JIDE Docking Framework User Guide

PURPOSE OF THIS DOCUMENT

Welcome to the *JIDE Docking Framework*, the most advanced framework for developing dockable windows in Swing.

This document is focused on introducing basic concepts of *JIDE Docking Framework*. It is more focused on the end-users' 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 Docking Framework,* or need a quick reference, please refer to *JIDE Docking Framework 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 be at least a pretty good starting guide.

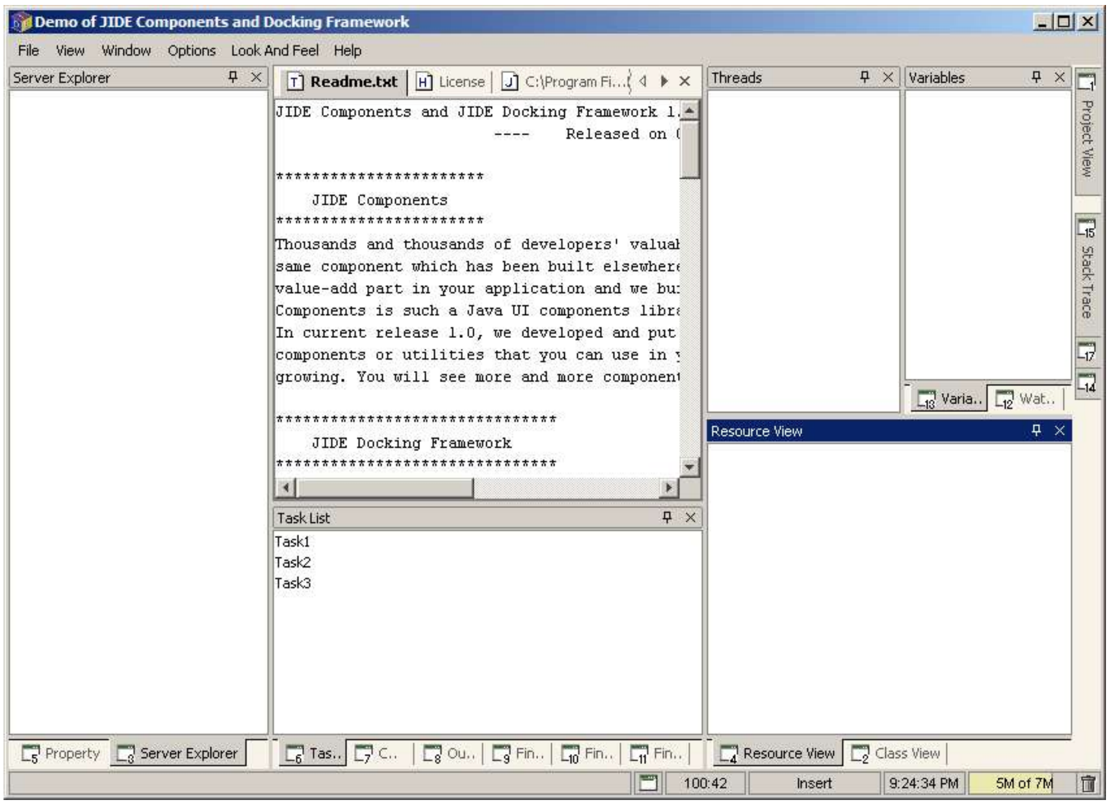
WHAT IS THE JIDE DOCKING FRAMEWORK

Since AWT/Swing was introduced, many companies have embraced this new technology and made many excellent user interfaces using it. However one thing that is obviously missing in most Swing applications is the dockable window. If you had ever used the Visual Studio .NET IDE, you already appreciate the value of dockable windows. Users have come to expect them because they greatly increase the application's ability to display information neatly. Without the ability to nicely group information into fixed areas, your application could look cluttered and confusing.

We found several open-source or commercial products that implement dockable window but unfortunately none of them are built using Swing. To meet this need, we created the *JIDE Docking Framework,* which allows you to produce similar docking functionality to what you see in the Visual Studio.NET IDE but using only Swing.

