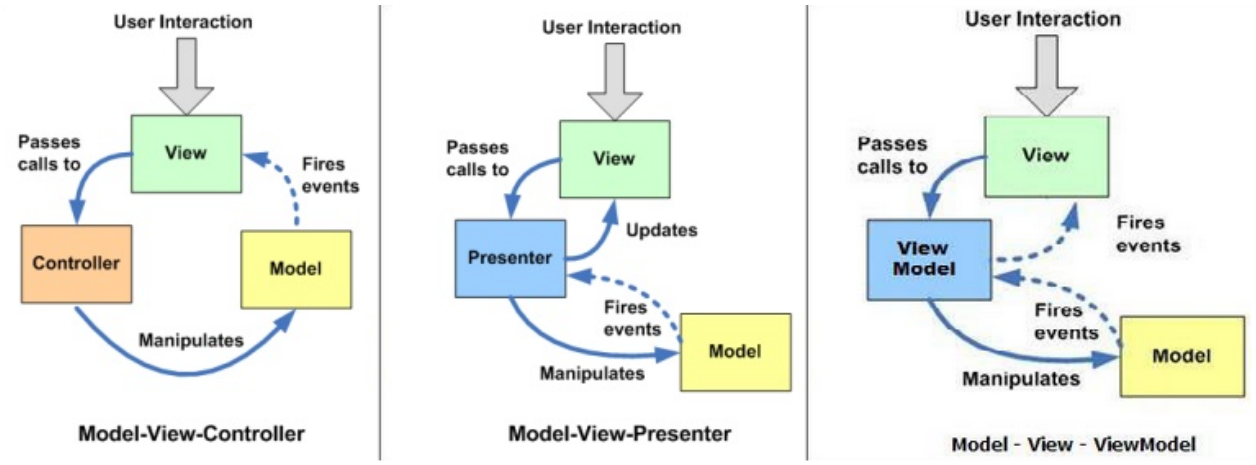
GUI入门

MVC vs MVP vs MVVM



MVC

关键点： 1、View是把控制权交移给Controller，自己不执行业务逻辑。 2、Controller执行业务逻辑并且操作Model，但不会直接操作View，可以说它是对View无知的。 3、View和Model的同步消息是通过观察者模式进行，而同步操作是由View自己请求Model的数据然后对视图进行更新。

MVC的优缺点

优点:

> 1、把业务逻辑全部分离到Controller中，模块化程度高。当业务逻辑变更的时候，不需要变更View和Model，只需要Controller换成另外一个Controller就行了（Swappable Controller）。
> 2、观察者模式可以做到多视图同时更新。

缺点:

> 1、Controller测试困难。因为视图同步操作是由View自己执行，而View只能在有UI的环境下运行。在没有UI环境下对Controller进行单元测试的时候，Controller业务逻辑的正确性是无法验证的：Controller更新Model的时候，无法对View的更新操作进行断言。 2、View无法组件化。View是强依赖特定的Model的，如果需要把这个View抽出来作为一个另外一个应用程序可复用的组件就困难了。因为不同程序的的Domain Model是不一样的

MVP

关键点:

> 1、View不再负责同步的逻辑，而是由Presenter负责。Presenter中既有业务逻辑也有同步逻辑。
> 2、View需要提供操作界面的接口给Presenter进行调用。（关键）

MVP的优缺点

优点：

> 1、便于测试。Presenter对View是通过接口进行，在对Presenter进行不依赖UI环境的单元测试的时候。可以通过Mock一个View对象，这个对象只需要实现了View的接口即可。然后依赖注入到Presenter中，单元测试的时候就可以完整的测试Presenter业务逻辑的正确性。 2、View可以进行组件化。在MVP当中，View不依赖Model。这样就可以让View从特定的业务场景中脱离出来，可以说View可以做到对业务逻辑完全无知。它只需要提供一系列接口提供给上层操作。这样就可以做高度可复用的View组件。

缺点：

> 1、Presenter中除了业务逻辑以外，还有大量的View->Model，Model->View的手动同步逻辑，造成Presenter比较笨重，维护起来会比较困难。

