

JIDE Components User Guide

PURPOSE OF THIS DOCUMENT

Welcome to the *JIDE Components*, the foundation product in JIDE Software's product line.

This document is focused on introducing basic concepts of *JIDE Components*. It is more focused on the end-user's experience than on all of the technical details necessary to complete a large project. If you want to learn how to use the API provided by *JIDE Components*, or need a quick reference, please refer to the *JIDE Components Developer Guide* that ships with the product.

This document can also serve as a starting point for your own User's Guide or sales literature. Please feel free to use any paragraphs, sentences, or screen shots from this guide to build your initial product documentation. We illustrate concepts using screen shots that are taken from our sample demo, so you will probably want to substitute in your own pictures as soon as possible, but this should at least be a pretty good starting guide.

WHAT IS JIDE COMPONENTS

Thousands and thousands of valuable development hours are wasted on rebuilding components that have been built elsewhere. Why not let us build those components for you so you can focus on the most value-added part of your application?

So what are those components and how do we choose them?

First of all, they are generic. Our components provide solutions for building IDEs (Integrated Development Environments). If you have ever worked to build an IDE or similar product, you will find our components are very familiar to you. Not because you've seen them before but because they are very generic components. Most likely you will say "hmm, I can use this component in my application".

Secondly, they are extensible. We never assume our components will satisfy all your requirements. So in addition to what we provide, we always leave an extension point so that you can write your own code to extend the component. Believe it or not, our whole product strategy is based on the extensibility of each component we are building. We try to cover all the requirements we can find and to build truly general, useful components, but at some point users will likely find a need we didn't address. That's fine! Our components allow you to "help yourselves".

Last but not least, they'll save you time. You use a component because you think it will be faster to build on top of it than to start from scratch. If the component is very easy, you probably will build it yourself. If you find a component is too complex and too hard to use, you probably will still build it yourself. With this in mind, we carefully chose what components to include in *JIDE Components*. You may notice that we didn't provide a large number of components. Why? Because we are "picky". Our pickiness means that all our carefully chosen components will definitely save your precious time.

PACKAGES

The table below lists the packages in the *JIDE Components*.

| Packages | Description |
|--|--|
| com.jidesoft.swing com.jidesoft.swing | Common components. We expose some of classes in our documentation. For those not exposed, either they are just used internally or they are not mature enough to be publicly used. Common components. We expose some of classes in our document. For those not exposed, either they are just used internally or they are not mature enough to be publicly used. |
| com.jidesoft.document com.jidesoft.document | DocumentPane – tabbed-document interface implementation similar to what you see in Visual Studio .NET IDE. DocumentPane – tabbed-document interface implementation similar to what you see in Visual Studio .NET IDE. |
| com.jidesoft.status com.jidesoft.status | StatusBar – A generic status bar implementation. StatusBar – A generic status bar implementation. |
| com.jidesoft.pane com.jidesoft.icon | CollapsiblePane, FloorTabbedPane and other pane components. Icon related classes |
| com.jidesoft.icon com.jidesoft.util | Icon related classes. Utilities classes |
| com.jidesoft.util com.jidesoft.tipoftheday | Utilities classes. Tip of the Day dialog component |
| com.jidesoft.options | Dialog for Options or Preferences (Coming soon) |
| com.jidesoft.wizard | Wizard (Coming soon) |

A general comment on our naming convention: If the class is modified from or based on an existing Swing/AWT class, we prefix the original Swing/AWT class name with Jide. For example, JideTabbedPane. From the name you can tell it's based on JTabbedPane. If it's a completely new component that doesn't exist in Swing/AWT, we don't prefix. For example, StatusBar, SidePane.

We will add more and more components to *JIDE Components* in the future. And we will keep the same package organization. If the component is complex enough, there will be a separate package for it. If the component is small or needs to be shared by other components, we will put them under com.jidesoft.swing.

DOCUMENT PANE

Background

Most software applications have a “document” concept. For example, in a typical Java IDE, the Java file is the “document”. In a photo processing application, an image file such as a gif or jpeg is the “document”. The document is usually the center of an application, so selecting the right model to manage those documents is very important.

Looking at the history of how applications organize documents, there are two popular models, known as SDI and MDI.

SDI – Single-Document Interface

Most of early IDEs used SDI. They can only deal with one document at a time. If you want to open another document, you must either close the current one or open a completely new instance of the IDE. Notepad.exe is an example of this. Simple as it is, the drawbacks of it are obvious. One big drawback is user can only view one document at a time. However there are still a lot of applications using SDI, especially applications for consumers and home users, because SDI is simple to use.

