Framework/Language for new web 2.0 sites (2008 and 2009)

Asked 16 years, 2 months ago Modified 15 years, 9 months ago

Viewed 1k times

4

I know I'll get a thousand "Depends on what you're trying to do" answers, but seriously, there really is no solid information about this online yet. Here are my assumptions - I think they're similar for alot of people right now:

1. It is now October 2008. I want to start writing an application for January 2009. I am willing to use beta code and such but by January, I'd like a site that doesn't have 'strange' problems. With that said, if a language is simply 10% slower than another, I don't care about those things as long as the issue is linear. My main concern is developer productivity.
2. I'll be using Linux, Apache, MySQL for the application.
3. I want the power to do things like run scp and ftp client functions with stable libraries (I only picked those two because they're not web-related but at the same time represent pretty common network protocols that any larger app might use). Technologies like OpenID and Oauth will be used as well.

4. Experienced web developers are readily available (i.e. I don't have to find people from financial companies and such).
5. Whatever the choice is is common and will be around for a while.
6. Here's a kicker. I'd like to be able to use advanced presentation layer tools/languages similar to HAML, SASS. I definitively want to use JQuery.
7. I will be creating a Facebook app and at some point doing things like dealing with SMS messages, iPhone apps, etc...

At this point, the choices for language are PHP (Cake,Symfony,Zend), Python (Django), Ruby (Merb). I'm really between Django and Merb at this point mostly because everybody else seems to be going that way.

Please don't put any technologies in here that aren't made for mainstream. I know Merb is untested mostly, but their stated goal is a solid platform and it has alot of momentum behind it so I'm confident that it's workable. Please don't answer with how great Perl is or .Net.

For Future References - these choices were already made:

- Debian (Lenny) - For converting CPU cycles into something useful. Trac
- 0.11 - For Project Management Gliffy - For wireframes and such

- Google Docs/Apps - For documentation, hosted email, etc...
- Amazon ec2/S3 - For hosting, storage.

Cheers, Adam

python ruby-on-rails django merb

Share

Improve this question

Follow

edited Mar 5, 2009 at 21:23

Mez
24.9k ● 14 ● 74 ● 93

asked Oct 8, 2008 at 18:07

Adam Nelson
8,090 ● 11 ● 47 ● 65

your 'previously made choices' are all formatted wrong, could you please edit it? thanks – davr Oct 8, 2008 at 19:01

13 Answers

Sorted by: Highest score (default)

18

Django!

Look up the DjangoCon talks on Google/Youtube - Especially "Reusable Apps" (www.youtube.com/watch?v=A-S0tqpPga4)

I've been using Django for some time, after starting with Ruby/Rails. I found the Django Community easier to get into (nicer), the language documented with *excellent* examples, and it's modularity is awesome, especially if you're wanting to throw custom components into the mix, and not be forced to use certain things here and there.

I'm sure there are probably ways to be just as flexible with Rails or some such, but I highly encourage you to take a long look at the Django introductions, etc, at http://www.djangoproject.com/

Eugene mentioned it's now at 1.0 - and therefore will remain a stable and backward-compatible codebase well through January 2009.

Also, the automatic admin interfaces it builds are *production ready*, and extremely flexible.

Share Improve this answer Follow

answered Oct 8, 2008 at 18:51

anonymous coward

12.8k ● 13 ● 58 ● 98

This was a great talk. It introduced me to two technologies: djangoplugables.com - A set of apps for Django pinaxproject.com - A ready-to-go app structure for web 2.0 sites using Django. I think I may start with from here.
– Adam Nelson Oct 8, 2008 at 21:16

I'm very glad I was able to help, no matter what direction you ultimately go. There was also a talk about Pinax itself, if you haven't already seen it. It may be a bit drawn out at points, but it goes into detail on some of the specific things included.
– anonymous coward Oct 9, 2008 at 13:52

9

Sorry, but your question is wrong. People are probably going to vote me down for this one but I want to say it anyway:

I wouldn't expect to get an objective answer! Why? That's simple:

- All Ruby advocates will tell to use Ruby.
- All Python advocates will tell to use Python.
- All PHP advocates will tell to use PHP.
- Insert additional languages here.

