

FLASH & FLEX

Vol.3 No.5 Monthly Issue 5/2010 (13) July ISSN 1898 -9136

DEVELOPER'S MAGAZINE

PROTECTING, LICENSING AND SELLING ADOBE FLEX & AIR APPS

REFACTORING ACTIONSCRIPT CODE WITH SOURCEMATE

GOOGLE TV

GETTING START TO DEVELOP ANDROID APPS

WITH ADOBE AIR

PREPARING A ROBOT TO

PLAY FARMVILLE AUTOMATICALLY




FLUID LAYOUTS WITH ACTIONSCRIPT
FLEX 4 – THE PROBLEM WITH CHILDREN
HOW TO: REMOTING WITH ZEND STUDIO AND FLASH BUILDER



sourcemate BETA
for Flash Builder 4



RELEASE THE INTERACTIVE BEAST

Application Hosting Services for the Adobe Flash Media Interactive Server 

STREAMING

Shred the restraints of traditional streaming video and free your inner beast to build the wildest apps.

COLLABORATION

Build any collaborative app you can imagine - social theatres, user recorded video, instant live data, and more.

EXPERTS

Our FMS global network has been growing since 2002. We know FMS and can help you achieve app greatness.



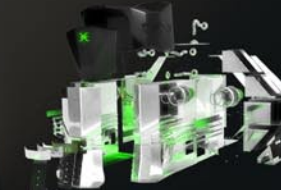
Basic plans from
\$9.95/mo



Advanced plans from
\$220/mo



Dedicated plans from
\$895/mo



Custom plans from
\$6/mo

Editor's Note



Dear Readers!

Summer is here. Time for relaxing and putting our powers off for this time. Do the brain reset and get back with doubled energy. Hopefully you won't forget to take this issue with you ;)

This magazine is mostly about learning new techniques and keeping up with the latest industry news, it is also made for practical knowledge. This time we have chosen a great article that we recommend for every Flash/Flex Developer – Protect, License and Sell Adobe Flex & AIR Apps. That's really crucial for those who would like to earn serious money from their passion.

For those who would like to start from the beginning – we have prepared a great article where you can learn basic steps for developing Android Apps with Adobe AIR. Yet many developers, even senior developers, don't truly understand the power of refactoring. In the article Refactoring ActionScript Code, you'll go over the refactoring features in SourceMate one by one – great, isn't it? Do you play FarmVille – the most popular game on Facebook? Read about the explanation of the game client flash vars list, and much more about the game. Besides that, Fluid Layouts with ActionScript 3.0, Flex4 and Zend and PHP articles.

There is much more inside – let's not waste your time and skip to the content. We look forward to sharing more information with the community in the coming weeks and months. If you have questions, comments, or concerns, please don't hesitate to contact me directly, or ask our authors about any details.

I want to thank you for your continued commitment to the FFD mag community, and I look forward to new opportunities to work together.

With best regards,

Ewa Samulska

ewa.samulska@ffdmag.com

Feedback

FLASH&FLEX

Editor in Chief: Ewa Samulska ewa.samulska@ffdmag.com

Proofreaders: Betsy Irvine, Patrick French

DTP Team: Ireneusz Pogroszewski

ireneusz.pogroszewski@software.com.pl

Art Director: Ireneusz Pogroszewski

ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher: Paweł Marciniak

Publisher: Software Press Sp. z o.o. SK

ul. Bokserka 1 02-682 Warszawa Poland Worldwide Publishing

Software Press Sp. z o.o. SK is looking for partners from all over the World.

If you are interested in cooperating with us,
please contact us by e-mail: cooperation@software.com.pl

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage.

All trade marks presented in the magazine were used only for informative purposes.
All rights to trade marks presented in the magazine are reserved by the companies which own them.

Thanks to the most active and helping beta testers:

Russell TangChoon, Lee Graham, Jassa Amir Lang, Ed Werzyn, Yann Smith-Kielland, Justus, Csomák Gábor, Kevin Martin, Charles Wong, Ali Raza, Almog Koren, Izcoatl Armando Estanol Fuentes, Lionel Low, Michael J. Iriarte, Paula R. Mould, Rosarin Adulseranee, Sidney de Koning

To create graphs and diagrams we used smartdraw.com program by



The editors use automatic DTP system **APOSE**

Mathematical formulas created by Design Science MathType™

ATTENTION!

Distributing current or past issues of this magazine – without permission of the publisher – is harmful activity and will result in judicial liability.

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

Stack Photo Gallery

Natural looking tool emulating a set of photos

- captions for photos
- dynamic shadows

Art Flash Gallery

Fully customisable and flexible flash gallery

- text descriptions for images
- fullscreen mode

Zen Flash Gallery

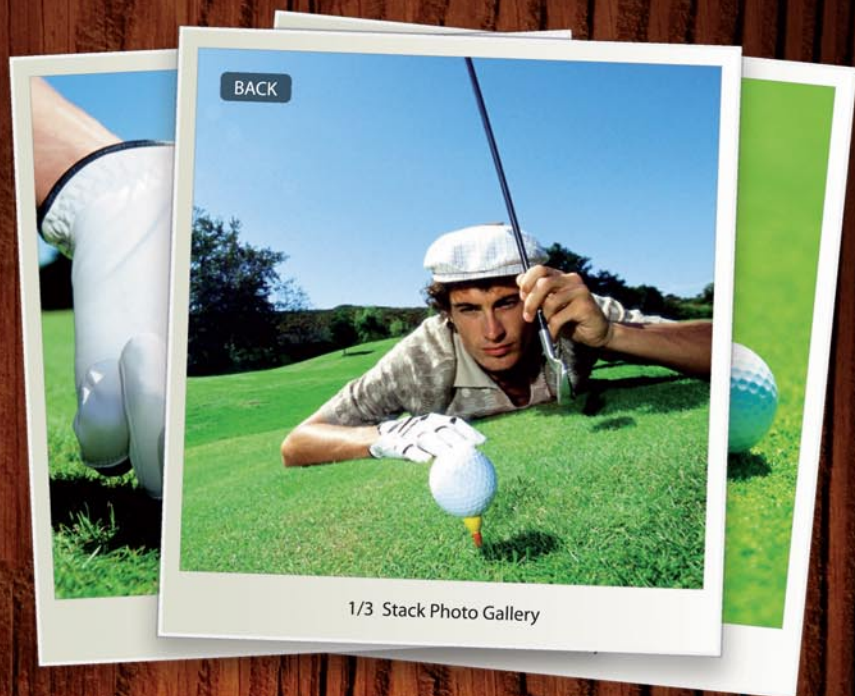
Flash gallery with incredible 3D rotation effect

- folder hierarchy support
- slideshow mode

PhotoFlow Flash Gallery

Famous streamline photo flow effect

- customizable interface
- dynamic reflections



COMMON FEATURES



XML Support



Flexible Configuration



Various Transition Effects



Complete Docs



Trial Version

CONTENTS

Special Report

08 GoogleTV

BY LEE GRAHAM

10 Refactoring ActionScript Code with SourceMate

BY CHRIS GROSS

14 Overcoming FUD Protecting, Licensing and Selling Adobe Flex & AIR Apps

BY CLIFF HALL

InBrief

22 News

BY CSOMÁK GÁBOR

APPS

24 Getting start to develop Android apps with Adobe AIR

BY WILLIAM TSANG

GAMES

30 Preparing a Robot to Play FarmVille Automatically

BY ELAD COHEN

ActionScript Development

32 Fluid Layouts with ActionScript3.0

BY RYAN D'AGOSTINO

Flex Development

36 Flex 4 – The Problem With Children

BY HUW COLLINGBOURNE

ZEND and PHP

42 How to: Remoting with Zend Studio and Flash Builder (PART 2)

BY KEVIN SCHROEDER

Interview

46 Interview with Chris Gross

PROFILE

48 Ryan D'Agostino

Books Review

50 Flex 4 Cookbook Real-world recipes for developing Rich Internet Applications

BY ALI RAZA

The issue 5/2010 sponsored by Advertisers:

Influxis www.influxis.com 2-3	Flex[er] www.flexer.info 29
Mediaparts Interactive S.A www.page-flip.com 5	Exsys www.exsys.com 41
FusionMaps for Flex www.fusioncharts.com/flex 7	FITC www.fitc.ca/SF 45
Kevin Ruse + Associatess Inc. www.kevinruse.com 9	Flash and Math www.flashandmath.com 49
Kindisoft www.kindisoft.com 13	ActionScriptJobs.com http://actionscriptjobs.com/ 51
ElementRiver www.elementriver.com 21	
Gamersafe www.gamersafe.com 23	

Tip of the issue



Event Listener for XML

Michael Greenhut, Arkadium Game Programmer

When populating a Flash application with xml, you may have *race condition* – your ActionScript might call a function to load the xml, then execute before the xml has time to load, and you'll be left with a lot of empty fields and null values that you weren't expecting. To avoid this, add an event listener in your ActionScript code that waits for the xml to load before it advances too far. Something like this:

```
private function loadXML()
{
    var xmlLoader:URLLoader = new URLLoader();
    xmlLoader.addEventListener(Event.COMPLETE, onLevelX
mlFinishedLoading);
    xmlLoader.load(new URLRequest("stuff.xml"));
}

private function onLevelXmlFinishedLoading(e:Event)
{
    ... //pick things up here
}
```

(this is also true for the .swf Loader class)

Need advanced data visualization in Adobe Flex?

FusionCharts helps you build stunning charts, gauges and maps for your Flex solutions in no time at all. Offering over 50 chart types and 300 maps, it offers you a powerful reporting experience for all your Flex projects.



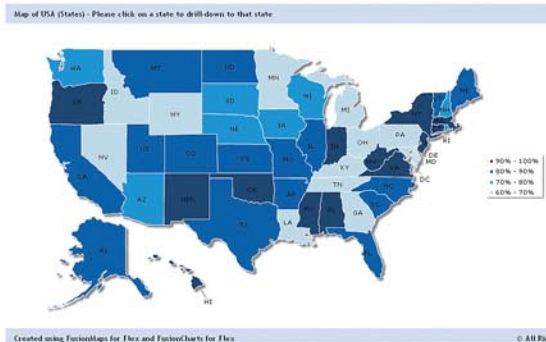
FusionCharts for Flex



- ▶ Animated & interactive charts
- ▶ Over 50 charts & gauges
- ▶ True 3D capabilities
- ▶ Export as Image/PDF/CSV
- ▶ Extensive drill-down supported



FusionMaps for Flex



- ▶ Interactive & data-driven maps
- ▶ Over 300 maps provided
- ▶ APIs for extensive drill-down
- ▶ Export as Image/PDF/CSV
- ▶ Real-life demos & code samples

Download now from
www.fusioncharts.com/flex

Google TV & Showcase

Well if you haven't heard, Google has announced that they are releasing Google TV (<http://www.google.com/tv/>). Think of it as mixing your computer into your TV. You can view anything on the web, including Flash content! Check out this demo (http://blogs.adobe.com/flashplatform/2010/05/flash_player_101_on_google_tv.html) from Adobe about Flash content on Google TV. Another HUGE bonus is that Google TV will support Android Apps within Google TV. So we are going to be able to access Android Apps, surf the web, view your favorite TV programs and interact with Flash content from the comforts of your couch. Pretty wicked, right?

Details are very limited at this point about the Adobe Flash on Google TV, but I've heard rumors that the AIR for Android (<http://labs.adobe.com/technologies/air2/android/>) apps will eventually be able to run on Google TV as well! Stay tuned for more on this!!!

Flex, Flash, AIR Showcase

This magazine is mostly about learning new techniques and keeping up with the latest industry news, but I thought I would do something a little bit and showcase a few apps developed with the tools we use everyday.

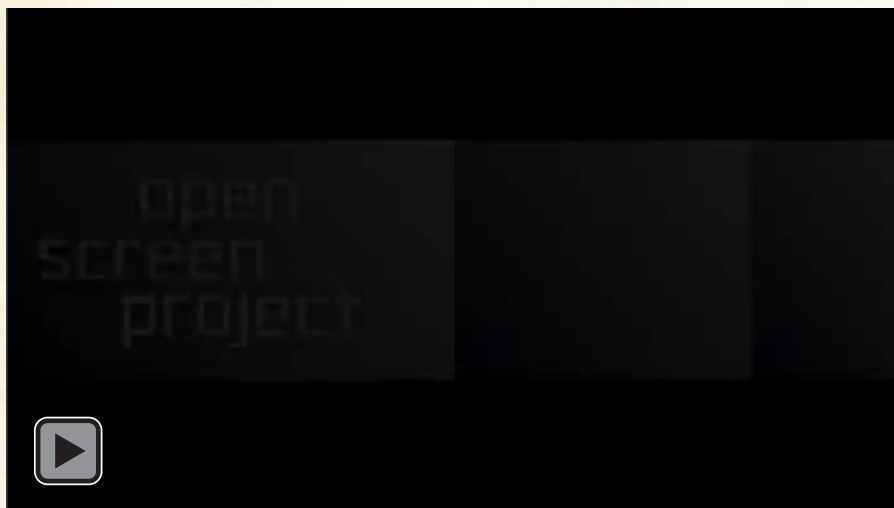
Atlantic Records' FanBase (Flash & Flex): <http://tv.adobe.com/watch/customer-stories-air/atlantic-records-fanbase-application>

Adelaide Football Club (Flash, Flex, ColdFusion, LiveCycle, Flash Media Server, etc...): http://www.adobe.com/cfusion/showcase/index.cfm?event=casestudydetail&casestudyid=1013470&loc=en_us

TweetDeck (Flex & AIR): <http://tv.adobe.com/watch/customer-stories-air/interview-with-tweetdeck-founder/>

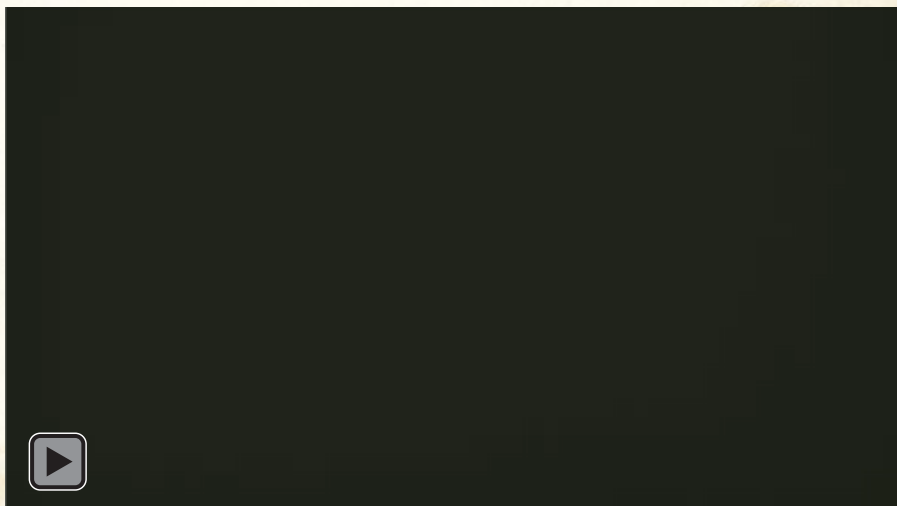
NCAA Vault (Flex, Flash Media Server, & Photoshop): <http://vault.ncaa.com/>

http://blogs.adobe.com/flashplatform/2010/05/flash_player_101_on_google_tv.html



LEE GRAHAM

Lee Graham is co-founder of TRImagination (<http://trimagination.tumblr.com/>), an educational app company based in the United States. He has been involved in developing interactive eLearning applications for five years and working with Adobe in beta testing Flash CS5, AIR 2.0, AIR for Android & Flash Player 10.1. You can connect with him on Twitter: <http://twitter.com/donaldleegraham> or his Blog: <http://l33.me/>.





Certified Instructor

+ Dreamweaver®

+ Contribute®

+ Flex with AIR®

+ Macromedia® Flash®



```
1  /*
2  ask this question of prospective students
3  */
```

What do you want to learn today?

```
6
7  /*
8  this tells 'em what we do
9  */
10 Learn Flash, Flex, Contribute, DreamWeaver, XML, XSLT, JavaScript,
11 XHTML, CSS and ColdFusion from an Adobe certified instructor, who:
```

- Specializes in training beginners to advanced learners
- Conducts corporate on-site training
- Provides consulting services

```
16
17 /*
18 let others sing your praises
19 */
20 "It's rare to find an instructor who has personality
21 and the ability to teach complex subjects. Kevin is great!"
22 -Kelley Sullivan, Power Integrations
```



K E V I N R U S E + A S S O C I A T E S

www.kevinruse.com

001.408.496.6846

kevin@kevinruse.com

Istevens@kevinruse.com

Refactoring Actionscript Code

with SourceMate

In 1999, Martin Fowler's Refactoring: Improving the design of existing code was published. Since then refactoring has become a staple of software development.

This year, ElementRiver released SourceMate and brought advanced code editing and code generation features, including a set of advanced refactoring capabilities, to Flash Builder 4. Yet many developers, even senior developers, don't truly understand the power of refactoring. Moreover, individual refactorings can be confusing and it's not easy to know when and where to use each refactoring. In this article, we'll go over the refactoring features in SourceMate one by one. By the end, you should understand the power of these features and you'll see how automated refactoring can save hours of development, often with just one use.

Refactoring: Convert Local Variable to Field

What it does: Converts the selected local variable into a class field

When to use it: After you realize a local variable will be needed to be accessed in other functions.

The Convert Local Variable to Field is one of the simpler refactorings, but also one of the more commonly used. Have you ever created a local variable only to realize you need that value later in another function? This can be especially common when working with asynchronous events. You've written a function only to realize that you need to wait for an event response before continuing processing. Of course, you need access to the variable or variables you've defined in the original function in the event handling function. In these situations you can use the Convert Local to Field refactoring to turn those local variables into class fields so they can be accessed in

both the original function and in the subsequent event handler. One of the nicest features of this refactoring is that it doesn't change your context. If you're working in a large file, having to scroll up to the top of the class definition to add a new class field, and then scrolling back to where you left off, is quite annoying. When you use Convert Local to Field, the scroll position never changes. The new field is created and you're ready to keep coding without having to find your place again. Of course, this refactoring isn't only useful with asynchronous code. In any situation where you need access to a local variable in another function, Convert Local to Field can help.

Refactoring: Extract Constant

What it does: Takes the selected literal value and turns it into a class constant

When to use it: When it's time to clean up those special values.

The programming orthodoxy says "Thou shalt not put hard coded literal values in thy code". Of course, we don't always listen to the programming intelligentsia. Sometimes we're working against deadlines, or maybe we don't see the immediate need to make these literal values into constants. However, there are real benefits to doing so. First, instead of seeing an obtuse literal value in code, we'll have a named constant that will provide at least a clue as to what the literal means (so instead of seeing a "3.14" we'll see a constant named PIE). It's also very important to turn these literal values into constants if they're used more than once. Otherwise, if the next developer after us has to make a change he/she might miss a spot or two. To use Extract Constant simply select or place your cursor inside a literal value and select the refactoring from the SourceMate

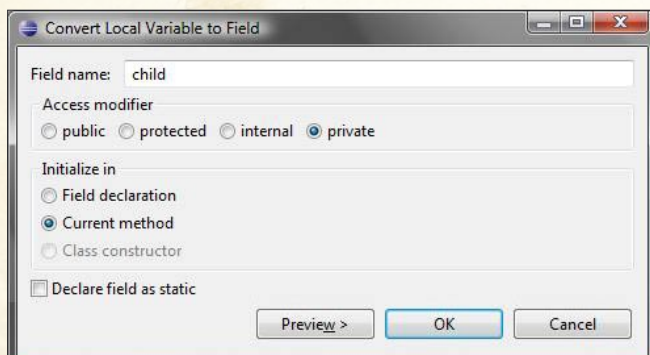


Figure 1. Convert local to field

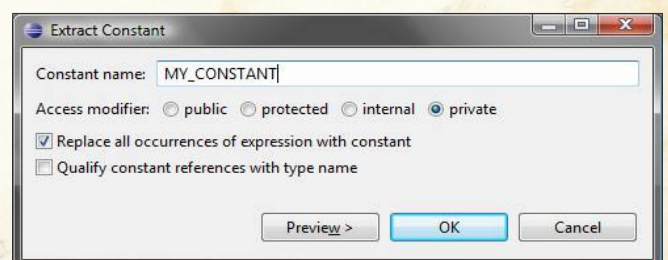


Figure 2. Extract constant

menu. SourceMate will create a new class level constant and replace the literal value under your cursor with this new constant. Most significantly, SourceMate will go through all the code in your class and replace any additional references to the same literal value with the new constant.

