Coding For Posterity - why quality matters

Posted At : June 17, 2005 12:46 PM | Posted By : Mark Kruger
Related Categories: Content Management, Follies and Foibles

I realize that you have a great coding style. I know you are sold on your personal framework, language and platform. I realize that when you look at your code you say to yourself "dang I'm good!". But I'm pleading with you, when you write an application, try to take pains to consider other developers. Here's 3 principles that you should keep in mind:

1. You will never anticipate all the eventual needs of the client - don't gamble on it.
2. You will not be the only one to work on this code. Other developers *will* see this code. You can *count on it!*
3. Your client trusts you - do what's right.

1. Anticipating Every Need

There are a lot of applications that try to anticipate every need a user might have. Take content management for example, we've worked with a few such systems and we've looked at about 10 different packages in the last few months. All of them have a great set of features. They all handle editing, approving, page layout and navigation. They have complex management tools that include cool things that I've never even considered. But none of them fully meets the needs of our client (a university). The result is that the question "how easy is it to modify?" comes bubbling to the top of the heap - and becomes the deciding factor in our decision.

Ecommerce is another example. We have worked with a large and expensive Coldfusion storefront with a huge set of features. It's clear from looking at the features that a team of developers has tried their best to meet every conceivable need. Why are we working on it then? Simple, there where things the client wanted to do with the web site that weren't possible with the existing code base. This particular package is written in such a way as to make meeting those needs a Herculean effort.

Am I saying you shouldn't try to meet every need? No - I'm saying to *write modifiable code*. Write code that simplifies the task of modification. Write modular, reusable and well documented code that invites tinkering. Remember, if you have a good relationship with your client there's a good chance that *you* will be the one tinkering. If you don't write code that is easy to work with, you will really hate it when you have to change it. As a side note regarding e-commerce, **CF Webstore** is a great example of easy to modify code.

2. You WILL be critiqued

It's an inescapable fact of life. Code lasts forever (except your current project which can be blown away by a crashed hardrive at any moment). Someone somewhere will see what you have done and pass judgment on it. How you code will invite blessing or cursing upon your head. Years ago when I left my first CF job I left behind my "learning applications". Mostly these were intranet widgets for doing small data tasks. I was surprised to learn recently that much of that code is still up and running - although it is 7 or 8 years old by now. Developers tasked with updating or maintaining it probably mutter unspeakable things about me on a regular basis. My suggestion is, make a pact with yourself to write code that you will be proud of someday - or at least you won't run away when someone mentions it.

3. Love Your Customer Like You Love Yourself

One of the stated goals of CF Webtools is to "...help the customer leverage technology in a *cost effective* manner." I've worked with companies who enter meetings with a vision for how they can extract the maximum amount of cash from a client with a minimum amount of effort. Although they listen carefully, they are really just listening for opportunities to pitch. My advice is *don't ever "pitch" your client*. There may be times when they need your advice and you have the opportunity to help make a decision that may benefit you as a company or developer, but it's not the best for them. In those cases *always choose what is best for the client*. Not only is it the right thing to do, but it's one of the reasons clients will come back to you again and again.

That applies to your code as well. Don't take the easy road. Don't slap things together and just "make it work". Let yourself be guided by the idea that this code must be maintained for *N* number of years. Write it so that someone else with a similar skill set can make use of it later. Write it so that if you died tomorrow they could hire a new developer and pick up where you left off with a minimum of effort. Don't force clients to sign draconian license agreements that leave a sour taste in their mouth. Be reasonable. Be fair. Be honest and incorruptible. I can promise you this - IF you manage to solve a pressing need, save them money and grow their business, you have just made a long term customer. That 25 grand you made on the application will pale in comparison to the work that will come your way. I love Google's company motto - "don't be evil". That makes a good rule for developers - and for life.