1.先想好命令的类型 需要数据就调用Command 改变视图的命令就用Mediator

一般：MonoBehaviour-Command

MonoBehaviour-Mediator

MonoBehaviour- Command- Mediator

MonoBehaviour- Mediator-Command

Command- Mediator

Mediator-Command

Ps: Destroy()和Instantiate只在MonoBehaviour中可以生效，注意消息传递

2.在NotificationConstant中编写消息名（注意不要重名，易读，不准用拼音，首字母大写，举例：JuageMonsterDistance 调用名和参数名一致），放到相应的消息类别中

3.如果是Command命令：

i 在RunFacade中注册Command消息，放到相对应的消息类别中

ii 在XXXCommand的能够接受不同消息的Switch中添加相应的case;

iii 在相应的XXXProxy中添加public函数（与数据操作相关），函数名要与消息名相关联，在此类中可以获取相应类的数据,通过调用Proxy中的Data。只能在Proxy中写改变、获取数据相关的代码

iiii 在XXXCommand的case中调用在XXXProxy中添加的函数

iiiii 使用该命令：

在MonoBehaviour类中(MVC类外)

RunFacade.getInstance.sendNotificationCommand(NotificationConstant.XXX);

在MVC类中可直接用SendNotification(NotificationConstant.XXX);

4.如果是Mediator命令：

i 在相应的XXXMediator中，在ListNotificationInterests()中将消息放到list列表中， list.add(XXXXX)

ii 在XXXMediator的HandleNotification的Switch中添加相应的case;

iii 在case中写相关的改变视图的代码

5. 传递参数：

在Tool中有一个Past类，在里面写需要传递参数的结构类；

传递：SendNotification(NotificationConstant.XXX，数据类);

接收： 数据类 past=notification.Body as 数据类

pppppps: 注意命名结构 可读性 重复性

mvc整体框架不能私自改变。

所有的string参数都要在Tags中定义