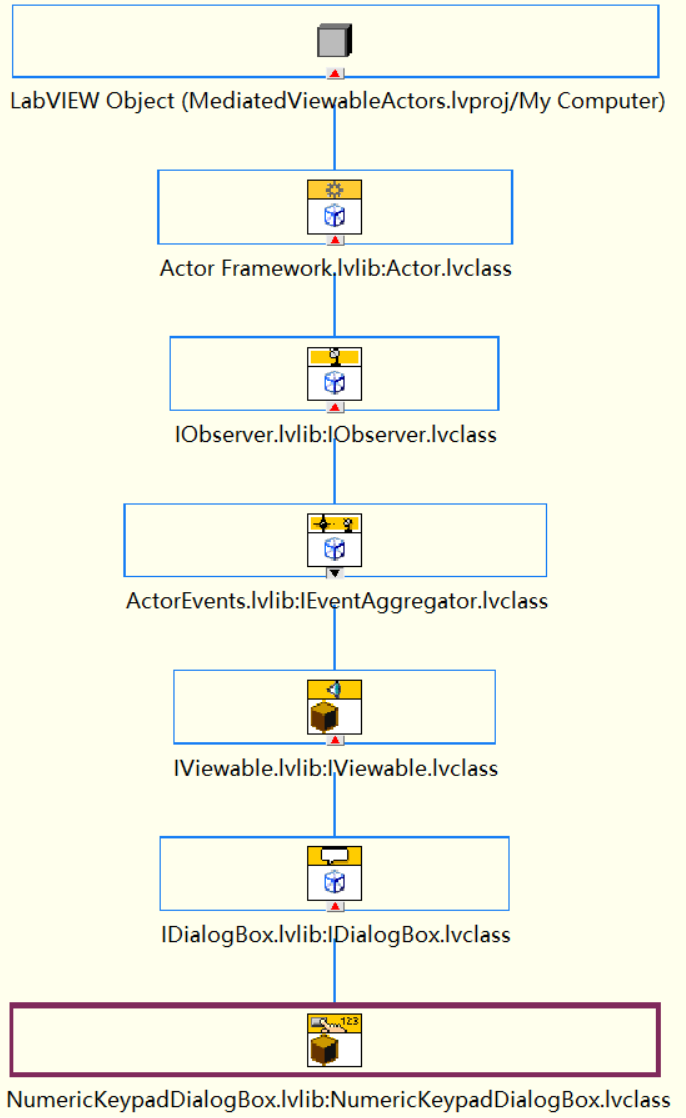
接收面板关闭事件

#### Receive Value Change Event.vi

接收值改变事件

## 1.13 NumericKeypadDialogBox：IViewable

数字键盘对话框



### 1.13.1重写AF

#### Actor Core.vi

设置消息文本字符串控件值

调用Register For Control Events注册所有按钮控件值改变事件，注册的事件由IEventAggregator的Launch Event Handlers方法启动IEventHandler的子Actor

