

DIGISLICE™

**DigiSlice AppComposer™
Installation Guide**

DigiSlice AppComposer Installation Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Copyright Notice

Copyright © 2003 DigiSlice Corporation Corporation.

All Rights Reserved.

This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent in writing from DigiSlice Corporation., 2020 SW 8th Ave Suite 343., West Linn, OR 97068, USA.

ALL EXAMPLES WITH NAMES, COMPANY NAMES, OR COMPANIES THAT APPEAR IN THIS MANUAL ARE IMAGINARY AND DO NOT REFER TO, OR PORTRAY, IN NAME OR SUBSTANCE, ANY ACTUAL NAMES, COMPANIES, ENTITIES, OR INSTITUTIONS. ANY RESEMBLANCE TO ANY REAL PERSON, COMPANY, ENTITY, OR INSTITUTION IS PURELY COINCIDENTAL.

Every effort has been made to ensure the accuracy of this manual. However, DigiSlice makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability and fitness for a particular purpose. DigiSlice shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual or the examples herein. The information in this document is subject to change without notice.

Trademarks

AppComposer, DigiSlice, and the DigiSlice logo are U.S. trademarks of DigiSlice Corporation.

Other product names mentioned in this manual may be trademarks or registered trademarks of their respective companies and are the sole property of their respective manufacturers.

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

C O N T E N T S

1	Introducing DigiSlice AppComposer	5
	What is AppComposer	5
	Advantages of Component-based Applications	5
	Benefits of Using AppComposer	6
	Using AppComposer	7
	Conventions Used in This Manual	8
2	Installing DigiSlice AppComposer	9
	System Requirements	10
	DigiSlice AppComposer Installation	11
	Installation Directory	11
	Shortcut Location	13
	Choosing the Java location	15
	Invoking AppComposer	17
	Licensing AppComposer	17
	Installing components	20
	Installing a Media Library	20
	Reinstalling and Upgrading AppComposer	21
	Database	21
	JBoss EJB Server	22
	Dreamweaver extensions	22
3	Where to go from here	25
	Learning to Use DigiSlice AppComposer	25
	Documentation	26
	Accessing AppComposer Documentation	26
	AppComposer Tutorials	26
	AppComposer Manuals	27
	Other AppComposer Documentation	27
	ObjectInfo Documentation	27
	Offline Documentation	29

Composer Installation Directories29

Introducing DigiSlice AppComposer

What is AppComposer

DigiSlice AppComposer offers a new approach to building applications using a visual authoring paradigm. Using open standards-based technology, AppComposer allows developers to build complex applications out of Java components. Developers will find that they do not have to learn any proprietary languages because AppComposer enables the developer to use components that adhere to open specifications. Developers can utilize components from any source. If the imported components comply with Java specifications, they can be utilized in Composer.

Advantages of Component-based Applications

The advantages of building applications out of components are well-known. These advantages include: dramatically increased reliability, supportability, scalability, and reusability of the code. Traditionally, the hardest part of building applications from components has been connecting the components together. DigiSlice AppComposer solves this integration problem by providing a powerful and comprehensible visual authoring environment that makes it easy to assemble components into applications.

Benefits of Using AppComposer

Developers can use DigiSlice AppComposer to build stand-alone java applications, browser-based applets, java servlets, JSP pages, and visual and non-visual components.

Projects developed in DigiSlice AppComposer are:

- ◆ **Standards-based:** AppComposer assembles applications from standards-based components, which can be acquired from a number of sources or built with any Java development tool. Conforming to standards when writing your own components vastly improves the ease of use and reusability of those components.
- ◆ **Portable:** Applications built in AppComposer run on any Java-enabled platform, including all major J2EE application servers. AppComposer itself is written in Java.
- ◆ **Powerful:** AppComposer allows the use of Java expressions or entire pieces of Java code, in case the predefined components do not provide the functionality required by the application.
- ◆ **Flexible:** Applications built with AppComposer are easy to maintain or change. The application can be easily modified over time as the needs of the business change. Applications can be built generically, such as a generic e-commerce application, and then easily customized to meet specific and individual needs.
- ◆ **Reusable:** Functionality and components built in AppComposer are completely reusable in other applications or development environments. The process of using AppComposer inherently creates new components to encapsulate pieces of application functionality.
- ◆ **Fast:** DigiSlice AppComposer provides a run-time editing and debug environment for iterative development and rapid prototyping.

