JIDE Shortcut Editor Developer Guide

Contents

Purpose of This Document

JIDE Shortcut Editor is a special module for shortcut schemas. This developer guide is designed for developers who want to learn how to use *JIDE Shortcut Editor* in their applications.

Basic Concepts

Almost all applications have toolbars and menu bar. On toolbars or menu bar, there are actions or commands. To make it easy to access, commonly used commands are assigned with shortcut keys (also know as accelerators). For example, shortcut key for *copy* command is usually Ctrl-C. There could be more than one shortcut keys for a particular command. For example, on Windows, *copy* also has shortcut Ctrl-Insert. If you put all commands and their shortcuts together, it becomes a kind of schema. We call it shortcut schema.

Different users may have different preferences of what shortcut keys to use. Taking text editor for example, some users are from UNIX background; they want vi or emacs compatible shortcut keys. Other users are Visual Studio users and want shortcut keys to be the same as in Visual Studio. This will make developing application harder because you have to deal with multiple shortcut schemas and allow users to define their own shortcut.

Based on the discussion above, we introduce the following concepts.

Shortcut

Shortcut represents the keyboard keys or mouse clicks. It is an abstract class. The two concrete classes are *KeyboardShortcut* or *MouseShortcut*.

ShorcutSchema

ShortcutSchema associates command with shortcuts. To make it simple, we represent command as a string. In the other words, it's named command. The string value is the name of the command. It has to be unique of course.

Shortcut schemas can form a hierarchy. An application can ship with a default schema. Users can create a new schema that extends the default schema and add their own shortcuts. This is made possible by setParent method in *ShortcutSchema*.

ShortcutSchemaManager

ShortcutSchemaManager manages several shortcut schemas. Only one shortcut schema will be active at one time.

Features of Shortcut Editor

ShortcutEditor is an editor for *ShortcutSchemaManager*. It has many advanced features that you will need in your application. Here are the main features.

- Multiple shortcut schemas support: User can define their own shortcut schemas and switch among those schemas.

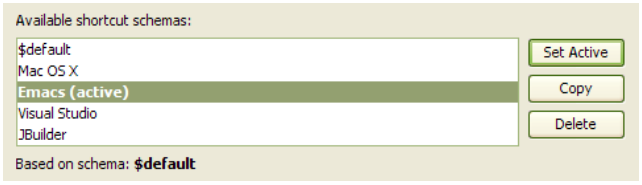


Figure 1 Shortcut Schema List

- Shortcut schema hierarchy: User can define a new shortcut schema extending another existing schema. So all shortcuts for existing schema can still be used while user can add additional shortcuts to the new schema or override existing shortcuts. Overridden shortcuts are displayed in blue text with (*) at the end.

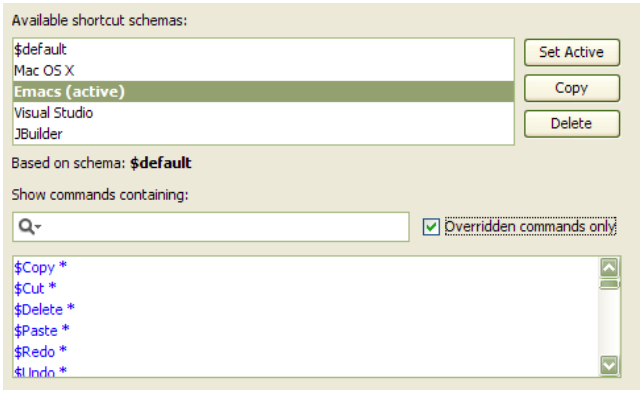


Figure 2 Overridden shortcuts

- Support multi-keystroke keyboard shortcut: For keyboard shortcut, it can support multiple-keystroke, up to three keystrokes. For example, in emacs, Ctrl-X Ctrl-C two keystrokes together will produce exit command.