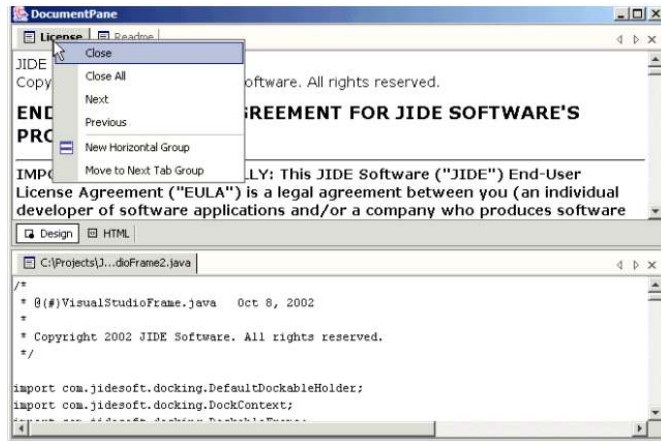
MDI – Multiple-Document Interface

In MDI, you can view/edit multiple documents at the same time. You can either use Windows menus or hotkeys (such as Ctrl-Tab) to navigate between those documents. MDI overcomes the major drawbacks of SDI, but it has some drawbacks of its own. The two biggest issues are that it wastes screen space if not maximized, and that it is difficult to navigate if maximized. Many users are frustrated when child windows are locked into the parent window in an MDI interface and find it very hard to use. Pure MDI got popular for several years and died down because those drawbacks. If your application is complicated and you are considering using an MDI interface, you should investigate alternative, MDI-like designs. We provide a Tabbed-Document Interface (TDI) to satisfy your need.

In *JIDE Components*, we took a new way to view/edit multiple documents – Tabbed-Document Interface. We didn’t invent this approach. Many IDEs already use an interface of this kind, such as Visual Studio .NET, IntelliJ IDEA and JBuilder. It’s an obvious trend in recent software applications.

What does [a](#) Tabbed-Document Interface look like?

To understand the tabbed-document interface, you should first understand tabbed panes. Each tab is a document or a document view. We call a tab dedicated to holding a document (as opposed to a side window) a document tab. A document tab shows the document title and an optional icon. You can use the icon to indicate the document type, such as Java file or XML file. Several document tabs form a document group. Our implementation allows multiple document groups so that you can view multiple sets of documents at the same time.

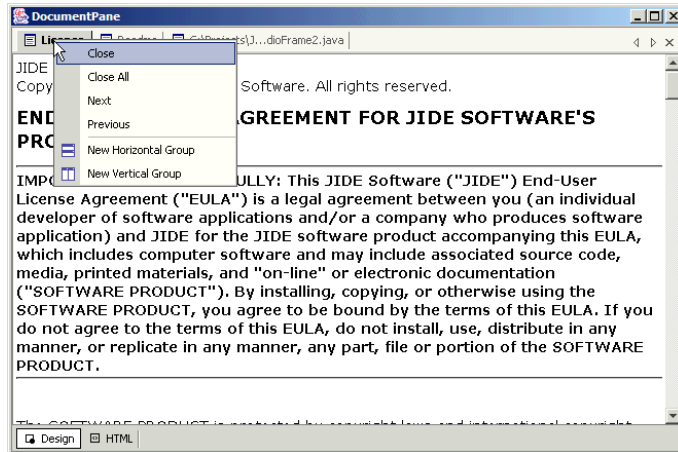


How will my users use it?

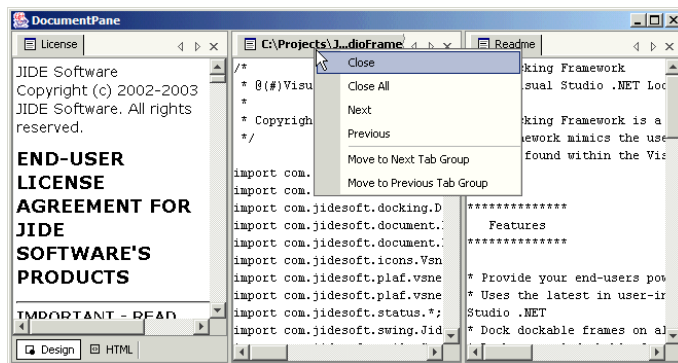
All “generic” operations related to a document pane can be done through the document tabs.