Using AppComposer

Using DigiSlice AppComposer to build an application, developers will often find themselves in the middle of the development process, communicating, on the one hand, with application user interface designers, and, on the other hand, with data intensive component constructors and suppliers. When using AppComposer, part of the task is to find the components needed to create the desired application. The other part of the task is to combine these components into a set of core functional pieces to create a fully functioning application. Often, the components needed will already exist within the organization or in third-party marketplaces. In some situations components will need to be created from scratch.

For example, when building a web application with DigiSlice AppComposer, the AppComposer developer typically works with other developers who have expertise in creating HTML/JSP web pages and also with developers in the IT department who can supply software components to access the organization's database store. The AppComposer developer often acts as the coordinator between different parts of the development team, orchestrating the integration of the user interface components with back-end data components.

DigiSlice AppComposer allows development professionals to easily assemble components from across the entire organization into finished applications.

Conventions Used in This Manual

This manual uses the following typographic conventions:

- ◆ Names of files, resources, classes, methods, and variables, as well as code fragments and information you type, appear in the `code` typeface.
- ◆ Metanames appear in *italic*. A metaname is a descriptive placeholder for a real name. For example, when referring to a capsule's `.zac` file, we say *capsulename.zac*, rather than specifying a specific capsule name.
- ◆ Names of menus, menu items, buttons, and other user interface elements appear in this **typeface**.
- ◆ Keys you press at the same time are shown as follows: Control-G (press and hold the Control key and then press the G key). Please note that even though the letter keys are listed in uppercase, you should not hold down the Shift key when executing these key combinations unless the Shift key is listed as part of the combination.

Installing DigiSlice AppComposer

This chapter explains the installation and configuration of DigiSlice AppComposer. This chapter focuses on installation and configuration on Windows (XP, 2000 & NT), Linux, Solaris 8 (SPARC & X86) and Mac OS X.

AppComposer uses the InstallAnywhere installer from ZeroG software to make installation as easy as possible. For most users, installing AppComposer consists of executing the self-contained installer program named `AppComposerInstall<OS> [VM] -<extension>`

The [VM] version of an executable will install a Java SDK along with the application, if you already have the required SDK, you won't need to download the version with the SDK. On Mac OS X the SDK is part of the operating system and provided by the system vendors, and therefore we won't be shipping a version that includes a SDK.

In addition to AppComposer, the installer also installs the JBoss application server for building EJB (Enterprise Java Bean) applications (Enterprise Edition only) and the Hypersonic SQL database.

System Requirements

- ◆ **Operating System** -High-end workstations running either Microsoft Windows (Windows NT SP6, Windows 2000 or Windows XP), Linux (RedHat 7.2 or Mandrake 9 or higher), Solaris 8 (Sparc and X86) or Mac OS X. Since AppComposer is a Java application, it should run on other platforms, however, it is only officially supported on platforms that we have in our lab. (**Note:** It will not currently work on Windows9x).
- ◆ **Processor Speed and Memory**- At least a 400 mhz Pentium II processor with 128 MB of memory, 256 MB is highly recommended.
- ◆ **Hard Disk Space**- A full install of AppComposer and its add-on component tools requires at least 110 megabytes of hard disk space, depending on the optional features chosen for installation.
- ◆ **Note**- You must have installed a Java Software Development Kit (SDK or JDK) before AppComposer can be run.You can download a version of J2SE SDK from Sun Microsystems from their website at <http://java.sun.com> (make sure you download and installed the SDK as the JRE is not sufficient). You must use version 1.3.1 or greater for this release. You have to have administrative rights on Windows NT, Windows 2000 and Windows XP in order to install software. On Linux/Solaris/Mac OS X you have to have write permissions in the directory where the software will be installed.
- ◆ We highly recommend that you use JDK 1.4.1 on all operating systems, on Mac OS X version 1.4.1 is the only official supported JDK.

