通过目录或导航窗格进行定向跳转

目录

[一、 FILENAME：单机版使用说明 3](#_Toc116166157)

[名词解释 3](#_Toc116166158)

[项目名关键词（mvc\_gui\_interact） 3](#_Toc116166159)

[二、src目录结构 4](#_Toc116166160)

[图片 4](#_Toc116166161)

[新增模块命名建议： 4](#_Toc116166162)

[三、com.frm概要 5](#_Toc116166163)

[四、demo.\* 构建过程 5](#_Toc116166164)

[四.1、登录模块demo 5](#_Toc116166165)

[1、初始的目录： 5](#_Toc116166166)

[2、创建包【nf.系统名.模块名】： 6](#_Toc116166167)

[3、创建必须类（Model） 6](#_Toc116166168)

[4、创建必须的插件类（View） 7](#_Toc116166169)

[5、创建必须类（Ctrl） 11](#_Toc116166170)

[6、至此，单个模块创建完毕 13](#_Toc116166171)

[创建程序的唯一入口main方法Main类 13](#_Toc116166172)

[创建包 13](#_Toc116166173)

[创建带main方法的Main类 14](#_Toc116166174)

[完善代码 15](#_Toc116166175)

[通过Main类启动程序 15](#_Toc116166176)

[界面一demo 16](#_Toc116166177)

[假设模块名为：Module1： 16](#_Toc116166178)

[界面初始内容： 17](#_Toc116166179)

[界面二demo 17](#_Toc116166180)

[假设模块名为：Module2： 17](#_Toc116166181)

[界面初始内容： 18](#_Toc116166182)

[五、demo混合使用演示 19](#_Toc116166183)

[打开&关闭模块的使用 19](#_Toc116166184)

[协议&&消息体定义&&使用 21](#_Toc116166185)

[其它细节先查阅代码自行调试. 25](#_Toc116166186)

[六、常用方法描述 25](#_Toc116166187)

# FILENAME：单机版使用说明

代码能跑，文档仅参考作用，细节看代码

## 名词解释

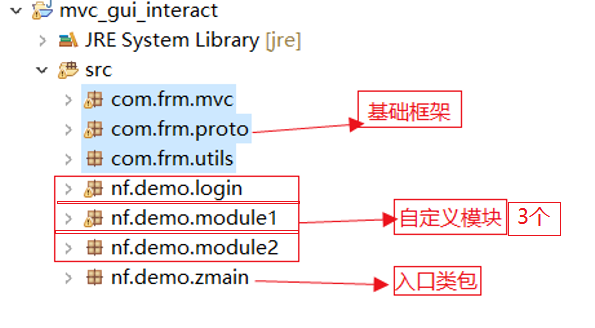
|  |  |
| --- | --- |
| 必须类 | 要习惯性地创建；必须有的类 |
| 插件类 | 通过插件创建的类，插件会生成一些必要的方法 |

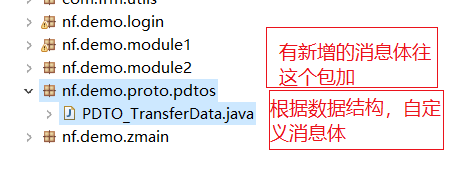
## 项目名关键词（mvc\_gui\_interact）

* MVC
  + M：Model（存放数据）
  + V：View（界面元素）
  + C：Controller[Ctrl]（控制器）
* GUI
  + 构建方式：
    - 通过WindowBuilder插件构建
    - 自己写，代码量少且灵活
* Interact
  + 通过消息唯一协议号接收数据进行交互

# 二、src目录结构

## 图片



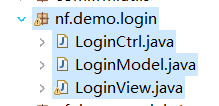


## 新增模块命名建议：

包名建议：nf . 系统名 . 模块名

类名建议：xxxModel \ xxxView \ xxxCtrl

如Login模块：



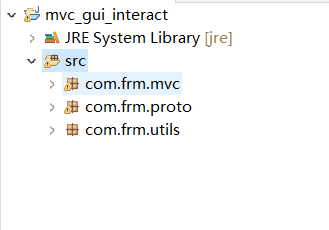
# 三、com.frm概要

* + 对各个模块M、V、C类的实例化和销毁的管理
  + Model类可以进行数据保存
  + 支持销毁界面（V）但保留数据部分（Model）
  + 支持通过协议号进行消息传递
  + 可扩展性强

