



Beginning

Adobe
AIR™

Building Applications for the Adobe Integrated Runtime

by Rich Tevold



Beginning
Adobe® AIR™
Building Applications for the Adobe Integrated Runtime

Rich Tretola



WILEY
Wiley Publishing, Inc.

Beginning Adobe AIR

Acknowledgments	xiii
Foreword	xxi
Introduction	xxv
Part I: Getting Started	1
Chapter 1: Introducing AIR (Adobe Integrated Runtime)	3
Chapter 2: Rich Internet/Desktop Applications	15
Chapter 3: The Programming Languages	21
Chapter 4: The AIR Development Tools	35
Chapter 5: Building an Application	53
Chapter 6: Packaging the AIR	65
Part II: Adding Data	87
Chapter 7: Working with Remote Data	89
Part III: AIR APIs	
Chapter 8: Accessing the File System	113
Chapter 9: The Windowing API	135
Chapter 10: Interacting with the O.S.	153
Chapter 11: The SQLite Database	169
Chapter 12: Communication between AIR and the Operating System.	197
Chapter 13: Application Status	217
Part IV: The AIR Components	233
Chapter 14: The File System Components	235
Chapter 15: The HTML Component	245
Appendix A: Taking a Flex App to the Desktop	259
Appendix B: Solutions for Exercises	281
Index	305

Beginning
Adobe® AIR™
Building Applications for the Adobe Integrated Runtime

Rich Tretola



Wiley Publishing, Inc.

Beginning Adobe® Air™

Published by

Wiley Publishing, Inc.

10475 Crosspoint Boulevard

Indianapolis, IN 46256

www.wiley.com

Copyright © 2008 by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-22904-0

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data is available from the publisher.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600.

Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at <http://www.wiley.com/go/permissions>.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Wrox Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Adobe and AIR are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.



Figure 9-14: The text being updated from the TextInput on the main file on Mac OS X.



component with additional properties

```
encoding="utf-8"?>  
http://www.adobe.com/2006/mxml"  
="200"  
,  
,  
e">  
Center="0" y="58"  
le Window" id="lbl"/>
```



Figure 9-16: The window on Mac OS X, with only the Close button enabled and no Resize handle



object previously discussed, Window also has a `NativeWindowType` property. Setting the `NativeWindowType` is a good idea. Listing 9-13 shows an example of a updated `mx:Window`. The results of this updated example can be seen in Figure 9-19.

Listing 9-13: A Utility window

```
<?xml version="1.0" encoding="utf-8"?>
<http://www.adobe.com/2006/mxml"
xmlns:mx="http://www.adobe.com/2006/mxml"
width="200"
height="200"
title="A Sample Window" id="lbl"/>
```



Figure 9-19: A Utility window on Mac OS X.



Create Window

Minimize

Maximize

Restore

Close

Set Text

E



A Sample Window



10

Interacting with the O.S.

AIR can interact with the underlying operating system by utilizing some of its existing features, including dock icons on Mac OS X, system tray icons on Microsoft Windows, and context right-click menus. The dock icons and system tray icons can also contain menus.

Dock Icons

Mac OS X has a dock that is mounted either at the right side, left side, or bottom of the screen. The dock contains shortcuts to open installed applications and also shows currently running applications. Chapter 6 demonstrated how to create custom icons for the installer. This icon would be the one shown in the dock when an application launches. AIR also gives us the ability to change the dock icon while the application is running. This can be very useful to display information to the user. Changing the icon is a simple process and simply requires the creation of a `flash.display.BitmapData` object. This can be created from an existing image displayed within the application or can be created from an embedded image class.

To get started, please create a new AIR project named *Chapter10_Dock*, which will create a new file named *Chapter10_Dock.mxml*. You will also need an image file to use as the dock icon. The source code that is part of this book has four PNG files representing four different-sized icons that I created. The image sizes are 16×16 , 32×32 , 48×48 , and 128×128 . As you will see shortly, the dock icon accepts an array of images of various sizes and then uses the best image to fit the situation; you can get away with using just one mid-sized image, which will be scaled accordingly.

We'll now proceed to the source code for adding a dock icon. Enter the code from Listing 10-1 into the newly created *Chapter10_Dock.mxml* file. If you only have one icon image that is 128×128 , you could alternatively use the code from Listing 10-2, but the quality may not be acceptable when it is scaled to meet all the system requirements.

The code in Listing 10-1 first embeds four image files and types them as class objects. Next, on `creationComplete` of the application, the `init()` function is called where four new `Bitmap` objects are created from the embedded images. Finally, the `NativeApplication.nativeApplication.icon.bitmaps` is set using the `BitmapData` properties from the `Bitmap` objects.

```

[Embed(source="e128.png")]
[Bindable]
private var Icon128:Class;

private function init():void{
    var bitmap128:Bitmap = new Icon128();
    NativeApplication.nativeApplication.icon.bitmaps =
[bitmap128.bitmapData];
}

]]>

</mx:Script>
</mx:WindowedApplication>

```



Figure 10-1: The dock icon on Mac OS X.



Figure 10-2: The Alt-Tab menu with the new icon.

Menu

Dock icons also support the addition of *custom menus*. Adding a custom menu is very similar to adding a `NativeMenu` or a `ContextMenu`. You will simply need to build a `NativeMenu` object and then apply it to the `DockIcon` object. For more information on `NativeMenus`, please take a look at Chapter 12.

To add a menu to the dock icon, add the two functions from Listing 10-3 to the `Chapter10_Dock.mxml` file's script block. If you examine these functions, you will see that there is a `createMenu()` function and a `handleMenuClick()` function. The `createMenu()` function first creates a `NativeMenu` object and then creates four `NativeMenuItem` objects. Each `NativeMenuItem` has an event listener added and assigned the `handleMenuClick()` function, the function that will be called when the menu is selected. The `NativeMenuItem` items are then added to the `NativeMenu` object, and finally, if the `NativeApplication.supportsDockIcon` returns `True`, the `NativeApplication.nativeApplication.icon` is cast as a `DockIcon` and the menu is set.

When a `NativeMenuItem` is selected, the `handleMenuClick()` function is called and the appropriate action is taken.

The last thing you need to do is add a call to the `createMenu()` function within the `Chapter10_Dock.mxml`'s `init()` function. The results can be seen in Figure 10-3.