Refactoring: Extract Local Var

What it does: Takes the selected expression and turns it into a new local variable

When to use it: You find yourself retyping the same expression over and over again

Arguably, refactoring is the process of changing code with the knowledge of how the code *should* have been written ("if I knew then what I know now"). Perhaps you started writing code and entered a quick if expression that checked an x,y coordinate that intersected the class's parent component. Something like this:

```
if (this.parent.hitTestPoint(x,y)) { ...
```

There's nothing wrong with that code. But perhaps you've continued writing code and you've had to check that same coordinate intersection more than once. Not only have you typed that same expression a couple of times, but you're also calling a potentially expensive function more than necessary. We also have the added concern that if we need to change the expression, we'll need to change each instance of it, potentially leading to bugs if we miss an instance. Instead, use the Extract Local Var refactoring. In this case, we'd select the entire expression `this.parent.hitTestPoint(x,y)` and select Extract Local Var from the SourceMate menu. Tell SourceMate what you'd like to name the new variable and then your code will look like this:

```
var intersects:Boolean = this.parent.hitTestPoint(x,y);
if (intersects) { ...
```

Just as with Extract Constant, SourceMate will also go through the entire function and find any places you've typed the same expression and replace it with the new local variable. The code has become a little more readable, more maintainable, and if we replaced more than one instance of the expression, more efficient.

Refactoring: Extract Interface

What it does: Creates an interface from a selected set of functions on the source class

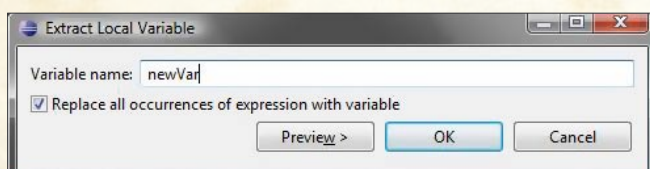


Figure 3. Extract local var

When to use it: When you need to replace a class with an interface

Sometimes you may create a class only to later realize you need to allow for multiple implementations of that feature. For example, you might have a logging class but you later decide to log to different places depending on whether your app is running in the web browser or running in AIR. Ordinarily, you'd need to create a new interface, glancing back at the original class from time to time to remember the functions to include. Then add the implements clause to the original class. And finally go through your existing code to replace references of the original class to the new interface. SourceMate can do all this for you with a few clicks. The Extract Interface dialog will present you with a list of functions from the original class. Choose which ones to include and SourceMate will create the interface for you. It will also add the implements clause to the original class. As with all the other *Extract* features, SourceMate will replace all the current references to the original class to the new interface if that referencing code calls only methods you've included in the new interface. SourceMate understands your intentions and handles it all for you.

Refactoring: Extract Method

What it does: Moves the selected lines of code into their own function

When to use it: When you find a certain set of code in an existing function is useful in its own right

Extract Method is one of the clever refactorings in SourceMate. How many times have you been writing some code only to think that a couple lines of code should be moved into a new function? Perhaps you've written the same lines of code multiple times or perhaps you believe the code is important enough to have its own method. Highlight the lines of code in question and choose Extract Method. The power of this refactoring is evident as it parses the selected lines and determines which variables will be needed by the new function. Those variables will become arguments to the new function. In a similar manner, SourceMate will parse the lines after the selection to determine if any variables declared within the selection are referenced. If

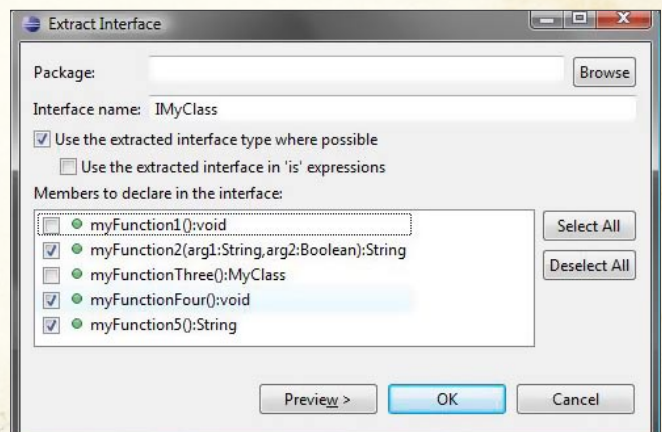


Figure 4. Extract interface

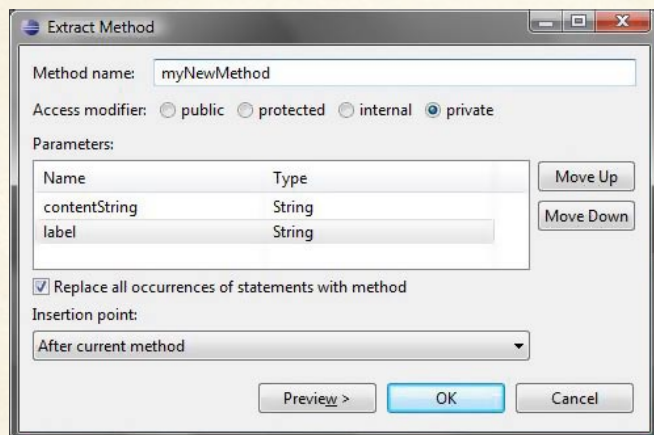


Figure 5. *Extract method*

SourceMate finds a variable referenced after the selection, it will return it from the newly created function. Essentially, SourceMate untangles the collection of variables and ensures that the existing code works the same after the selected lines are encapsulated in a new method. When complete, the selected code will be replaced with a call to the new function, sending in the appropriate arguments and capturing the return value. Of course, it bears repeating that Extract Method, like all *Extract* refactorings, will find any other lines of code that duplicate the ones being extracted and replace them with a call to the new method.

Refactoring: Change Method Signature

What it does: Updates a function signature

When to use it: Anytime you need to change a function signature without breaking calling code

Change Method Signature is arguably the most powerful and useful refactoring but also the most misunderstood. After all, changing a method signature seems like an easy, straightforward task. But Change Method Signature's power comes from its ability to update code throughout your project. Normally, when changing a method signature you risk breaking any code in your project that calls that method. Fortunately, Actionscript does allow you to provide default values for arguments added to the end of the argument list, but beyond that you have to go through and update each place that calls the given method. Want to add a new argument to the beginning of the argument list? Want to change the order of the arguments? Want to remove an argument? Get ready to click through a mass of files, finding and fixing the newly broken code. If you use the Change Method Signature dialog, SourceMate will update all the calling code appropriately and leave your

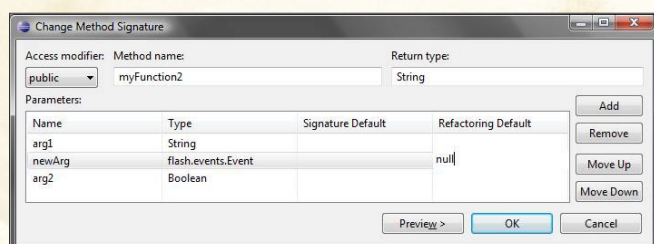


Figure 6. *Change method signature*

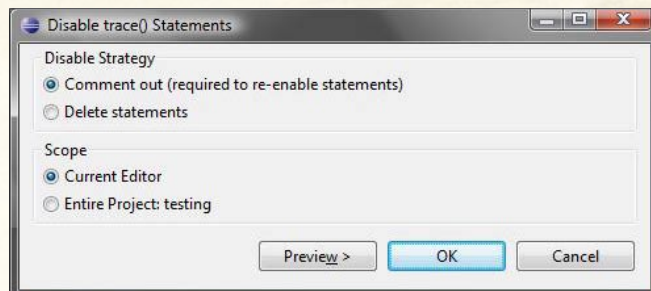


Figure 7. *Disable trace() Statements*

code still fully functioning. This feature gives a developer great power to go back and clean up their APIs after the fact. No longer should developers feel that their function signatures are cast in stone.

Refactoring: Disable trace() Statements

What it does: Removes or comments `trace()` statements
When to use it: After a debugging session or before creating a production build

From the most misunderstood to the easy to understand, the Disable `trace()` refactoring is as useful as it is understandable. When debugging difficult problems, developers often resort to littering their code with trace statements. Sometimes these trace statements can be placed in numerous files throughout the project. Once the developer finally resolves the bug they likely need to remove these trace statements. Does the developer remember where each trace was added? Let SourceMate do it for you and you'll be sure all the errant trace statements are gone. The Disable `trace()` refactoring can either remove the statements completely from your code or, if you choose, just comment them out. You then have the option through the corresponding Enable `trace()` refactoring to turn them back on. In your next debugging sessions, go ahead and add `trace()` statements with abandon. SourceMate will help you remove them in one click.

Conclusion

The power of these refactoring features should be evident. Rather than typing in a mindless text editor, SourceMate's refactoring tools become your intelligent code collaborator. SourceMate understands not just the characters you're typing, but your intentions. SourceMate intelligently updates your code based on the knowledge of both what you're trying to do and the existing structure of your project. The time saved by using these features can be huge. It's not uncommon to hear stories of one developer saving over 3 hours by using just one automated refactoring. Considering how valuable our time is as developers, we should all strive to have a full set of automating refactoring tools available to us. SourceMate is an indispensable aide for every Flex and Flash developer.

CHRIS GROSS

KINDISOFT

Protect Your ActionScript 1/2/3 from All SWF Decompilers.

Protect your ActionScript 3 with secureSWF v3

secureSWF continues to be the most sophisticated ActionScript obfuscation, protection, and encryption solution available today. Version 3.0 features ActionScript 3 support for Adobe's Flash, Flex and AIR applications and is available for Windows, Mac OS X and Linux.

Now Supports
Flash CS4

Main Features:

- Identifiers Renaming
- Control Flow Obfuscation
- Dynamic Code Wrapping
- Statement-level Randomization
- Literal Strings Encryption
- Encrypted Domain Locking
- Encrypted Loader Creator
- SWC Files Support
- Project Files and Command-line Interface



secureSWF
Publish with Confidence.

Learn More About secureSWF and Request Your Fully-Functional Copy At:

WWW.KINDISOFT.COM

Save 15% With Promotion Code: FFDM15

Overcoming FUD

Protecting, Licensing and Selling Adobe Flex & AIR Apps

Monetizing Your Cognitive Surplus

In a recent talk (<http://blip.tv/play/gshVtNIUAg>), author Clay Shirkey discussed the concept of *cognitive surplus*. The basic idea is that all the time we used to spend watching Gilligan's Island on TV (a one-way medium) is now something we can harness for creative endeavors via the two-way, participatory medium of the Internet. That aggregate cognitive surplus is a vast resource that has appeared out of nowhere and can be used to achieve nearly anything. Some use that time to build public works like Wikipedia or blog the depths of their wisdom and experience to help place the knowledge of mankind within everyone's reach.

For those who happen to develop software for a living and have creative minds and entrepreneurial inclinations, that cognitive surplus can easily be directed toward creating side-incomes from software products created in their spare time.

But when we begin to think about what it would take to control licensing of our software and keep it from being cracked and redistributed for free, or see our hard work stolen by a competitor with a decompiler, the attractiveness of trying to produce and sell our own software can quickly fade. Having been there, I'm here to tell you it can be done.

In the Flex and AIR software development community, there is much FUD (fear, uncertainty and doubt) surrounding the related issues of code protection and license control. When all you want to do is focus on making your app the very best it can be, these questions are pretty depressing. You know it will take a lot of time to evaluate and implement something that will reliably control and protect the fruit of all the hours of hard work you'll put in on your app.

While the quandary vexes developers in every field, this article specifically focuses on Flex and AIR developers and their options for monetizing some of their cognitive surplus. Some of the big questions to be answered along the way are:

- Just what the dangers and the options for mitigating them?
- Are license control and code protection issues that developers should attempt to build their own solutions for?
- How reliable and secure are the off-the-shelf options and what approaches do they use?

- Should these issues be handled by a single product or multiple products?
- Once my product is ready to go, how do I sell and market it?

Spy vs. Spy

It is easy enough to dismiss the risks to your intellectual property as fringe hackers that won't really impact your potential business. In an online discussion I started in preparation for writing this article, a well known and respected industry blogger (who will remain nameless) had this to say:

"I wouldn't worry too much about it with Flex and AIR apps. Have you ever decompiled them and sifted through the pile of crap that is puked up by your decompiler? Flash applications are a different story, but the Flex compiler's method of breaking down MXML classes and all of the objects that are declared within it into AS objects, essentially obfuscates the code so much that it is nearly impossible to make sense of it all."

Even though I once held a similar view, I was a little shocked by this statement, considering the source. It completely marginalizes the issue. Developers who make a living at it tend to be used to high-level code that is readable and well organized. The internal code of the Flex framework itself can at times be challenging to understand, and the output of a decompiler can be even more difficult to follow since comments and local variable names are usually lost. So if a method is not clearly documented with meaningful variable names, these high-level developers dismiss it as unreadable garbage.

But this brand of thinking doesn't take into account the large number of people who apparently have nothing better to do with their time than write malware or crack retail software for fun and profit. They cut their teeth churning out bot-nets and viruses and build their *net-crad* creating cracks or key generators for nearly every licensed software product on the market; regardless of the complexities involved.

The real secret to the cracker's chances at owning your app lies in the last statement made by the commenter: "...it is nearly impossible to make sense of it all."

Newsflash: The cracker doesn't have to unravel it all, just key parts; specifically those that keep you from using it for free.

I'll give you a simple, first hand example. A long time ago, back in the days when dinosaurs ruled the earth and the 51" floppy was common, there was a very popular spreadsheet program that was as easy to learn as uno, dos, tres...

Anyway, they had a license control scheme that forced you to put the original installation disk in the floppy drive and hit enter each time you ran the program, even though it was installed on your hard drive. It read this *key disk*, presumably checking to see if a certain sector had a certain value or something along those lines. If any other disk than the key disk was inserted (including an exact copy made using sector copying *nibblers*), the program would exit to DOS; otherwise it would proceed to the functionality. Since the nibbler programs copied every sector and bit, not just those allocated by the official file table, the authors probably implemented a scheme that expected not only key data but also certain errors to be present on certain sectors. In some early copy-protection schemes, the errors were introduced by physically damaging a sector with a laser.

Whatever the actual scheme entailed, it was obviously Quite Clever, and clearly a lot of effort had been put into it. But it just bothered me that I had to drag out the key disk every time. I'd paid for the software but this step was a speed bump I didn't care for. So I decided to *fix* it. This was before the Internet added the possibility of sharing the result widely with the world, so morality didn't enter into it; it was inconvenient and as a competent developer, I figured I had the right to correct the experience, at least for myself.

Defeating their system was really easy. From a 386 assembly program, there was only one way safe way to *Exit to DOS* – Interrupt 21 / Function 4C. So I searched the executable for that interrupt and function and found only two instances; just what I would expect. One is the program exiting when the key disk isn't found; the other is when the user chooses to exit from the file menu.

Comparing a hex dump of the executable code to the processor's opcode table, it was simple to see that one of these interrupt calls was bounded by a subroutine jump followed by conditionally proceeding to the exit interrupt or jumping over the exit and into more unknown code. Probably going off to the subroutine to check the disk and then exiting or working based on the result. So I simply made it jump over the exit regardless of the return from the subroutine by poking in an unconditional branch opcode with a hex editor and that's all it took. It looked for a disk, didn't find one, and then brought up the program.

The point of this little trip down memory lane was: I didn't even have to understand the protection scheme or the actual program in order to defeat the license control. And I was a paying user, not a cracker! If you want to monetize your app, you cannot afford to marginalize the potential for your software to be cracked.

Aside from the software delivery methods, nothing has really changed. Today, the landscape of code protection and licensing still resembles a SPY vs. SPY cartoon. The Black Hats and White Hats are pitted in an ongoing struggle for code dominance. What does the playing field look like?

BLACK HATS

Client Cracking

Decompilers exist that make it dirt simple to steal the intellectual property of any compiled Flash movie (.swf), including those created by Flex and AIR. This includes the ability to create an editable source file which can be modified to bypass any licensing scheme. As with all products some are better than others, but the cream of the crop gives the Black Hats power akin to commanding the Death Star when it comes to cracking your app.

Key Generation

Even if they don't crack your client app, they can approach the problem from a key generation tack. Instead of trying to crack your app and patch it to ignore licensing, they can try to generate keys that it will accept as good. Thus it is important for the licensing scheme not to rely on magical keys that are deemed valid because they match the output of some secret algorithm.

Server Hacking

If you use an *activation server* for licensing that returns a yes/no response to whether your key is good, they can often use techniques like host file redirection to point the app to their own server which always says the key is good. Or they can try to break into your activation server and gain control of it. Obviously if there is a server part, you are vulnerable to attack on both the client and server side. And the more complicated the server apparatus, the more exploits there are for it.

Key Sharing

Even with code protection in place, your license control scheme must prevent people from easily sharing their license keys. Surely there will always be a paying audience for a product that offers good functionality, support and upgrades at a reasonable cost, but will that audience be large enough to offset the number

of people who choose to simply search the net for a key and run the software for free? Many people will be happy to share the key info with a friend if it is only a big impersonal string of numbers that holds the magic. If they must include personal information along with that key for it to work, they'll be less likely to share.

WHITE HATS

Securing the Client

Code protection products exist that combat decompilers by obfuscating and/or encrypting the code they are trying to steal. This process is inherently more difficult and fraught with danger since by scrambling the actual code enough to stump the decompiler or the reader of its output; it is possible to negatively impact the performance and reliability of the program you are trying to protect. Again, some tools are more effective than others at this task; some are nearly useless while others can cause the decompiler itself to crash or fall into a loop making it much harder if not impossible for crackers to get the goods.

License Control

Combined with code protection, license control allows you to reliably ensure that the application will only run and enable the features that are associated with the license issued. There are varying types of license control, but almost all rely upon a server component, or otherwise the application is highly susceptible to key generation attacks. Some approaches combine license control with code protection, some don't. Some place a lot of logic on the server and some don't. As with any system, the more complex the server component, the more prone to attack it is. There are more potential chinks in the armor. And beyond the threat of attack, there is also reliability to be concerned with. Finally, it is also important to consider the fees or percentages associated with the use of the server component.

Options for Publishing Your Application

Let's review your options as a company or individual developer or looking to monetize a new Flex or AIR application; from worst to best:

Forget it, it's too much hassle

Sadly this will be the choice that many developers will make. If the effort needed to build it and the cost and/or additional effort needed to protect and control it reach a certain threshold, it doesn't seem reasonable to try and tackle. Another good idea down the drain (at least until someone with more funding and time thinks of it).

Build it and give it away as freeware or open source

When the desire and energy to build it is there, but the time or inclination to protect and control it isn't, many developers

will choose this route. At least the idea will see the light of day, and perhaps if done well, it will be used by others. It's a very gratifying thing, knowing you've somehow contributed to the greater good, but it doesn't buy the baby a new pair of shoes. While there is a remote possibility that occasionally some people will take a moment to donate a few bucks to your project fund, I wouldn't hold my breath.

Build it and sell it without code protection or license control

This is the honor system. It relies entirely on the goodness of your fellow computer owner to pay you for the download and not share it with anyone. The argument for this approach is that if you put out a good product, charge a reasonable price and offer great upgrade and support options, and have a great marketing department, you might make enough revenue to justify the effort. Of course you must accept that anyone could easily snag the decompiled source, remove any logos or copyright messages and resell it or share the modified version. If you found out about it, they're probably in another country, and there's little you could do about it, short of starting a legal battle with them. Good luck with that.