Got the idea?

I recommend you to try each of the languages you mentioned for yourself. At least a few days each. Afterwards you should have a much better foundation to make your final decision.

That said, I would choose Ruby (because I am a Ruby advocate).

Share Improve this answer Follow

answered Oct 8, 2008 at 18:51

Christoph Schiessl
6,858 ● 4 ● 35 ● 45

This is the answer I expected – Adam Nelson Oct 8, 2008 at 20:05

All of them will get the job done.

7

Use the one that you and your team are most familiar with

This will have a far greater impact on the delivery times and stability of your app than any of the other variables.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot
1 ● 1

answered Oct 9, 2008 at 20:02

Orion Edwards
123k ● 66 ● 245 ● 339

5

✓

it depends.

php - symfony is a great framework. downsides: php, wordy and directory heavy. propel gets annoying to use. upsides: php is everywhere and labor is cheap. well done framework, and good support. lots of plugins to make your life easier

python - django is also a great framework. downsides: python programmers can be harder to find, django even harder. changing your db schema can be somewhat difficult since there are no official migrations. doesn't quite do mvc like you'd expect. upsides: does everything you need and has the great python std library and community behind it.

ruby - i've never used merb, so I'll address rails. upsides: there is a plugin, gem, or recipe for almost anything you could want to do. easy to use. downsides: those plugins, gems, and recipes sometimes fail to work in mysterious ways. monkey patching is often evil. the community is.. vocal. opinionated software, and sometimes those opinions are wrong (*lack of foreign keys*). rails itself seems like a tower of cards waiting to explode and take hours of your life away.

with all of that said, I'm a freelance php/symfony and ruby/rails developer. I've worked on several projects in both languages and frameworks. My latest project is in Rails solely because of ActiveMerchant. I've been looking for a reason to develop a django app for a while. If there were an ActiveMerchant like library for django, I probably would have used it.

Share Improve this answer Follow

answered Oct 8, 2008 at 19:08

jcoby
4,250 ● 2 ● 30 ● 25

4

I would go with Django, if you are comfortable with a Python solution. It's at version 1.0 now, and is maturing nicely, with a large user base and many contributors. Integrating jQuery is no problem, and I've done it without any issues.

The only thing is, as far as I can tell, Ruby is much more popular for web development nowadays, so it's easier to find Ruby developers. I get this impression from browsing recent job advertisements - there aren't that many for Python or Django. I don't know much about Merb, so I can't give a fair comparison.

I've done enough PHP to not recommend starting a new project with it.

Share Improve this answer Follow

answered Oct 8, 2008 at 18:19

Eugene
825 ● 1 ● 9 ● 14

2

Based in your reasons, I would go with Ruby. I see that you want some administration tools (scp, ftp client) and Ruby has it (net/sftp and net/ftp libraries).

Also, there are great gems like God for monitoring your system, Vlad the Deployer for deploying, etc. And a lot of alternatives in Merb's field, just use whatever you find it's better for your needs (Thin, Mongrel, ebb, etc).

Share Improve this answer Follow

answered Oct 8, 2008 at 18:26

Dario
951 ● 9 ● 18

2

To get a feeling of where the Django ecosystem is at currently, you might want to check out

- djangopeople.net (try djangopeople.net/us/ny for New York state)
- djangogigs.com

Share Improve this answer Follow

answered Oct 9, 2008 at 10:55

akaihola
26.8k ● 7 ● 63 ● 74

1

I have to preface this with my agreeing with Orion Edwards, choose the one your team is most familiar with.

However, I also have to note the curious lack of ASP.NET languages in your list. Not to provoke the great zealot army, but where's the beef? .NET is a stable, rapid development platform and the labor pool is growing daily. VB.NET and C# are transportable skill sets, and that can mean a lot when you're building a team of developers to work on a diverse set of tasks. .NET also allows you to separate your presentation layer from your backend code, like other languages, but also allows you to expose that backend code as web service for things like your iPhone and Facebook applications.

Take every suggestion with a grain of salt, and pick what suits the application best. Do your research, and design for function and not the zealots.