DigiSlice AppComposer Installation

During installation, the installer will present several choices to you. The following sections describe these selections and the suggested default choices.

To start the installation, launch the installer either from the CD or the file you downloaded.

Installation Directory

The installer installs AppComposer in the following directory as default.

Windows - C:\AppComposer

Linux - /opt/AppComposer

Solaris - /opt/AppComposer

Macintosh - /Applications/AppComposer

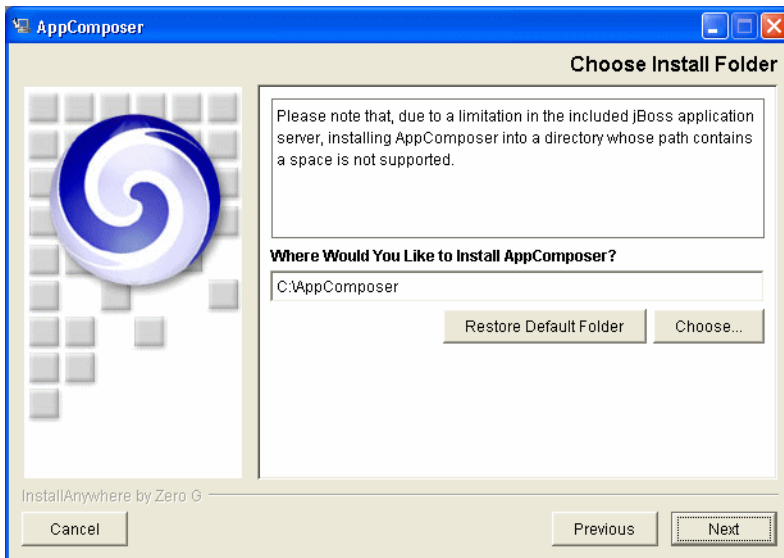


Figure 2-1 Windows directory



Figure 2-2 Linux/Solaris directory



Figure 2-3 Mac OS X directory

Shortcut Location

Windows - The installer will create a new program group called DigiSlice AppComposer by default as seen in the following screen, but you can change the location by specifying a different one.

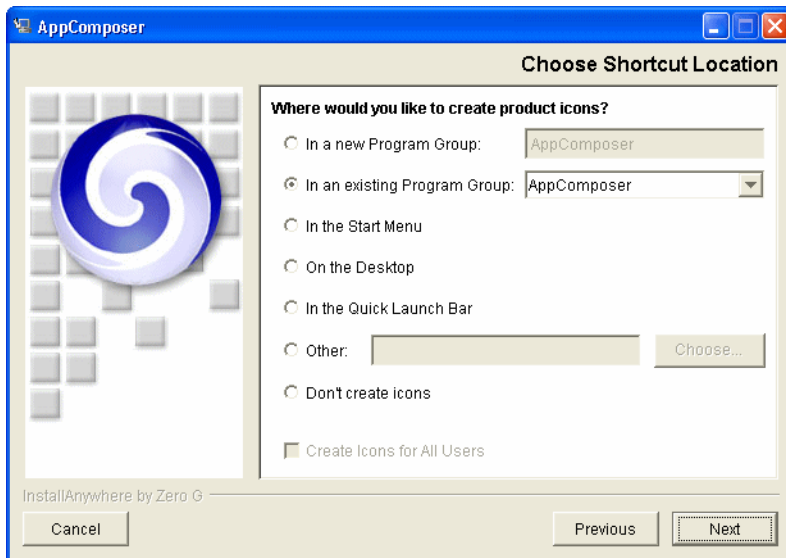


Figure 2-4 Windows Shortcut Location

Linux/Solaris - On Linux, Solaris and other Unix systems, the following dialog is displayed to allow you to have the installation program create symbolic links to special files and important documents. These files can be found in the installation directory, so it is not required that you create these links.

