GUI入门

GUI Key elments

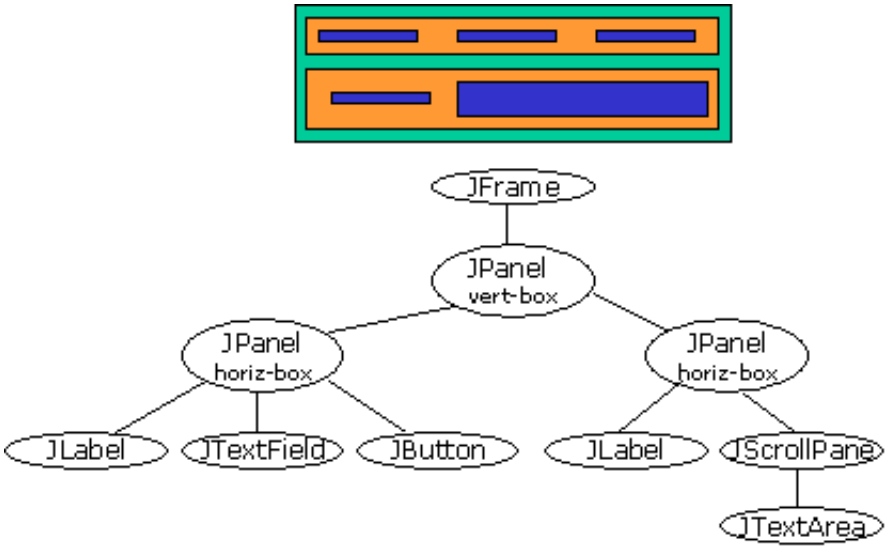
任何GUI（Graphical User Interface）都有三个关键元素：组件、布局和事件。

Component

组件是整个GUI的基础。常见组件有Label、TextField、Button等。

Layout

有了组件之后，下面要考虑的就是组件的布局。组件之间是Composite模式，可以转化为一个树状结构。

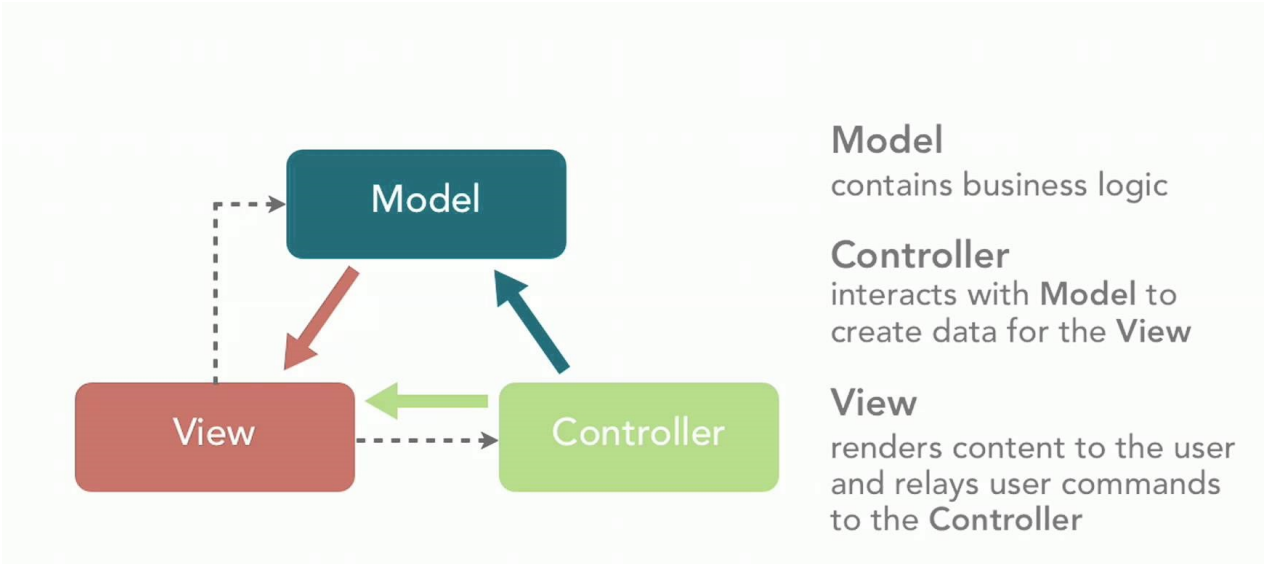


Event

最后，每个组件都要有事件响应的处理。比如，按按钮，获得焦点，被选择等。

Act that Results in an Event	Listener Type
User clicks a button, presses Return while typing in a text field, or chooses a menu item	ActionListener
User closes a frame (main window)	WindowListener
User presses a mouse button while the cursor is over a component	MouseListener
User moves the mouse over a component	MouseMotionListener
Component becomes visible	ComponentListener
Comoponent gets the keyboard focus	FocusListener
Table or list selection changes	ListSelectionListener

MVC Style



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。

MVC in Swing

Model

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

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

```
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();
   }
```

