Creating Actions

`Action` objects are used for menu items, toolbar buttons and plain buttons. In Genuine, an `Action` object has to inherit from `GenuineAction`. Every `GenuineAction` object must be initialized with a proper resource item (see Using Resources). You therefore have to create both some resource entries in a properties file and an appropriate sub class of `GenuineAction`.

Have a look at the following example of the `CopyAction`. First, the resources are defined:

```
copy.text=&Copy
copy.image=crystal/16/actions/editcopy.png
copy.key=control C
copy.tooltip=Copy
```

As you can see in the example, the following resources are meaningful in combination with an `Action` class:

- `text`: The text that is shown on a button or menu item. If the text contains an ampersand (&), the character following the ampersand is set as mnemonic.
- `image`: A reference to an image file that has to be available on the classpath. The image is shown as icon.
- `key`: The keyboard shortcut to activate the action. See `javax.swing.KeyStroke#getKeyStroke(String)` for an explanation of the syntax.
- `tooltip`: The tooltip for the action

The class `CopyAction` is implemented as follows:

```
public class CopyAction extends GenuineAction {

  private ClipboardService clipboardService;

  /**
   * Passes an instance of the ClipboardService to be called when the
action
   * is performed.
   */
  CopyAction(ClipboardService clipboardService) {
    super(DefaultResources.COPY);
    this.clipboardService = clipboardService;
  }

  /**
   * Calls the clipboard service.
   */
```

```
  public void actionPerformed(ActionEvent e) {
    if (clipboardService != null) {
      clipboardService.copy();
    }
  }
}
```

The `Action` object is initialized by passing the name of the resource item to be used to the super constructor and performing internal initializations.