AppComposer - Link to the executable

Documentation.html - Link to the top level documentation ReadMe.txt -

Important information... this is display during the installation process

Uninstall_AppComposer - Link to an executable that will uninstall the application

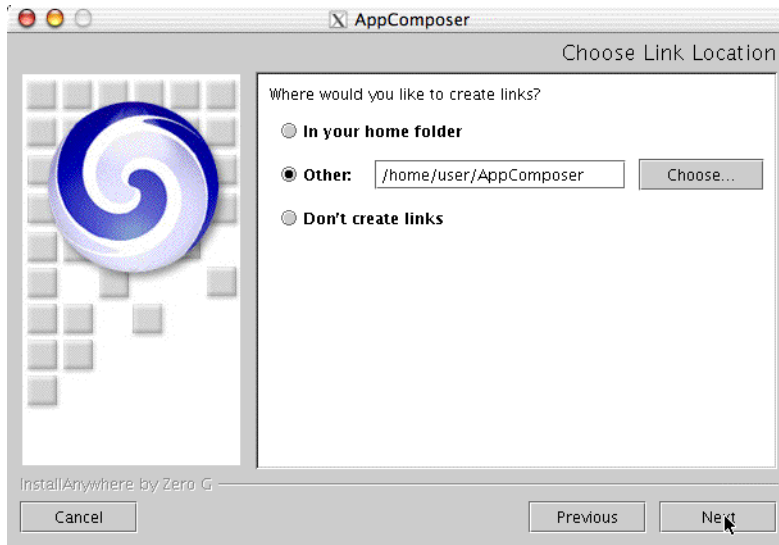


Figure 2-5 Linux/Solaris - Link Location

The Mac OS X Alias location shortcut asks you to place a shortcut in either the dock, the desktop or in your home folder as indicated by the dialog on the next page.

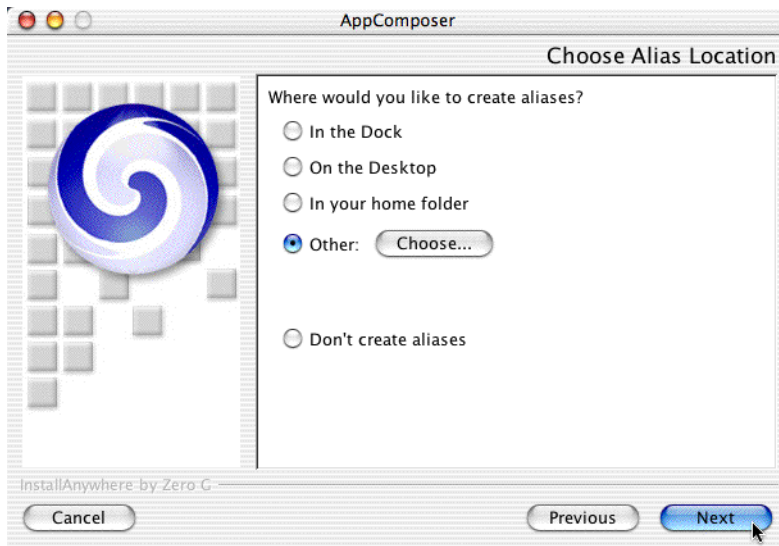


Figure 2-6 Mac OS X Alias Location

Choosing the Java location

During the installation of AppComposer a valid Java JDK or SDK location must be specified. It is important to note that a compiler is needed for AppComposer to function and a JRE (Java Runtime Environment) is not sufficient. If multiple JDK's are installed on the machine, the following dialogs will display all the ones found during a quick search. If you want to specify a JDK not found, you can browse to the proper location or run an exhaustive search which will take some time. On a Mac OS X machine a Java location does not have to be specified as it is provided by Apple and is a fixed location.