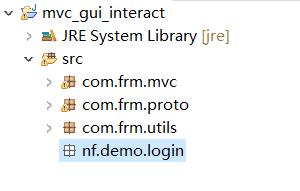
# 四、demo.\* 构建过程

## 四.1、登录模块demo

### 1、初始的目录：

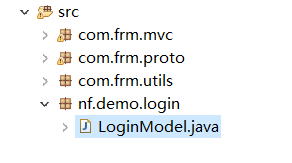


### 2、创建包【nf.系统名.模块名】：



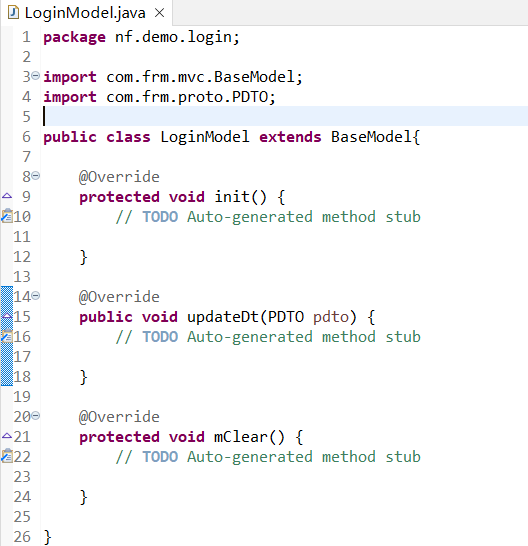
### 3、创建必须类（Model）

#### 创建类LoginModel：



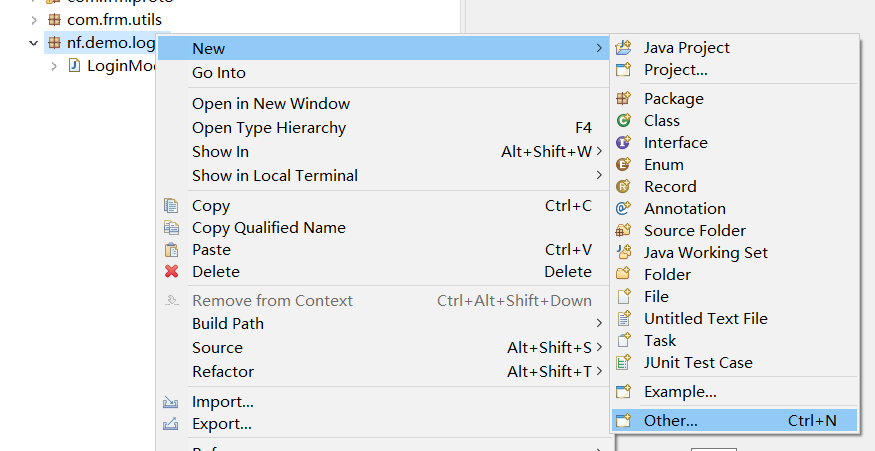
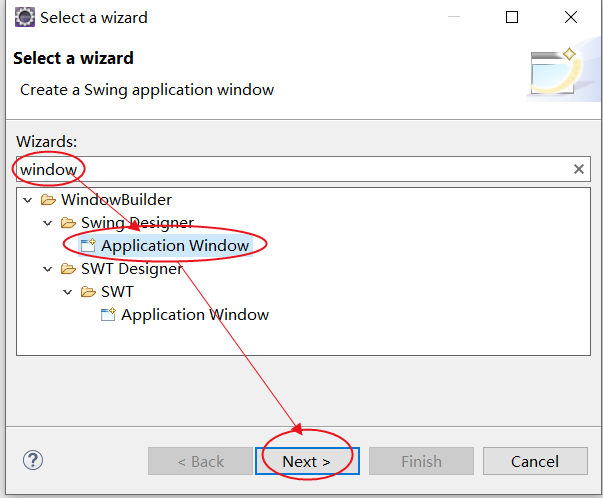
#### 继承BaseModel