Build it, without code protection, but implementing your own license control

Not that different from the previous option, since any effort you put toward rolling your own license control could still be defeated by decompilers. You're still relying mostly on people's good will in paying for a license, it's just that you also have to spend a bunch of time figuring out how your license control will work and implementing it. Having done this myself, I can only say beware, it's more complex than you might think. It really is more work than you want to attempt on your own, but license control is still the simpler of the two issues.

Build it and sell it using off-the-shelf license control and no code protection

While you still risk having your license control bypassed by decompilers, at least you don't have to waste any time figuring out how to implement it. It gives you control over your product to the extent that people who play by the rules (or are at least sufficiently concerned about viruses that they won't run cracked software) can choose different licensed experiences (such as basic or premium functionality), depending on the license control software. Different off-the-shelf license control products use different approaches and charge differently including one-time licensing fees, or even annual and per license fees or percentages.

Build it and sell it using a combined license-control and code-protection solution

Making sure you have both code protection as well as license control is the only way you can build a

business model around your Flex or AIR app that doesn't leak like a submarine with a screen door. But while combining the licensing and code protection in one product may seem attractive and even logical, the fact is you're tying two unrelated problems together – the very opposite of the loose-coupling we seek with object oriented programming. If the license control part is great but the code protection is lousy (or visa versa), then you're married to both for better or for worse. Migrating all of your license holders and their old installations of your product to a new licensing scheme may well be impossible, and you're at the mercy of the vendor to update their code protection to beat the latest decompilers.

Build it and sell it using off-the-shelf

license control and off-the-shelf code protection

This is the approach that makes the most sense to me. A license management solution need only fulfill a particular scope of functionality. If it fits the way you want to sell your app, then it doesn't need to change over time. If you're satisfied with the way it lets you manage your users, issue licenses and control your software, then you want that to remain stable. If it's built right, license management doesn't need new features or rethinking of how the old ones work. However, the business of code protection evolves continuously. The Black Hats figure out the latest obfuscation tricks the White Hats throw at them and the decompilers and the obfuscators are updated to cope.

Wash, rinse and repeat. This is why your virus protection software has to be continually updated; there are a flood of new viruses coming out all the time. The same goes with code protection. Therefore, it is something that inherently requires updating over time and having the option to choose or switch to the best code protection on the market at any time (without losing all the license holders you had signed up already) is really important. If your code protection vendor doesn't keep up with the latest threats, his competitor will. Thus a new code threat doesn't upset your sales and licensing process. You can switch your code protection vendor at any time and simply re-protect your code and redeploy or auto-update the app your customers are running.

License Control Solutions

There are several solutions available to you today. Some are only available for AIR. Some run their own activation servers, thus taking on the onus of security and uptime reliability for your license data. Some charge a fee per license, some either don't or are not up front about their pricing scheme. Ultimately it isn't reasonable to provide a comparison table, since the approaches are so different.

A combined License Control/Code Protection solution, they operate their own activation servers which your applications communicate with and they can encrypt your software as well (although this has been reportedly (<http://www.codingforums.com/showthread.php?t=195270>) been cracked). They offer a trial version for both Flex and AIR, but the details of pricing and fees associated with the premium versions are only available by talking to their sales staff.

Sharify by Luck Laboratories

– <http://www.sharify.it/>

An AIR only solution, Sharify provides a library to embed in your application and implements a one-time registration scheme, where the user only has to be online when they register the app (thus ruling out the ability to remotely shut down a license that has been compromised). They operate their own activation server and your application interacts with their API by making a call to their REST service. They charge a 3% fee on each license you issue, and interestingly they collect that fee by invoicing you monthly.

Shibuya by Adobe

– <http://labs.adobe.com/technologies/shibuya/>

Still in a pre-release phase, Shibuya was announced at Adobe Max in October 2009. The website doesn't make clear how their scheme works other than that they are the payment vendor, you can see some analytics and issue trial versions of your AIR apps. There is no mention about how they get paid, if it is a percentage, flat fee, recurring fee, or some combination thereof. Currently you can sell your applications on the Adobe AIR Marketplace.

Zarqon by Futurescale

– <http://zarqon.net>

Rather than operate private activation servers (with the attendant potential for hacking and downtime), Zarqon stores its licenses in an encrypted format in your own Amazon S3 account where you pay literally pennies a month for storage. This means you control the data and the oldest and most reliable cloud storage provider on the Internet is protecting it and serving your licenses. You administer licenses via a desktop app, and embed a library in your product for checking licenses. The unique system defeats license key generation attacks and makes key sharing highly unlikely. You can issue expiring trials, one-time registration, or your app can check every time it is run, allowing you to shut down abused licenses remotely at any time. You purchase a onetime license for use of the software. There are no other fees or percentages on sales and you can sell your software wherever you like.

Decompilers and Code Protection Solutions

You really have to see what you're up against from the decompilers in order to evaluate the effectiveness of the available code protection solutions. Fortunately, the top contenders on both sides of the fence offer fully functional versions of their products for evaluation. You can truly enlighten yourself by spending an afternoon downloading the decompilers, testing them against your app and seeing the results for yourself. Protect your app using any of the available code protection solutions, then point the decompilers at the result and see what happens.

The Leading Flash Decompilers

Trillix by Eltima Software – <http://www.decompiler-swf.com/>

Sothink by SourceTec Software – <http://www.sothink.com/>

I did quite a bit of research on the matter in order to protect my own product (Zarqon) and to cut to the chase, I found that Trillix is currently the best decompiler, but I was still able to crash it using Kindisoft's secureSWF. Honestly, the other code protection solutions didn't stand a chance against Trillix. Far from generating *unreadable garbage*, the Trillix Flash Decompiler does an unbelievable job of getting the goods from an unprotected app.

For example below is some of the sensitive source code from the Zarqon Flex Demo (<http://futurescale.com/v3/ZarqonFlexDemo/srcview/>) and the same methods as decompiled by Trillix. I would show you what Trillix made of the obfuscated code, but it crashed when I tried to decompile the swf see (Figure 1 and Figure 2).

As you can see, an unprotected Flex or AIR application can be easily decompiled. The source code can be extracted from the swf, saved, modified and recompiled into a version with license control bypassed.

As the author of Zarqon, I'm really highlighting the vulnerability of code licensed with my product here, but my mission is to point out that *you must have code protection in addition to license control*. And since the state of the art for code protection is advancing all the time, you need to decouple code protection from

license control so that you can zig when the bad guys zag without losing your license holder data.

The last thing I want is for a developer to purchase my license control product, go to the effort of building and selling their app only to be undermined by crackers because the application was unprotected. Therefore, I suggest combining Zarqon license control with a solid code protection solution such as secureSWF, which I use and endorse.

Leading Standalone Code Protection Solutions

secureSWF by Kindisoft – <http://kindisoft.com/>

SWF Protector by DCOMSoft – <http://dcomsoft.com/>

SWFEncrypt by Amayeta – <http://amayeta.com/>

I advocate testing all these products to see for yourself how effective they are, but as I mentioned before, the only one I was able to stump Trillix with was secureSWF.

All three products are run against your final swf and obfuscate the compiled bytecode in different ways. The thing that stands out about Kindisoft's secureSWF is that it provides 4 separate types of protection, each of which has multiple levels of *intensity* for fine-grained control of the final product. It even has the ability to change these settings for individual class methods, so that you can maximize protection while minimizing the obfuscation's impact on runtime behavior. And finally, you can encrypt sensitive strings within the application such as credentials. Basically you can put full-blast protection on, test your app, and if there are problems, peel back the layers and their settings until you get the right balance between protection and performance. And recently secureSWF has been integrated with Powerflasher's FDT development environment making the final step of code protection part an integral part of the development process.

Your goal is to make the decompilers crash while still producing a product that operates according to spec. As tedious as this may sound, it really isn't all that difficult to do in practice with the advanced tools available.

Selling Your Application

So once you've decided on how you'll do license control and code protection, you have to figure out how to sell

```
/**
 * Check the license at startup.
 */
private function startup():void
{
    // Your Zarqon Key (YOU MUST USE YOUR ZARQON LICENSE KEY HERE)
    var issuerKey:String = "ZQN-0F9579673774792FE4CA50D3D02C06D7";

    // The License key (YOU MUST CREATE A PRODUCT AND ISSUE A SITE LICENSE)
    var licenseKey:String = "ZFD-E324F2E4AD83064553A90A440640B04E";

    // Create the LicenseManager
    licenseManager = new LicenseManager( issuerKey, licenseKey );

    // Add Event Listeners
    licenseManager.addEventListener( LicenseManagerEvent.LICENSE_INVALID, licenseInvalid );
    licenseManager.addEventListener( LicenseManagerEvent.LICENSE_VALID, licenseValid );

    // Validate the License
    licenseManager.validateLicense();
}

private function licenseInvalid( lme:LicenseManagerEvent ):void
{
    pnlLicense.status = "License Invalid!";
    lblSite.text = licenseManager.site;
}
```

Figure 1. Zarqon Flex Demo Source

```
private function enableFeature(arg1:net.zarqon.api.entity.License, arg2:String):Boolean
{
    if (arg1 == null || !arg1.enabled)
    {
        return false;
    }
    return arg1.hasFeature(arg2) && arg1.getFeature(arg2).enabled;
}

private function licenseInvalid(arg1:net.zarqon.api.events.LicenseManagerEvent):void
{
    pnlLicense.status = "License Invalid!";
    lblSite.text = licenseManager.site;
    return;
}

private function startup():void
{
    var loc1:**"ZQN-0F9579673774792FE4CA50D3D02C06D7";
    var loc2:**"ZFD-E324F2E4AD83064553A90A440640B04E";
    licenseManager = new LicenseManager( loc1, loc2 );
    licenseManager.addEventListener( LicenseManagerEvent.LICENSE_INVALID, licenseInvalid );
    licenseManager.addEventListener( LicenseManagerEvent.LICENSE_VALID, licenseValid );
    licenseManager.validateLicense();
    return;
}
```

Figure 2. Unprotected Zarqon Flex Demo as Decompiled by Trillix

your product. How will you get your product into your users' hands and their payment into your bank account? The mechanics of selling the software and issuing the license will vary depending on the license control software you use.

Zarqon makes it easy to add an in-application link to your menu that takes you to your payment vendor's order management page, making it easy to sell anywhere.

Leading Payment Vendors

Google Checkout – <http://checkout.google.com/>

Paypal – <http://paypal.com>

SWREG – <http://www.swreg.org/>

Cleverbridge – <http://cleverbridge.com/>

You could sell by mail order if you like, but in reality, you need to be able to accept payment for your application online. Each payment vendor has a different approach to helping you sell your product, but in the end, all vendors take a percentage of each sale. Google and Paypal are neck and neck with their pricing, and are acceptably low for the service they provide. I chose Google because it's easy to setup and use, and because Paypal has a reputation (<http://www.google.com/search?q=paypal+freezing+accounts>) for freezing accounts too often for a variety of reasons. Cleverbridge is attractive if you plan to have a high volume of high-priced sales. They have affiliate programs, partner cross-sell and tight integration with your website. They also charge a fee for setting up your cart and a higher percentage of each sale.

Marketing Your Application

Finally, you need to get the word out about your application. Beyond blogging about it yourself, there are a number of things you can do to get the party started.

Get Bloggers to Write About Your Product

Getting others excited enough to blog about your product without paying them or giving them special treatment can be a challenge. And some applications will naturally be more tractable to getting bloggers pecking away. If your product is simple and easy enough to play around with for a few minutes you can likely pull it off. If it is an extreme niche product requiring more than a few minutes to get your head around and try out, you might find it more difficult. So don't get discouraged if you don't get a lot of advance blogging about your product. Once it's in the marketplace, and users get their hands on it, they'll eventually post about their experiences good or bad.

For instance, with Zarqon, I discovered that license control products are not so easy to solicit posts about because it just isn't that something you can test drive by poking a few buttons, be amazed and toss off a post

about it. Although it is easy enough to build into your app, you have to be working on an app or harboring an idea you'd like to monetize or it's just not that interesting. Although there wasn't much advance blogging about it, users have good things to say (<http://futurescale.com/v3/blog/120-why-we-built-zarqon#IDComment76171561>), though and that can be worth more than a solicited review.

"I am so happy I found Zarqon! 6 months ago I was desperately looking for an affordable, easy to implement, scalable air licensing solution and it was a dead end.

I paused the development of my project – I had put a lot of work and love into it and I did not want to give it away for free, or be tied to a commission based license scheme. I lost all hope and decided to give it up.

Just yesterday I accidentally discovered Zarqon. I couldn't believe my eyes! I tied it to my application within minutes and it was working perfectly. A dream come true! And when I saw the price... I was blown away! I was expecting a price tag of \$1000+

I bought it right away! As an independent & small developer I need to thank you for developing this amazing licensing tool." – Konstantinos

However the more review-friendly Balsamiq Mockups (<http://balsamiq.com/blog/?p=198>) product had incredible success by sending out a simple email to bloggers pointing them to the app, which they could run, do some doodles with and immediately start telling people about it. And Peldi, the founder of Balsamiq did a great job of documenting his app's rise to fame and all the various tips and tricks he picked up on the way on his company blog. It is an inspirational and instructive read for any budding entrepreneur.

Issue a Press Release

There are a number of companies out there who can help make sure that the appropriate media outlets are aware of your product. Magazines and industry blogs always have a *news hole* and they often fill it with relevant press releases. You can track them down and submit your press release directly or you can pay a service to do it for you. Some of these services even offer a free level that still ensures your message gets out on the web to some top sites.

Leading PR Vendors

PRLog – <http://www.prlog.org/pub>

PR.com – <http://www.pr.com/promote-your-business>

PR9.net – <http://www.pr9.net/press>

IMNewsWatch – <http://www.imnewswatch.com/PressRelease.php>

PRWeb – <https://account.prweb.com/>
i-NewsWire – <http://www.i-newswire.com/>
MarketWire – <https://www.marketwire.com/mw/>
PR Newswire – <https://portal.prnewswire.com/>

Run Advertisements

There are also quite a few options for advertising out there. You can run text ads, graphic ads, even interactive ads built with Flash. You can buy a slot on a specific website, or you can run keyword ads that appear in search results or on content sites deemed relevant.

Zarqon also makes it easy for you to put a handy in-application link to your ad vendor on your menu so that you can easily access your ad campaigns.

Leading Ad Vendors

AdReady – <http://www.adready.com/>
AdBrite – <http://www.adbrite.com/>
BuySellAds – <http://buysellads.com/>
FaceBook Ads – <http://www.facebook.com/ads/>
Google AdWords – <http://adwords.google.com/>

Regardless of the ad vendor you chose, if you will be making ad placements on specific websites, you need a way of gauging the value of the target site. The ad vendor may supply some amount of metrics to help you decide on the right choices, but I found an independent analysis tool to be pretty effective.

The free Website Value tool by Website Traffic Agents correlates a number of metrics like Google page rank, Alexa rank and others to assign a dollar value to the input website. Just put the name of the website you're interested in at the end of the URL (Example valuation: <http://www.webtrafficagents.com/WebSiteValue/puremvc.org>):

Running this against the various websites you're interested in running ads on (or sending blogger pre-release notice emails to) can give you a quick ranking of the sites in your industry.

Conclusion

It can be blindingly simple to see your completed world-shaking app in a single flash of insight. But then you have to implement it, protect it, license it, market it and sell it. It's all those parts after the initial vision that often keep developers from forging ahead and making their dreams come to life. After all, you could be playing Xbox with your cognitive surplus. But if you're still motivated to press on, it is well within your reach to monetize your Flex and AIR apps and see the hard work pay off.

My goal with Zarqon was to enable myself and others to easily control the licensing of their applications. I wanted the most secure and reliable cloud storage without having to write and defend a server component.

I built it with the tacit understanding that it could be broken. I took the initial stance that any lock can be broken, so ignore that segment of the population and focus on those who'll pay money for a great product, sold for a reasonable price and well supported. As time wore on, this idea seemed increasingly naive to me.

It was only near the end of the project that I really focused enough energy on the research into code protection and the dangers of not using it. I realized that although Zarqon itself might survive without code protection due to its design which anticipated crackers from the outset, the simple apps that I and others would publish using Zarqon for license control could not make the same assumption. If you just want to put out a great app, license and sell it, you should be able to do so and not worry that sales will fall to zero quickly because the pirated version is available on all the popular *warez* sites.

I discovered that the realm of code protection is huge problem unto itself and best left to dedicated experts in the field. Certainly not something for me to attempt to add to Zarqon in order to compete with the feature list of the one competitor that does bundle the two. In fact I realized how important it was for the two issues to be decoupled. With Zarqon and secureSWF, it is affordable and easy to combine secure and reliable license control with world class code protection and be ready to ship your Flex or AIR app within the day.

The Zarqon Desktop Control Center AIR application itself uses the Zarqon API for licensing and is protected by secureSWF.

Within a month after finishing Zarqon, I had the bulk of my next product (a simpler offering that targets a much broader audience) built, license-controlled and protected. It is in private testing now, but the process of getting it out there will involve the same issues I've described above, only it will be much simpler for having gone through it once and not having to invent the licensing.

I hope you'll consider letting your next idea come to life. Feel free to contact me if you'd like a hand with making it happen.

CLIFF HALL

A Flex/AIR consultant for hire through his company Futurescale and the author of the open source PureMVC Framework (<http://puremvc.org>), he recently learned more than he ever wanted to know about the necessity for code protection when he created the Zarqon Active License Control System (<http://zarqon.net>). His consuming hobby is making music which he does through his recording project Sea of Arrows (<http://seaofarrows.com>).

He can be reached via <http://contact.futurescale.com>



sourcemate

for Flash Builder 4

SourceMate, the new must-have companion tool for Flash Builder 4, increases an Actionscript developer's productivity exponentially. As Flash Builder's use in the enterprise continues to increase the importance of being a productive developer has never been greater.

With SourceMate refactoring features, code templates, and more you can take your Flash development to the next level.

Features Include:

Code Templates (i.e. code snippets)

- // Code templates available from both the content assist popup and the templates view
- // Works in both .as and .mxml files
- // Create your own templates and share them with other developers

Code Generation

- // Improved Getter/Setter generation
- // Generate Override/Implement methods
- // Generate Constructor from Fields
- // Generate toString()
- // Generate ASDoc comments on all elements

Refactoring

- // Extract Method
- // Extract Variable
- // Extract Interface
- // Convert Local Variable to Field
- // Extract Constant
- // Change Method Signature

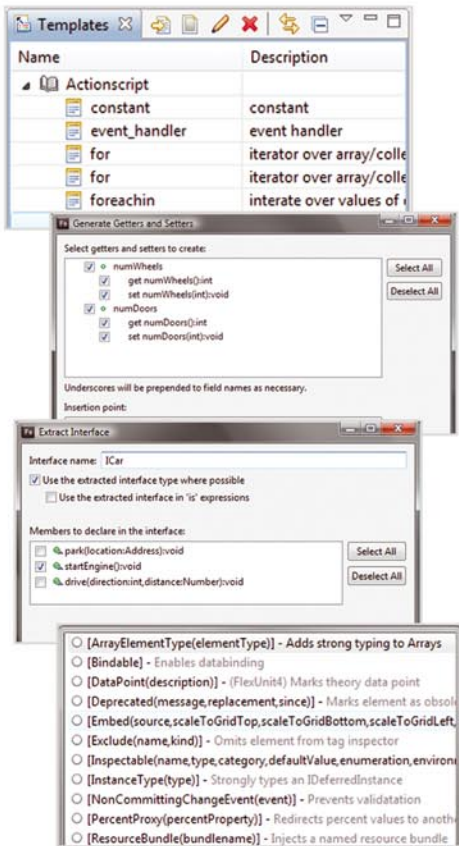
Metadata Tag Support

- // Content assist for metadata
- // Metadata validation during compilation
- // Support for 3rd party or framework metadata tags

Other features

- // Globally Disable/Remove trace() statements
- // Generate Ant build.xml from project settings
- // Build ASDoc Wizard
- // TODO/FIXME/XXX task creation

only
\$79



twitter

What they're saying about SourceMate

"... incredibly impressed with SourceMate ..."

