

Tips & Techniques

Bringing development best practices to Flash, Flex, and Air

Learn up

- Always be learning - try something new on every project if you can.
- Expand beyond Flash/Flex/Air - even if it's just learning "Hello World" in a new environment
 - Learn a database system or two (MySql, MS SQL, etc)
 - Web platform (ColdFusion, PHP, Ruby on Rails, etc)
 - General purpose programming language (Java, any of the .NET languages, etc)
- Most SDKs from Sun, Microsoft, and others are free...
- The more varried environments you know, the more you can start to think in meta-concepts and patterns.

Plan your attack

- Be clear on requirements - make a list of features, review it with the client.
- Have a high level architecture in mind before you code. Step away from your development tools.
- Often missed aspects
 - Are there logging/tracking requirements?
 - What are the important performance characteristics?
 - If a service/resource fails, what's the fall back?
 - What about security (encryption, privacy, etc)
- Think about what technologies/techniques are best ahead of time - always be researching alternatives.

Stay organized

- Naming - standardize, case, numbering, separators
- Standard project structure (adapt to suit)
 - docs - specs, diagrams, business rules, etc.
 - src - your main source code (FLA, AS, MXML, etc)
 - lib - any 3rd party libraries needed for this project
 - assets-src - original design files
 - gen-src - any code generated by tools
 - test - your test code
 - test-data - any data needed for tests to run
 - assets - final assets for embed or at runtime
 - build - publish/compile/finalize to here
 - dist - zip everything up/package to here

Think ahead - configuration

- Plan for change
 - Text WILL change. Assets WILL change.
 - Links WILL change. Environments WILL change.
- Things that should almost always be configurable
 - All URLs - links out, service endpoints, asset paths
 - Most numbers your code uses - (max this, min that...)
- Seriously think about these
 - On/Off switches for optional logic like filtering of results, logging, tracking, showing of intro, etc.
 - Other strings in your piece - labels, messages, etc.
Makes internationalization easier down the road too.

OK, you can write some code now...

- Stay consistent with your code formatting, have a standard within a team.
- Name variables based on what they are, and include units in the name when applicable.
 - Bad: length, Good: lengthInInches
 - Bad: max, Good: maxRetryCount
 - Don't use abbreviations often (some are ok - min, max)
- Use packages to organize classes according to a system.
- Don't forget to document as you go.

Testing, testing, 123

- Once you've written some code, write a test harness to run it. (Or, if you're adventurous, write the test first!)
- Use a testing framework, or roll your own simple one for the project.
- Write tests for important functionality - commonly called functions, key classes, etc.
- Run the full test suite often.
- Once you get comfortable with testing, you'll have more confidence in your code, and you can change things without worry.

Learn the art of debugging

- Use trace or logging to output variables/events during runtime execution.
 - Works well initially, but as things get more complex, this becomes less scaleable.
- Use the debugger in your environment
 - Flash has a simple debugger
 - FlexBuilder has a very powerful, easier to use one
 - Take the time to learn these tools - it will be time well spent. A day or two spent learning your debugger will pay off forever.

Always be refactoring

- Refactor when code is becoming too complex.
- Classes should be focused on one or two things.
- If methods get really long, consider breaking them up into more methods that do focused parts of the original.
- If you can't test a certain part of your system, it's a good sign you need to refactor it into more classes.

Use a version control system

- In Flash
 - Use a VCS built into your OS's file browser
 - Tigris.org has TortoiseSVN & CVS
 - Use a standalone client
 - oxygen has a great one that works on all platforms
- In Flex
 - There are Eclipse plug-ins for both CVS and SVN - both are stable at this point.
- If possible, locate your repository remotely, that way you also have the added benefit of off site backup.

Backup, backup, backup

- It's not fun when you loose days/weeks worth of work, and there are many cheap/free ways to backup...
 - Save your work to a flash drive, external drive, or burn it to DVD.
- Take the time to setup automated backups
 - Use a syncing program rsync, Easy2Sync or similiar to sync your files to another computer/FTP site
 - Use an online backup service - Mozy works well, many others too.
 - On OSX, setup Time Machine

Make robots do the work...

- Automate as much as possible
 - Project build
 - Testing (Unit, Functional, Load)
 - Deployment (to each environment)
 - Monitoring
- Use templates, macros, key commands, etc. These all add up to huge productivity gains.
- If you're dealing with lots of data objects, look into code generation and/or model driven development - can be a huge time saver.
- Automate conversion/slicing of source assets

Closing / Most important points

- Always be learning
- Plan/Think ahead
- Do some form of testing
- Master debugging
- Make robots do the work

Thanks!

Who wrote "the book" on refactoring?

Who wrote "the book" on refactoring?

Martin Fowler

What is a code smell?

What is a code smell?

Wikipedia says: In the community of computer programming, code smell is any symptom that indicates something may be wrong. It generally indicates that the code should be refactored or the overall design should be reexamined.