### 1.13.2静态方法

#### Append Character.vi

附加字符

#### Launch Numeric Keypad Dialog Box.vi

启动数字键盘对话框

### 1.13.3动态方法

#### Receive Value Change Event.vi

接收值改变事件

## 1.14 IModel：IObserver

存在实现业务逻辑的模型。它们不应该直接向ViewModel或Viewables发送Actor消息，因为这样做会将它们紧密地耦合到这些类。

必须由Viewables使用/观察（或由ViewModel反应）的模型数据应通过数据中介发布。、

### 1.14.1重写AF

#### Pre Launch Init.vi

调用父方法

调用Send Auto-Launch Models向自己发送启动启动时的嵌套模型消息

### 1.14.2静态方法

#### Remove Nested Model.vi

删除嵌套模型

#### Stop Nested Models.vi

停止嵌套模型

#### Track Nested Model.vi

追踪嵌套模型

#### Defer Message Until Cleaned Up.vi

将消息推迟到清理完毕

#### Read Lock Publication Msg.vi

读取锁定发布消息

#### Read Publish Msg.vi

读取发布消息

#### Read Unlock Publication Msg.vi

读取解锁发布消息

#### Transfer Model Resources.vi

转移模型资源

#### Lock Data.vi

锁定数据

#### Mutexed Increment Data.vi

互斥增量数据

#### Mutexed Merge Errors.vi

互斥合并错误

#### Mutexed Publish Data.vi

互斥发布数据

#### Publish Data.vi

将数据（值）发布到中介总线。任何有权访问中介总线的参与者都可以观察到已发布的数据。

#### Publish If True.vi

仅当布尔值为TRUE时才发布该值。

此发布允许多个参与者或多段代码发布一个标志，该标志在中介总线上保持为真，直到另一个参与者有意处理它。（全局标志设置的简单多对一选项）Unlock Data.vi

解锁数据

#### Model Cleanup Done.vi

模型清理完成

#### Write Auto-Launch Nested Models.vi

写入自动启动嵌套模型

#### Write Pre-Addressed Publisher Messages.vi

写入预先写好地址的发布者消息

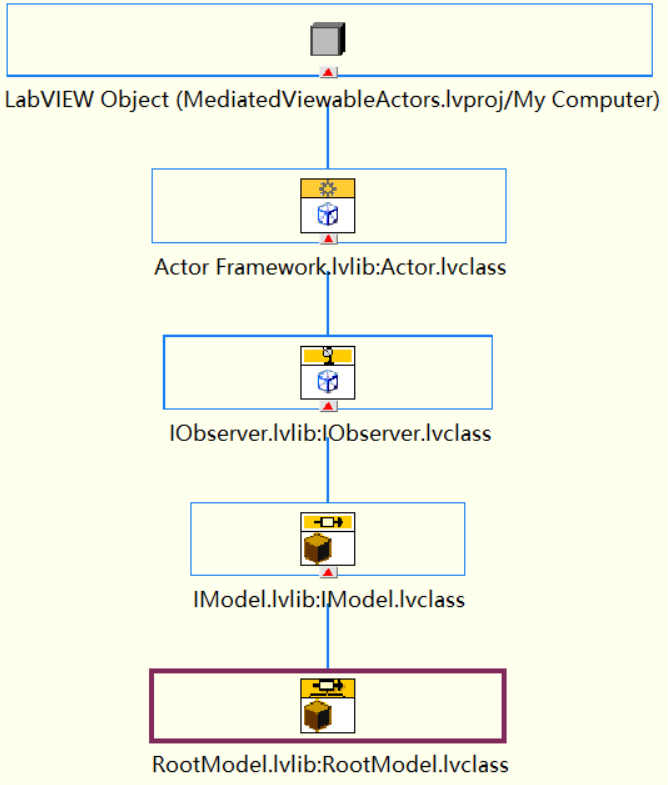