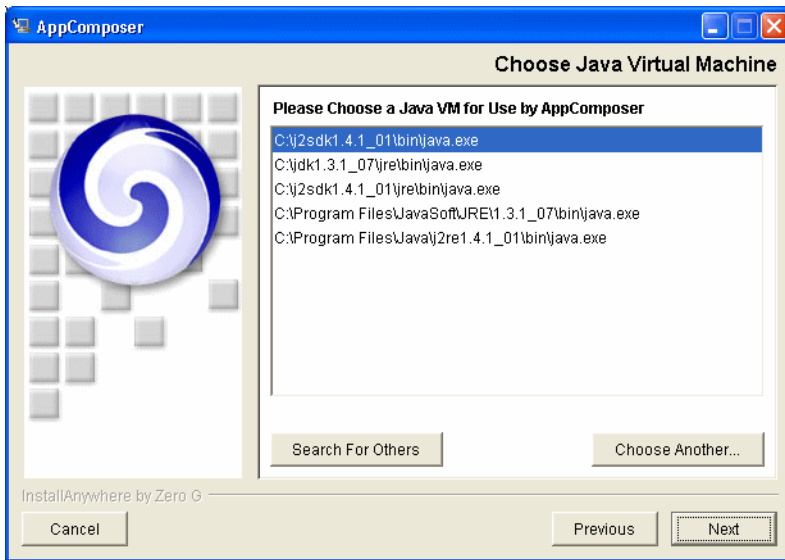


Figure 2-7 Windows Java Location

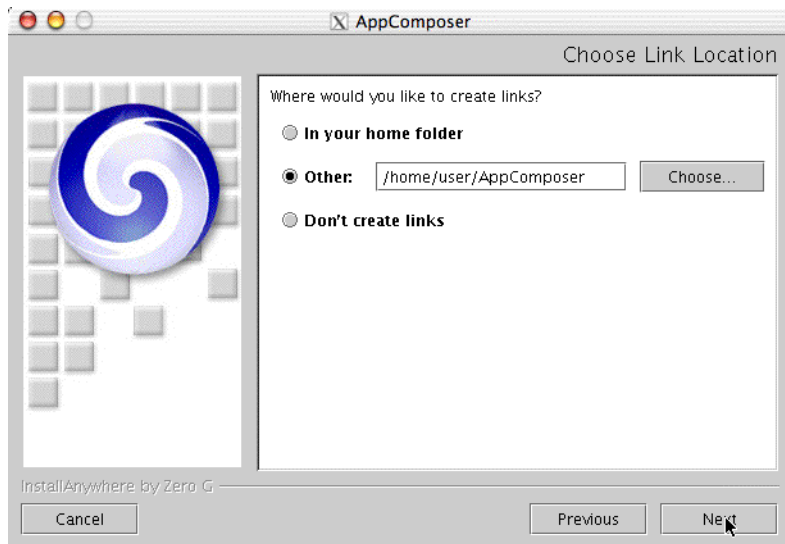


Figure 2-8 Linux/Solaris Java Location

Invoking AppComposer

Once AppComposer is installed, you start AppComposer the following way:

Windows: Start -> Programs -> DigiSlice AppComposer -> DigiSlice AppComposer

Linux/Solaris: /opt/AppComposer/DigiSlice_AppComposer or by using the symbolic link created earlier such as /home/user/AppComposer.

Mac OS X: Click on AppComposer, Inside the Applications/, AppComposer folder. -- Please note that on the Mac OS X version the icon will appear, then disappear from the dock, but will reappear a few seconds later. This is normal behavior and does NOT indicate that the application has crashed.

Licensing AppComposer

On initial launch or until AppComposer is licensed the following screen will be displayed.



Figure 2-9 AppComposer Licensing

If you already have your license information, you can click the “Enter Key” button and enter your information. If you have not yet acquired a license, click the “Get Key” button or point your browser to:

<http://support.digislice.com/registration>

If you click the “Enter Key” button, you will be presented with the following screen.



The image shows a dialog box titled "AppComposer Registration" with a close button (X) in the top right corner. The text inside the dialog box reads "Please enter the following Registration Information". Below this text are three input fields: "Full Name:", "Email Address:", and "Registration Key:". At the bottom of the dialog box are two buttons: "Quit" and "OK".

Enter your name, your email address used to register and your license key, then click the OK button and your registration will be completed. AppComposer will now launch.

Installing components

Installing new components in AppComposer is easy. For simple components, you only need to copy the components into the `beans` subdirectory of the AppComposer installation directory. If the components come with JavaDoc documentation, that documentation should be copied into the `docs` subdirectory of the AppComposer installation directory.