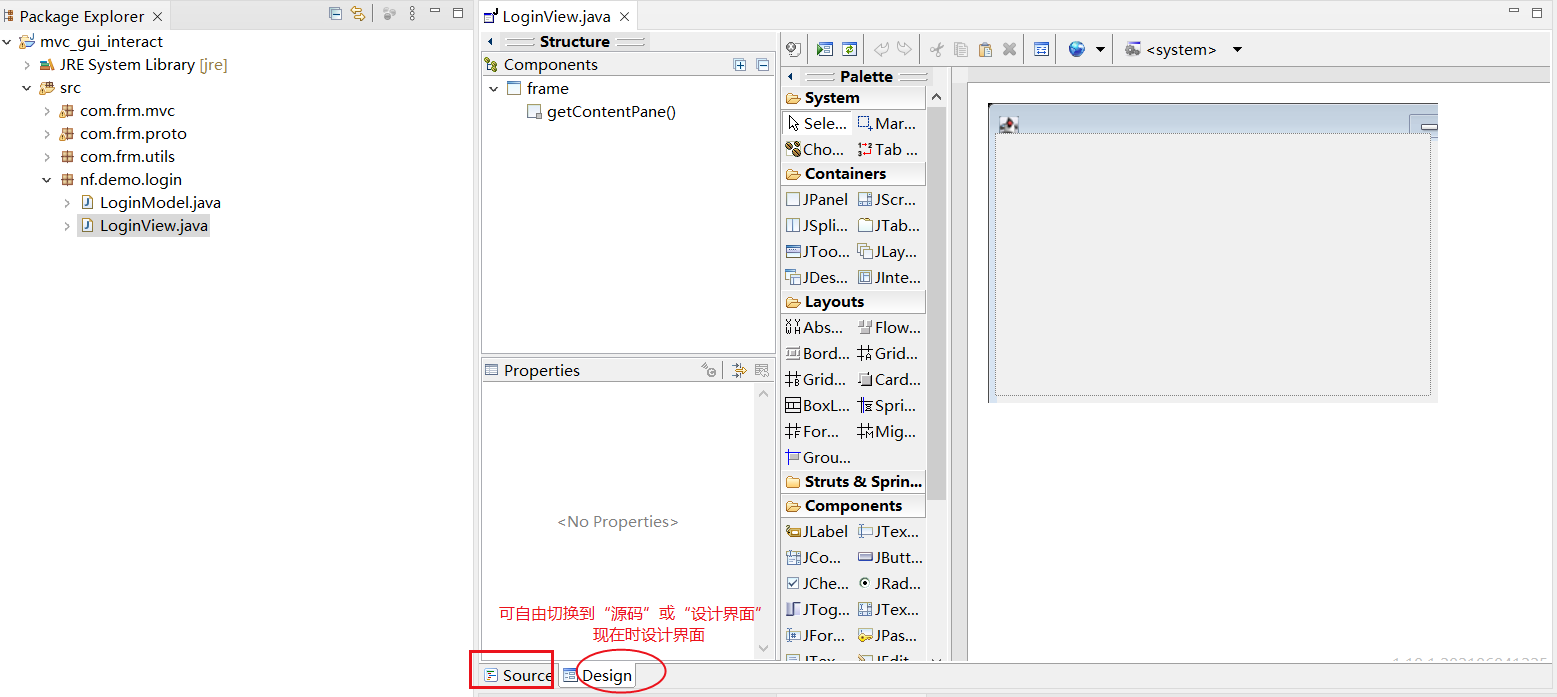
* + 导包
  + 实现方法
  + 完成如下图所示



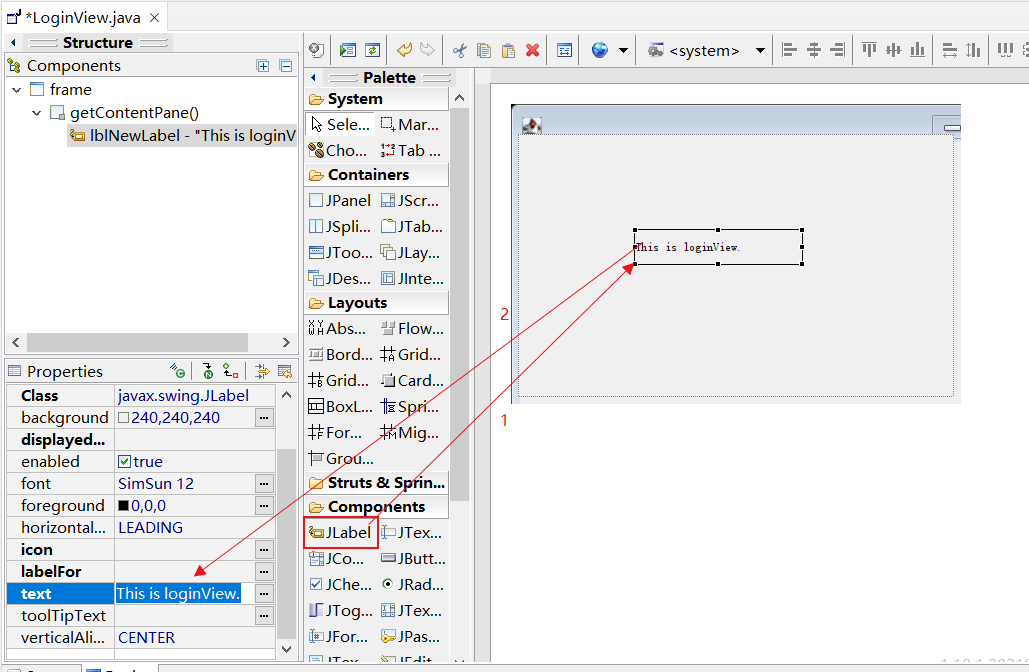
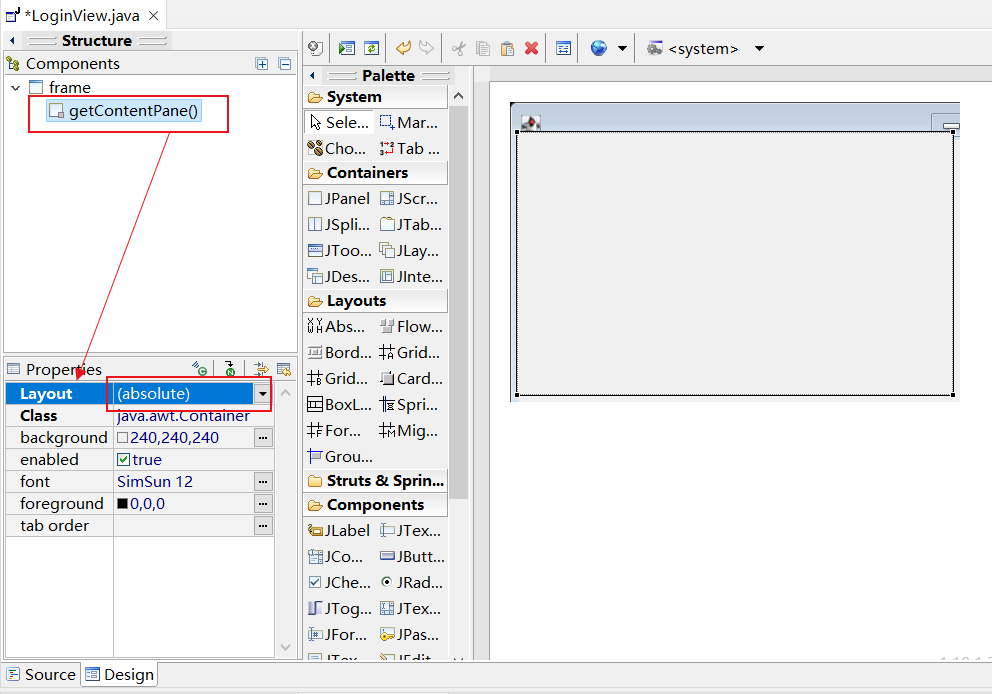
### 4、创建必须的插件类（View）

#### 创建LoginView（用装好的插件来创建，看图走）

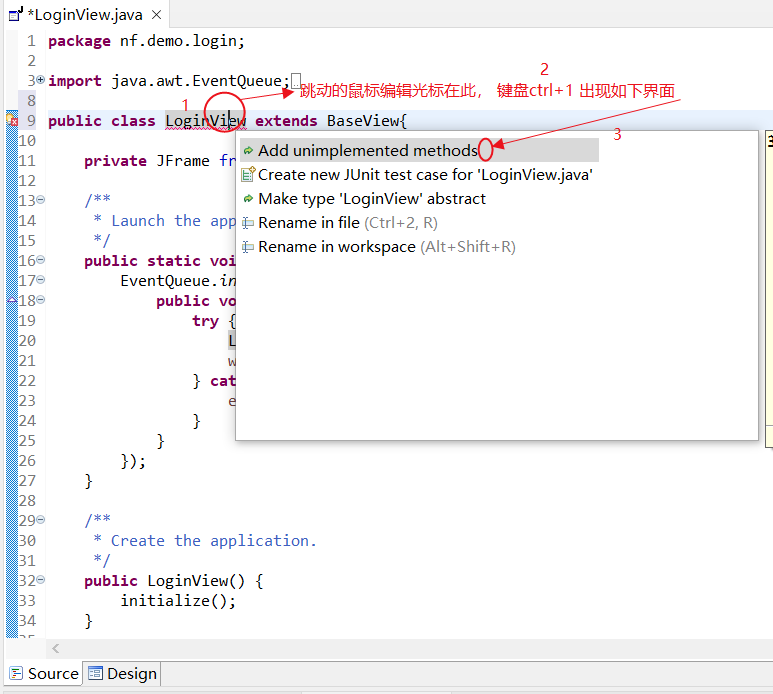


#### 改为absolute布局，添加一个页签



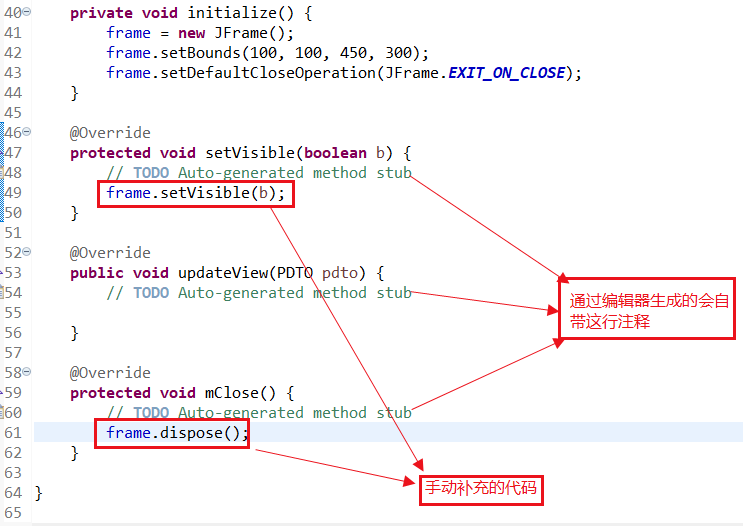
#### 继承BaseView（切换到source）

Step1. 老规矩，导包、实现未实现的方法（可将鼠标编辑光标置于报错处，然后Ctrl+1快速唤出报错解决方案）



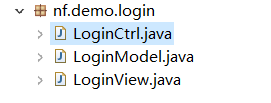
Step2. 在生成的方法里补充上代码（图片红框部分为补充的）

|  |  |
| --- | --- |
| 对应的方法 | 需要手动加的代码 |
| **protected** **void** setVisible(**boolean** b) { | frame.setVisible(b); |
| **protected** **void** mClose() { | frame.dispose(); |