- Clicking on the tab will active that document.
- Right clicking on a document tab will popup a context menu. The built-in menu already comes with a lot of built-in functionality and can be extended.

The screen shot above shows a context menu that appears when a user right clicks on the “Readme” tab. In the menu, “Close” means close the currently selected document and “Close All” means close all documents (i.e. all the tabs that are within the document tabbed pane). The “x” button to the right is the equivalent to “Close”. “Next” and “Previous” mean set the focus on the next and previous document, respectively. “New Horizontal Group” means create a document group and put the currently selected document into that new group. Using the screen shot above as an example, if you choose “New Horizontal Group”, there will be three document groups, with each group having one document. “Move to Next Tab Group” means move the selected document to next document group. Please note that the context menu only lists actions that are allowed. See below for another example. Since there is only one document group in the example below, you are allowed to create a new document group either vertically or horizontally. That’s why you see both “New Horizontal Group” and “New Vertical Group” in the context menu.



In the example below, there is no menu choice for either “New Horizontal Group” or “New Vertical Group”. Since there is only one document in the current document group, it doesn’t make sense to create another group. However you can move it to another group.



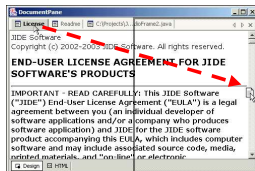
Another, simpler way to move windows and create groups is to use drag and drop.

- Drag a document tab horizontally within a tab area and the tab order will change.

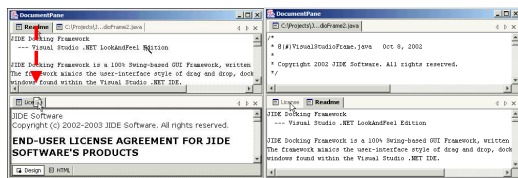


COPYRIGHT © 2002-2003 JIDE SOFTWARE. ALL RIGHTS RESERVED

- Dragging a tab and dropping it near the south border or east border will create a new horizontal or vertical tab group, respectively.



- Dragging a tab and dropping it in another document group's tab area will move the document to that document group.



- Dragging a tab and dropping it in the middle of a document will popup a menu allowing you choose.



- Anytime during dragging, pressing ESC will cancel the dragging.

You can also customize the appearance of the title. In the example below, the title of the Java file is too long, so we put “...” in the middle to omit some characters. You can still tell or guess where the file is located and what the file name is.



We understand you may have different requirements, so we allow you to customize both the content and format of the title. Depending on what you want to customize, you can use one of two methods.

If you just want to display a title with different text but keep the color and style the same (bold for active document and plain for other documents), you can call `DocumentPane.setTitleConverter(StringConverter)`. `StringConverter` is an interface that can convert from an input string to an output string. The input string is the actual title and the output string is what is to be displayed.

COPYRIGHT © 2002-2003 JIDE SOFTWARE. ALL RIGHTS RESERVED

If you want a complete customization of the title, including color, style and text, you can write your own class extending `DocumentComponent` and override the `getDisplayTitle()` method. Since you can use html in the returned display title, you have a lot of flexibility. Below is example source code for coloring the title. In this example, not only “...” is in the beginning of the title, rather than the middle, but also the color is changed.



STATUS BAR

StatusBar is never a central part of an IDE but every IDE has it. A well-designed status bar can make a user interface user-friendly because it gives immediate, non-intrusive responses to user actions.

StatusBar and StatusBarItem

The main class of StatusBar is com.jidesoft.swing.StatusBar. It is divided into a series of items of type StatusBarItem. In the example below, the whole thing is the StatusBar and each gray box is a StatusBarItem. You can use one of several StatusBarItem's that we developed or you can create your own StatusBarItem.



ProgressStatusBarItem

ProgressStatusBarItem is the most visible item. It can display a text message or it can display progress in an optional progress bar. You can activate (display) the progress bar by calling setProgressStatus("Running"), and you can set the percentage finished using setProgress(int). In the below status bars, the second one is in running mode.



LabelStatusBarItem

If you just want to display a text on status bar, you can use this. It is simply a JLabel inside a JPanel. Below is an example.



ButtonStatusBarItem

This is similar to LabelStatusBarItem, just replace the JLabel with a JButton.



TimeStatusBarItem

TimeStatusBarItem is used to display a clock or calendar on the status bar. You can customize the format of how to display that the time and date are displayed in.



