

## ■ ■ ■ 9.4.4 DocumentEvent事件

### 1. DocumentEvent事件源

- 用户在文本区中进行文本编辑操作，使得文本区中的内容发生变化，这将导致文本区所维护的文档模型中的数据内容发生变化。导致文本区所维护的文档触发DocumentEvent事件。



## ■ ■ ■ 9.4.4 DocumentEvent事件

### 2. 注册监视器

- 使用addDocumentListener(DocumentListener listener) 将实现DocumentListener接口的类的对象注册为事件源的监视器。

### 3. DocumentListener接口

在javax.swing.event包中，该接口中有三个方法：

**public void changedUpdate(DocumentEvent e)**

**public void removeUpdate(DocumentEvent e)**

**public void insertUpdate(DocumentEvent e)**

- 事件源触发DocumentEvent事件后，监视器将发现触发的DocumentEvent事件，然后调用接口中的相应方法对发生的事件作出处理。

**例9-9演示**



## 9.4.5 MouseEvent事件

### 1. 使用MouseListener接口处理鼠标事件

➤ 可以处理以下5种操作触发的鼠标事件：

在事件源上按下鼠标键。

在事件源上释放鼠标键。

在事件源上点击鼠标键。

鼠标进入事件源。

鼠标退出事件源。

## 9.4.5 MouseEvent事件

➤ MouseEvent中有下列几个重要的方法：

getX() 获取鼠标指针在事件源坐标系中的x坐标。

getY() 获取鼠标指针在事件源坐标系中的y坐标。

getClickCount() 获取鼠标被单击的次数。

getSource() 获取发生鼠标事件的事件源。

## ■ ■ ■ 9.4.5 MouseEvent事件

### ➤ 事件源注册监视器

使用addMouseListener(MouseListener listener)

MouseListener接口方法：

mousePressed(MouseEvent)

mouseReleased(MouseEvent)

mouseClicked(MouseEvent)

mouseEntered(MouseEvent)

mouseExited(MouseEvent)

例9-10演示



## ■ ■ ■ 9.4.5 MouseEvent事件

### 2. 使用MouseListener接口处理鼠标事件

- 可以处理以下两种操作触发的鼠标事件。

在事件源上拖动鼠标

在事件源上移动鼠标

- 注册监视器的方法是addMouseListener(MouseMotionListener listener)。

- MouseMotionListener接口中有如下方法。

mouseDragged(MouseEvent)

mouseMoved(MouseEvent)

例9-10演示(更新内容)



## 9.4.6 焦点事件

- 组件使用 `addFocusListener(FocusListener listener)` 注册焦点事件监视器。
- 当组件获得焦点监视器后，如果组件从无输入焦点变成有输入焦点，或从有输入焦点变成无输入焦点都会触发 `FocusEvent` 事件。
- 创建监视器的类必须要实现 `FocusListener` 接口，该接口有两个方法：

```
public void focusGained(FocusEvent e)
```

```
public void focusLost(FocusEvent e)
```

**FocusEventDemo 演示**

## ■ ■ ■ 9.4.7 键盘事件

- 当按下、释放或敲击键盘上一个键时触发了键盘事件。当一个组件处于激活状态时，敲击键盘上一个键就导致这个组件触发键盘事件。
- 使用KeyListener接口处理键盘事件，存在3个方法：  
`public void keyPressed(KeyEvent e)`  
`public void KeyReleased(KeyEvent e)`  
`public void keyTyped(KeyEvent e)`