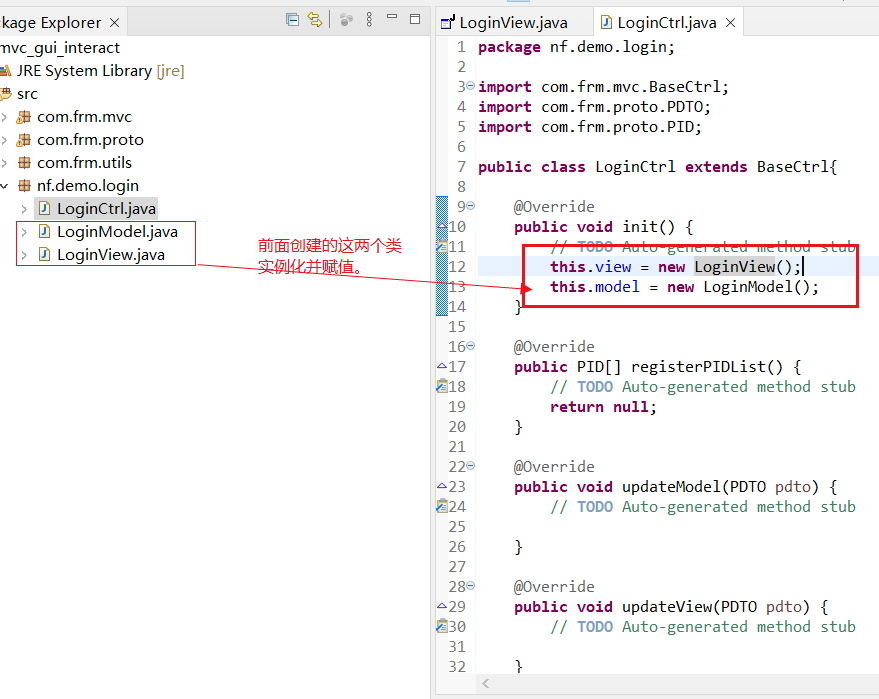
### 5、创建必须类（Ctrl）

#### 创建LoginCtrl

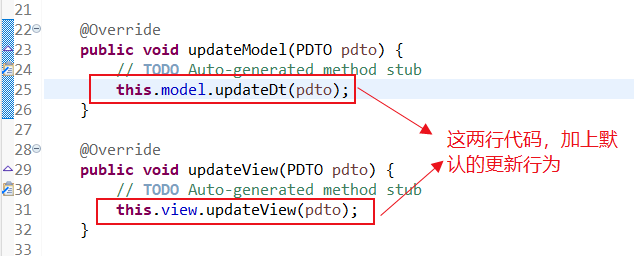


#### 继承BaseCtrl

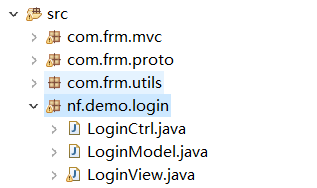
老规矩：导包、实现unimplemented方法

重点：此处要在init()方法里，new出前面创建的xxxModel类和xxxView类，并赋值到对应的xxxCtrl类里。看图【红框为新增的部分】

必须习惯性地加上下面的两行（不然后面协议更新不了界面）



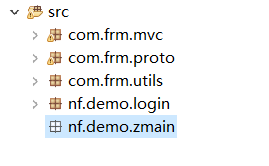
### 6、至此，单个模块创建完毕



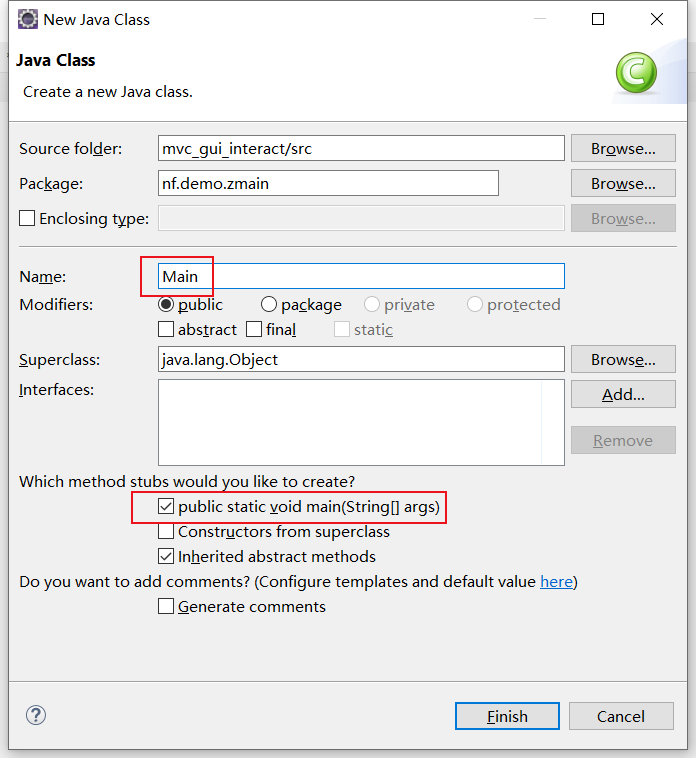
