Chapter 11 Bean Introspection and Customization

1. The BeanInfo class, the property editor class, and the customizer class are not necessary at runtime.

2. To expose properties whose get and set methods do not follow standard naming patterns, create a new BeanInfo class that extends the SimpleBeanInfo class and implement the getPropertyDescriptors method.

3. To expose events whose registration and removal methods do not follow standard naming patterns, create a new BeanInfo class that extends the SimpleBeanInfo class and implement the getEventSetDescriptors method.

4. To programmatically inspect bean properties and events at runtime, use the classes Class, Field, Method, Modifier, and Inspector in the Java Reflection API.

5. The getAsText method returns the property value as a string to be displayed in the component inspector. The setAsText method can be used to retrieve the value as text from the component inspector, parse the text to an appropriate value, and then invoke the setValue method to set the value in PropertyEditor. This method may throw java.lang.IllegalArgumentException if the text is badly formatted or if this kind of property cannot be expressed as text. The getValue method returns the value as Object. The setValue method sets a value of Object. The getTags method returns a list of allowed property values to be displayed as a drop-down menu in the component inspector.

6. The PropertyEditorSupport class is often used to create custom editors. This support class implements all of the methods in the PropertyEditor interface. Browse the source code for java.bean.PropertyEditorSupport to find out the implementation of the methods getAsText, setAsText, getValue, setValue, and getTags implemented in the PropertyEditorSupport class.

7. To create a custom property editor to display property values in a choice menu in the component inspector, extend the PropertyEditorSupport class, and implement the getTags method.

8. To specify a custom property editor in the BeanInfo class, use the setPropertyEditorClass method in an instance of PropertyDescriptor.

9. The setDisplayName method sets a display name for this feature (property, event, or method). The setValue method associates a named attribute with this feature.

10. Click the Property Editor tab to create property editors from BeansExpress. To generate a BeanInfo class using the BeansExpress, click the BeanInfo tab to display the page for generating Bean Info.

11. The getCustomEditor should be overridden to return an instance of the GUI editor, and the supportsCustomEditor should be overridden to return true. To start a GUI property editor in the component inspector, click the ellipsis in the component inspector for the property. A GUI property is displayed in a dialog box with three buttons OK, Cancel, and Help.

12. The methods setObject passes the bean to be customized to the customizer, so that it can find the state of the bean, update the bean, and interact with the bean. The builder tool starts a customizer by invoking this method. The addPropertyChangeListener method registers a listener for the PropertyChange event. When a build tool starts a customizer, it creates an instance of the customizer and registers a listener to listen for property changes by invoking this method. Whenever a property is changed, the customizer should fire a PropertyChange event to notify the builder tool so that it can update the bean accordingly. The removePropertyChangeListener method removes a listener for the PropertyChange event. To start a customizer, right-click the object in the component tree and choose Customizer from the popup menu.