"Loving SourceMate snippets."

"SourceMate > the product is awesome..."

Download SourceMate:

www.elementriver.com/sourcemate

1818 Library Street Suite 500 | Reston, VA 20190 | (571) 308-9475

 Element River

Flash Player 10.1: Live and Ready for Android

This month marks an exciting milestone for Flash Player. On Tuesday, June 22, Adobe announced the availability of the shipping version of Flash Player 10.1 for mobile. The final release has been posted to the Android Market. With Andy Rubin's announcement about Android 2.2 („Froyo”) being released as open source, Adobe expects that selected devices can be upgraded to Froyo and can install Flash Player. This release was a major undertaking to get the same Flash Player to work across various smartphones and desktop operating systems and browsers.

source: *Adobe Flash Platform Blog*

Tour de Flex 2.0 – Nearly 500 Flex Examples!

Adobe has just launched the new AIR 2 based Tour de Flex version 2.0 which now contains almost 500 Flex examples! The new version has new AIR 2 examples (only available in the AIR version of Tour de Flex) including: File Promises, Mass Storage Detection, Native Process, Open with default app, Socket Server. Also there are some great examples of the new Flash Player 10.1 and AIR 2 APIs including: Gestures, Global Error Handler, Globalization / Internationalization, Microphone access.

source: *Adobe Flash Platform Blog*

Adobe AIR 2 SDK Now Available for Download

After Adobe AIR 2 runtime came available, the Adobe AIR 2 SDK is also available for download!

The SDK and the Adobe AIR 2 documentation can be found on adobe.com

source: *Adobe Flash Platform Blog*

News selected by Gábor Csomák

Flash Camp Manchester

Date: Thursday 8th July 2010

Time: 12-8pm

Venue: Manchester Metropolitan University Business School, Aytoun Street (Manchester City Centre)

Flash Camp Manchester is a free event organised by the Midlands Flash Platform User Group in association with MMU Business School. Over an afternoon and evening, experts in Flash, Flex and AIR will share their knowledge through presentations and talks. Come and meet some of the Adobe team, professionals and community leaders and network with developers and designers. Whether you're just getting started with the Flash Platform or consider yourself a pro, there's something for you.

Confirmed speaker sessions include:

- Open Source Media Framework – Edwin van Rijkom (Adobe OSMF Development Team)

- Mobile User Experience – Anthony & Jerome Ribot (Ribot)
- FDT as a development environment – Michael Plank (PowerFlasher Solutions)
- PowerFlasher are also giving every attendee a free full copy of FDT Pure 3.5
- Flash Player for Mobile Devices – Mark Doherty (Adobe Flash Platform Evangelist)
- Flex 4, Spark components & Flash Builder 4 – Mike Jones (Adobe Flash Platform Evangelist)
- PaperVision 3D – Seb Lee-Delisle (Plug-in Media)

For more information visit <http://flashmidlands.com/flashcamp/manchester.html> or to book your place on-line go to <http://flashcampmanchester.eventbrite.com/>

Further enquiries please contact Trevor Ward on 07973 923494 or email

trevor@flashmidlands.com

Adobe AIR Plug-In for Aptana Studio

Aptana, in partnership with Adobe, released the Adobe AIR Plug-In for Aptana Studio. The updated version of the plug-in allows JavaScript developers to easily build rich, out-of-browser applications powered by the latest capabilities availabilities in AIR 2. The plug-in includes advanced support for debugging, profiling, code hinting and more. For instructions on how to install the plug-in, see aptana.com

Flash developer Grant Skinner recently published two very impressive experiments using Adobe AIR and Android. In his first experiment, he built a wireless slot car gas pedal. In his second experiment, he built a multi-screen game called Androidroids that, well, must be seen to be believed. You can see them on gskinner.com

Serge Jespers not only posted a video tutorial demonstrating how

to create native installers in AIR 2, he built an application using AIR that helps you package your native installers. The application is called Package Assistant Pro and is currently available for download. If interested, go to webkitchen.be

Adobe recently released the 1.0 version of the Open Source Media Framework (OSMF). This framework simplifies the development of media players by allowing developers to assemble components to create high-quality, full-featured video playback experiences in Flash Player or Adobe AIR. The framework is now available for download. There are also many resources for developers interested in getting started with OSMF.

Tour de ColdFusion is in beta stage. More info available on tourdecf.adobe.com

Source: *Adobe Flash Platform Blog*

SIMPLIFY GAME DEVELOPMENT BY USING GAMERSAFE

GamerSafe is a powerful tool that allows you to add quality features to your Flash game quickly and easily.

▶ **Universal Saved Games**

So players can resume their game even on another website!

▶ **Advanced Trophy System**

Reward your players with points they can spend on game stuff!

▶ **Customizable Leaderboards**

Bring out the competitive spirit .. painlessly!

▶ **Newsletter Support**

Keep the players updated on your newest games!

▶ **Micro-Transactions**

Create another revenue stream for your game!

▶ **LevelVault**

Players can store & share levels that they create in your game.



GAMERSAFE
Your gaming life, simplified

FIND OUT MORE 
GAMERSAFE.COM

Getting Started

to develop Android apps with Adobe AIR

In recent months, the Adobe-Apple war was ongoing. Apple has blocked Flash entirely from iPhone OS. No Flash on browser and no Flash on apps development. Adobe has stopped developing the iPhone export feature in the new Flash CS5.

What you will learn...

- Setup Android Emulator in Mac/Windows
- Basic steps for developing Android Apps with Adobe AIR

What you should know...

- Basic ActionScript
- Basic usage of Flash CS5
- Working with command line in Mac or Windows

But Adobe didn't stop creating tools for development of other mobile platforms. Since mobile platform are the future of web development, Adobe has just released the tools for developing apps on Android devices – AIR for Android.

The Android platform is a rising star of future mobile platforms. At the time i wrote this article, about 60 thousand Android apps were listed in the Android Market.

So, the development of Android apps is becoming more important to developers. And Adobe has started helping developers enter this platform with the AIR for Android. That means, we can develop Android apps using Flash platform – Flash CS5 or Flash Builder.

In this article, you will learn the necessary steps to start developing Android Apps by using Air for Android. I will focus on using the AIR for *Android Extension for Flash CS5*. If you are interested in developing using FlashBuilder, you can visit the Adobe Prerelease Program site for more information.

Tools

In order to start development with AIR for Android, we need the following tools.

Adobe Flash CS5

Adobe newest tool for creating Flash Platform content.

You can download the trial version from Adobe's website if you haven't upgrade to this version yet.

Android SDK

The SDK for Android platform development. We need the emulator included in the SDK for testing our apps.

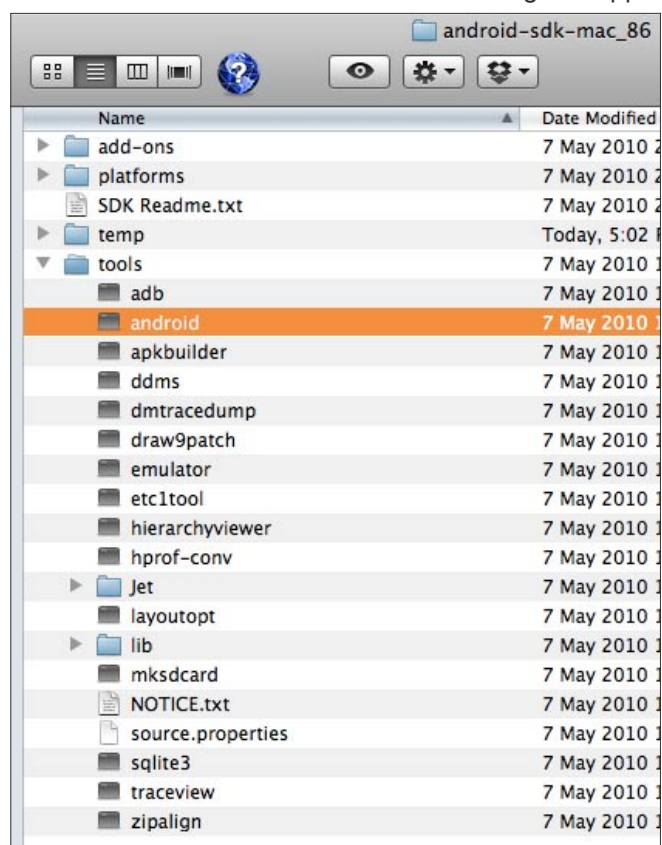


Figure 1. Android SDK

You can download it at <http://developer.android.com/sdk/>.

AIR for Android Prerelease Extension

The extension for Adobe Flash CS5 for developing Android apps. It will add the feature for exporting Flash content to Android devices. You can download it at prerelease.adobe.com.

AIR for Android Runtime

The runtime which is necessary for running apps in Android. You can download it at prerelease.adobe.com.

Android device

The Android device must be running Android OS 2.1 (Eclair) or 2.2 (Froyo).

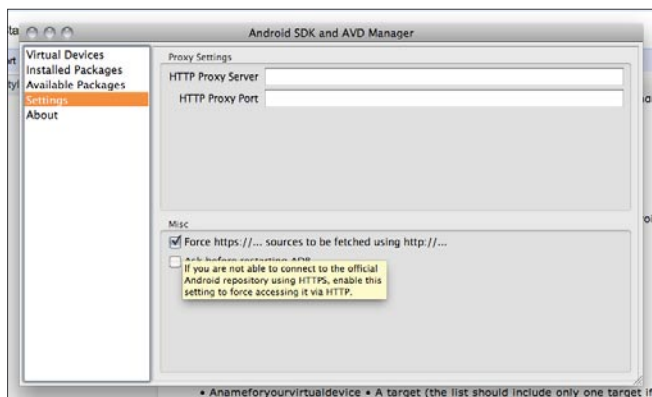


Figure 2. Android SDK and AVD Manager

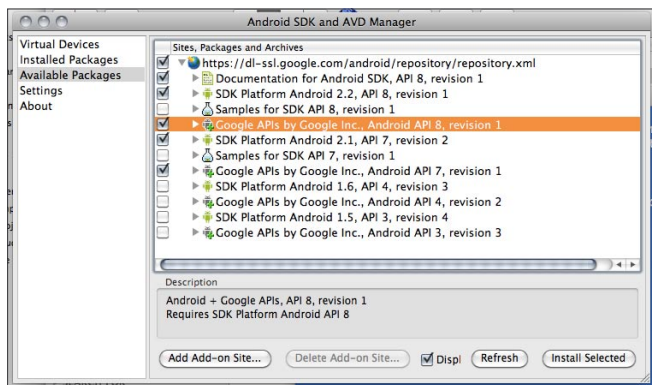


Figure 3. Install SDK API

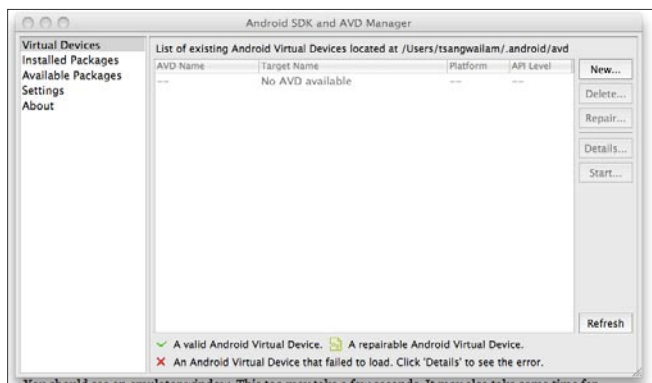


Figure 4. Start adding Virtual Device

Preparation

A. Install the AIR for Android Extension for Flash CS 5

To install, simply double click on the downloaded mxp and the extension will install via extension manager.

B. Install Android SDK and setup Android Virtual Devices Emulator

First, you need to install the Android SDK for transfer the apps to Android device. Also, you may want to test your apps with the Android emulator included in the SDK. You can download the latest Android SDK at <http://developer.android.com/sdk>. To install the SDK, simply

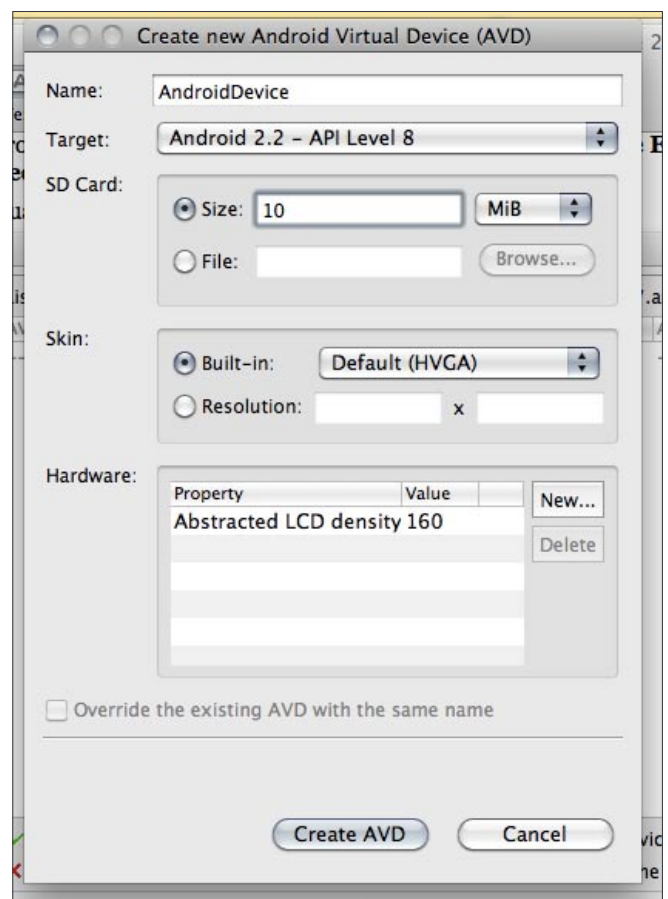


Figure 5. Setup a new Virtual Device

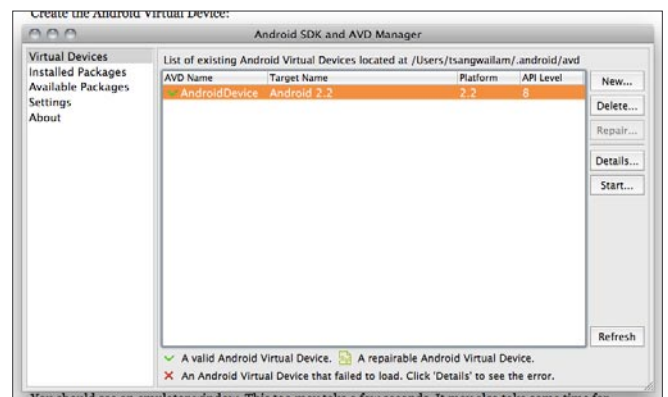


Figure 6. Virtual Device Ready

unpack the starter package to a safe location and then add the location to your PATH.

Setup Android Virtual Devices Emulator

1. Launch the Android SDK and AVD Manager application:

- In Windows, run the SDK Setup.exe file, at the root of the Android SDK directory.
- In MacOS, run the android application, in the tools subdirectory of the Android SDK directory see (Figure 1).

2. Select the Settings page and select the `Force https://.....//` option see (Figure 2).

3. Select the Available Packages page. You should see a list of available Android SDKs.

4. Select the *SDK Platform Android 2.2* (or if you want to develop on a Android 2.1 device, you can select 2.1) and click the Install Selected button see (Figure 3).



Figure 7. Android Emulator

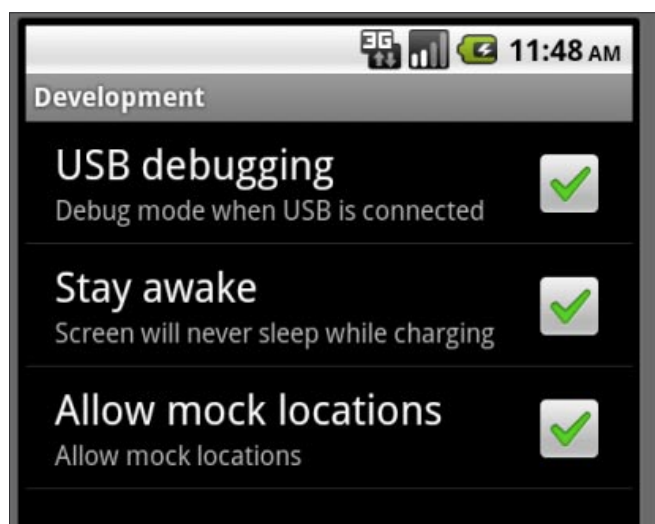


Figure 8. Enable USB debugging

5. Select the VirtualDevices page and click the New button see (Figure 4).

6. Fill in the necessary setting shown below. In the skin option, you may select different screen resolutions according to your target device. HVGA (640×240) or WVGA800 (800×480) see (Figure 5).

7. Click the Create AVD button. And a Virtual Device is created.

8. Now you can launch the new Virtual Device by clicking the Start button see (Figure 6).

9. Launching the Virtual Device may take about a minute. After the virtual device is launched successfully, you will see the virtual device like screen shot below see (Figure 7).

Install Adobe AIR Runtime for Android on the Virtual Devices

To install the runtime, simply put the apk in webserver which can accessed by any browser or send the apk via email and get it from email within the Virtual Devices.

After downloading the apk, the Android OS will prompt for installation. Follow the steps on screen and finish the installation.

Alternatively, you can install the apk via ADB tools. First, make sure the *USB Debugging* option is enabled in your Android device. Connect the device via USB or start Virtual Device if you using emulator see (Figure 8).



Figure 9. Create a new file from Flash CS5

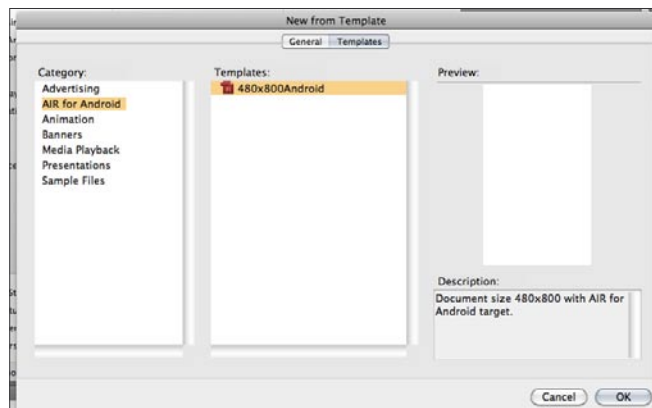


Figure 10. Template for Android

Launch Command (Windows) or Terminal (MacOS).
Run the command below to

```
adb install pathToRuntime/Runtime.apk
```

Install the AIR runtime, if you have not already done so, using the ADB install command:

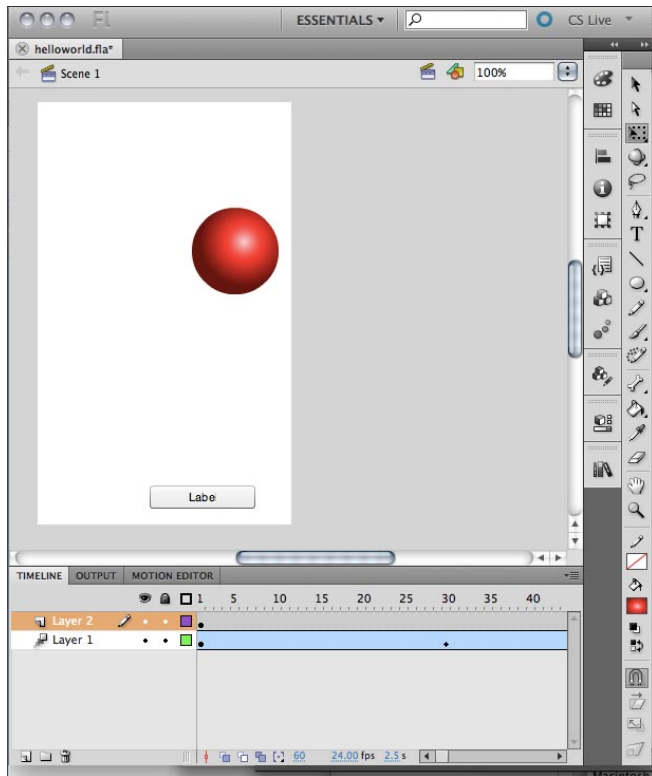


Figure 11. Create animation and content like normal Flash

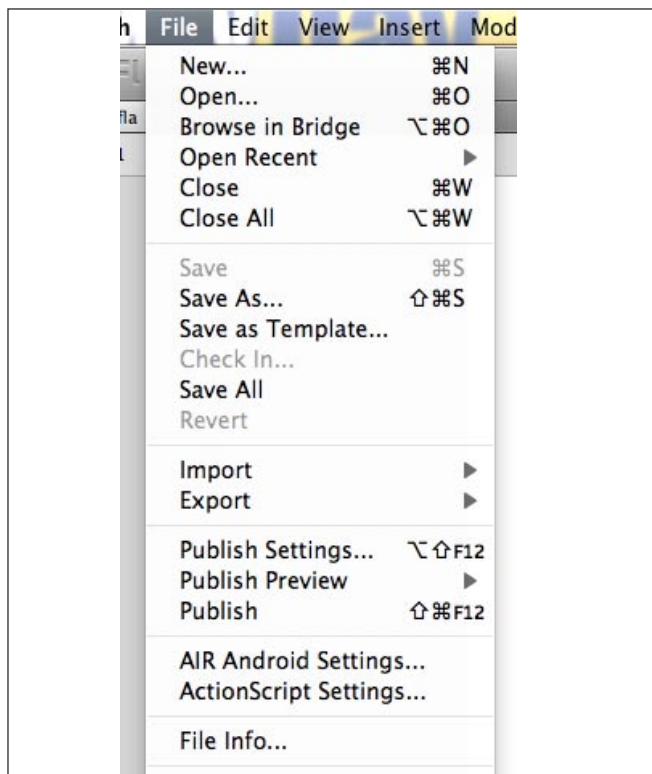


Figure 12. Select „AIR Android Settings“

For more information about ADB tool, see <http://developer.android.com/guide/developing/tools/adb.html> for ADB documentation on the Android Developers website.