可以自己实践熟悉下过程。

## 创建程序的唯一入口main方法Main类

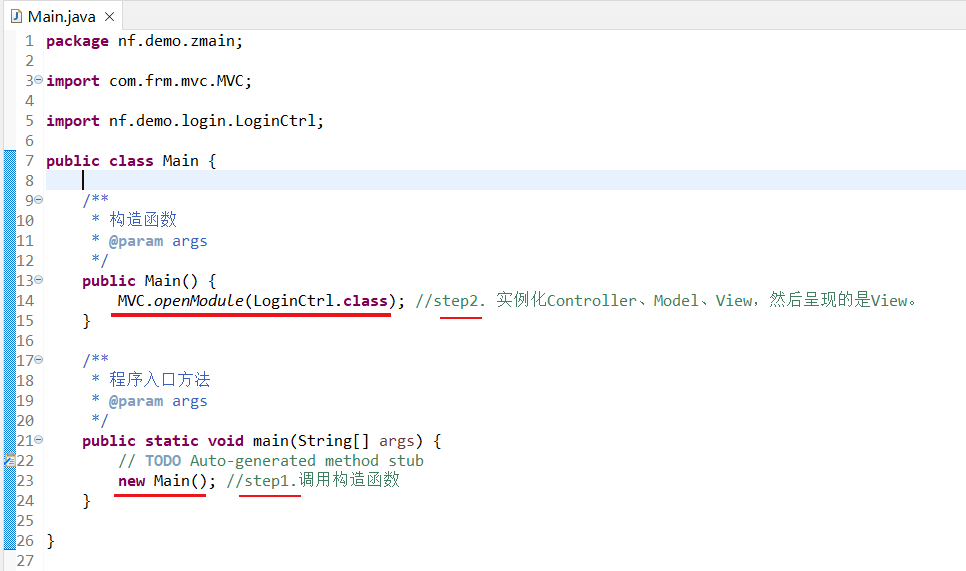
### 创建包



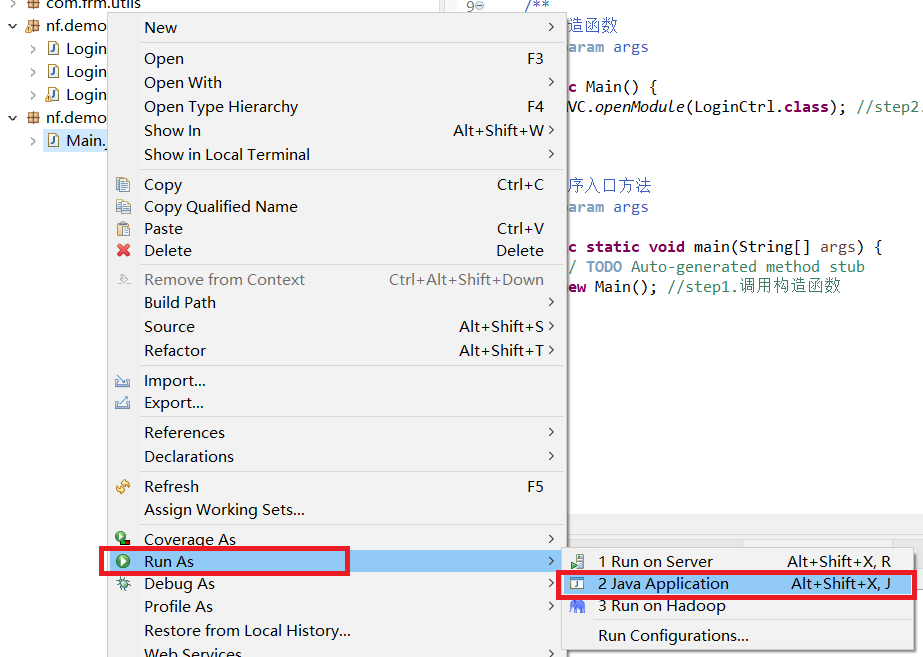
### 创建带main方法的Main类



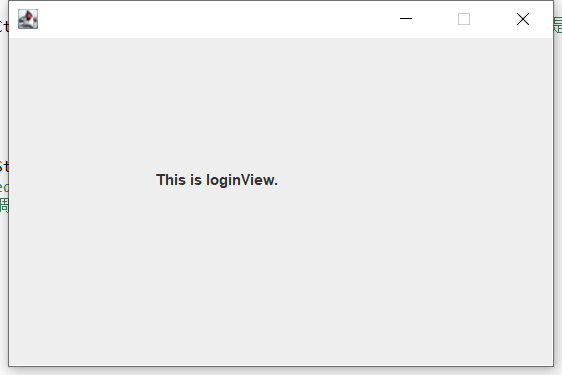
### 完善代码



### 通过Main类启动程序



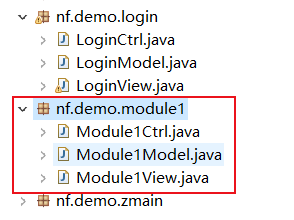
运行结果（成功效果图）：



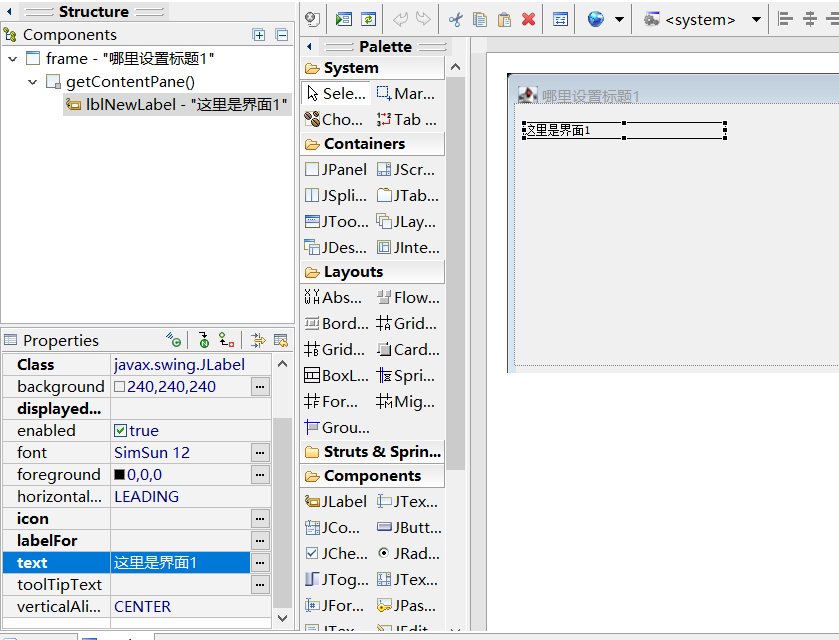
## 界面一demo

### 假设模块名为：Module1：

重复四.1、登录模块demo的2~6步，完成后包结构如图。



