What is the best approach to both modularity and platform independence? [closed]

Asked 16 years, 3 months ago Modified 2 years, 2 months ago

Viewed 567 times

1

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 2 years ago.

Improve this question

I hope this question does not come off as broad as it may seem at first. I am designing a software application that I would like to be both cross-platform and modular. I am still in the planning phase and can pick practically any language and toolset.

This makes things harder, not easier, because there are seemingly so many ways of accomplishing both of the goals (modularity, platform agnosticism).

My basic premise is that security, data storage, interaction with the operating system, and configuration should all be handled by a "container" application - but most of the other functionality will be supplied through plug-in modules. If I had to describe it at a high level (without completely giving away my idea), it would be a single application that can do many different jobs, all dedicated to the same goal (there are lots of disparate things to do, but all the data has to interact and be highly available).

I find myself wrestling with not so much how to do it (I can think of lots of ways), but which method is best.

For example, I know that Eclipse practically embodies what I am describing, but I find Java applications in general (and Eclipse is no exception) to be too large and slow for what I need. Ditto desktop apps written Python and Ruby (which are excellent languages!)

I don't mind recompiling the code base for different platforms as native exectables. Yet, C and C++ have their own set of issues.

As a C# developer, I have a preference for managed code, but I am not at all sold on Mono, yet (I could be convinced).

Does anyone have any ideas/experiences/ specific favorite frameworks to share?

compiled interpreted-language

Share

Improve this question

Follow

edited Oct 7, 2022 at 0:50

starball

48.3k • 28 • 187 • 847

asked Aug 28, 2008 at 22:03

Wing

6,108 • 3 • 22 • 12

6 Answers

Sorted by: Highest score (default)

2

Just to cite an example: for .NET apps there are the CAB (Composite Application Block) and the Composite Application Guidance for WPF. Both are mainly implementations of a set of several design patterns focused on modularity and loose coupling between components similar to a plug-in architecture: you have an IOC framework, MVC base classes, a loosely coupled event broker, dynamic loading of modules and other stuff.

So I suppose that kind of pattern infrastructure is what you are trying to find, just not specifically for .NET. But if you see the CAB as a set of pattern implementations, you can see that almost every language and platform has some form of already built-in or third party frameworks for individual patterns.

So my take would be:

1. Study (if you are not familiar with) some of those design patterns. You could take as an example the CAB framework for WPF documentation: Patterns in the Composite Application Library
2. Design your architecture thinking on which of those patterns you think would be useful for what you want to achieve **first without thinking in specific pattern implementations or products**.
3. Once you have your 'architectural requirements' defined more specifically, look for individual frameworks that help accomplish each one of those patterns/features for the language you decide to use and put together your own application framework based on them.

I agree that the hard part is to make all this platform independent. I really cannot think on any other solution to choose a mature platform independent language like Java.

Share Improve this answer

Follow

answered Aug 28, 2008 at 22:45

Sergio Acosta

11.4k ● 12 ● 63 ● 91

1

Are you planning a desktop or web application?

Everyone around here seems to think that Mono is great, but I still do not think it is ready for industry use, I would equate mono to where wine is, great idea; when it works it works well, and when it doesn't...well your out of luck.

mod_mono for Apache is extremely glitchy and is hard to get running correctly.

If your aiming for the desktop, nothing beats the eclipse RCP (Rich Client Platform) framework: http://wiki.eclipse.org/index.php/Rich_Client_Platform.

You can build window, linux, mac all under the same code and all UI components are native to the OS. And RCP wins in modularity hands down, it has a plug-in architecture that is unrivaled (from what I have seen)

I have worked with RCP for 1.5 years now and I dunno what else could replace it, it is #1 in it's niche.

If your totally opposed to java I would look into wxWidgets with either python or C++

Share Improve this answer Follow

edited Aug 28, 2008 at 22:40

answered Aug 28, 2008 at 22:34

mmattax
27.6k ● 42 ● 117 ● 151

1

If you want platform independence, then you'll have to trade off between performance and development effort. C++ may be faster than Java (this is debatable FWIW) but you'll get platform independence a lot more easily with Java. Python and Ruby are in the same boat.

I doubt that .NET would be much faster than Java (they're both VM languages after all), but the big problem with .NET is platform independence. Mono has a noble goal and surprisingly good results so far but it will *always* be playing catch-up with Microsoft on Windows. You might be able to accept its limitations but it's still not the same as having identical multiplatform environments that Java, Python, and Ruby have. Also: the .NET development and support tools are heavily skewed towards Windows, and probably always will be.

IMO, your best bet is to target Java... or, at the very least, the JVM. If you don't like the Java language (and as a C# dev I'm guessing that's not the case) then you at least have options like Jython, JRuby, and Scala. With the JVM, you get very good platform independence, good performance, and access to a huge number of libraries and support tools. There's almost always a Java library, port or implementation that will do what you need it to do. I don't think any other platform out there has the same number of options; there's real value in that flexibility.

As for modularity: that's more about how you build the software than what platform you use. I don't know much about plugin architectures like you describe but I'm guessing that it will be possible in pretty much any modern platform you pick.

answered Aug 28, 2008 at 22:51

Craig Walker

51.6k ● 57 ● 155 ● 212

If you plan on doing python development, you can always use pyrex to optimize some of the slower parts.

1

Share Improve this answer

Follow

answered Aug 28, 2008 at 22:55

Jason Baker

198k ● 138 ● 382 ● 520

With my limited Mono experience I can say I'm quite sold on it. The fact that there is active development and a lot of ongoing effort to bring it up to spec with the latest .Net technologies is encouraging. It is incredibly useful to be able to use existing .Net skills on multiple platforms. I had similar issues with performance when attempting to accomplish some basic tasks in Python + PyGTK -- maybe they can be made to perform in the right hands but it is nice to not have to worry about performance 90% of the time.

0

Share Improve this answer

Follow

answered Aug 28, 2008 at 22:08

Luke

19k ● 24 ● 88 ● 112

For desktop applications, writing it in an interpreted language, and using a cross-platform UI toolkit like wxWidgets will get you a long way towards platform independence (you just have to be careful not to use any

0

other modules that aren't cross-platform, use things like Python's `os.path` module, in place of doing things like `config_path = "/home/$USER"`)

That said, to make a *good* cross-platform application, you will have to do some things differently on each platform..

For example, OS X is probably the most different - preferences are usually stored in ~/Library/Prefernces/ as .plists, UI's are generally based around floating windows, with a single menu-bar docked at the top-of-screen.

I suppose this is where the modularity comes into play.. With the preferences example above, you could have a class `UserConfig`, of which you have OS-specific versions of. The Windows one stores config data in the appropriate `Application Data` folder, or the registry. The Mac OS one uses .plist files on `~/Library/Preferences/`, and the unix'y one uses ~/.dotfiles.

Share Improve this answer Follow

edited Oct 10, 2008 at 7:51

answered Oct 10, 2008 at 7:40

dbr

169k ● 69 ● 283 ● 347