DOCKABLEFRAME

We made a demo application that basically imitates Visual Studio.NET IDE to illustrate the usage of the *JIDE Docking Framework*. This demo is part of the *JIDE Docking Framework* product, and its source code is available. Figure 1 is a screenshot of the demo.



```
JIDE Components and JIDE Docking Framework 1.
                        ----    Released on (

***********************
    JIDE Components
***********************
Thousands and thousands of developers' valuab
same component which has been built elsewhere
value-add part in your application and we bu:
Components is such a Java UI components libra
In current release 1.0, we developed and put
components or utilities that you can use in y
growing. You will see more and more component

********************************
    JIDE Docking Framework
********************************
```

```
Task1
Task2
Task3
```

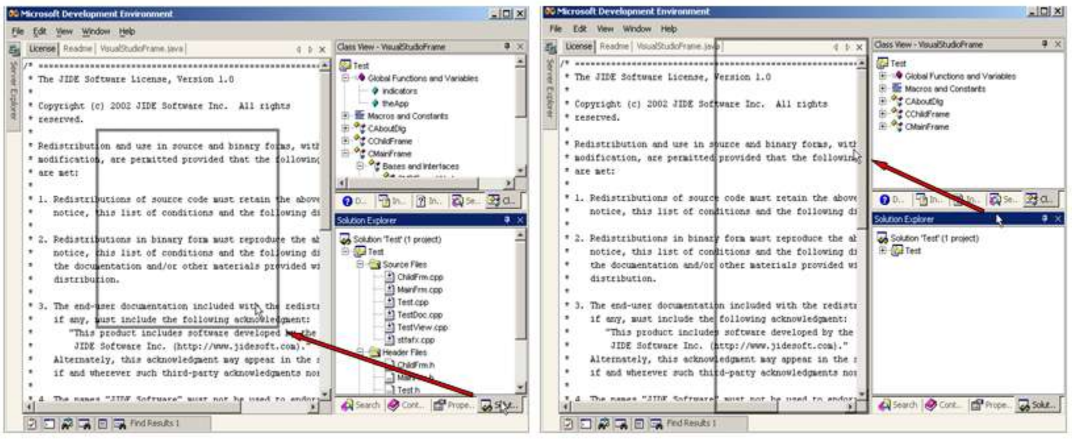
Figure 1 A demo IDE based on JIDE Docking Framework

JIDE Docking Framework provides a rich customization experience for users of your application, allowing the users to move dockable frames to get their favorite screen configurations. For example, some users prefer to have all of an application's options out in plain sight, while other prefer to concentrate solely on the task at hand without being distracted by options they don't need right now.

Dockable frames in the *JIDE Docking Framework* support all kinds of distinct arrangements to maximize the user's experience. They can be:

- Docked to the all sides of the central client area of your main JFrame.
- Floating above your main JFrame.
- Collapsed to all sides of your main JFrame, only appearing when the user moves their mouse over the button on the side bar.
- Docked with other Dockable frames in a tabbed pane.
- Put in a complex hierarchy along with other dockable frames in any area.

You can dock windows to any area of your frame simply by dragging the dockable window's title bar tab (if tab-docked) to the desired area. The outline will change shape based on the current destination. If it snaps to another window's edge, then when you release the mouse, it will dock to that area. You can hold CTRL key at any time to avoid docking.



```
/* ************************************************************
 * The JIDE Software License, Version 1.0
 *
 * Copyright (c) 2002 JIDE Software Inc.  All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with
 * modification, are permitted provided that the following
 * are met:
 *
 * 1. Redistributions of source code must retain the above
 *    notice, this list of conditions and the following di
 *
 * 2. Redistributions in binary form must reproduce the ab
 *    notice, this list of conditions and the following di
 *    the documentation and/or other materials provided wi
 *    distribution.
 *
 * 3. The end-user documentation included with the redistr
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        JIDE Software Inc. (http://www.jidesoft.com)."
 *    Alternately, this acknowledgment may appear in the s
 *    if and wherever such third-party acknowledgments nor
 *
```

Microsoft Development Environment
File Edit View Window Help
License Readme VisualStudioFrame.java
Class View - VisualStudioFrame
Solution Explorer
Server Explorer
Find Results 1

Figure 2 Dock or float window to any place

You can even dock dockable frame inside another frame to create tab-docked frames. Basically you just move the mouse pointer over other frame's title bar during dragging, the contour will change to a tab shape, indicating it will become a tab pane within the destination frame.

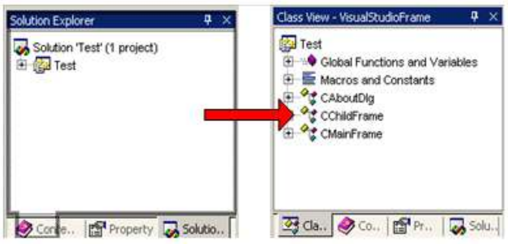


Figure 3 Tab-dock

If you click on the auto-hide button indicated by red rectangle below, all frames in the same tabbed pane will collapse to the nearest side of the JFrame. Clicking on a button or hovering the mouse over it causes the auto-hid frame to "slide out". This docking technique allows you to provide easy access to infrequently used frames, thereby maximizing the screen real estate. You can resize the sliding frame as well. The cursor will change shape when the mouse stops over the border of the sliding frame.