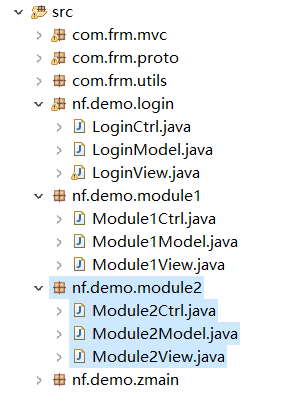
### 界面初始内容：



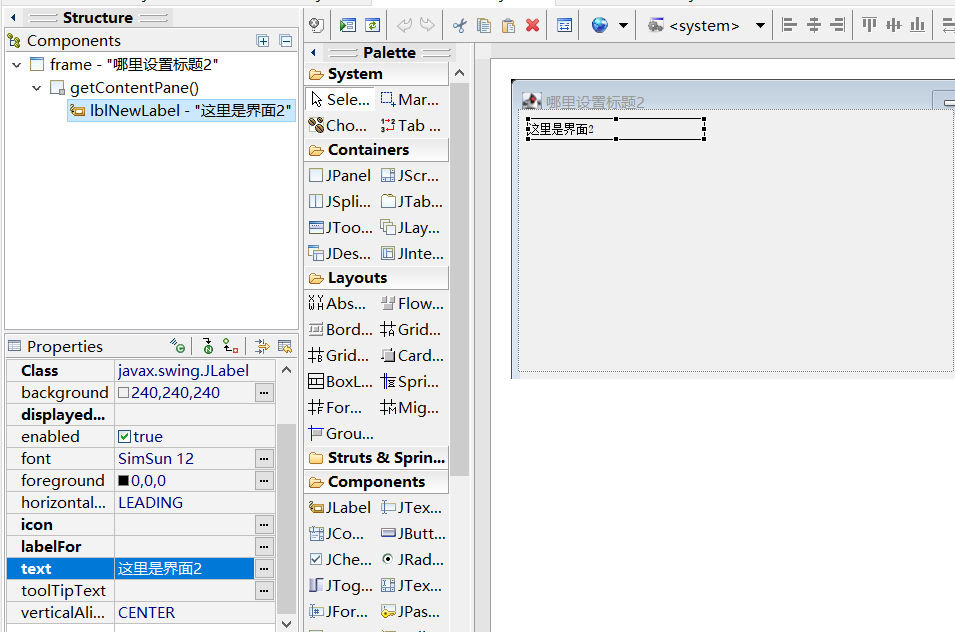
## 界面二demo

### 假设模块名为：Module2：

重复四.1、登录模块demo的2~6步，完成后包结构如图。



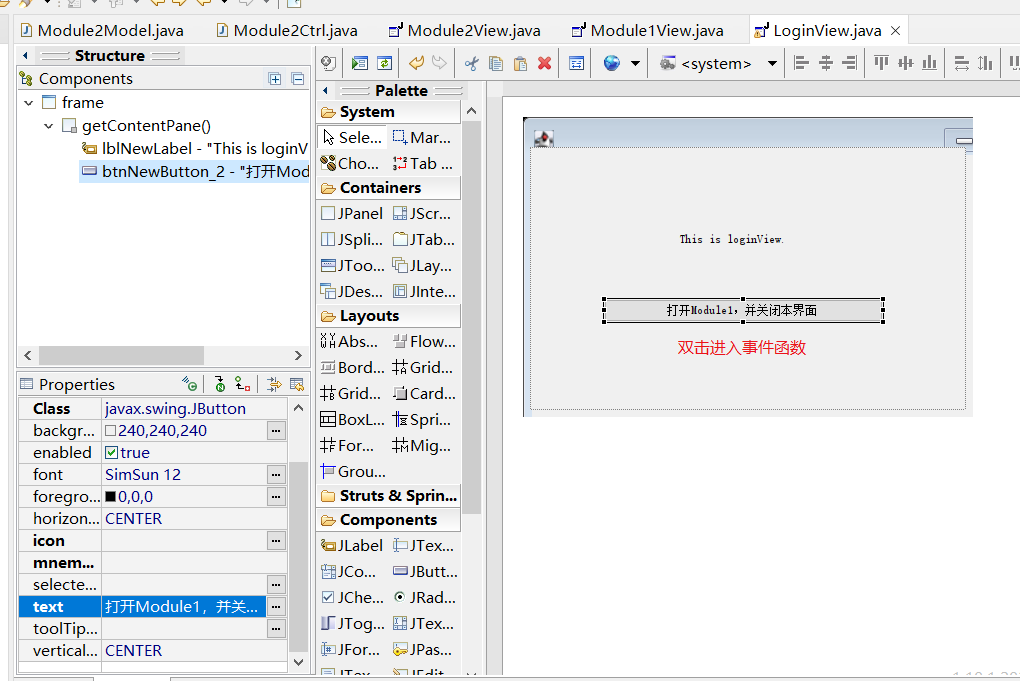
### 界面初始内容：



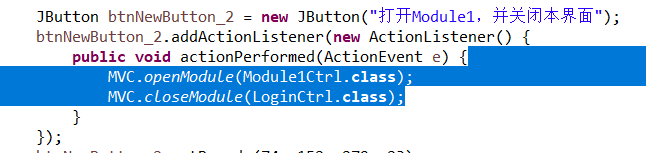
# 五、demo混合使用演示

### 打开&关闭模块的使用

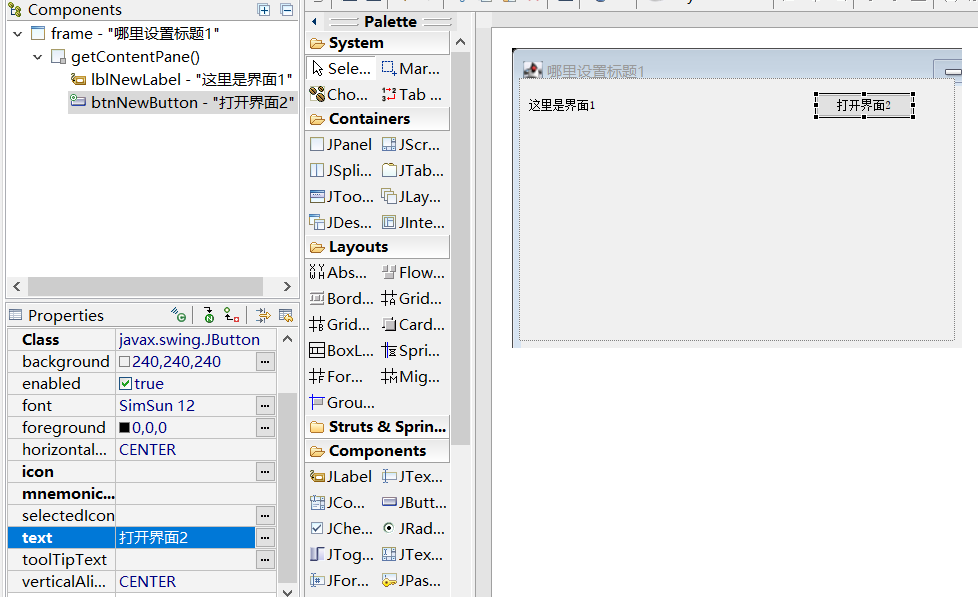
#### LoginView添加按钮



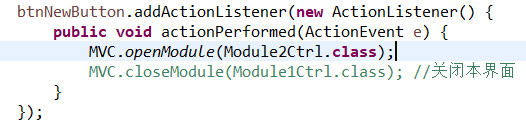