If the beans use any assets, they can be placed into the `asset` subdirectory of the AppComposer installation directory (but this is only a convention, and is not required).

If the components have any libraries, these will need to be installed onto your system using the instructions provided with the components.

Installing a Media Library

If you want to use AppComposer to build client-side rich media applications, you must install a media library and components to access that media library. AppComposer has been tested with the Quicktime libraries from Apple and with the Java Media Framework, but these libraries are not officially supported.

To download quicktime visit <http://www.apple.com/quicktime>

Installing QuickTime is not especially difficult, there are just lots of steps. First, you must install the latest version of Quicktime on your system. For playback you don't need the Pro version, the free version is fine. Note that there are versions of QuickTime only for Macintosh and Windows, so if you are on some other platform, you can try the Java Media Framework instead.

When you install QuickTime, be sure to install the QuickTime for Java libraries. You can also download and install QuickTime for Java separately. This is a set of Java classes that give access to QuickTime from Java. These classes must be added to your system CLASSPATH so that AppComposer can find them.

At this point, you can access QuickTime calls from within AppComposer scripts, but what you really want is QuickTime-enabled JavaBean components. There are two sources for QuickTime components: Elegant Chaos and Tiger Island. These two sources have different capabilities. Finally, you need to install these components into the beans subdirectory of the AppComposer installation directory.

Reinstalling and Upgrading AppComposer

You can install a new version of AppComposer in the same directory as a previous version. We recommend that you uninstall your existing AppComposer installation before installing a new version over it. The uninstallation process will not delete any files that you have added to the installation directory, so your project files are safe.

Another option is to install the new version of AppComposer in a different directory, allowing you to have more than one version of AppComposer on your system at the same time. The only tricky part of this technique is that the file associations for AppComposer files (i.e. the action to take when double-clicking on a composer-generated file) will point to the AppComposer version that was installed most recently.

Database

AppComposer uses the HSQL Database Engine (HSQLDB) for use with the examples and SQL Components that are included with AppComposer. HSQLDB is an open source, lightweight 100% Java SQL Database Engine. Documentation and examples for HSQLDB can be found at

<http://hsqldb.sourceforge.net>

AppComposer installs shortcut to a start HSQLDB within AppComposer. To start HSQLDB, set the DatabaseServiceEnabled preference in the General tab of the Edit - > Preferences dialog.

AppComposer comes preconfigured HSQLDB. To use AppComposer with other databases, you must configure them in the `sql.properties` file. AppComposer should work with any database that has JDBC drivers. See the AppComposer Database documentation for more information.

JBoss EJB Server

AppComposer ships with a full version of the JBoss J2EE server. Use JBoss to deploy Enterprise Java Beans (EJB) for use with AppComposer. JBoss is a free open source J2EE server.

The AppComposer User Guide describes how to work with EJBs in AppComposer. That chapter also describes how to deploy EJBs in JBoss. In addition, the JBoss installation directory contains complete documentation on JBoss to provide the in-depth information needed to fully utilize JBoss.

The AppComposer User Guide also contains information about how to use other popular J2EE servers instead of JBoss.

Note: To take full advantage of integrating EJBs within AppComposer, you may want to consider using DigiSlice's EJB Plugin, part of Enterprise Suite. If your version of AppComposer did not come with this plugin, you can purchase an upgrade (you will see a "Import EJB" menu item under the "File" menu if this plugin is installed)

Dreamweaver extensions

AppComposer ships with extensions for Macromedia Dreamweaver. If you use Dreamweaver and would like to use these extensions, use the following procedure.

To install Dreamweaver Extensions:

- 1 If you have not already done so, download the Dreamweaver Extensions Manager from:
http://www.macromedia.com/exchange/em_download/
- 2 Make sure that Dreamweaver is not open, then install the Extensions Manager that you downloaded.