例9-12演示



## ■ ■ ■ 9.4.8 事件处理总结

### ➤ 1. 授权模式

Java的事件处理是基于授权模式，即事件源调用方法将某个对象注册为自己的事件监视器。

### ➤ 2. 接口回调

Java使用接口回调技术，实现处理事件过程。

`addXXXListener(XXXListener listener)`

方法中的参数是一个接口，`listener`可以引用任何实现了该接口的类所创建对象。当事件源发生事件时，接口`listener`回调被类实现的接口方法。



## 9.4.8 事件总结

### ➤ 3. 方法绑定


Java将某种事件的处理绑定到对应接口，即绑定到接口的方法。当事件源触发事件发生后，监视器准确知道去调用哪个方法。

### ➤ 4. 保持松耦合

监视器和事件源尽可能保持是一种松耦合关系，也就是说尽量让事件源所在的类和监视器所在类是关联关系。



## 第9章 GUI编程

1. Java Swing概述
  2. 窗口
  3. 常用组件与布局
  4. 事件处理
  5. 使用MVC结构
  6. 对话框
  7. 发布GUI程序
- 


## 9.5 使用MVC结构

- MVC是一种通过三个不同部分构造一个软件或组件的理想办法：
- **模型（Model）**：用于存储数据的对象。
- **视图（View）**：为模型提供数据显示的对象。
- **控制器（Controller）**：处理用户的交互操作。  
通过视图修改和更新模型中的数据；  
当模型中数据变化时，让视图更新显示。

例9-15演示



## 第9章 GUI编程

1. Java Swing概述
  2. 窗口
  3. 常用组件与布局
  4. 事件处理
  5. 使用MVC结构
  6. 对话框
  7. 发布GUI程序
- 

## 9.6 对话框

- JDialog类和JFrame都是窗口子类，二者的实例都是底层容器，JDialog类创建的对话框依赖某个窗口。
- 对话框分为**无模式**和**有模式**两种：
  - 有模式对话框是指对话框处于激活状态时，只让程序响应对话框内部的事件，用户不能再激活对话框所在程序中其它窗口，直到该对话框消失不可见。
  - 无模式对话框处于激活状态，能再激活其它窗口，也不堵塞其它线程执行。

## 9.6.1 消息对话框

- 消息对话框是有模式对话框，进行一个重要的操作动作之前，最好能弹出一个消息对话框。可以用javax.swing包中的JOptionPane类的静态方法：
- `public static void showMessageDialog(Component parentComponent, String message, String title, int messageType)`
- 创建一个消息对话框。

例9-16演示

## 9.6.2 输入对话框

- 输入对话框含有供用户输入文本的文本框、一个确认和取消按钮，是有模式对话框。当输入对话框可见，要求用户输入一个字符串。

javax.swing包中的JOptionPane类的静态方法：

- `public static String showInputDialog(Component parentComponent, Object message, String title, int messageType)`
- 可以创建一个输入对话框。

例9-17演示



### ■ ■ ■ 9.6.3 确认对话框

- 确认对话框是有模式对话框，可以用javax.swing包中的JOptionPane类的静态方法：
- `public static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType)`
- 得到一个确认对话框。

例9-18演示

## ■ ■ ■ 9.6.4 颜色对话框

- 可以用javax.swing包中的JColorChooser类静态方法:
- `public static Color showDialog(Component component, String title, Color initialColor)`
- 创建一个有模式的颜色对话框。

例9-19演示

## 9.6.5 自定义对话框

- 创建对话框与创建窗口类似，通过建立JDialog的子类来建立一个对话框类，这个类的一个实例，即这个子类创建的一个对象，就是一个对话框。
- 对话框是一个容器，它的默认布局是BorderLayout，对话框可以添加组件，实现与用户的交互操作。

## 9.6.5 自定义对话框


以下是构造对话框的2个常用构造方法：

- `JDialog()` 构造一个无标题的初始不可见的对话框，对话框依赖一个默认的不可见的窗口，该窗口由Java运行环境提供。
- `JDialog(JFrame owner)` 构造一个无标题的初始不可见的无模式的对话框，`owner`是对话框所依赖窗口。

例9-20演示



## 第9章 GUI编程

1. Java Swing概述
  2. 窗口
  3. 常用组件与布局
  4. 事件处理
  5. 使用MVC结构
  6. 对话框
  7. 发布GUI程序
- 

## 9.7 发布GUI程序

- 可以使用Eclipse导出功能，把一些源程序文件压缩成一个JAR文件，来发布我们的应用程序。
- 比如，我们生成一个Tom.jar，然后使用java解释器（使用参数-jar）执行这个压缩文件。
- `java -jar Tom.jar`
- 也可以用鼠标双击该文件，执行这个压缩文件。