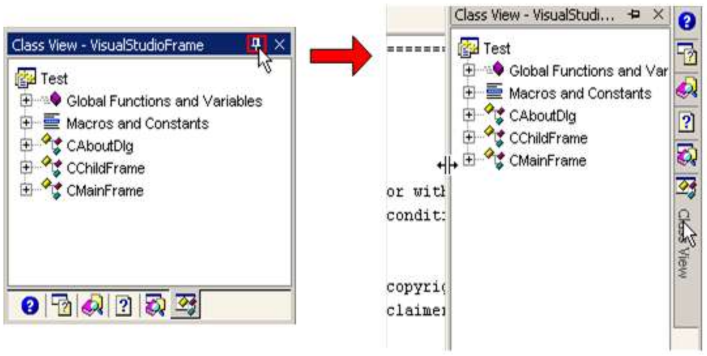
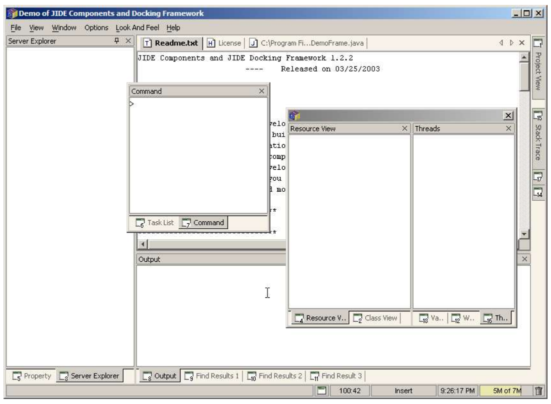

Class View - VisualStudioFrame
Test
Global Functions and Variables
Macros and Constants
CAboutDlg
CChildFrame
CMainFrame
Class View - VisualStudi...
Class View

Figure 4 Collapse Dockable frames to any side of JFrame

You can even float one frame and several frames out of the main JFrame.

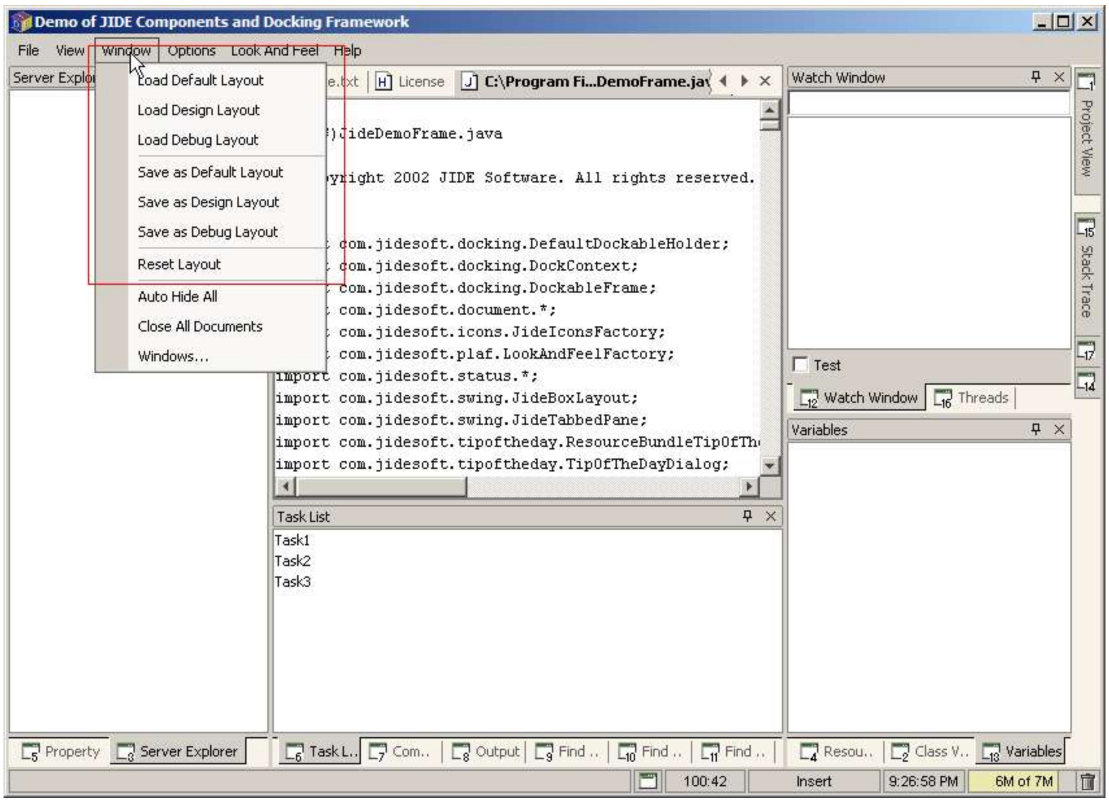


```
JIDE Components and JIDE Docking Framework 1.2.2
                    ----   Released on 03/25/2003
```

Figure 5 Floating windows

At any time, double clicking on title bar or tab will toggle between the floating state and the docked state.

