

What is the best way to migrate an existing messy webapp to elegant MVC? [closed]

Asked 16 years, 3 months ago Modified 15 years, 11 months ago


Viewed 2k times



5



Closed. This question needs to be more [focused](#). It is not currently accepting answers.

 **Want to improve this question?** Update the question so it focuses on one problem only by [editing this post](#).

Closed 6 years ago.

[Improve this question](#)

I joined a new company about a month ago. The company is rather small in size and has pretty strong "start-up" feel to it. I'm working as a Java developer on a team of 3 others. The company primarily sells a service to for businesses/business-type people to use in communicating with each other.

One of the main things I have been, and will be working on, is the main website for the company - from which the service is sold, existing users login to check their service and pay their bills, new users can sign up for a trial, etc.

Currently this is a JSP application deployed on Tomcat, with access to a database done thru a persistence layer written by the company itself.

A repeated and growing frustration I am having here (and I'm pretty happy with the job overall, so this isn't an "oh no I don't like my job"-type post) is the lack of any larger design or architecture for this web application. The app is made up of several dozen JSP pages, with almost no logic existing in Servlets or Beans or any other sort of framework. Many of the JSP pages are thousands of lines of code, they `jsp:include` other JSP pages, business logic is mixed in with the HTML, frequently used snippets of code (such as obtaining a web service connection) is cut and paste rather than reused, etc. In other words, the application is a mess.

There have been some rumblings within the company of trying to re-architect this site so that it fits MVC better; I think that the developers and higher-ups are beginning to realize that this current pattern of spaghetti code isn't sustainable or very easily scalable to add more features for the users. The higher-ups and developers are wary of completely re-writing the thing (with good reason, since this would mean several weeks or months of work re-writing existing functionality), but we've had some discussions of (slowly) re-writing certain areas of the site into a new framework.

What are some of the best strategies to enable moving the application and codebase into this direction? How can

I as a developer really help move this along, and quickly, without seeming like the jerk-y new guy who comes into a job and tells everyone that what they've written is crap? Are there any proven strategies or experiences that you've used in your own job experience when you've encountered this sort of thing?

java

model-view-controller

jsp

architecture

Share

Improve this question

Follow

edited Sep 2, 2008 at 18:40



travis

36.4k ● 21 ● 72 ● 97

asked Sep 2, 2008 at 18:39



matt b

140k ● 66 ● 284 ● 350

8 Answers

Sorted by:

Highest score (default)





12



Your best bet is probably to refactor it slowly as you go along. Few of us have the resources that would be required to completely start from scratch with something that has so many business rules buried in it. Management really hates it when you spend months on developing an app that has more bugs than the one you replaced.

If you have the opportunity to build any separate apps from scratch, use all of the best practices there and use it to demonstrate how effective they are. When you can, incorporate those ideas gradually into the old application.

Share Improve this answer

edited Jan 6, 2009 at 17:48

Follow

answered Sep 2, 2008 at 18:43



[Eric Z Beard](#)

38.4k ● 27 ● 101 ● 147



4



First pick up a copy of Michael Feather's Working [Effectively with Legacy Code](#). Then identify how best to test the existing code. The worst case is that you are stuck with just some high level regression tests (or nothing at all) and If you are lucky there will be unit tests. Then it is a case of slow steady refactoring hopefully while adding new business functionality at the same time.

Share Improve this answer

answered Sep 2, 2008 at 21:48

Follow



[John Channing](#)

6,591 ● 7 ● 47 ● 57



2



In my experience, the "elegance" of an app typically has more to do with the database design than anything. If you have a *great* database design, including a well-defined stored procedure interface, good application code tends to follow no matter what platform you use. If you have *poor* database design, no matter what platform you use you'll have a very difficult time building elegant application code, as you'll be constantly compensating for the database.

Of course, there's plenty of room between *great* and *poor*, but my point is that if you want good application code, start by making sure your database is up to snuff.

Share Improve this answer

answered Sep 2, 2008 at 18:46

Follow



Joel Coehoorn

415k ● 114 ● 577 ● 813



2



This is harder to do in applications that are only in maintenance mode because it is hard to convince management that rewriting something that is already 'working' is worth doing. I would start by applying the principles of MVC to any new code that you are able to work on (ie. move business logic to something akin to a model, put all your layout/view code in one place)

As you gain experience with new code in MVC you can start seeing opportunities to change the existing code subtly so that it also falls in line. It can be a very slow

process, but if you can show the benefits of this way of doing it then you will be able to convince others and get the entire team on board.

Share Improve this answer

answered Sep 2, 2008 at 18:48

Follow



[Dan Cramer](#)

685 ● 1 ● 9 ● 14



2



I would agree with the slow refactoring approach; for example, take that copy-and-pasted code and extract it into whatever the appropriate Java paradigm is (a class, perhaps? or better yet, use an existing library?). When your code is really clean and terse, yet still lacking in overall architectural strategy, *then* you'll be able to make things fit into an overall architecture much more easily.

Share Improve this answer

answered Sep 2, 2008 at 18:50

Follow



[Domenic](#)

113k ● 42 ● 226 ● 273



1



Iteratively refactor. Also look for new features that can be done completely in the new framework as a way to show the value of the new framework.

Share Improve this answer

answered Sep 15, 2008 at 18:52

Follow



[James A. N. Stauffer](#)

2,669 ● 3 ● 22 ● 32



1



My suggestion would be to find the rare pages that do not require an import of other JSPs, or have very few imports. Treat each JSP imported as a black box, and refactor these pages around them (iteratively, testing each change and making sure it works before continuing). Once these are cleaned up, you can continue finding pages with more and more imports, until finally you have refactored the imports.

When refactoring, note the parts that try to access resources not given to the page and try to take this out to a controller. For instance, anything that accesses the database should be inside a controller, let the JSP handle the display of the information the controller gives to it via a forward. In this way you will develop several servlets, or things like servlets, for each page. I would suggest using a front-controller based framework for this refactoring (from personal experience I recommend Spring and its Controller interface), so that each controller isn't a separate Servlet but is rather delegated to from a single servlet that is mapped appropriately.

For the controller, it is better to do database hits all at once rather than attempt them piecemeal. Users can and generally do tolerate a page load, but the page output will be much faster if all the database data is given to the rendering code rather than the rendering code hanging and not giving data to the client while it is trying to read yet another piece of data from the database.

I feel your pain and wish you luck in this endeavor. Now, when you have to maintain an application that abuses Spring Webflow, that's another story :)

Share Improve this answer

answered Sep 16, 2008 at 3:16

Follow



MetroidFan2002

29.9k ● 16 ● 65 ● 81



0



The best way is to print out the code, crumple it up, and throw it out. Don't even recycle the paper.

You've got an application that is written in 1,000+ line long JSPs. It probably has a god-awful Domain Model (if it even has one at all) and doesn't just MIX presentation with business logic, it BLENDS it and sits there and keeps stirring for hours. There is no way to take out the code that is crappy and move into an MVC Controller class and still be doing the right thing, you'll just end up with a MVC app with an anemic domain model or one that has stuff like Database calls in Controller code, you're still failing.

You could try a new app that does the right thing and then have the two apps talk to each other, but that's new complexity in itself. Also you're likely going to be doing the same amount of work that you were going to do if you just started from scratch, but you might have an easier time trying to convince your bosses that this is a better approach.

Share Improve this answer

answered Sep 2, 2008 at 19:35

Follow



[bpapa](#)

21.5k ● 25 ● 100 ● 147