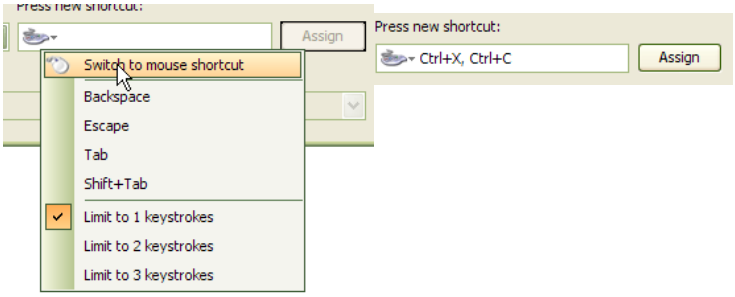


Figure 3 Keyboard shortcut

- Support mouse shortcut: It allows you to define command for mouse single click or double click with or without the modifiers such as Ctrl, Alt, Shift or Command, Option key on Mac OS X. Mouse shortcut is not a common thing in application. However it is widely used in IntelliJ IDEA and improves the usability of IntelliJ IDEA's code editor greatly.

Figure 4 Mouse shortcut

- Automatically resolve shortcut conflicts when user tries to assign the same shortcut key to another command.

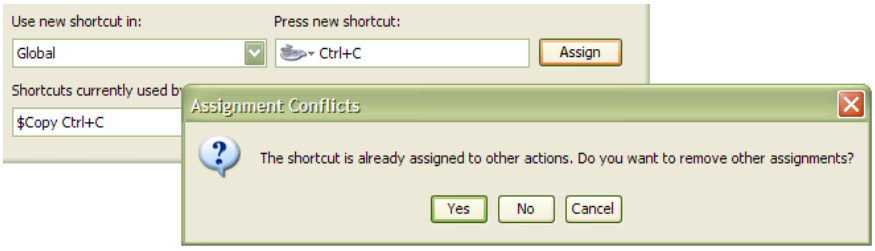


Figure 5 Resolve shortcut assignment conflicts

- Quick filtering function: allow user find the command quickly.
- Context support: Allow user to define different shortcuts for the same shortcut under different contexts.

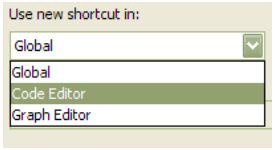


Figure 6 Shortcut context

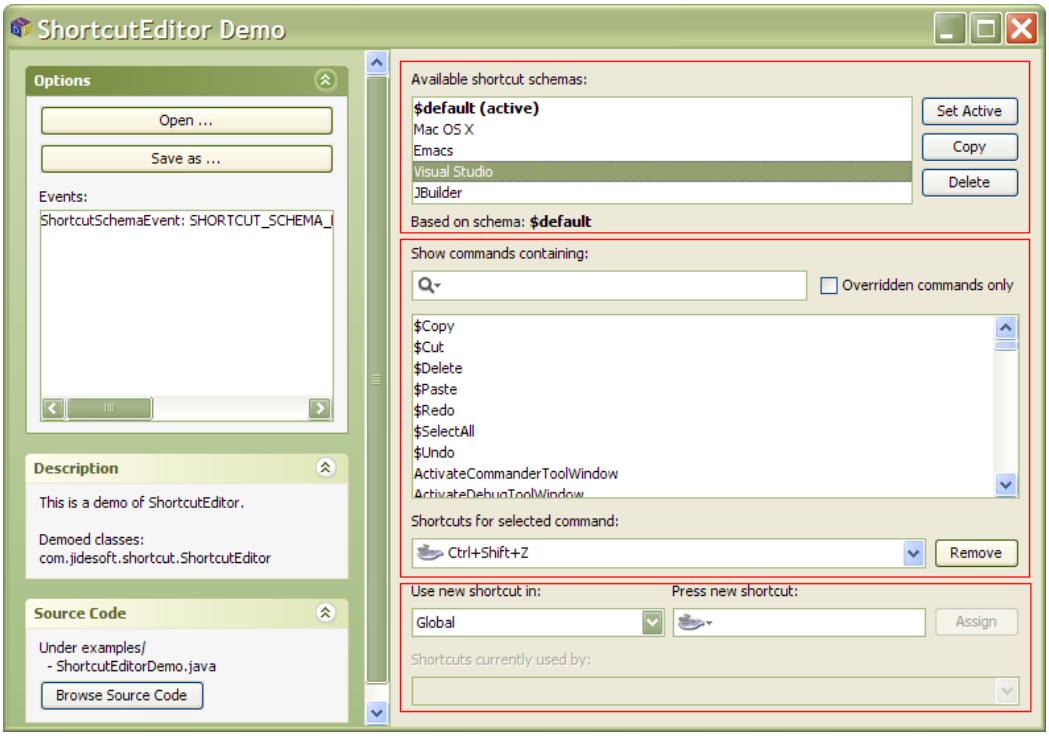
- Listener/Event mechanism.
- Save/load shortcut schemas to/from xml file.