In additional to the features above, layout information can be persisted automatically when you exit the application. So when you close the application and open it again, the layout is exactly the same as it was before closing. You can also programmatically save the current layout data of your dockable frame arrangements into a named profile and load them back later. As an example of why this is important, in Visual Studio.NET, the arrangements of windows are different during design-time and run-time. *JIDE Docking Framework* provides the same feature.



```
)JideDemoFrame.java

yright 2002 JIDE Software. All rights reserved.


com.jidesoft.docking.DefaultDockableHolder;
com.jidesoft.docking.DockContext;
com.jidesoft.docking.DockableFrame;
com.jidesoft.document.*;
com.jidesoft.icons.JideIconsFactory;
com.jidesoft.plaf.LookAndFeelFactory;
import com.jidesoft.status.*;
import com.jidesoft.swing.JideBoxLayout;
import com.jidesoft.swing.JideTabbedPane;
import com.jidesoft.tipoftheday.ResourceBundleTipOfTh
import com.jidesoft.tipoftheday.TipOfTheDayDialog;
```

Figure 6 Different layouts

STATE AND MODE OF DOCKABLEFRAME

A dockable frame can have five states – Docked, Floating, Auto-hide (Hidden), Auto-hide (Showing) and Hidden. These states are mutually exclusive. While most states are easy to understand, there may be some confusion between the docked and floating states. For example, a frame can be docked with other frames within the same floating window. As long as dockable frame is in the main JFrame, it is in docked state. Otherwise, it's floating state.

You can transfer almost from any one state to any other, but there are a few exceptions. For example, you cannot jump from auto-hide (Hidden) or auto-hide (showing) directly to floating mode and vice versa. To switch states:

From Docked to Floating state	• Drag on title bar or tab and drop when the outline isn't in a docked shape. • Drag on title bar or tab and drop when the outline is in a floating window. • Right mouse click on title bar and select "Floating" • Double click the mouse on a docking frame's title bar or tab
From Floating to Docked state	• Drag on title bar or tab and drop when the outline is in a docked shape. • Right mouse click on the title bar and unselect "Floating" • Double click the mouse on the floating frame's title bar or tab
From Docked to Auto-hide (hidden)	• Click on the auto-hide button on a docking frame's title bar. • Right mouse click on the title bar and select "Auto Hide"
From Auto-hide to Auto-hide (Showing)	• Hover the mouse over or click on the auto-hide (hidden) frame's button on the side bar. Clicking will show the frame and activate it.
From Auto-hide (Showing) to Docked	• Click on the stop auto-hide button on an auto-hide (showing) frame's title bar. • Right mouse click on the title bar and unselect "Auto Hide"
Any state to Hidden	• Click on the close button on title bar
Hidden to previous state	• This can only be achieved programmatically. For example, you can create a JMenuItem to do so. In the action of JMenuItem call showFrame(String title)

ADDITIONAL OPTIONS

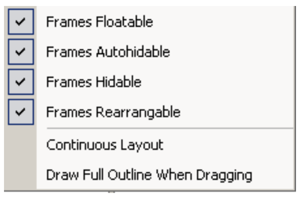


Figure 7 Options

The sample application has a few options that you can select to change the frame behaviors globally. Those options can be selected via the Edit menu. See above.

Floatable: This indicates whether the application's frames can be undocked. This is a global option, meaning it will modify each frame's setting.

Autohidable: This indicates whether the application's frames can be set to the autohide (hidden) state.

Hidable: This indicates whether the application's frames can be closed (or hidden).

Rearrangable: This indicates whether the application's frames can be moved around.

ContinuousLayout: This indicates whether the components continuously redraw themselves when the user resizes the split pane or moves a pane. The default is false. Note that performance could suffer when this is set to true.

OutlineMode: When a dockable frame is dragged, an outline of the frame is painted. In the initial version of JIDE Docking Framework, this outline was clipped to the main JFrame, so the pane could be only partially visible. Since version 1.2.1 of Docking Framework, if OutlineMode is true (1), the dockable frame is painted completely even if it strays outside the main window borders. To maintain compatibility with prior versions, the default OutlineMode setting is 0.