### 1.14.3动态方法

#### Launch Nested Model.vi

启动嵌套模型

## 1.15 RootModel：IModel

根模型



### 1.15.1重写AF

#### Pre Launch Init.vi

判断是否为根模型，发送Startup Run Headless Msg，仅启动中介

### 1.15.2静态方法

#### Write Startup Run Headless Msg.vi

写入启动运行无头消息

#### Construct Headless MVA Application.vim

构建无头MVA应用

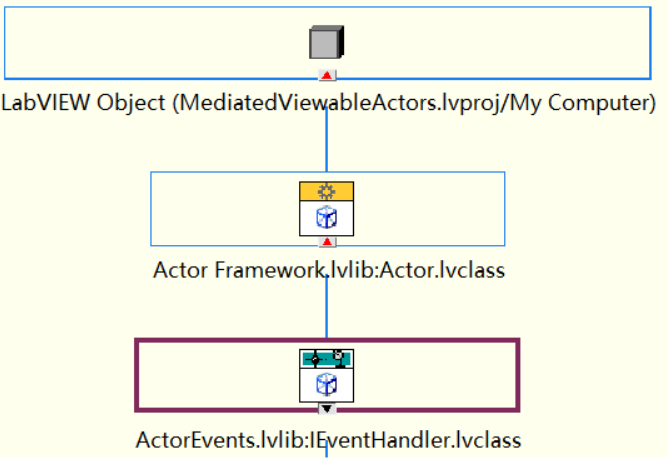
#### Run Headless.vi

运行无头

### 1.15.3动态方法

## 1.16 IEventHandler：Actor

事件处理器父类



### 1.16.1重写AF

#### Actor Core.vi

调用Create Stop Event创建停止事件，停止辅助循环；

调用Register Dynamic Events(抽象方法)注册动态事件，根据启动的具体类调用具体方法；

调用Registration Complete注册完成发送注册完成通知；

调用Handle Events处理事件，根据启动的具体类调用具体方法；

### 1.16.2静态方法

#### Create Stop Event.vi

创建停止事件

#### Registration Complete.vi

注册完成

#### Stop Event Handler.vi

停止事件处理器辅助循环

#### Read Registrations.vi

读取注册信息

#### Read Stop Event.vi

读取停止事件

#### Read Timeout Message Info.vi

读取超时消息信息

#### Write EventFilter Reference.vi

写入事件筛选器引用

#### Write Registration Types.vi

写入注册类型

#### Write Timeout Message.vi

写入超时消息

#### Launch Event Handler.vi

启动事件处理程序

### 1.16.3动态方法

#### Handle Events.vi

处理事件

#### Read Valid Registration Types.vi

读取有效的注册类型

#### Register Dynamic Events.vi

抽象方法

注册动态事件

### 1.16.4静态属性

#### Event Filter @Read

事件筛选器

#### EventAggregator Enqueuer @Read

事件聚合器消息队列

#### Timeout [ms] @Read

超时[ms]