- 3** Start Dreamweaver, and select **Manage Extensions** from the Dreamweaver Commands menu.
- 4** Select **Install Extension** from the Extension Manager File menu.
- 5** Select the file **AppComposer/Dreamweaver Extensions/ComposerTags.mxp**
- 6** Accept the Macromedia Extensions Disclaimer.
- 7** Restart Dreamweaver.

Where to go from here

Learning to Use DigiSlice AppComposer

DigiSlice AppComposer provides a unique visual authoring driven application development environment. AppComposer introduces several new concepts into application development, most notably the notion of *actors* and *behaviors*. These concepts are well explained in the DigiSlice AppComposer Getting Started guide. You may further explore them by putting them to use in concrete examples in the tutorials. If you are installing AppComposer for the first time, we strongly encourage you to read the Getting Started guide and work through the various tutorials included with AppComposer. These tutorials start by showing you how to build simple applications with AppComposer, and then progress in complexity all the way up to building a full-fledged web-based e-commerce application. These tutorials will get you started with the basic operation of AppComposer, and then take you deeper into the more subtle and powerful aspects of application development with AppComposer.

The available tutorials can be found by choosing the AppComposer Documentation item from AppComposer's Help menu, or as described in more detail in the following section on AppComposer documentation.

Documentation

There are three types of documentation that come with AppComposer. The first type includes the AppComposer manuals, like this one. Additionally, there are tutorials and examples that show how to use AppComposer by example. Finally, there is JavaBean documentation generated by JavaDoc, displayed in a tab of each actor's Details pane of the AppComposer window.

Accessing AppComposer Documentation

The AppComposer documentation is included online in the product distribution. You can access it in one of two ways:

- ◆ From the AppComposer Help menu, select the AppComposer Documentation option.
- ◆ Documentation online at:
<http://support.digislice.com/docs/AppComposer>

AppComposer Tutorials

These tutorials give an excellent introduction to AppComposer, starting at the most basic operating instructions up to sophisticated use of EJB components in a full-fledged e-commerce application.

- ◆ **Introductory Tutorial** — a collection of short tutorials showing how to build various types of projects using AppComposer, including servlet and JSP pages. These tutorials have the most basic information about how to use AppComposer.
- ◆ **Mutual Fund Tutorial** — In this tutorial you build pieces of a web site for Reliance Mutual Funds, a fictitious investment company. In this scenario, you are provided with the database and EJBs to access the data, while graphic artists have built the front end with JSP pages. Your job is to use AppComposer to build the glue that binds the pieces together.

- ◆ **Using Enterprise JavaBeans** — a simple tutorial showing you how to import and work with EJBs in AppComposer.
- ◆ **Using SQL Components** — a simple tutorial that provides experience working with the AppComposer SQL beans.
- ◆ **Creating Session EJBs** — walks you through creating two simple session bean capsules, one stateless and the other stateful, and shows you how to deploy session beans to an EJB container.

AppComposer Manuals

The documents in this section are traditional, indexed user manuals in PDF format.

- ◆ **Getting Started Guide** — explains underlying concepts to help you come to a rapid understanding of the AppComposer assembly model
- ◆ **User's Guide** — the answers to your questions about AppComposer.

Other AppComposer Documentation

These documents, in HTML format, cover a variety of areas not addressed in other AppComposer documentation.

- ◆ **Examples** — descriptions of example programs that come with AppComposer. While shorter and less verbose than the tutorials, these examples show off different features and capabilities of AppComposer, and are an excellent learning tool.
- ◆ **Component Specification** — JavaDoc documentation for components that come with AppComposer.

ObjectInfo Documentation

AppComposer can display documentation for any JavaBean component used as an actor in AppComposer. This documentation is generated by JavaDoc from the source for each component.

In general, AppComposer components come from three places:

- ◆ Components that come with AppComposer, including the HTML beans, visual beans like Text and Rect, and non-visual beans like Clock and SqlSelect.
- ◆ Components that come with Java, including AWT components like Button and Scrollbar.
- ◆ Components you add later, or that you write yourself.

JavaBean documentation is displayed from a tab in the Details pane. Select an item from the outline or toybox, and click on the Object Info tab in the Details pane to see its JavaDocs.