Start building your first Android app by Air

Now, we start developing our first hello world Android app.

1. Launch Flash Professional CS5. Create a new document.

From the Home screen, you can click the link *Air for Android* to create an AIR for Android application see (Figure 9).

Or, You can select *menu>edit>new* to open a new document. From the new document dialog, select the Templates section tab. You will find a category *Air for Android* (You may check your installation of the .zxp if you can't find this option).

Select it and you will find the *480x800 Android templates*. This will create a project with with dimension 480x800px and 24 fps. You can change the dimension in the document setting if you want to develop in a different resolution. (Remark: The content will scale to fit the screen if the target device has a different resolution) see (Figure 10).

2. Now, you can create the app inside the Flash authoring tool like any normal Flash project. You can add animation, components or ActionScript. I create a

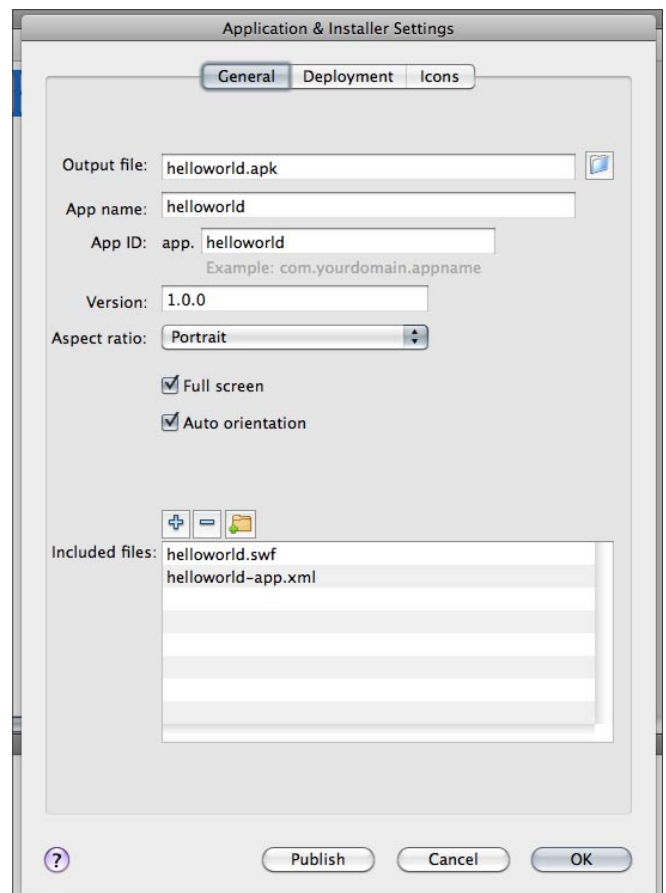


Figure 13. Output setting for Android apps

timeline animation and include a button component to demo the capacities of using original Flash content.

3. After finishing creating the Flash content, we going to publish it to Android.

Open the AIR Android Settings via *menu>file>AIR Android Settings* see (Figure 12).

In the *General* tab, you will set the basic settings for the Android app see (Figure 13).

OutputFile: The output filename of the Android app. Should end with *.apk*.

Appname: The name of the app. It also sets the icon caption.

AppID: The unity name of the app.

Version: The version of the app.

Aspectratio: The orientation of your app when developed. Can be Portrait or Landscape.

Fullscreen: Enable it if you want the app to run in fullscreen (i.e. no notification bar)

Auto orientation: Enable it if you want your app to automatic resize the view when the device change orientation.

Include files: Additional files included with the built apk. If you have additional resources which need to loaded, you can include them by clicking the add icon.

In the *Development* tab, you will set the options for development see (Figure 14).

Certificate: Each Android app needs to sign with a certificate. This option specifies the certificate file for signing the app. if you didn't want to purchase a certificate from a valid certificate provider, you can generate the certificate by clicking the *Create* button see (Figure 15).

Android Development type: The type of generated apk, Release or Debug. If you want to do debugging with the device, you need to select *Debug*.

After Publish: The action to take after publishing. Normally, we have to take both options.

Optional, you can specify the icon use for the Android apps. You can prepare 3 icons for resolution 36px,48px and 72px and the icon supporting png format see (Figure 16).

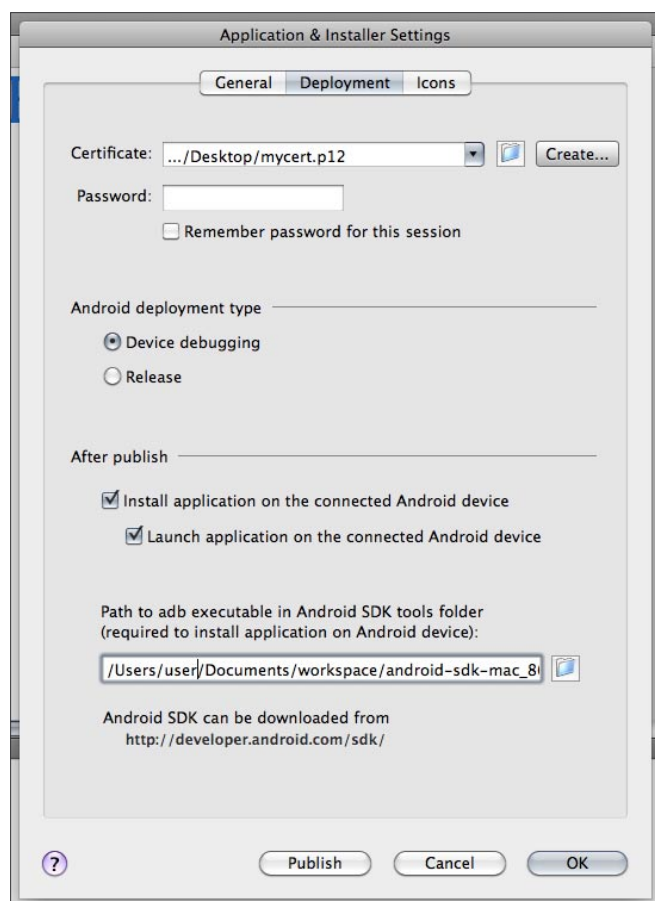


Figure 14. Development settings

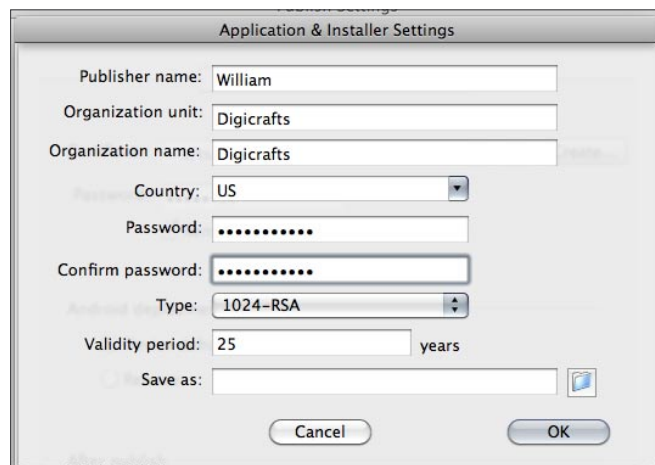


Figure 15. Create cert for signing Android app

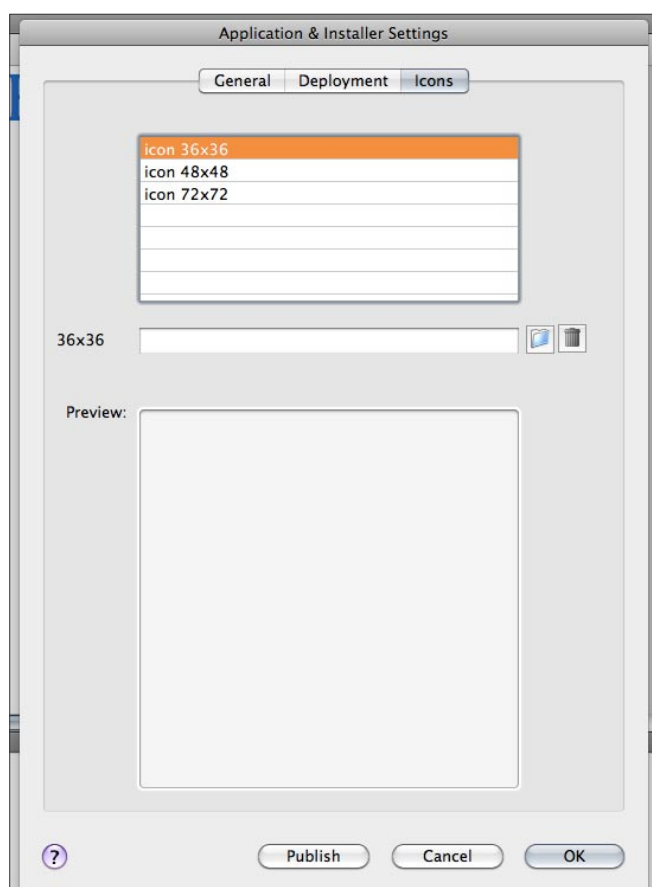


Figure 16. Setup Android apps icon

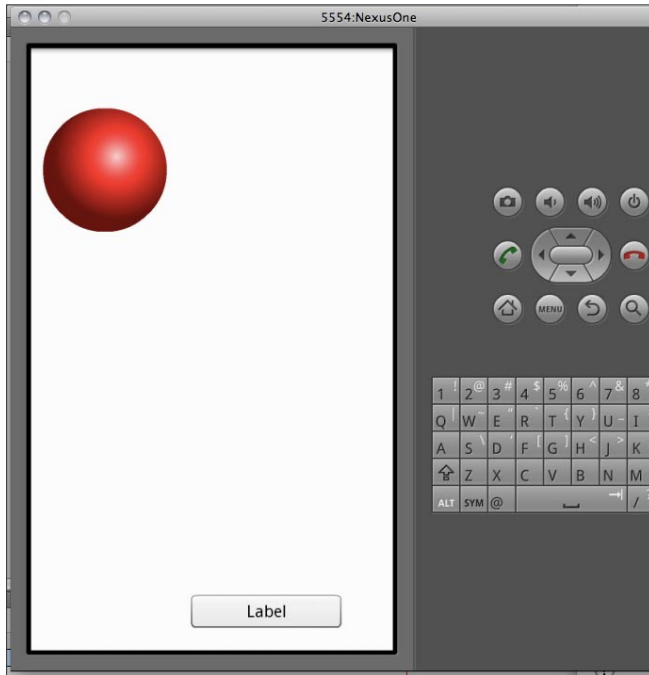


Figure 17. App created by Flash run on Android

Resources

- Adobe Prerelease Program – <https://prerelease.adobe.com/>
- Android Developer – <http://developer.android.com/>

4. Now, you can click the Publish button to generate the apk file and install the apk file to the Android device.

And the app should start automatically. Else, you can start the app from the program drawer of Android OS see (Figure 17).

Alternatively, you can manually transfer the apk file to Android System. Simply put the apk in webserver which can be accessed by any browser or send the apk via email and get it from email within the Virtual Devices.

After download the apk, the Android OS will prompt for installation. Follow the steps on screen and finish the installation.

5. After thought: Adobe made a such great tool for developers to step into mobile app development. It is a great start on the future trend of mobile development. Flash is a great tool for developing multimedia projects and has a bright future on mobile platform development. It can speed up the process of mobile app development. Also, there exists thousands of ready-to-use libraries for developing with AS3. Developers and designers can really focus on creativity rather than learning a new language.

WILLIAM TSANG

William Tsang is the Chief Developer of Digicrafts. Digicrafts is a leading solution provider of Adobe Flash Platform. <http://digicrafts.com.hk/components>



www.flexer.info

Preparing a Robot

to Play FarmVille Automatically

FarmVille is the most popular game on Facebook. The purpose of the game is to establish a flourishing farm. In the course of the game the player acquires seedlings, plants them, and later harvests them. If the player fails to harvest on time, the seedlings rot and the player receives no coins for them.

What you will learn...

- how the game client communicates with the server
- how the server identifies the player who made each request
- explanation of the game client flash vars list

What you should know...

- The robot is written in ActionScript 3
- familiar with classes in flash.net.* package such as NetConnection, URLLoader and URLRequest

These actions (acquiring seedlings, planting and harvesting them) are repeated in the course of the game, rewarding the player with coins.

In the next part, I will explain how to prepare a robot which will perform these actions automatically.

To prepare the robot we have to know what goes on behind the scenes when the actions are performed.

The game consists of two parts: the client's side (written in ActionScript 3) and the server side (written in php). The data is sent and received in amf format.

This architecture allows sending and receiving complex objects.

Explanation of Robot's Code

You can download the robot's original code here: <http://ffdmag.com/system/files/files/410/original/FarmVille.rar>

The robot is a Flash application written in ActionScript 3. The user interface consists of one button and a status bar. Pressing the button starts the robot which logs into the Facebook game page (the user name and password



Figure 1. The two images are displayed in a chronological order, first the seedlings grow (left image), then they are harvested and re-planted (right image). the robot is doing all the actions without the player doing anything, in the upper-part of the right image, the coins number raised from 5,057 to 7,387

to the Facebook site are taken from a Cookie file which is saved on the computer, so that when logging into

Facebook it is important to choose the option “automatic login next time” for the Cookie file to be saved on the computer). The messages sent from the game to the server include data from the Flash Vars variables (variables which are defined in SWF-file loading code and which the SWF file can use).

You can reach all of the Flash Vars variables by loading several HTML pages in a certain order, the same pages which would have been loaded by the player if he opened the game with the browser see (Figure 2).

Details of Flash Vars Variables

- flashRevision – the game version
- token – player’s current session identifier
- fb_sig_time – one of the variables which Facebook transmits to every application opened in it
- master_id – the ID number of the Facebook user
- app_url – the server-side address in the game, where all messages are sent during the game

```
55 var UrlLoader:URLLoader = new URLLoader();
56 UrlLoader.addEventListener(Event.COMPLETE, UrlLoaderCompleteEvent);
57 UrlLoader.load(new URLRequest("http://apps.facebook.com/onthefarm/"));
58
59 private function UrlLoaderCompleteEvent(e:Event):void
60 {
61     trace("UrlLoaderCompleteEvent");
62
63     var Html:String = e.target.data;
64
65     var StartIndex:int = Html.indexOf("http://fb-uc-1.farmville.com/flash.php");
66     var Uri:String = Html.substr(StartIndex, Html.indexOf("\n", StartIndex) - StartIndex);
67
68     var UrlLoader:URLLoader = new URLLoader();
69     UrlLoader.addEventListener(Event.COMPLETE, UrlLoader2CompleteEvent);
70     UrlLoader.load(new URLRequest(Uri));
71
72 private function UrlLoader2CompleteEvent(e:Event):void
73 {
74     trace("UrlLoader2CompleteEvent");
75
76     var Html:String = e.target.data;
77
78     var StartIndex:int = Html.indexOf("http://www.facebook.com/login.php");
79
80     if (StartIndex == -1)
81     {
82         GotHtml(Html);
83     }
84     else
85     {
86         var Uri:String = Html.substr(StartIndex, Html.indexOf("\n", StartIndex) - StartIndex);
87
88         var UrlLoader:URLLoader = new URLLoader();
89         UrlLoader.addEventListener(Event.COMPLETE, UrlLoader3CompleteEvent);
90         UrlLoader.load(new URLRequest(Uri));
91     }
92 }
93 private function UrlLoader3CompleteEvent(e:Event):void
94 {
95     trace("UrlLoader3CompleteEvent");
96
97     var Html:String = e.target.data;
98     GotHtml(Html);
99 }
100 }
```

Figure 2. Creating a group of requests in a specified order to get the flash vars list

```
137 var ZippedWorld:ByteArray = Base64.decodeToByteArray(World);
138 ZippedWorld.uncompress();
139 var JsonWorld:JSONDecoder = new JSONDecoder(ZippedWorld.toString(), false);
140 var SerializedWorld:Object = JsonWorld.getValue() as Object;
141
142 WorldObjectsArray = SerializedWorld.objectsArray;
```

Figure 3. Creating an objects array that includes all the elements in the farm based on the g_world variable

```
271 var FunctionArray:Array = new Array();
272 if (WorldObj.className == "Plot")
273 {
274     if (WorldObj.state == "plowed")
275     {
276         var ClearParamsArray:Array = new Array();
277         ClearParamsArray.push("clear");
278         ClearParamsArray.push(direction, seedName, WorldObj.seedName, className, WorldObj.className, deleted:false, state:"plowed", isProduct:isProduct);
279         ClearParamsArray.push(isFromWorld, isFromWorld, energyCost:0);
280         FunctionArray.push(functionName:"WorldService.getFoundation", params:ClearParamsArray, sequence:DefaultSequenceNumber());
281     }
282     if (WorldObj.state == "garden")
283     {
284         var HarvestParamsArray:Array = new Array();
285         HarvestParamsArray.push("harvest");
286         HarvestParamsArray.push(direction, seedName, WorldObj.seedName, className, WorldObj.className, deleted:false, state:"garden", isProduct:isProduct);
287         HarvestParamsArray.push(isFromWorld, isFromWorld, energyCost:0);
288         FunctionArray.push(functionName:"WorldService.getFoundation", params:HarvestParamsArray, sequence:DefaultSequenceNumber());
289     }
290     if (WorldObj.state == "planted") || WorldObj.state == "garden" || WorldObj.state == "fallen"
291     {
292         var FertilizeParamsArray:Array = new Array();
293         FertilizeParamsArray.push("fertilize");
294         FertilizeParamsArray.push(direction, seedName, WorldObj.seedName, className, WorldObj.className, deleted:false, state:"planted", isProduct:isProduct);
295         FertilizeParamsArray.push(isFromWorld, isFromWorld, energyCost:0);
296         FunctionArray.push(functionName:"WorldService.getFoundation", params:FertilizeParamsArray, sequence:DefaultSequenceNumber());
297     }
298     if (WorldObj.state == "fallen") || WorldObj.state == "garden" || WorldObj.state == "fallen" || WorldObj.state == "plowed"
299     {
300         var PlaceParamsArray:Array = new Array();
301         PlaceParamsArray.push("place");
302         PlaceParamsArray.push(direction, seedName, WorldObj.seedName, className, WorldObj.className, deleted:false, state:"plowed", isProduct:isProduct);
303         PlaceParamsArray.push(isFromWorld, isFromWorld, energyCost:0);
304         FunctionArray.push(functionName:"WorldService.getFoundation", params:PlaceParamsArray, sequence:DefaultSequenceNumber());
305     }
306 }
307 if (FunctionArray.length > 0)
308 {
309     HttpCall("FlashDevice.sendMessage", Req, DefaultSequenceNumber(), FunctionArray, 1);
310 }
```

Figure 4. checking the current state of a Plot and then sending a request to the server accordingly

```
272 private function GetSignedParamsObj():Object
273 {
274     var SignedParamsObj:Object = new Object();
275     SignedParamsObj.masterId = Uid;
276     SignedParamsObj.snId = "1";
277     SignedParamsObj.uid = Uid;
278     SignedParamsObj.wid = "-1";
279     SignedParamsObj.flashRevision = int(FlashRevision);
280     SignedParamsObj.token = Token;
281     SignedParamsObj.sigTime = SigTime;
282
283     return SignedParamsObj;
284 }
```

Figure 5. Creating the object which is sent in every request to the server and allows the server to identify the player who made the request

Besides these variables, there is also the g_world variable. This variable is defined in JavaScript code on the HTML page where the Flash Vars are located and it includes information on every object in the farm (like seedlings, trees, and animals). The variable value g_world is encoded in Base64, so that you need to convert the encoded string into an array of objects see (Figure 3).