#### Timeout Message @Read

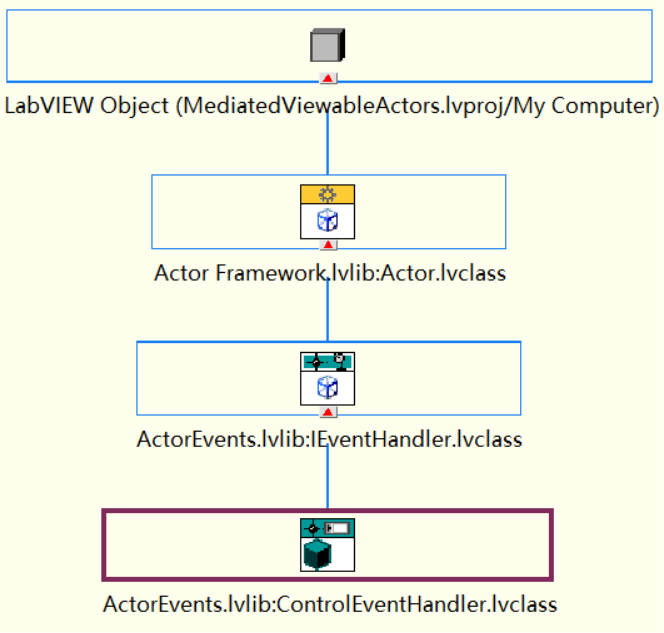
超时消息

#### Timeout Message Priority @Read

时消息优先级

## 1.17ControlEventHandler：IEventHandler

控件事件处理器



### 1.17.1重写AF

### 1.17.2静态方法

#### Read Registration Refnums.vi

读取注册控件引用

#### Unpress Latching Boolean.vi

解除锁定布尔值

#### New ControlEventHandler.vi

新建控件事件处理程序

### 1.17.3动态方法

#### Handle Events.vi

处理事件

#### Read Valid Registration Types.vi

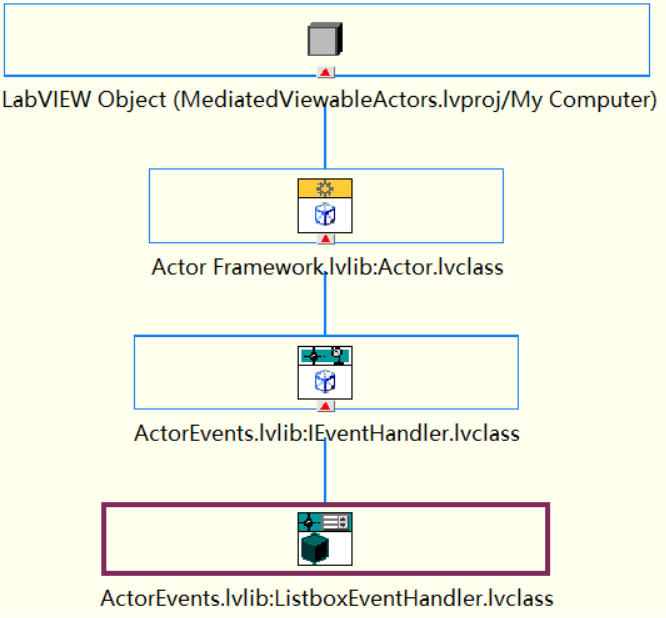
读取有效的注册类型

#### Register Dynamic Events.vi

注册动态事件

## 1.18 ListboxEventHandler：IEventHandler

列表框事件处理器



### 1.18.1重写AF

### 1.18.2静态方法

#### Read Registration Refnum.vi

读取注册控件引用

#### New ListboxEventHandler.vi

新建列表框事件处理程序

### 1.18.3动态方法

#### Handle Events.vi

处理事件

#### Read Valid Registration Types.vi

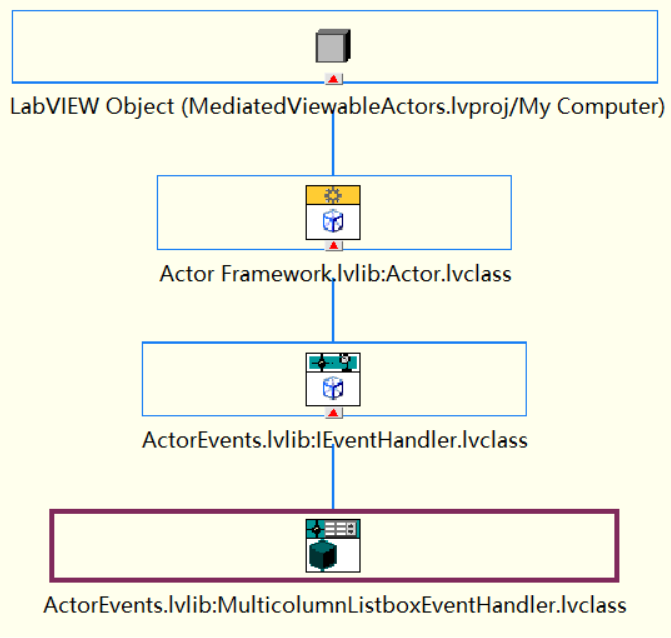
读取有效的注册类型

#### Register Dynamic Events.vi

注册动态事件

## 1.19 MulticolumnListboxEventHandler：IEventHandler

多列列表框事件处理器



### 1.19.1重写AF

### 1.19.2静态方法

#### Read Registration Refnum.vi

读取注册控件引用

#### New MulticolumnListboxEventHandler.vi

新建多列列表框事件处理程序

### 1.19.3动态方法

#### Handle Events.vi

处理事件

#### Read Valid Registration Types.vi

读取有效的注册类型

#### Register Dynamic Events.vi

注册动态事件