To display online documentation for a JavaBean, the pane must be able to find the documentation:

- ◆ For the JavaBeans that come with AppComposer, AppComposer automatically installs documentation docs subdirectory of the AppComposer installation directory.
- ◆ For JDK documentation, AppComposer accesses this documentation from Sun's website, or you can set a local path from the **Edit -> Preferences** dialog.
- ◆ You can add JavaBeans to AppComposer by placing them in the beans subdirectory of the AppComposer installation directory. When you do this, you should also add any corresponding JavaDoc documentation to the docs subdirectory of the AppComposer installation directory. JavaBeans you acquire usually come with JavaDoc documentation.

JavaDoc documentation is always in HTML format, but there are two ways that this documentation can be named: using dot notation or using directory notation. For example, the Clock JavaBean in the com.digislice.misc package can be named `com.digislice.misc.Clock.html`, or it can be named `com/digislice/misc/Clock.html`. Use backslashes (\) instead of forward slashes with Windows. AppComposer can use either format. The AppComposer JavaDoc documentation uses directory-based filenames because the dot notation results in file names that exceed the maximum allowable file name length on some platforms.

Offline Documentation

You can access the JDK JavaBean documentation from Sun's website. To install the JDK JavaBean documentation locally, you need to download it, install it, then set a preference so that AppComposer can find it.

First, you need to download the JDK documentation from Sun. This can be downloaded from <http://java.sun.com/j2se/1.3/docs.html>. Like all JavaDoc documentation it is in HTML format, and is identical for all platforms. If you are using AppComposer with another version of Java you need to download its documentation instead.

Unpack this documentation into an appropriate directory, such as the docs subdirectory of the AppComposer directory, or (if you are using some other version of Java) the docs subdirectory of your JDK directory. The JavaDoc documentation we are interested in is in the `api` subdirectory of the unpacked docs directory.

Finally, you need to change a preference so that AppComposer can find this documentation. From inside AppComposer, go to the **Edit** menu and choose **Preferences**. Set the AppComposer preference `JDKDocsDirectory` (in the **General** tab) to the `api` subdirectory of the JDK docs directory.

Composer Installation Directories

AppComposer is installed into the directory you specified during the installation. This directory contains the following subdirectories:

- ◆ `asset` - Contains any images, sounds, animations, and other asset files used by the example programs.
- ◆ `beans` - Contains all JavaBeans used as actors. To add a new actor JavaBean, simply drag it into this directory and restart AppComposer. Also contains EJB .jar files that have been copied here and expanded by AppComposer.
- ◆ `databases` - contains the database used in the SQL examples, along with scripts to rebuild these databases.

- ◆ docs - Repository for all AppComposer documentation. Product manuals and tutorials can be found in docs/manuals. Also contains JavaDoc documentation for JavaBeans. If you add a new JavaBean to the beans directory, you can store its JavaDoc documentation here, and AppComposer will find it.
- ◆ DreamweaverExtensions - Contains AppComposer extensions for Macromedia Dreamweaver.
- ◆ dtd - Contains dtDs used by AppComposer's XML facilities.
- ◆ examples - Contains example programs. This is a convention only; you can put AppComposer projects and capsules anywhere.
- ◆ icons - AppComposer icons
- ◆ jBoss - Contains the JBoss EJB server and its associated files.
- ◆ libraries - third-party java libraries used by AppComposer.
- ◆ org - Contains Apache Tomcat and Jasper files for the JSP and debug server.
- ◆ preferences - holds the properties files that maintain 'startup state', such as the projects and capsules you had open, your preferences settings, and so forth.
- ◆ public_html - Where AppComposer's debug web server expects to find html files. The location of this directory can be changed using a AppComposer preference.
- ◆ sources - AppComposer uses some Java code whose source is freely available. We include that source code here for you.
- ◆ temp - AppComposer uses this directory to hold intermediate results when "grinding beans" or processing .jsp's.
- ◆ UninstallerData - Contains files used to remove AppComposer from your system, including a log of all actions taken during the installation of AppComposer. This directory only exists on certain platforms.
- ◆ WEB-INF - Contains the ComposerTags.jar tag library.