#### LoginView添加事件：开启模块1、关闭登录模块的代码



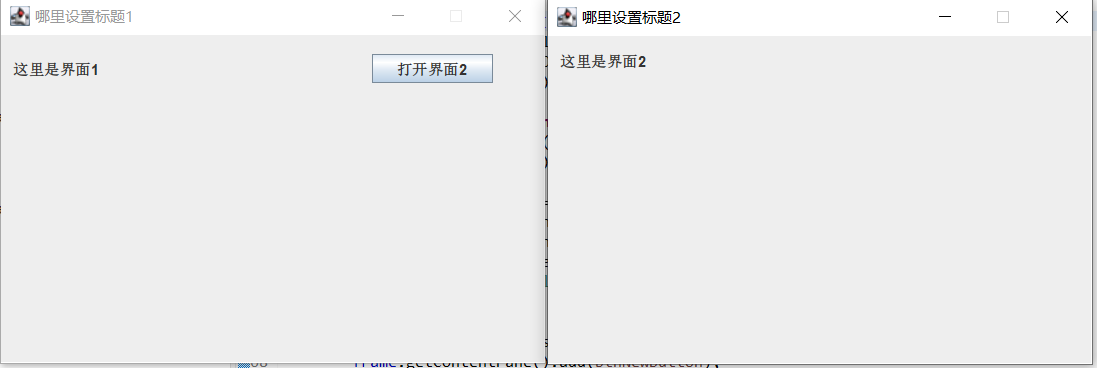
#### Module1View添加按钮：



#### Module1View添加事件：开启模块2



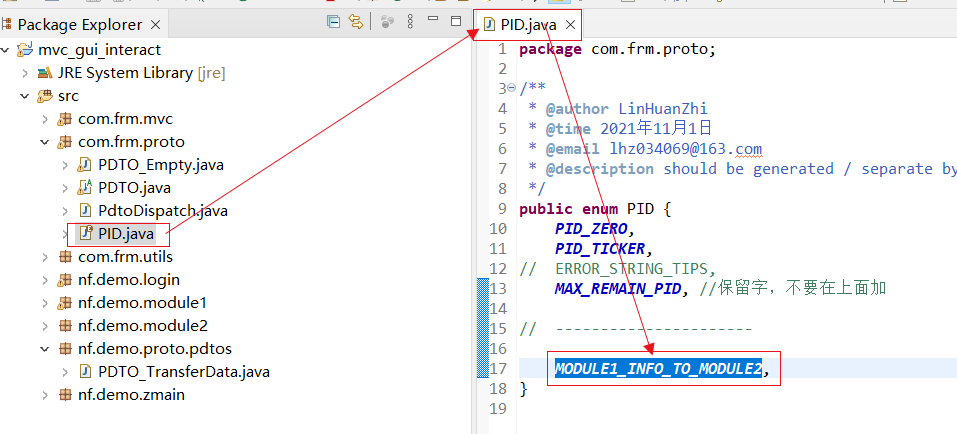
#### 同时打开多个界面



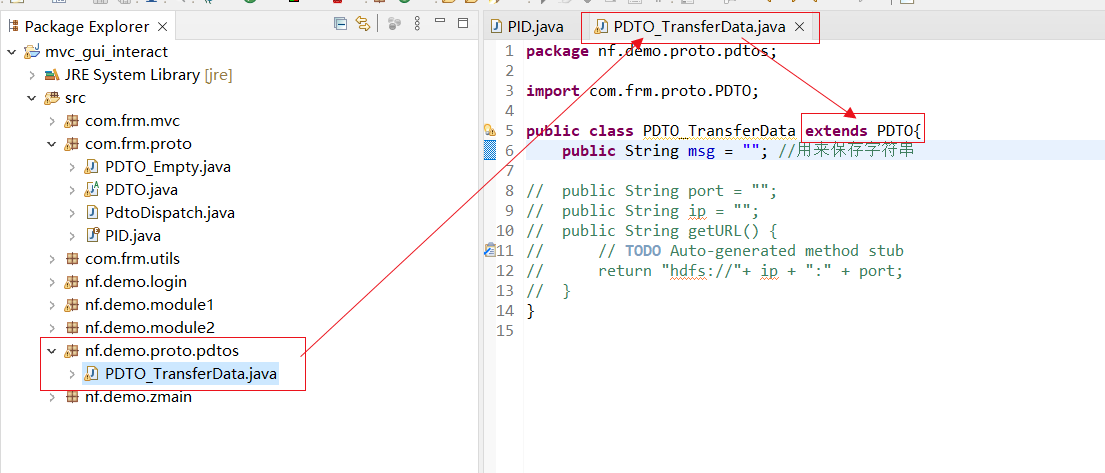
### 协议&&消息体定义&&使用

假设现在界面1和2同时打开了，要从界面1发送消息到到界面2。

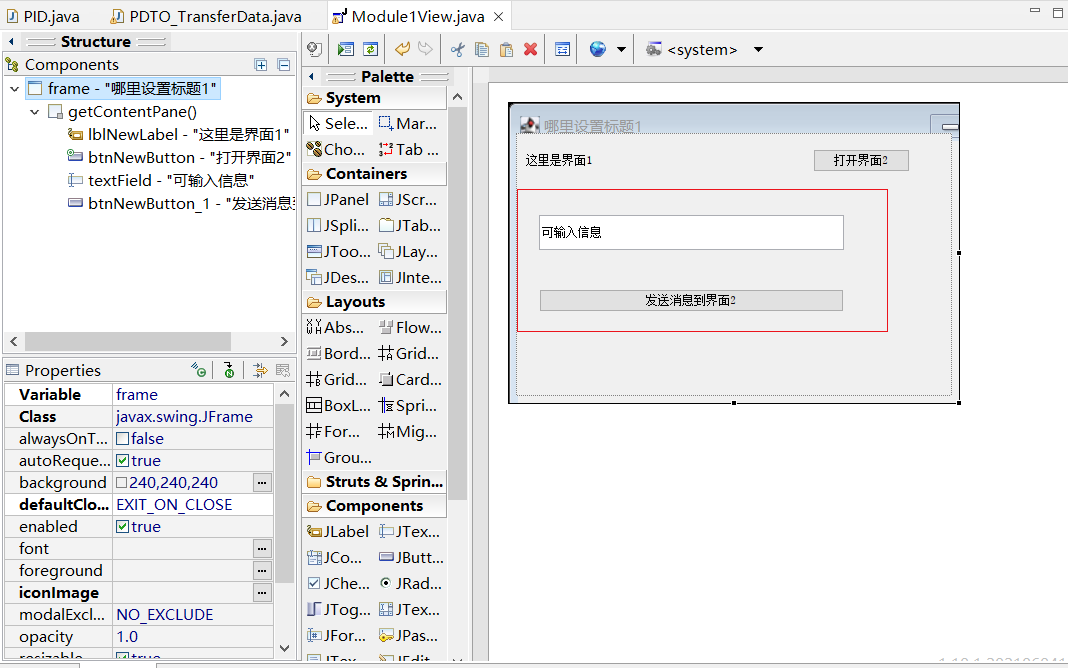
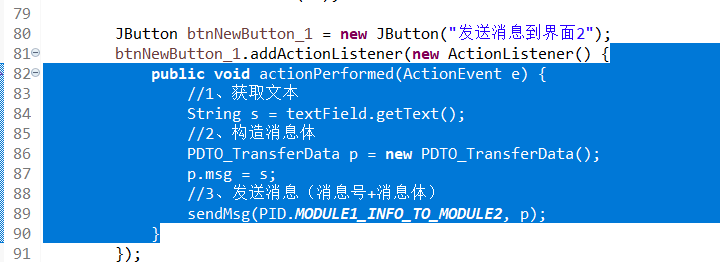
#### 协议号定义