Classes, Interfaces and Demos

Classes – Model Layer	
Shortcut (com.jidesoft.shortcut)	Abstract class representing shortcut. Concrete classes for it are *KeyboardShortcut* and *MouseShortcut*.
ShortcutSchema (com.jidesoft.shortcut)	A data structure for commands and their shortcuts.
ShortcutSchemaManager (com.jidesoft.shortcut)	A manager class that can manage several shortcut schemas.
Classes – View Layer	
ShortcutEditor (com.jidesoft.shortcut)	The main UI component
Classes – Event/Listener	
ShortcutEvent (com.jidesoft.shortcut)	This event is fired by ShortcutSchema to notify interested parties

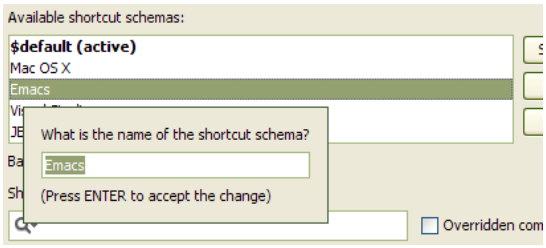
	that shortcut is changed in ShortcutSchema.
ShortcutListener (com.jidesoft.shortcut)	Listener for ShortcutEvent. Call ShortcutSchema#addShortcutListener to register a new listener for ShortcutEvent.
ShortcutSchemaEvent (com.jidesoft.shortcut)	This event is fired by ShortcutSchemaManager to notify interested parties that shortcut schema is changed in ShortcutSchemaManager.
ShortcutSchemaListener (com.jidesoft.shortcut)	Listener for ShortcutSchemaEvent. Call ShortcutSchemaManager#addShortcutSchemaListener to register a new listener for ShortcutSchemaEvent.
Utilities	
ShortcutPersistenceUtils (com.jidesoft.shortcut)	A utility class that supports saving ShortcutSchemaManager to xml file and loading it back.
Demos	
ShortcutEditorDemo (examples\M1. ShortcutEditor)	

User Interface of Shortcut Editor



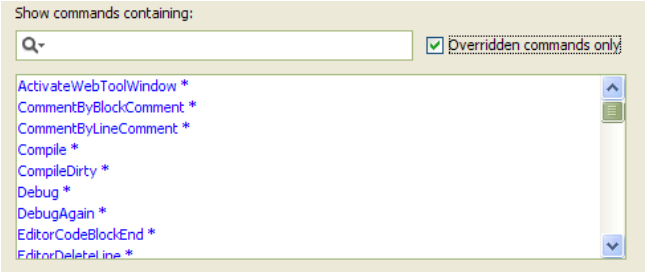
There are three parts in the UI of *ShortcutEditor*. See above. See the screenshot above. Each red rectangle is one part.

The top part is the shortcut schema list section. It lists all available shortcut schemas in *ShortcutSchemaManager*. Active one is in bold. You can select a schema from the list and do things like “Delete”, “Copy”, “Set Active”. You can also double click on the schema to rename it. See below.

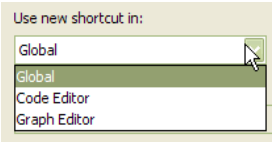


The middle part is the command list section. It will update base on the selected schema in shortcut schema list. All commands are list in a flat list. There is a quick filter field. You can type in part of the command name and the list will only display those that match. When user selects a command, the combo box below the list will display all shortcuts for that particular command. The remove button next to it will remove the shortcut that is currently selected in the combo box. By default, a command can be assigned multiple shortcuts. That’s why it uses combo box to show all shortcuts. But if you pass in false in the “allowMultipleShortcuts” parameter in *ShortcutEditor* constructor, only one shortcut is allowed. If so, the combo box will be replaced by a read-only text field.

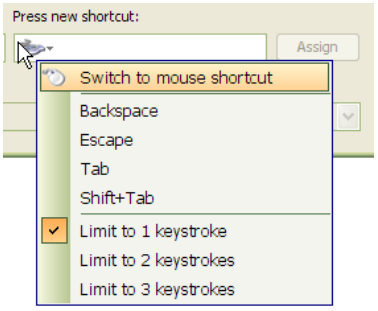
There is also a check box “Overridden commands only”. See below. Overridden commands are those commands that are present in parent schema but child schema overridden them to add or remove shortcuts. In the command list, overridden commands are display in blue with (*) at the end of the command name.