Formatted: Emphasis

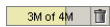
Formatted: Caption

Formatted: Caption

TimeStatusBarItem extends LabelStatusBar so it has all methods of LabelStatusBarItem.

MemoryStatusBarItem

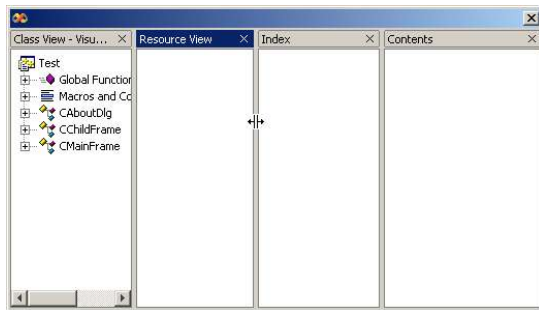
We borrowed the idea of MemoryStatusBarItem from IntelliJ IDEA. The MemoryStatusBarItem is used to display currently used memory vs. total memory in the current JVM. It also allows a user to manually run garbage collection by pressing the garbage can button. In the example below, 3M is the amount of used memory and 4M is the total memory.



PANE

JideSplitPane

JSplitPane is a useful Swing component. However it has one limitation – it can only split into two windows. If you want to split into three windows, you have to use two JSplitPanes. That may be OK. But if you want to split into four or five or more, you will quickly get into trouble maintaining so many JSplitPanes. As you can see in *JIDE Docking Framework*, we need to be able to split a panel into any number of windows. JSplitPane obviously cannot meet this need gracefully, so we developed JideSplitPane.



Above is an example of JideSplitPane which is split into four parts. Each divider can be moved by mouse to resize the components beside it.

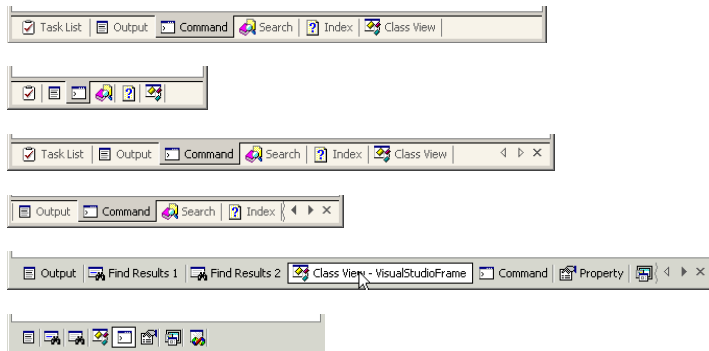
JideTabbedPane

JideTabbedPane is similar to JTabbedPane. The difference is JideTabbedPane:

- Has an option to hide the tab area if there is only one component in a tabbed pane.
- Has an option to resize the tab width so that all tabs can fit in one row.

- Has an option to show scroll left and scroll right buttons and a close button in the upper right of the tab area.
- Only allows TOP and BOTTOM tab placement in the current version.

Below are various examples of JideTabbedPane with different configuration settings.



CollapsiblePane

CollapsiblePane, as the name indicates, is a pane which can be collapsed. See below for an example.

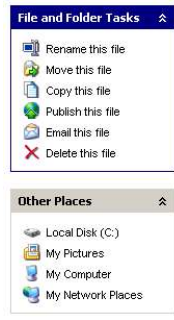
Before collapse



After collapse



Here is the screenshot of an example in JideDemoFrame. You can see this screenshot when clicking on “View” Menu – “CollapsiblePane”.



In additional, there are different styles of CollapsiblePane. See below for another example of CollapsiblePane.



FloorTabbedPane



FloorTabbedPane is another type of tabbed pane. Typical tabbed pane has many panels and corresponding tabs. User can click on tab to choose which panel to view. FloorTabbedPane also have many panels. However, instead of using tabs, it just uses buttons to switch between panels. Buttons are organized vertically, as floors in a storied building. That's how it gets the name of FloorTabbedPane. One famous example of it is the Outlook Bar in Microsoft Outlook product.

Left is a screenshot which you can find in JideDemoFrame when clicking on "View" Menu – "FloorTabbedPane".

TIPS OF THE DAY

A lot of applications use Tips of the Day dialog to get a new user up and running quickly. The content of Tips of the Day is little different from the online help document, but most users prefer to read Tips of the Day because they are concise and they only take a short amount of time per day. To allow more applications to easily create their own Tips of the Day, we created the TipOfTheDayDialog component.