After converting the string into an array of objects, you can go over every object, check its condition, and perform an automatic action, if necessary.

For example: if the object is a vacant garden bed, you can plant a seedling in it, and if the object is a seedling, you can check if it can be harvested, and if yes, send a message about it to the server see (Figure 4).

Every message sent to the server includes a signature object. This object allows the server to know who performed the request. The signature object includes the Flash Vars variables see (Figure 5).

ELAD COHEN

Elad Cohen, the chief programmer of www.GammonShark.com and one-on-one game platform. In the past, Elad served in the Mamram (IDF's computing unit) and received certificates of excellence and recognition.

Contact details: elad@GammonShark.com

Fluid Layouts

with ActionScript 3.0

With the emergence of HTML 5, Flash sites might start seeming like a thing of the past. I still see value in creating all out sites that use Flash. In this article you will learn how create a SWF that fills 100% of your browser and responds to the browser being resized.

What you will learn...

- How to setup your SWF for full screen
- How to handle browser resizing – Event.RESIZE

What you should know...

- How to setup a new project in Flex Builder 3
- Familiar with general AS3 programming concepts
- How to add a SWF to a HTML page

You've probably seen some really awesome sites out there that utilize all the real estate the browser has to offer. You might have even noticed that if you resize your browser window, that the content inside reacts or adjusts to that action. In this article you'll learn how to create a SWF that will utilize 100% of the browser and responds to the browser being resized.

Getting Started

First things first. Create a new folder *called* **Full-Screen** anywhere on your machine. Open up Flex Builder and create a new ActionScript project. For project name field, *type* **Main** and place the project into your newly created folder. We're going to use Tweeners (<http://code.google.com/p/tweener/>) for this project. To use any resource we'll need to add the folder to our source path. *Click next>Add Folder>Browse to the Tweeners folder>Click choose>Click OK>Finally, click finish.* Your project should now be setup. Let the coding begin!

Creating the background

We're going to create a seamless background from a bitmap that we're going to load in. If you've downloaded the source files that accompany this article you'll find *preview.jpg* in the *src* folder. Copy and paste that image into your *src* folder. Then add these three private properties to the class.

The `_bgImage` property is data typed to `BitmapData` and will be used to draw the seamless background. `_overlay`

will be used to draw a gradient over the seamless background to give it some color. Finally `_loader` is the property that will be loading in our bitmap.

Now that we have our properties established lets add the code to actually create the background. Copy and paste the code from Listing 2 and I'll explain what's going on (comments are removed).

The `init()` method is where everything starts. We first declare a variable that stores the path to the image. Then we instantiate `_loader` and pass in path to the begin loading the image.

The `imageComplete()` method is executed once the image has been completely loaded. When this method is called, we instantiate `_bgImage` and call the `draw()` method of the `BitmapData` class. The `draw()` method draws the source display object onto the bitmap image (<http://www.adobe.com/>). Next we call two methods, `drawBackground()` and `drawOverlay()`.

The `drawBackground()` doesn't do anything besides calling `beginBitmapFill()` on this object. We pass in the `BitmapData` object `_bgImage` to the method and draw a rectangle to the stage width and stage height. We

Listing 1. Class properties

```
private var _bgImage:BitmapData;
private var _overlay:Shape;
private var _loader:Loader;
```


finish by calling the `endFill()` method. Important: The fill won't be rendered without the call to `endFill()` method.

Here's what the compiled file will look like before we call the `drawOverlay()` method (see Figure 1).

The `drawOverlay()` method adds a nice gradient over the tiled background. We start this method off by creating three variables – colors, alphas, and ratios. Next we create the matrix object and call `createGradientBox()`. Passing in the current width and height of the stage to

the method. Finally, we instantiate `_overlay`. Then draw a radial gradient over the tiled background and set the blend mode of the shape to multiply (see Figure 2).

Setting up the stage

At this point we have our background created. Everything is looking great. If you haven't tried to already resize the window. You might see something like Figure 3. Not really appealing if you ask me. Let's fix that now.

Listing 2. Background creation methods

```
public function Main()
{
    init ();
}

private function init ():void
{
    var path:String = "preview.jpg";
    _loader = new Loader ();
    _loader.contentLoaderInfo.addEventListener (Event.COMPLETE, imageComplete, false, 0, true);
    _loader.load (new URLRequest (path));
}

private function imageComplete (e:Event):void
{
    _bgImage = new BitmapData (_loader.width, _loader.height);
    _bgImage.draw (_loader);

    drawBackground ();
    drawOverlay ();
    _loader.contentLoaderInfo.removeEventListener (Event.COMPLETE, imageComplete);
    _loader.unload ();
    _loader = null;
}

private function drawBackground ():void
{
    this.graphics.beginBitmapFill (_bgImage);
    this.graphics.drawRect (0, 0, stage.stageWidth, stage.stageHeight);
    this.graphics.endFill ();
}

private function drawOverlay ():void
{
    var colors:Array = [0x003465, 0x000F1D];
    var alphas:Array = [1, 1];
    var ratios:Array = [0, 255];
    var matrix:Matrix = new Matrix ();
    matrix.createGradientBox (stage.stageWidth, stage.stageHeight);
    _overlay = new Shape ();
    _overlay.graphics.beginGradientFill (GradientType.RADIAL, colors, alphas, ratios, matrix);
    _overlay.graphics.drawRect (0, 0, stage.stageWidth, stage.stageHeight);
    _overlay.graphics.endFill ();
    addChildAt (_overlay, 0);
    _overlay.blendMode = BlendMode.MULTIPLY;
}
```

Head back to the `init()` method and add the following lines of code from Listing 3. And add the code from Listing 4 after `drawOverlay()`; We need to stop the content inside the SWF from scaling when the window is resized. Setting the scale mode of the stage to `NO_SCALE` does exactly that. We then align the content of the SWF to the top left corner. Next we register the stage for the `Event.RESIZE` event.

`Event.RESIZE` is an event that is triggered each time the browser or Flash Player is resized. It must be registered with the stage in order to function. In all fluid layouts this event is used to retrieve the new width and height of the stage and then use these two values to reposition or scale the objects on stage accordingly.

Every time the `RESIZE` event is dispatched the `stageResize` method is called. In this method we set the width of `_overlay` to the width of the stage. We also set the height of `_overlay` to the height of the stage. Next we call the `drawBackground()` method again. This redraws the background every time the window is resized. Test the application again and resize the

window. Everything is working like it should. The window stretches the overlay and redraws the tiled background (see Figure 4).

Animated Repositioning

Creating and redrawing a background isn't the only thing you can do with `Event.RESIZE`. What if you had a content box that you wanted to be centered on screen? When the user resizes the window you just don't want the box to be stuck in the middle, you want it to animate. No problem. Add the following property to the class (under the other 3).

```
private var _container:Sprite;
```

Next add the following block of code under the `stageResize()` method (Listing 5).

Finally call `createContainer()` inside `imageComplete()` and call `repositionContainer` from `stageResize()`.

The previous code is pretty straightforward. We create a new property called `_container` and data type it to `Sprite`.

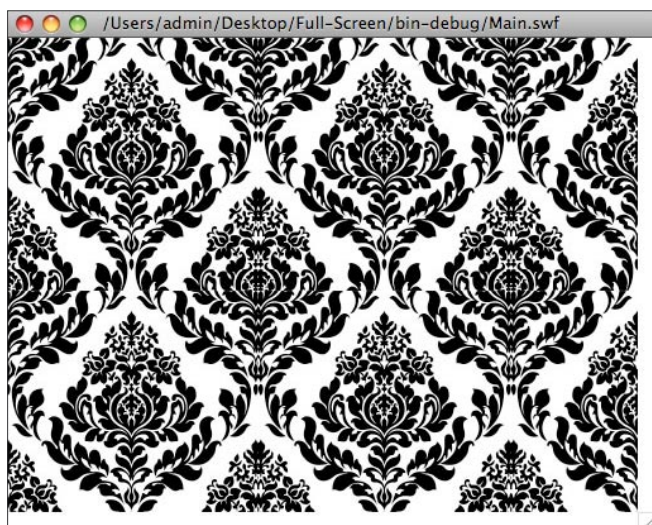


Figure 1. The tiled background that was created with the `drawBackground()` method

Listing 3. Setup the stage

```
stage.scaleMode = StageScaleMode.NO_SCALE;
stage.align = StageAlign.TOP_LEFT;
stage.addEventListener (Event.RESIZE, stageResize,
                        false, 0, true);
```

Listing 4. The `stageResize()` method that handles `Event.RESIZE`

```
private function stageResize (e:Event):void
{
    _overlay.width = stage.stageWidth;
    _overlay.height = stage.stageHeight;

    drawBackground ();
}
```



Figure 2. The fully created background

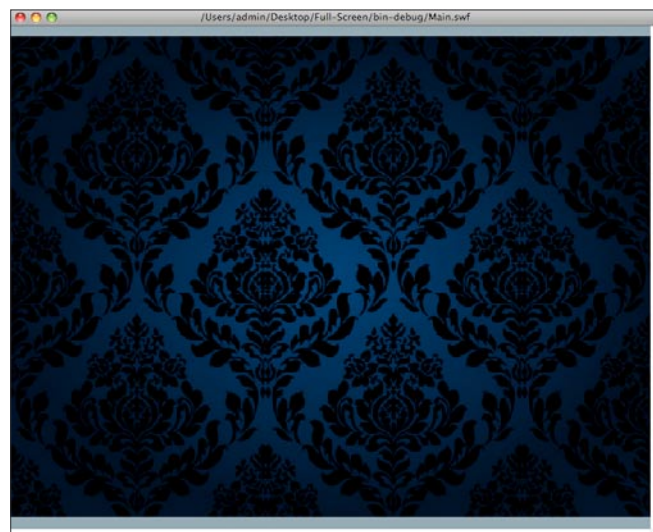


Figure 3. The window being resized without setting the stage up

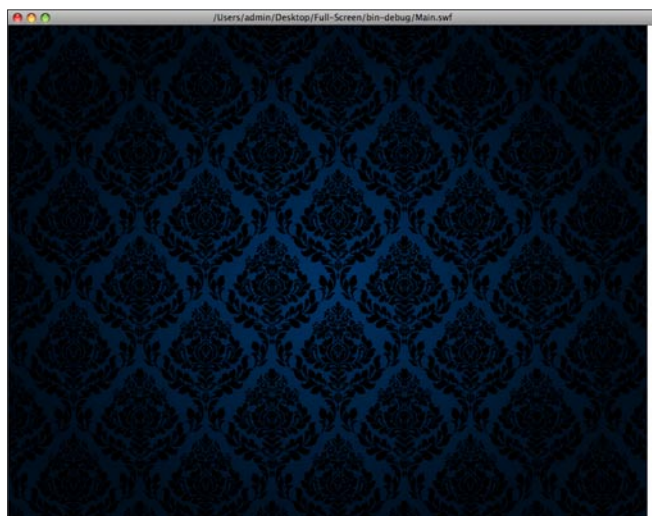


Figure 4. No problems now! The stage is now set

We then create a method called `createContainer()`. This method draws a rectangle with a white stroke and see through background. We then center it on screen. Finally we call the method from the `imageComplete()` method. Next we create a method called `repositionContainer()`. This method creates two variables; one for the new x center point and one for the new y center point. We then create a call to Tweeners and pass those two variables in. Now the container animates to the center of the screen when the window is resized.

If you don't call `repositionContainer()` from within `stageResize()`, the box will be stuck in position (see Figure 5). Figure 6 shows how everything should look.

That wraps up this article. Be sure to download the source files. The code is fully commented.

Listing 5. Container creation methods

```
private function createContainer ():void
{
    _container = new Sprite ();
    _container.graphics.lineStyle (3, 0xFFFFFF);
    _container.graphics.beginFill (0x000F1D, 0.25);
    _container.graphics.drawRect (0, 0, 250, 300);
    _container.graphics.endFill ();

    _container.x = (stage.stageWidth - _container.width) / 2;
    _container.y = (stage.stageHeight - _container.height) / 2;
    addChild (_container);
}

private function repositionContainer ():void
{
    var newX:Number = (stage.stageWidth - _container.width) / 2;
    var newY:Number = (stage.stageHeight - _container.height) / 2;
    Tweener.addTween (_container, {x:newX, y:newY, time:1,
        transition:"easeInOutQuart"});
}
```

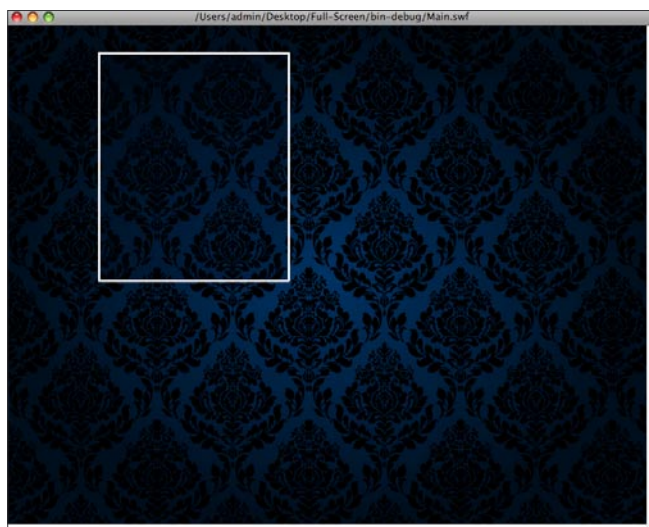


Figure 5. Adding a display object but not setting it to reposition

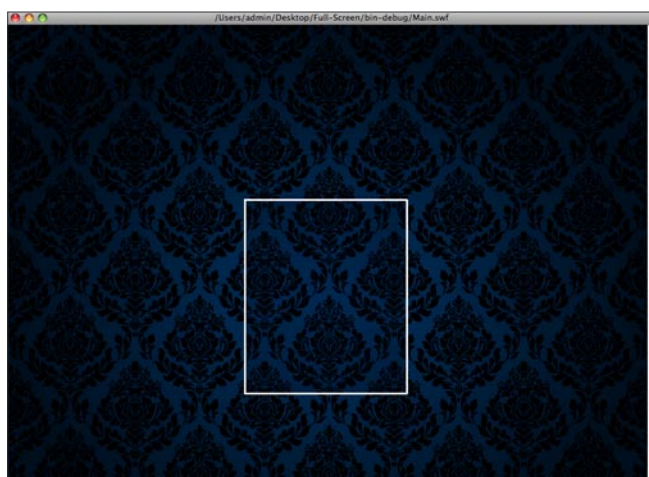


Figure 6. Setting the display object to reposition

I like to close these articles with a challenge. Can you think of a way to make the background image changeable? Right now the image is coded in such a way that it can't be changed. Sure you can go ahead and add a new image to the directory with the same name but what fun is that. If anyone has an idea for another article or has a general question, please email me at codedbyryan@gmail.com.

RYAN D'AGOSTINO

Ryan D'Agostino is an Interactive Developer at NAS Recruitment Communications in Cleveland, OH, part-time instructor at Virginia Marti College of Art and Design in Lakewood, Ohio, and freelance designer and ActionScript developer. For more information about Ryan please visit <http://www.codedbyryan.com/>

Flex 4

The Problem With Children

Flex 4 containers are not close relatives of Flex 3 Containers. If you want to avoid family feuds, you need to take care of the children. In this article, we'll find out how.

What you will learn...

- Some important features of the Flex 4 class library
- How use MX and Spark Containers in the same application
- How to avoid problems when manipulating children at runtime

What you should know...

- You need a solid understanding of Flex 3 and its classes
- You should be familiar with parent/child relationships of Containers
- Some familiarity with Flex 4 would be an advantage

Flex 4 introduces a whole new set of *Spark* containers which have the rather interesting and, at times, problematic ability to contain or be contained by the Flex 3-style MX containers. I say *problematic* due to the fact that Spark containers and MX containers are fundamentally different from one another. Each MX Container descends from the Container class, its contents are called its *children* and its parent is the Container that contains it. For example, if an MX Canvas contains an MX Panel, then the Canvas is the parent of the Panel and the Panel is the child of the Canvas.

Spark containers don't descend from the Container class, their parents are not necessarily the containers which contain them and their contents are called *elements* rather than *children*. When you want to

access the elements inside a Spark container you need to use a different set of methods than when accessing the children inside MX containers. And if that sounds confusing, things get even trickier when your application contains a mix of both MX and Spark containers. A small example may help to illustrate the problem.

Parents...

Let's assume you create a Flex 4 application. You put a Spark panel inside an MX Panel. You want to find all the parents of the Spark panel so you also add a TextArea and a button. You then write a simple method, `showAllParents()`, to display the parent of the Spark panel and all its parents until there are no more parents to be displayed. The code is shown in Listing 1.

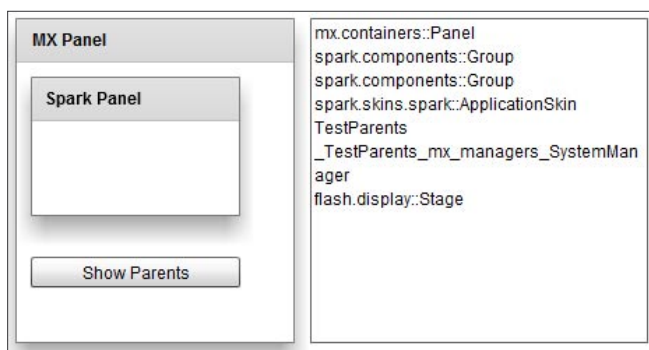


Figure 1. When a Spark Panel is inside an MX Panel, its parent is the MX panel

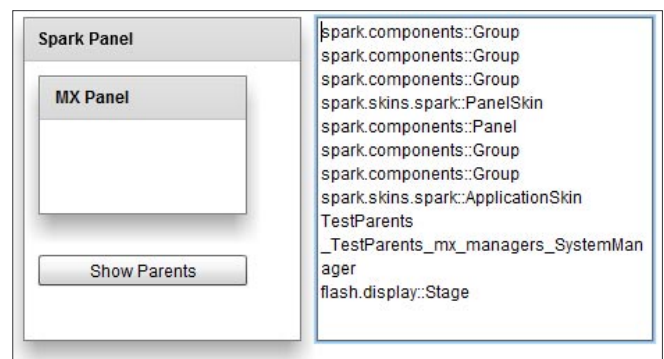


Figure 2. But when an MX Panel is inside a Spark Panel its parent is a Group object!