The bottom part is the new shortcut assignment section. It will be enabled only when a command is selected in the command list. A shortcut can assign to a command under certain context. That’s why there is a combo box to allow you to choose which context to use. In the demo, we hard coded three contexts as an example. Just so you know, the context combo box is optional.



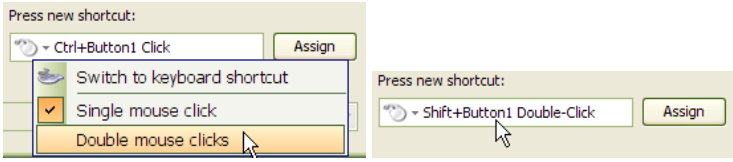
You can type in a keyboard shortcut in the shortcut field. There are a few special keystrokes that you have to use context menu to input them because of they are used for other purposes. For example, Backspace key is used delete keyboard shortcut you typed in. Escape will cancel the dialog since most likely you will use this shortcut editor in a dialog. Tab/Shirt-Tab are used to navigate to next/previous focusable component.



The keyboard shortcut field also supports multiple keyboard shortcuts. See below for an example. You need to switch to multiple keystroke mode using context menu first, then type in keystrokes in order.

Mouse shortcut is also supported. Again, you need to use context menu to switch to mouse shortcut mode. Then click on mouse shortcut field. The mouse shortcut field will detect which mouse button is clicked as well as any modifier keys such as Ctrl, Shift, Alt etc.

However it will not detect double mouse click automatically. You need to use context menu to switch to double mouse click mode.



Shortcut assignment section also can tell you if the new shortcut conflicts with existing ones. When you enter a new shortcut, the conflict combo box below it will show all the conflicts. You can either choose new shortcut that doesn't conflict or you press assign button and it will prompt to remove any conflict ones.

Event

There are two types of events in *ShortcutEditor*.

The first type is *ShortcutEvent* and *ShortcutListener*. Those are for changes inside a shortcut schema. You can add *ShortcutListener* to *ShortcutSchema* and listen to events such as

- *SHORTCUT_ADDED*
- *SHORTCUT_REMOVED*

The second type is *ShortcutSchemaEvent* and *ShortcutSchemaListener*. Those are for shortcut schema change. You can add *ShortcutSchemaListener* to *ShortcutSchemaManager* and listen to events such as

- *SHORTCUT_SCHEMA_ADDED*
- *SHORTCUT_SCHEMA_REMOVED*
- *SHORTCUT_SCHEMA_ACTIVATED*
- *SHORTCUT_SCHEMA_DEACTIVATED*
- *SHORTCUT_SCHEMA_RENAMED*

Persistent ShortcutSchemaManager as XML file

ShortcutPersistenceUtils is the class that can save *ShortcutSchemaManager* as xml file and load them back. The reason we didn't build this feature into *ShortcutSchemaManager* itself but provide a separate class for it is because we think you might have your own to persist it as xml format in your application.

All methods on *ShortcutPersistenceUtils* are static. There are several overload *load()* methods which can load shortcut definition from an xml file, or InputStream, or Document (org.w3c.dom.Document). You can also load shortcut definition from multiple files by setting "append" parameter to true in the *load()* method.

The save() method will save shortcut definition as Document or file. Please note *save()* method will use classes *XMLSerializer* and *OutputFormat*. Those two classes are part of xerces.jar before but now they are also part of JDK5. If you are using JDK5 and above, there is nothing you need to do. However if you are still using JDK1.4.2 or below, you need to include xerces.jar in your classpath. Otherwise you will get *NoClassDefFoundError* during runtime.

Internationalization Support

All of the Strings used in *JIDE Shortcut Editor* are contained in properties file as listed below.

com/jidesoft/shortcut/shortcutEditor.properties

Some users contributed localized version of this file and we put those files inside jide-properties.jar. If you want to support languages other than those we provided, just extract this properties file, translated to the language you want, add the correct postfix and then jar it back into jide-properties jar. You are welcome to send the translated properties file back to us if you want to share it.