

Actions

Actions are convenient ways to associate more than one event with an action. For example, the menu cut item command can be invoked in many applications with a button click or using accelerator keys. Instead, the code does not need to be duplicated as an Action object can be created that encapsulates the command intended by the event along with other related information.

An Action object provides a centralized location for handling activities. The object handles not only the action but also common text, icons, mnemonics, and the enabled or disabled status of the action component. Actions tend to be more expensive than the code used for typical ActionListener interface implementation but provide the benefit of centralized control.

```
import javax.swing.AbstractAction

class ExitAction extends AbstractAction {
    public ExitAction() {
        super("Exit");
        putValue(MNEMONIC_KEY, new Integer(KeyEvent.VK_X));
        putValue(SHORT_DESCRIPTION, "Exit the application");
    }

    public void actionPerformed(ActionEvent e) {
        doAction();
    }

    public void doAction() {
        if(isReadyToTerminate()) {
            System.exit(0);
        }
    }
}
```

The ExitAction object is defined.

```
private ExitAction exitAction;
```

The ExitAction object is created.

```
exitAction = new ExitAction();
```

The ExitAction object is used instead of the System.exit(0) statement.

```
exitMenuItem = new JMenuItem(new ExitAction());
```

or to allow its use in multiple places:

```
ExitAction exitAction = new ExitAction();  
...  
exitMenuItem = new JMenuItem(exitAction);  
...  
exitButton = new JButton (exitAction);  
...  
exitAction.doAction();
```

Another way of invoking it is to use the doAction method:

```
frame.addWindowListener(new WindowAdapter() {  
    public void windowClosing(WindowEvent e) {  
        exitAction.doAction();  
    }  
});
```

The actionPerformed method of an Action object is executed in the same way that it is performed when implementing the ActionListener interface. An Action object is passed and can be accessed as needed. Some implementations of the Action object use a doAction method that is invoked from the actionPerformed method.

The advantage of the doAction method is that it allows the actions to be executed without having to be able to create an ActionEvent and then calling the actionPerformed method. This restricts the activities in the doAction method to those that can be performed without the aid of the ActionEvent object.

A more detailed coverage of Actions can be found at <http://java.sun.com/docs/books/tutorial/uiswing/misc/action.html>.

The AbstractAction class is found in the package javax.swing.

The ExitAction class is used to capture the application exiting activities. Key points include:

- **putValue** – Sets the value for an action. The table below (<http://java.sun.com/docs/books/tutorial/uiswing/misc/action.html>) list the properties supported, the class it is used for, and its purpose

Property	Auto-Applied to: Class (Method Called)	Purpose
ACCELERATOR KEY	JMenuItem (<i>setAccelerator</i>)	The <code>KeyStroke</code> to be used as the accelerator for the action. For a discussion of accelerators versus mnemonics, see Enabling Keyboard Operation . ♦ Introduced in 1.3.
ACTION COMMAND KEY	AbstractButton, JCheckBox, JRadioButton (<i>setActionCommand</i>)	The command string associated with the <code>ActionEvent</code> .
LONG DESCRIPTION	None	The longer description for the action. Can be used for context-sensitive help.
MNEMONIC KEY	AbstractButton, JMenuItem, JCheckBox, JRadioButton (<i>setMnemonic</i>)	The mnemonic for the action. For a discussion of accelerators versus mnemonics, see Enabling Keyboard Operation . ♦ Introduced in 1.3.
NAME	AbstractButton, JMenuItem, JCheckBox, JRadioButton (<i>setText</i>)	The name of the action. You can set this property when creating the action using the <code>AbstractAction(String)</code> or <code>AbstractAction(String, Icon)</code> constructors.
SHORT DESCRIPTION	AbstractButton, JCheckBox, JRadioButton (<i>setToolTipText</i>)	The short description of the action.
SMALL ICON	AbstractButton, JMenuItem (<i>setIcon</i>)	The icon for the action used in the tool bar or on a button. You can set this property when creating the action using the <code>AbstractAction(name, icon)</code> constructor.