When you click the button (Figure 1), you will see that the first parent found is the MX Panel, `mx.containers::Panel`. Nothing surprising here, then!

However, now let's suppose that you rewrite the MXML so that the MX panel is inside the Spark Panel (Listing 2) and you recode the `button1_click()` function to call `showAllParents(mxPanel)`. This time you will find that the first parent found is not (as you might expect) the Spark Panel containing the MX Panel. It is, in fact, a Group object. What's more there are four levels of parent between the MX Panel and the Spark panel that contains it – three of these are Groups and one is a PanelSkin (Figure 2). Dealing with these *intermediate levels* of parent objects poses some special problems when manipulating Spark containers in ActionScript and I'll be looking at that topic in more detail next month.

...and Children

Now let's consider the other side of these relationships – namely, a container's children. Any controls placed inside an MX container such as a Canvas or a Panel must be descendents of `UIComponent`. They can be manipulated in ActionScript using methods such as `addChild()` and `removeChild()`, while the total number of children is returned by the `numChildren` property.

However, the contents of a Spark container may either be `UIComponents` or graphic objects such as `Rect` and `Ellipse`. The methods used to manipulate these are mostly similar to those used by an MX Container apart from the fact that the *Child* or *Children* part of the method name or property is replaced by *Element* or *Elements* – for example, `addElement()`, `removeElement()` and `numElements`. For a reference to equivalent methods in MX and Spark containers, see Table 1.

There is one extra complication you need to bear in mind. If you are an experienced Flex or Flash

programmer you will be aware of the fact that most of the MX methods aimed at handling children of containers are defined way up the Flash hierarchy in the `DisplayObjectContainer` class. All Flex `UIComponent` classes descend from `DisplayObjectContainer`. This is the line of descent of the MX Container class:

```
Container->UIComponent->FlexSprite->Sprite-
    >DisplayObjectContainer
```

And these are the lines of descent of the Spark containers:

```
Group->GroupBase->UIComponent->FlexSprite->Sprite-
    >DisplayObjectContainer
SkinnableContainer->SkinnableContainerBase-
    >SkinnableComponent->UIComponent-
    >FlexSprite->Sprite->DisplayObjectContainer
```

Given the fact that all children inherit the methods of their ancestors, it must be the case that Spark Groups and `SkinnableContainers` implement the standard *child* methods such as `addChild()` and `removeChild()` in addition to the new methods such as `addElement()` and `removeElement()`. You can verify this by looking at the code of `SkinnableComponent` in the Flex 4 SDK. All the default child-related methods are overridden. The overridden code has the sole function of throwing an error. Here, for example, is `SkinnableComponent`'s version of `addChild()`:

```
override public function addChild(child:DisplayObject):
    DisplayObject
{
    throw(new Error(resourceManager.getString("compo
        nents", "addChildError")));
}
```

Table 1.

Spark container	MX container	Description
<code>numElements</code>	<code>numChildren</code>	Number of children the container.
<code>addElement()</code>	<code>addChild()</code>	Adds a child to the container as the last child.
<code>addElementAt()</code>	<code>addChildAt()</code>	Add a child at a specific index in the container.
	<code>getChildren()</code>	Returns an Array containing all children.
<code>getElementAt()</code>	<code>getChildAt()</code>	Return a child at the specified index.
	<code>getChildByName()</code>	Return a child with the specified id.
<code>getElementIndex()</code>	<code>getChildIndex()</code>	Returns the index of a child.
<code>removeAllElements()</code>	<code>removeAllChildren()</code>	Removes all container children.
<code>removeElement()</code>	<code>removeChild()</code>	Remove the first child.
<code>removeElementAt()</code>	<code>removeChildAt()</code>	Remove the child at the specified index
<code>setElementIndex()</code>	<code>setChildIndex()</code>	Set the index of a child.
<code>swapElements()</code>	<code>swapChildren()</code>	Swap the indexes of two children
<code>swapElementsAt()</code>	<code>swapChildrenAt()</code>	Swap the indexes of two children.

In effect, the Spark container classes are *removing* behaviour that was defined by their ancestor classes. They can't literally remove those methods so they reimplement them to make them unusable. This is, it has to be said, rather a surprising thing to do. Normally, you would expect any class whose ancestors provide fully functional methods to provide the same fully functional methods themselves. The important thing to bear in mind is that while Spark containers cannot make use of the various *child* manipulation methods, those methods are

available to be called. As a consequence, it is legitimate to use a Spark container's *child-handling* methods in your code and your program will compile. However, if you then try call those methods, this will result in an error. I'll explain shortly a way of avoiding these problems.

Handling Children At Runtime

If your application's interface is fully defined at design-time and will never change at runtime, it is sufficient to take care to use the right methods for each type of

Listing 1. Find the Parent of a Spark Panel

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application height="400" width="800"
  xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:mx="library://ns.adobe.com/flex/mx"
  xmlns:s="library://ns.adobe.com/flex/spark">
  <fx:Script>
    <![CDATA[
      import mx.core.UIComponent;
      import flash.display.DisplayObjectContainer;
      import mx.collections.ArrayCollection;

      private function showAllParents ( aCtrl:UIComponent ):void {
        var aParent:DisplayObjectContainer;
        if ( aCtrl != null ) {
          aParent = aCtrl.parent;
          while ( aParent != null ) {
            ta.text += "\n" + getQualifiedClassName( aParent );
            aParent = aParent.parent;
          }
        }

        private function button1_click ( ):void {
          showAllParents( sparkPanel );
        }
      ]]>
    </fx:Script>
    <mx:Panel height="300" id="mxPanel" layout="absolute" width="250" x="10" y="10">
      <s:Panel height="100" id="sparkPanel" width="150" x="10" y="10"/>
    </mx:Panel>
    <mx:TextArea height="330" id="ta" width="425" x="300" y="10"/>
    <mx:Button click="button1_click()" height="22"
      id="Button1" label="Show Parents"
      width="150" x="35" y="255"/>
  </s:Application>
```

Listing 2. Find the Parent of an MX Panel

```
<s:Panel height="300" id="sparkPanel" width="250" x="10" y="10">
  <mx:Panel height="100" id="mxPanel" layout="absolute" width="150" x="10" y="10"/>
</s:Panel>
```


container. Any mistakes should show up quickly when you compile or run your application. But if your code manipulates components at runtime, potential errors may not be so easy to spot. In fact, you probably won't know you've made a mistake until your program goes unexpectedly wrong, quite possibly crashing. Say, for example, you move a component into a container in response to user interaction such as a button-click or a mouse-drag. If your code tries to add the component as a *child* of a Spark container or as an *element* of an MX container, you are in trouble!

In Flex applications it is not unusual to manipulate controls at runtime. My company's Flash Platform IDE, Amethyst, is an extreme example of this. At the heart of Amethyst is a Flex application called the Amethyst Designer, which allows users to create visual front-ends for their Flex programs by dragging, dropping, moving and resizing components. Every time components are added, removed or dropped into a container, the Amethyst Designer has to calculate a (sometimes quite complex) series of parent/child relationships.

For most of the development of Amethyst over the past couple of years, the Designer was written entirely in Flex 3. When the time came to rewrite it to support Flex 4 we faced a number of tricky problems. One of the trickiest was finding a way of supporting MX and Spark in the same design so that users could drag and drop any control (MX or Spark) into any container (MX or Spark) without the whole thing blowing up in their faces.

If you want to create Spark applications containing MX controls that can be manipulated safely in ActionScript you will face the same problems we did. Maybe I can save you some time and trouble by explaining how we solved these problems.

When a Container Is Not A Container

The first thing we did was to write some code to identify the type of each container. MX containers are easy. They all derive from the Container class. Spark containers, however, may either derive from `SkinnableContainer` or from `Group`. To categorize a container type you can define a few constants and write a very simple method to test the ancestry of a container passed to it as an argument (Listing 3).

This makes it easy to test the container type before calling either the Spark or the MX method required to manipulate its elements or its children. At the simplest level you could check if a Container is a `SkinnableContainer` or a `Group` and, if so, call a method such as `addElement()`, otherwise call the corresponding MX method such as `addChild()`. For a slightly more complicated, and more flexible, example, see Listing 4. This shows how you could test the precise Spark class type (`Group` or `SkinnableContainer`) and even deal with other class types if required. My `getClassType()` method returns a `Class` object which may be a class of some other type; this `Class` can be used to create objects of the actual class type. Using similar techniques you can write a whole library of Spark/MX compatible methods to add, remove and retrieve controls from containers without having to worry about the Spark or MX ancestry of those containers. Doing this will save you from accidental (and potentially catastrophic) errors when manipulating controls at runtime in Spark applications.

But dealing with children solves only half of the problem. You may also need to handle their parents. As I explained earlier, a Spark parent is not the same as an MX parent. So doing something as simple as moving `button1` into the same container as `button2` could, if not handled correctly, result in unexpected errors. If the

Listing 3. Determine whether a component is a Spark or an MX container

```
private static const SPARK_CONTAINER:String = "SPARK_CONTAINER";
private static const MX_CONTAINER:String = "MX_CONTAINER";
private static const NOT_A_CONTAINER:String = "NOT_A_CONTAINER";

private static function getContainerType ( aCtrl:DisplayObjectContainer ):String {
    var cType:String = NOT_A_CONTAINER;
    if ( aCtrl is Container ) {
        cType = MX_CONTAINER;
    } else if ( aCtrl is SkinnableContainer ) {
        cType = SPARK_CONTAINER;
    } else if ( aCtrl is Group ) {
        cType = SPARK_CONTAINER;
    } else {
        cType = NOT_A_CONTAINER;
    }
    return cType;
}
```

Listing 4. Given any type of container, find its class and call either MX or Spark methods

```

private static function getClassType ( anOb:Object ):Class {
    var containerClass:Class;
    containerClass = anOb.constructor;
    return containerClass;
}

// return a Class object of current object
private static function getSparkContainerClass ( anOb:Object ):Class {
    var containerClass:Class;
    if ( anOb is Group ) {
        containerClass = Group;
    } else if ( anOb is SkinnableContainer ) {
        containerClass = SkinnableContainer;
    } else {
        containerClass = getClassType( anOb );
        // maybe display warning msg here?
    }
    return containerClass;
}

private static function addCtrlToMXContainer ( ctrl:UIComponent, aContainer:Container):void {
    aContainer.addChild( ctrl );
}

private static function addCtrlToSparkContainer ( ctrl:UIComponent, aContainer:Object):void {
    var sparkClass:Class;
    sparkClass = getSparkContainerClass( aContainer );
    ( aContainer as sparkClass ).addElement( ctrl );
}

public static function addChildOrElement ( ctrl:UIComponent, aContainer:UIComponent ):void {
    var containerType:String = getContainerType( aContainer );
    try {
        if ( containerType == MX_CONTAINER ) {
            addCtrlToMXContainer( ctrl, ( aContainer as Container ) );
        } else if ( containerType == SPARK_CONTAINER ) {
            addCtrlToSparkContainer( ctrl, aContainer );
        } else {
            // Error! Unknown container type. Handle error here...
        }
    } catch ( e:Error ) {
        // handle Error here
    }
}

```

container of button2 is an MX container, then button2's parent is the container itself but if button2's container is a Spark container, its parent is not the container.

In the next article, I'll explain how to find the visible *parent* of any control ignoring any non-visible elements which intervene between it and its container. This will enable you to refer to Spark *parents* in a way that is entirely compatible with MX parents. Once again, this will simplify the coding of Flex 4 applications and make them far less error-prone.

HUW COLLINGBOURNE

Huw Collingbourne is Director of Technology at SapphireSteel Software. Over more than 20 years, he has programmed in languages ranging from Ruby to C# and has been a technical columnist for computer magazines such as PC Pro and PC Plus. He is the software lead of the Amethyst Designer, the Flex user interface design environment of the Amethyst IDE for Visual Studio. SapphireSteel Software: <http://www.sapphiresteel.com/>



Add complex interactive decision-making logic to your Flash applications.

Integrate your Flash apps with the proven knowledge automation expert system Inference Engine used by thousands of businesses including Fortune 100 companies, government agencies and the military. Go beyond graphics and multimedia to a unique complex probabilistic problem-solving analytical core!

Build systems that dynamically emulate a back-and-forth conversation with an expert to provide detailed, user-specific recommendations and advice for diagnostics, support, pre-sales recommendations, regulatory compliance and many other decision-support tasks.

How...your Flash application posts data to the Exsys Corvid® Runtime, a Java-based Servlet program, which uses heuristic rules for analysis. That sends XML back to your Flash application for user follow-up questions, to display results and recommendations, or perform other tasks. Systems can have a single or multiple SWF files. The underlying rule logic is developed in the Corvid Development environment - an easy to learn tool with a 26-year track record of handling complex rule-based decision support logic. All of the Action Script needed for the Servlet interactions is provided with Corvid and easily plugged into your Flash applications.

Download a free 30-day demo version of Corvid with tutorials - get your first systems built in just a few hours. Corvid systems can be delivered with Flash; or Corvid's Java applet and HTML based servlet runtimes.

For more information and links to sample systems go to: <http://www.exsys.com/flash.html>



Powered By




EXSYS
www.exsys.com

Remoting with Zend Studio and Flash Builder – Part 2

With Zend Studio, Zend Framework and Adobe Flash Builder

In our previous article, we looked at building the remote side of a PHP/Flash application using Zend Framework. Today we're going to look at matching up what you have on the PHP side with what you want on the Flash side.

In our first article, we looked at building the remote side of the interface with a Zend Framework application. Before we continue with that, we're going to explore a feature in Flash Builder that you will run across.

Let's look first at how to create the data. If you have an existing Flex application, you can right click on the project name and select *Properties*. From there, click on Flex Server and state the server side language with which you'll be working see (Figure 1). We have chosen PHP as the application server type, but there are a few more things to consider. The first is the web root, the public document root of your web server. In this case, it is set up as the document root for a specific virtual host, because there are several projects running simultaneously. Before you click *OK*, you need to validate the configuration, which will determine that

the document root you entered is accurate. The last text box to fill in, Output Folder, should be pre-filled for you (In our example we changed it slightly. It is usually entered as `<ProjectName>-debug`. We removed the `-debug`.) This aligns with the folder structure we created in the previous article.

Auto-generating your service layer

Among Flash Builder's features is the ability to connect to the database and create the class file needed to directly access the table. This allows you to build very simple CRUD applications with minimal coding from the PHP side.

To auto-generate your code, start at the same place you would if you were connecting to an existing service layer, but instead of browsing for the class, ask Flash Builder to build it for you see (Figure 2).

Doing this brings up another screen that allows you to specify the connection parameters to connect to the database and generate the class based on the columns in the specified table see (Figure 3).

This will generate a basic class file for your use, and a warning message see (Figure 4).

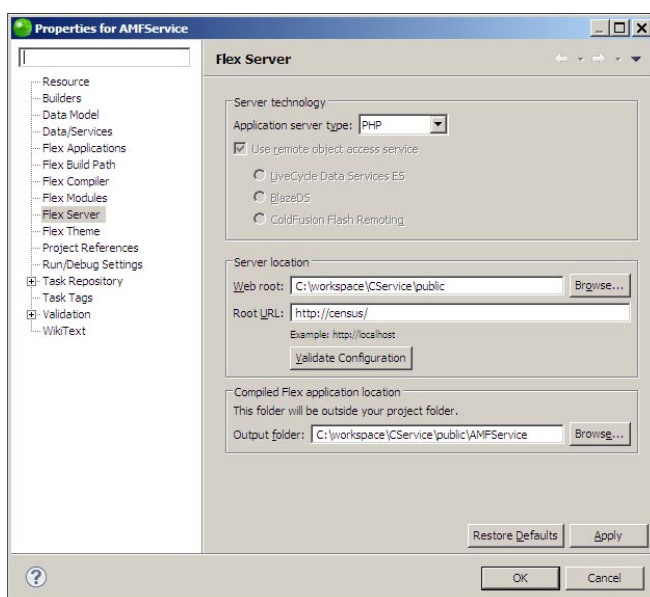


Figure 1.

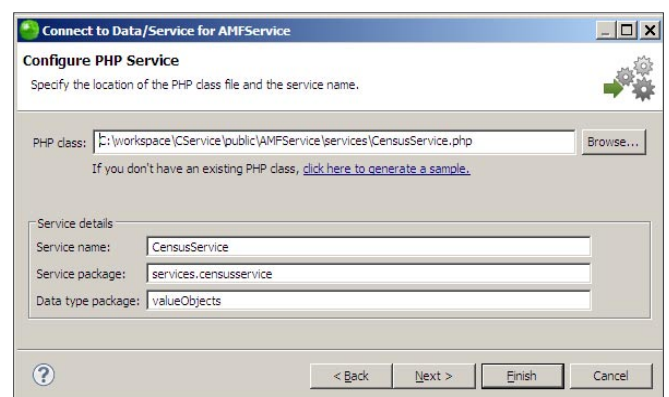


Figure 2.

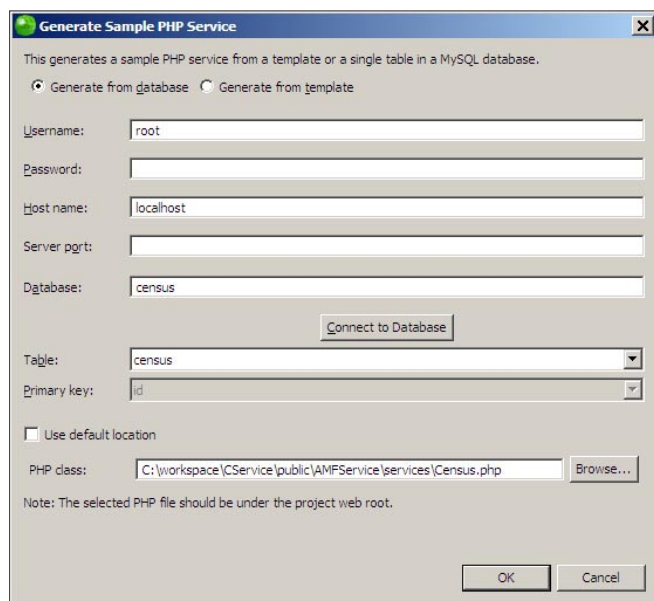


Figure 3.

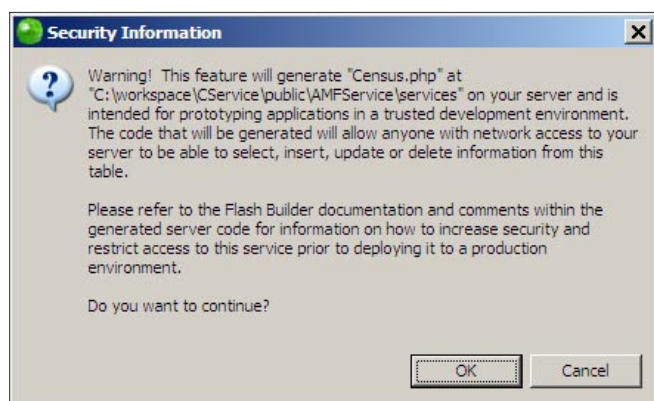


Figure 4.

This is an important warning, which corresponds to the previous screen's statement that this is a sample class. Here is a snippet.

```
class Census {
    var $username = "root";
    var $password = "";
    var $server = "localhost";
    var $port = "";
    var $databasename = "census";
    var $tablename = "census";
    var $connection;
    ...
}
```

There are three things to note about this code. First, it is generally not a good idea to have your username and password stored directly in a class file. That means that changing a database password requires a full redeployment of your application. Second, this code is compatible with PHP 4, which reached its end-of-life in 2008. And third, the generated code may not fit well within the context of a larger application see (Listing 1).

The code quality is fine, but does not lend itself well to re-use. However, as Flash Builder said, this should be considered sample code and not used in a production environment. When you compare this to our code previously written using `Zend_Db`, you can also see a marked difference in the amount of code that needs to be maintained.