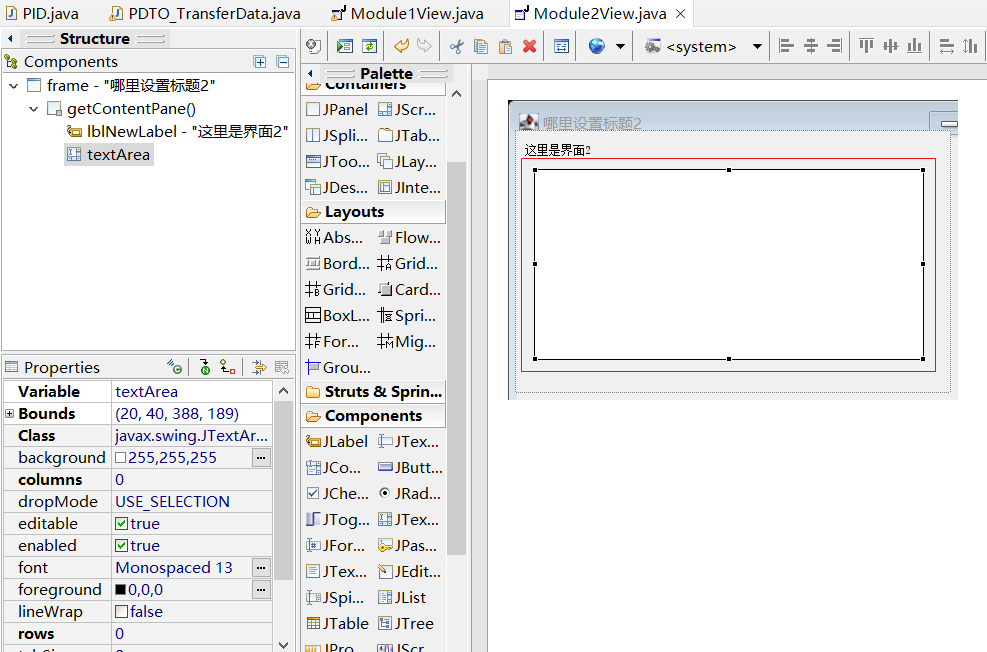
#### 消息体定义

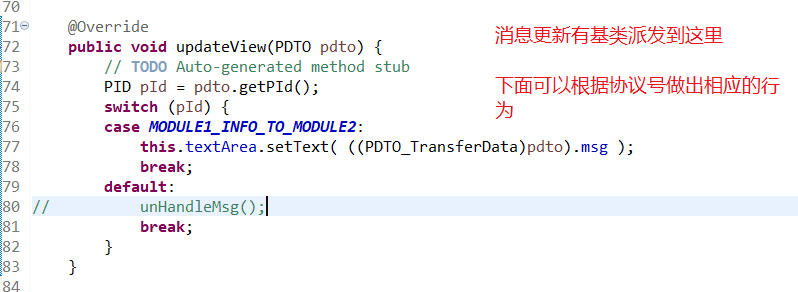


#### 界面1（Module1View）增加组件+代码

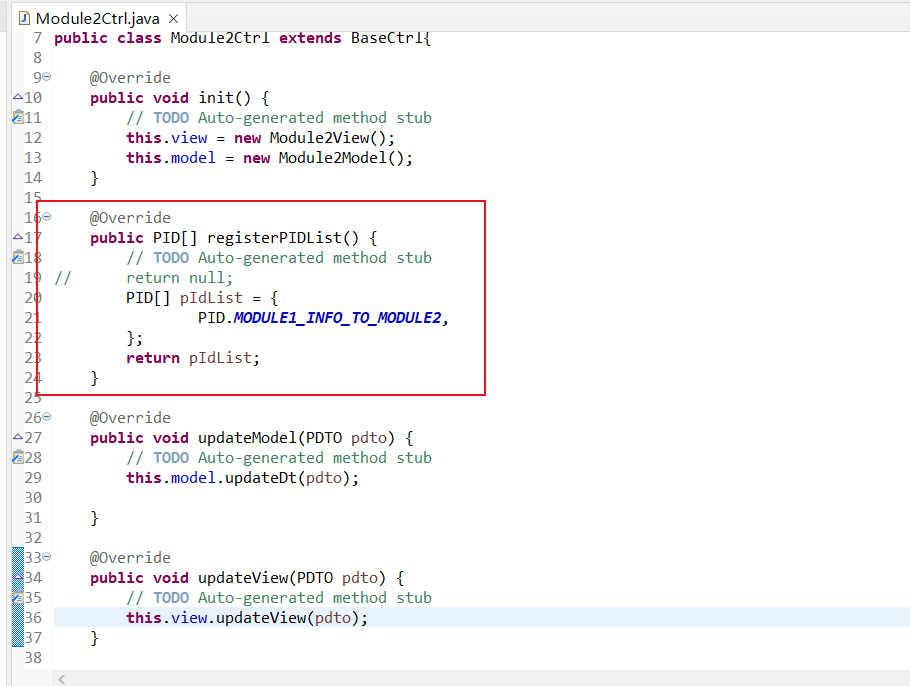
 

#### 界面2（Module2View）增加组件+代码

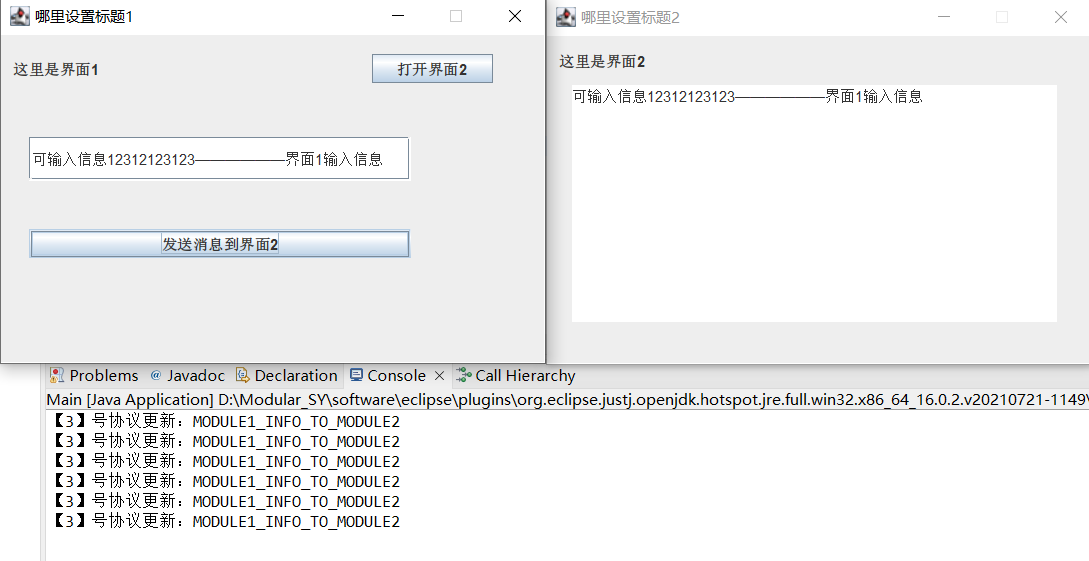


#### 在Module2Ctrl上注册消息



#### 一切就绪，可以实现预期效果（本地消息派发）



### 其它细节先查阅代码自行调试.

文档写得不是很全，有兴趣可以自己探索下。//TODO

# 六、常用方法描述

|  |  |
| --- | --- |
| 功能描述 | 示例 |
| 打开一个模块的MVC | MVC.*openModule*(LoginCtrl.**class**); |
| 关闭一个模块的MVC | MVC.*closeModule*(LoginCtrl.**class**); |
|  | 未完待续 //TODO；有需要先自行探索下 |
|  | 有更丰富更多内容的客户端和服务器端，需要可联系 |