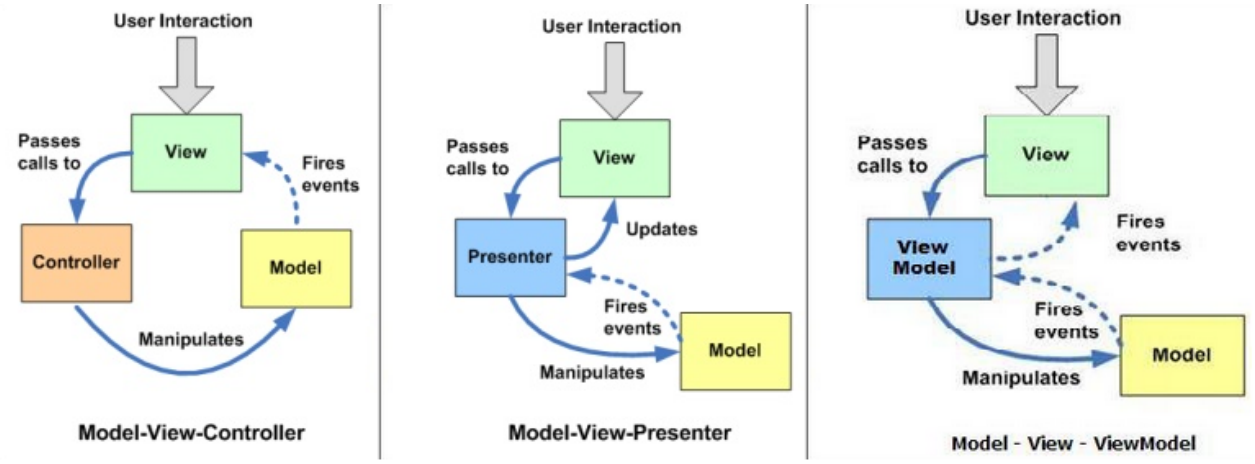
MVC in Web GUI

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>
```

MVC vs MVP vs MVVM

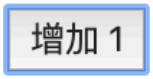


MVVM in Vue

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>
```



这个按钮被点击了 3 次。

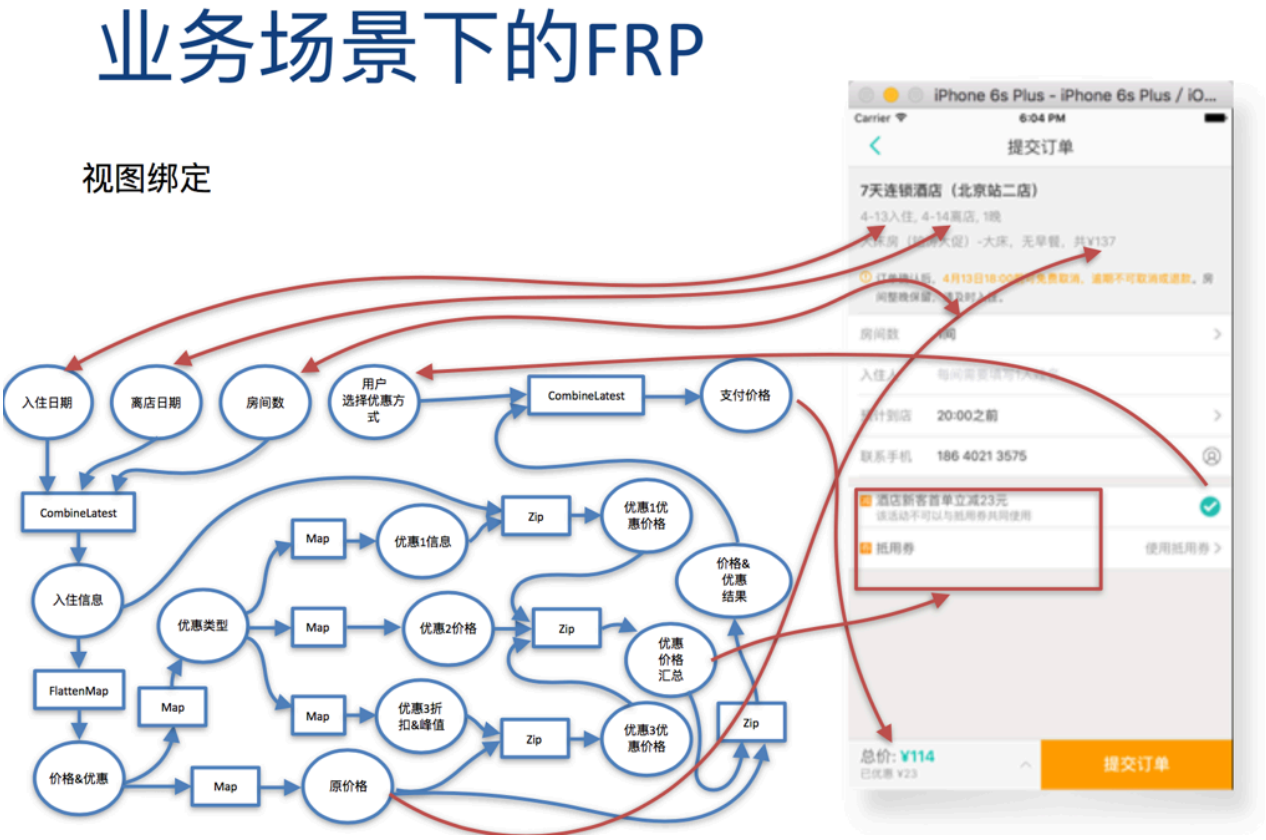
ViewModel有时候会异步请求Model的数据。当服务器回答后，再根据response数据设置ViewModel，从而达到View的改变。

```
<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>
```

网站列表

Google
Runoob
Taobao

MVVM可以和函数式编程范式下的流式编程相结合，生成复杂的数据处理。



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