```
public function getCensusByID( $itemID )
{
    $tbl = new Model_DbTable_Census();
    return $tbl->find($itemID)->current();
}
```

So, while you can use the auto-generated code, our advice (and Adobe's) is to use the code generation for scaffolding your application, at most.

Mirroring your remote server

To access data on the remote side in your Flash application, mirror the class. Here is where you might

Listing 1.

```
public function getCensusByID($itemID) {

    $stmt = mysqli_prepare(
        $this->connection,
        "SELECT * FROM $this->tablename where id=?"
    );
    $this->throwExceptionOnError();

    mysqli_stmt_bind_param($stmt, 'i', $itemID);
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    mysqli_stmt_bind_result(
        $stmt,
        $row->age,
        $row->classofworker,
        $row->education,
        $row->maritalstatus,
        $row->race,
        $row->sex, $row->id
    );

    if(mysqli_stmt_fetch($stmt)) {
        return $row;
    } else {
        return null;
    }
}
```

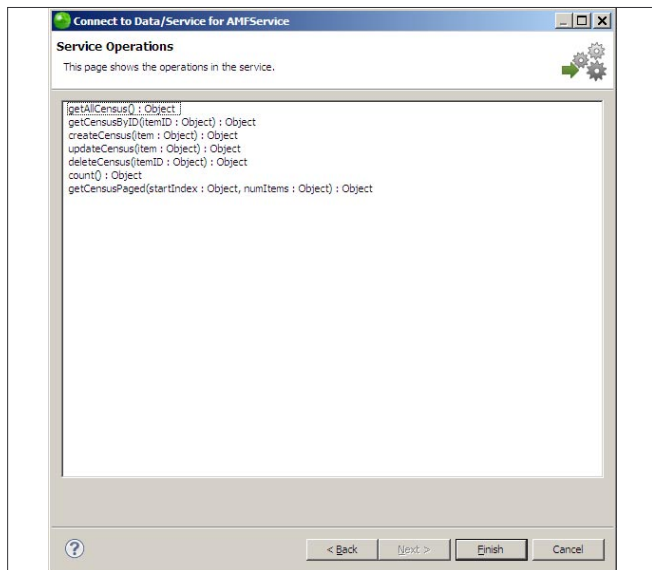


Figure 5.

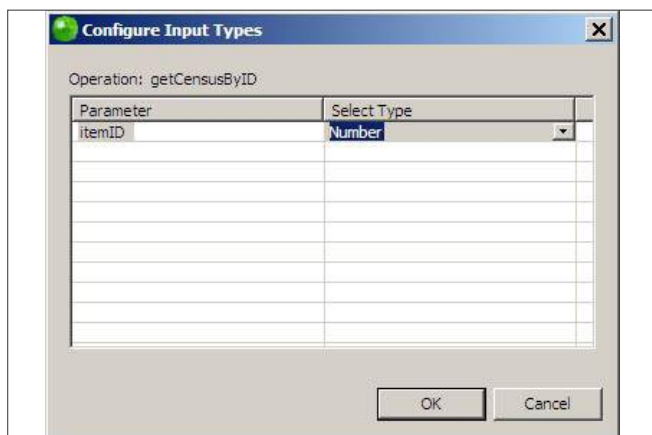


Figure 6.

want to do some thinking. In this basic example, we only have one database table that we need to provide an interface to, but there are two classes that we need to be aware of. The first is the CensusService class, which will provide access to the necessary functionality and a gateway between what you need to do and the model representing the data in the table.

Two types of introspection are needed. First, introspect the service class, taking the same route as when we used the auto-generated code, but rather than generating the code, introspect it see (Figure 5).

This is the same format as the generated class, but it is integrated more tightly with the application structure that we presented in our first article.

What we see here is the introspection of our PHP service class. But because we didn't define the parameter types for the individual service calls, though we could do that for when a certain object type is required. However, for the general PHP scalar types, that information cannot be determined, so we will need to specify the type see (Figure 6).

Here we are simply stating that the parameter being passed will be a number.

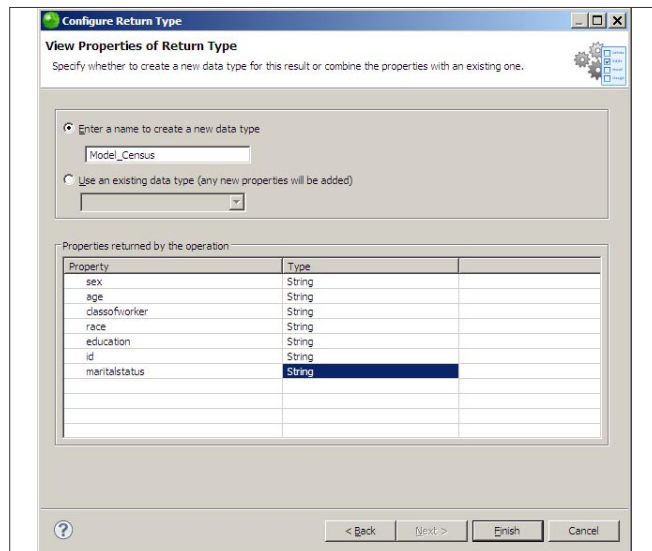


Figure 7.

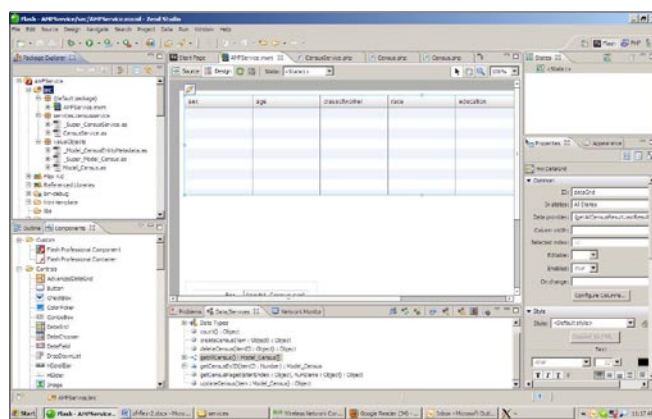


Figure 8.

The return value, however, is much more important. Flash Builder calls the service call and determines, based on the return type from PHP, what the corresponding Flash class needs to be see (Figure 7).

Two important things to consider here. First, the name of our class has been taken and is defined as a new class. Second, each of the individual properties has been found. Because this is a new data type, Flash Builder will automatically build out a class structure designed to directly interface with the PHP class see (Figure 8).

This generated code contains both the code for interfacing with the service class that we had defined, and definitions for the individual properties that align with the PHP class properties.

Conclusion

This article has demonstrated how you can bridge the gap between your Flash and PHP applications. In our next installment we will look at tying everything together and creating a working multi-tier application.

KEVIN SCHROEDER



fitc

San Francisco
2010

fitc.ca/sf

August 17-19

Over 70 presenters including:

Yugo Nakamura / Robert Hodgins / Erik Natzke / Mario Klingemann / Brendan Dawes / R Blank /
Jer Thorp / Theo Watson / Ralph Hauwert / Eugene Zatepyakin / Keith Peters / Stacey Mulcahy /
Brandon Hall / Didier Brun / Grant Skinner / Joa Ebert / André Michelle / Deepa Subramaniam /
Seb Lee-Delisle / Kevin Lynch / Jim Corbett / Lee Brimelow / Mike Chambers / Ted Patrick /
Richard Galvan / Mark Anders / Ryan Stewart

Interview with Chris Gross

Founder of ElementRiver

ElementRiver, and more specifically SourceMate, are quickly becoming household names to Flex and Flash developers. This year Element River released SourceMate, an advanced add-on to Flash Builder 4 that adds features like code generation, refactoring, snippets, an ASDoc wizard, an Ant build wizard, and more. Flex developers quickly took notice. In this interview, we chat with Chris Gross, Founder and President of ElementRiver.

It seems like the Flex world was waiting for an add-on like SourceMate. What gave you the idea to create SourceMate?

Yes, we believe SourceMate fills a real need for advanced IDE capabilities for Flex and Flash developers. We created SourceMate because, just like others, we also wanted those features. All of us at ElementRiver are both Java and Flex developers. In the Java world, we were used to the advanced features offered by IDE's like Eclipse and we wanted those for Flex. We also have significant experience developing Eclipse plug-ins and development tools. So it made perfect sense for us to fill the need for these tools in the Flex ecosystem.

From code generation (ex. Generate Getters/Setters) to refactoring, to metadata code hinting and TODO markers, SourceMate contains a wide range of time-saving features. Which features are the most popular and why?

The code templates, which could be described as insertable snippets, are probably the most popular. The feature is very powerful and very flexible. Just enter the first few characters of the snippet's name, trigger Flash Builder's content assist via [CTRL]+[space], and SourceMate will insert the snippet into your code. The power really comes from the variables inside those snippets. The snippets can contain variables which will be evaluated when the snippet is inserted. For example, `${enclosing_method}` is a variable that will be resolved to the name of the function the snippet was inserted into. Variables can also provide a tabbed completion mode when snippets need additional input. For example, SourceMate includes a snippet named *constant*. Type *const* and [CTRL]+[Space] and the snippet is inserted, but remember that SourceMate needs you to fill in the new constant's name, data type, and value. This is made easy by provide highlighting around the spots that need values and by allowing you to tab between these spots. Since you can create your

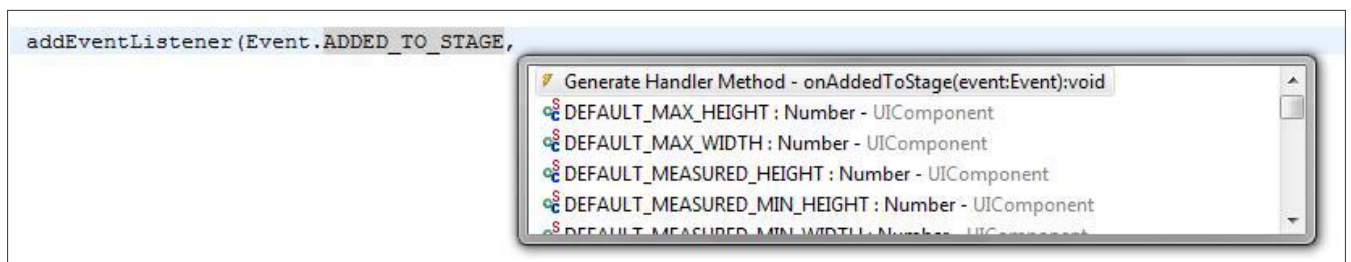


Figure 1. `addEventListener`

own snippets, and even export and share them, you make use of these powerful features too.

There are two other features I'd also mention that aren't the most obvious features but seem to garner a lot of praise. The first is our automatic event handler generation for Actionscript. Flash Builder 4 provides a similar feature in MXML. Put your cursor in an event property of an MXML tag and Flash Builder will offer to create an event handler for you. SourceMate can do the same thing after you type `addEventListener(Event.TYPE, .`. Just trigger content assist via `[CTRL]+[Space]`, and you'll see the event handler option (Figure 1).

The second feature is part of our metadata tag support. SourceMate provides metadata tag code hinting but there's one part of this feature that users seem to love the most. If you enter the `[Embed]` metadata tag, SourceMate can provide hints for the `source` attribute. So you no longer have to dig through your project to remember where your images are or how to path those correctly to the source file. SourceMate will provide a popup of images for you to select from, and insert the correct path for you.

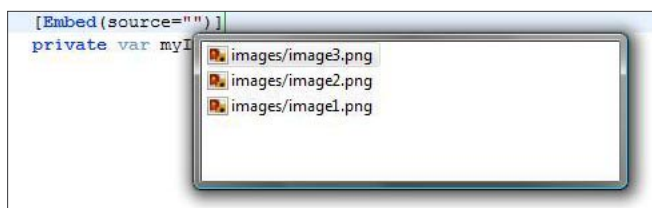


Figure 2. *Embed*

When do you plan to release the next version of SourceMate? What features will it include?

We've just announced the public beta for SourceMate 1.1. SourceMate 1.1 includes a handful of new features primarily around customizing the code SourceMate generates. SourceMate now includes options on where brackets are placed, how event handlers are named, the order of the function modifiers, and more. We've also snuck in a few more useful features like automatic hyperlinking in stack traces in the Flash Builder console. Users of our enterprise Flex framework, Potomac, will also be excited by the new Potomac/SourceMate integration.

You can read more about v1.1 on our website: <http://www.elementriver.com/announcing-sourcemate-v1-1-beta/>

We're also making plans for a follow up release, SourceMate 2.0, later this year.

Software developers are notoriously stingy when it comes to spending on tools. With so many free and open source options out there, why should they spend money on SourceMate?

While there are a few successful open source add-ons for Flash Builder, none of them come

close to the functionality provided by SourceMate. SourceMate's price tag, currently only \$79.99, is very reasonable. Given the high compensation of most software developers, SourceMate only has to save you a few hours to justify the investment. With all the time saving and high efficiency features, we're confident SourceMate will pay for itself in less than two weeks.

You mentioned Potomac. Can you tell us more about Potomac and what it does?

Potomac is our open source framework for modular enterprise Flex applications. Flex provides the basic building blocks for a modular application. Potomac builds on those features to provide a full framework to take away the pain and complexity that arises in large modular Flex applications. Potomac is loosely modeled after OSGi and the base Eclipse platform. OSGi is the defacto standard for large modular applications in the Java world. The key feature provided by OSGi and Potomac, is to abstract away the loading and management of modules. You simply declare the dependencies between modules, or declare pieces of your application (in Potomac we call them *parts*) and Potomac will load modules when necessary. Has the user requested a certain 'part' on screen? If so, Potomac will load the module that contains that part, and any modules it depends on. Potomac will do this all without requiring special coding from the developer, only the declarative dependencies.

Potomac also includes many features necessary in large enterprise applications like dependency injection, a sophisticated UI framework, and a Flash Builder plug-in to assist in Potomac development.

You can find more information about Potomac at <http://www.potomacframework.org>.

Thanks Chris. Any closing thoughts?

I would reiterate that SourceMate will save you time and money, increase your efficiency and eliminate some programming headaches and you can download and install it for free using our 30 day free trial to prove it. If you are intrigued by what people have been saying on their blogs and on twitter about SourceMate, then give it a try. You have nothing to lose and everything to gain. Once you start coding with SourceMate, it's hard to go back to just plain Flash Builder.

You can see an overview of features, download the free trial, and purchase SourceMate at: <http://www.elementriver.com/sourcemate>.



Ryan D'Agostino

Ryan D'Agostino is a Interactive Developer for NAS Recruitment Communications in Cleveland, Ohio. He has worked with Flash since 2004. Ryan currently teaches web design, flash cartooning, and ActionScript 3.0 at Virginia Marti College of Art and Design in Lakewood, Ohio. Ryan currently lives and works in Cleveland, Ohio. In addition to teaching and writing tutorials, he also does freelance work for web development companies.

He is very passionate for teaching and adapting to the constantly changing web frontier. Ryan has previously written tutorials on creating an XML Photo Gallery with AS3. In this issue he has another tutorial on creating Fluid Layouts in AS3. He plans to continue to write tutorials to help with different aspects of Flash and Flex.

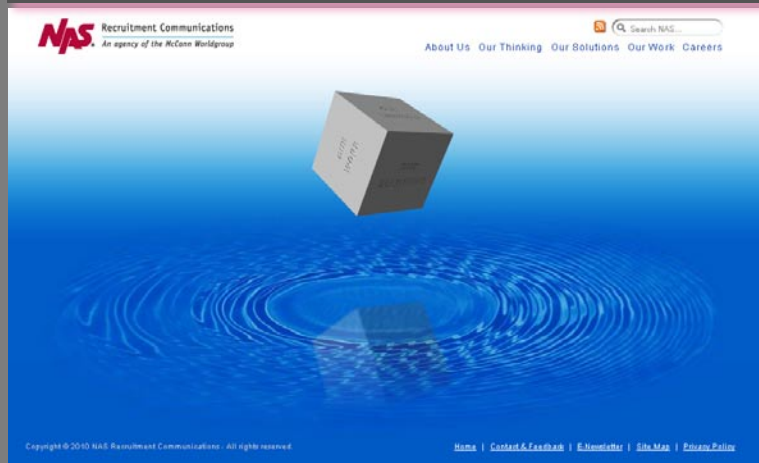
In his spare time, Ryan enjoys learning as much as possible. He is currently learning ASP.net with C#, HTML 5, JQuery, and Java. Ryan strives to be well rounded in development, design, and problem solving.

Projects:

www.precisionsupply.com – Design/HTML/CSS/Flash
www.nasrecruitment.com – Design/HTML/CSS/Flash/Papervision3D

Contact Info:

Ryan D'Agostino
codedbyryan@gmail.com
www.codedbyryan.com



Flash & Math

www.flashandmath.com

ActionScript 3
Tutorials

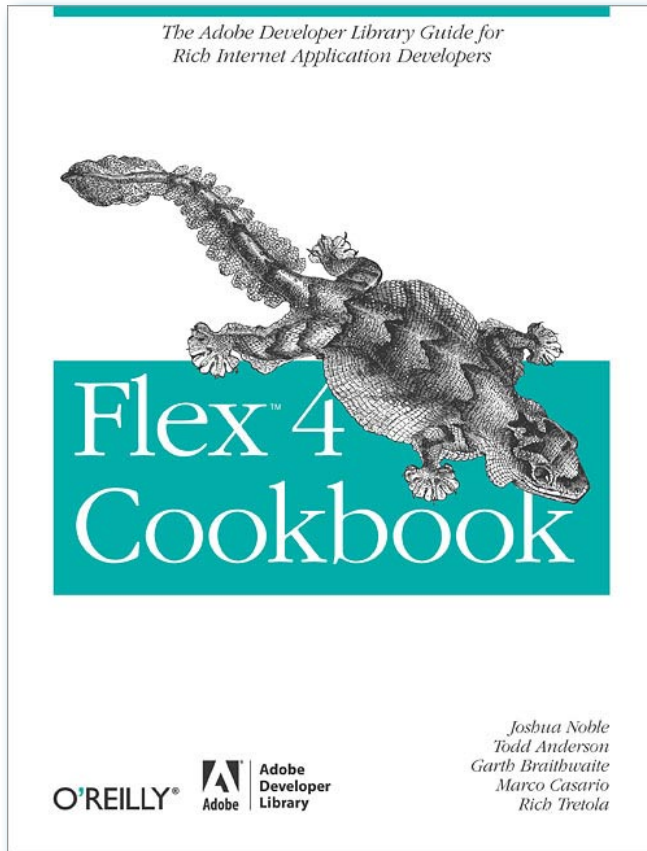
Dazzling Effects
in Flash CS4 and CS5



Read sample
chapters from
our book at
flashandmath.com

Flex 4 Cookbook

Real-world recipes for developing Rich Internet Applications



Authors: Joshua Noble, Todd Anderson, Garth Braithwaite, Marco Casario, Rich Tretola

Publisher: O'Reilly Media / Adobe Dev Library

ISBN: 978-0-596-80561-6

Pages: 768

Website: <http://oreilly.com/catalog/9780596805623>

O'Reilly cookbook series does not need any words of introduction. These series are satiated with tips, tricks and practical recipes for accomplishing different tasks. In case of Flex 4 cookbook, eminent standard and quality has been maintained for which the far-famed authors behind it are famous for. This book lists quick & complete recipes from basics to intermediate level which you can integrate in your projects or perhaps you can learn how different things are done in Flex 4.

I won't write much on it, because it's 24 chapters and table of contents are more than enough to make you fall in love for this book. There are not many books in print about Flex 4, so missing this book is not going to be a wise decision for your flex growth.

Highly recommend for beginners and intermediate level developers though advanced developers could benefit as well.

by Ali RAZA

Adobe Certified Instructor

Sun Certified Java Programmer

Zend Certified Engineer



ActionScriptJobs.com

KickASS Flex & Flash Jobs

NEED A JOB?

We work with companies to bring *KICKASS* Flash/Flex jobs to you through our Job Boards & Resume Tools.

NEED TO HIRE?

We can *HELP...* not only do we know how to recruit, we are Flash/Flex developers ourselves.

CREATED by ActionScript Developers ...

... FOR ActionScript Developers

www.ActionScriptJobs.com