MVVM

关键点 MVVM把View和Model的同步逻辑自动化了。以前Presenter负责的View和Model同步不再手动地进行操作，而是交由框架所提供的Binder进行负责。只需要告诉Binder，View显示的数据对应的是Model哪一部分即可。

MVVM的优缺点

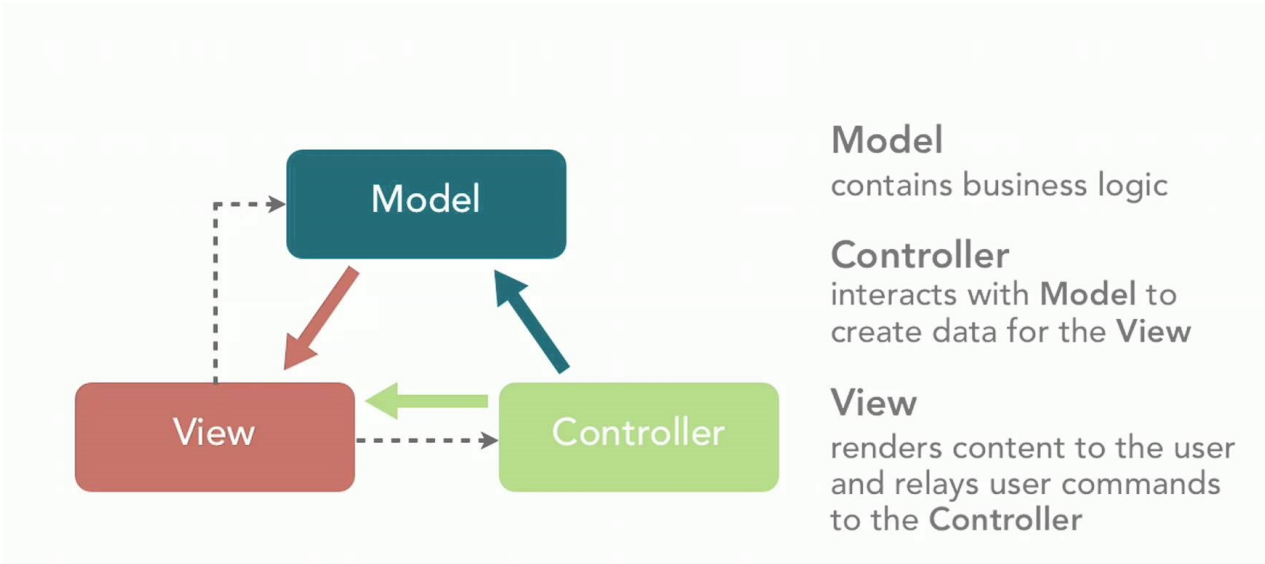
优点：

> 1、提高可维护性。解决了MVP大量的手动View和Model同步的问题，提供双向绑定机制。提高了代码的可维护性。 2、简化测试。因为同步逻辑是交由Binder做的，View跟着Model同时变更，所以只需要保证Model的正确性，View就正确。大大减少了对View同步更新的测试。

缺点：

> 1、过于简单的图形界面不适用，或说牛刀杀鸡。 2、对于大型的图形应用程序，视图状态较多，ViewModel的构建和维护的成本都会比较高。 3、数据绑定的声明是指令式地写在View的模版当中的，这些内容是没办法去打断点debug的。

MVC in Swing



View和Model之间是Observer模式。View要先向感兴趣的Model进行注册、Model改变之后通知View，View来Model取数据。View和Controller之间是事件机制。一次View的交互，对应一个响应的Controller。Controller负责修改Model，和选择View的显示。