Disclaimer: Once a PHP, ColdFusion and Perl developer. Flex zealot, and Adobe lover. Now writing enterprise .NET applications. ;)

Don't forget Mono, which will let you run .NET under *nix. Not that I'm saying it will be perfect, just playing devil's advocate.

Share Improve this answer Follow

answered Oct 9, 2008 at 20:13

willasaywhat
2,372 ● 20 ● 23

Stack Overflow itself is written on Microsoft tech - so it must be pretty good. Nonetheless, I put .NET in the same boat as Java - great if you have a million dollars for a formal development process or you previously worked in a larger dev team and you already know the tech (6 FT devs is $1M/yr) – Adam Nelson Mar 11, 2009 at 14:12

1

Don't get stuck in the mindset of server-side page layout. Consider technologies like SproutCore, GWT or ExtJS which put the layouting code fully on the client, making the server responsible only for data marshalling and processing (and easily replaced).

And you really, really need to know which server platform you want. Don't pick one because it's the flavor of the month, pick one because you're comfortable with it. Flavors don't last, a solidly built codebase will.

Share Improve this answer Follow

answered Oct 9, 2008 at 21:13

Joeri Sebrechts
11.1k ● 3 ● 38 ● 50

1

Having built apps in Django, I can attest to its utility. If only all frameworks were as elegant (yes Spring, I'm looking at you).

However in terms of betting the farm on Django, one thing you need to factor in is that Python 3 will be released shortly. Python 3 is not backwards compatible and there's a risk that it will fork the language and end up slowing momentum for all Python projects while they deal with the fallout. To be fair, Ruby 2.0 is due soon too, but I don't think it will be as disruptive.

Share Improve this answer Follow

answered Oct 16, 2008 at 15:07

Rob
5,612 ● 11 ● 43 ● 45

0

My experience with various new technologies over the last ten years leads me to recommend that you make stability of the platform a serious criterion. It's all well and good developing with the latest and greatest framework, but when you find it's moved forward a point version and suddenly the way you have done everything is deprecated, that can turn out to result in extra unnecessary work. This was particularly my experience working with rails a little ahead of version 1. For that reason alone I would avoid any platform that wasn't at least at 1.0 when you start work on it.

Ruby is great to work with and will keep your developer productivity high, but if Django is the more stable platform

I would favour that for sure.

Share Improve this answer Follow

answered Oct 9, 2008 at 11:05

glenatron
11.3k ● 13 ● 66 ● 120

0

It pays not to be biased about your server setup. Any modern web framework worth it's weight in source code has a SQL abstraction layer of some sort. PostgreSQL gets much better performance, and this is coming from a former MySQL partisan.

Apache is a beast, both to configure and on your server's resources. Why not go with something light-weight, like nginx or lighttpd?

(For the record, I'm a big Django user, but as the accepted answer said, go with whatever your team knows. Quick turn-arounds are not the time to be learning new frameworks. If you're hiring a team from scratch, go with Django.)

Share Improve this answer Follow

answered Oct 14, 2008 at 16:48

Ellen Teapot
1,610 ● 1 ● 15 ● 22

0

Update: I ended up using, and loving, Django. I'm totally done with PHP - sorry about that. Future readers trying to create a new web 2.0 site (assuming they have a programming background), should greatly consider this setup:

Amazon ec2 for hosting ($80/month - not cheap but worth it if you can afford it) Django/Python (Python is the most powerful scripting language on the planet - and Django just makes it work on the web)

Development should be done with SQLlite and the development server that comes with Django. Don't waste time with Nginx, Apache, MySQL until you're withing a few weeks of a beta.

Oh, and I now develop on a Mac, which works great for local Django development.

Finally, [Pinax]() is a great start for [Django]() development.

Share Improve this answer

Follow

answered Mar 5, 2009 at 21:09

Adam Nelson

8,090 ● 11 ● 47 ● 65

Happy to hear you've chosen Django. A note about DBMS though; I was thinking the same thing until it bit me. There are different behaviours for different DBMS's. I'd start with MySQL day one. – muhuk Mar 5, 2009 at 21:38