Model in Swing

ButtonModel接口的属性

- ActionCommand
- Mnemonic
- Armed
- Enabled
- Pressed
- Rollover
- Selected

同样的模型DefaultButtonModel可以用于不同视图

- 下压按钮
- 单选按钮
- 复选框
- 菜单项

View in Swing

JButton

- 继承JComponent
- 包含DefaultButtonModel、一些视图数据（标签和图标）、一个负责按钮视图的BasicButtonUI对象

Controller in Swing

```
public class ButtonUIListener
   implements MouseListener, MouseMotionListener,
              ChangeListener
{
   public void mouseMoved(MouseEvent mouseevent)
   public void mouseDragged(MouseEvent mouseevent)
   public void mouseClicked(MouseEvent mouseevent)
   public void mouseEntered(MouseEvent mouseevent)
   public void mouseExited(MouseEvent mouseevent)
   public void mousePressed(MouseEvent mouseevent)
   public void mouseReleased(MouseEvent mouseevent)
   public void stateChanged(ChangeEvent changeevent)
}
```

Controller修改Model状态

```
 public void mousePressed(MouseEvent mouseevent)
   {
      Button button = (Button)mouseevent.getSource();

      ButtonModel buttonmodel = button.getModel();

      buttonmodel.setPressed(true);

      buttonmodel.setArmed(true);
   }
```

状态改变后，异步事件响应，View拿Model数据，重画。

```
public void stateChanged(ChangeEvent changeevent)
   {
      Button button = (Button)changeevent.getSource();

      button.repaint();
   }
```

MVC in Web GUI

HTML表达了基础组件，生成DOM树，CSS表达样式生成CSS树。从而可以通过layout生成出渲染树。Javascipt代表触发事件机制之后的处理。Javascipt脚本去操作Dom、更改CSS样式时，浏览器又要重新构建DOM、CSSOM树，重新render，重新layout、paint；

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
```

```
<body>

<h1>JavaScript 验证输入</h1>

<p>请输入 1 到 10 之间的数字：</p>

<input id="numb">

<button type="button" onclick="myFunction()">提交</button>

<p id="demo"></p>

<script>
function myFunction() {
    var x, text;

    // 获取 id="numb" 的值
    x = document.getElementById("numb").value;

    // 如果输入的值 x 不是数字或者小于 1 或者大于 10，则提示错误 Not a Number or less
than one or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "输入错误";
    } else {
        text = "输入正确";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

MVP in Android

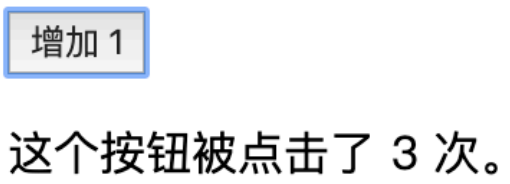
MVVM in Vue

counter数据和View中的点击次数进行双向绑定。

事件响应每次按按钮，counter数据+1。View中的点击次数也随之改变。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Vue 测试实例 - 菜鸟教程(runoob.com)</title>
<script src="https://cdn.staticfile.org/vue/2.2.2/vue.min.js"></script>
</head>
<body>
<div id="app">
  <button v-on:click="counter += 1">增加 1</button>
  <p>这个按钮被点击了 {{ counter }} 次。</p>
</div>

<script>
new Vue({
  el: '#app',
  data: {
    counter: 0
  }
})
</script>
</body>
</html>
```



ViewModel有时候会异步请求Model的数据。当服务器回答后，再根据response数据设置ViewModel，从而达到View的改变。MVVM可以将对数据的处理成函数式编程范式的流式编程。

```
<div id="app">
```

```
  <h1>网站列表</h1>
  <div
    v-for="site in info"
  >
    {{ site.name }}
  </div>
</div>
<script type = "text/javascript">
new Vue({
  el: '#app',
  data () {
    return {
      info: null
    }
  },
  mounted () {
    axios
      .get('https://www.runoob.com/try/ajax/json_demo.json')
      .then(response => (this.info = response.data.sites))
      .catch(function (error) { // 请求失败处理
        console.log(error);
      });
  }
})
</script>
```

Reference:

https://www.cs.utexas.edu/users/dsb/SwingTutorial/1_Basics/lecture.html

https://www.runoob.com/vue2/vuejs-ajax-axios.html

https://www.runoob.com/try/try.php?filename=tryjs_validation_number

https://segmentfault.com/a/1190000014070240

https://zhuanlan.zhihu.com/p/38108311

https://juejin.im/post/58870cc2128fe1006c46e39c