My natural development pattern - software layer-ification

Posted At : 6 January 2012 09:33 | Posted By : Shaun McCran
Related Categories: Software Architecture

Whilst reflecting on some of the projects I'd been through in 2011 I noticed a pattern that became more and more prominent as the year wore on.

All of my project functionality was naturally splitting itself into independent service layers.

All the projects I've architected for the last few years have followed one of several design patterns, whether they are project management or development patterns they have all followed similar principles. Similarly I am intimately familiar with software platforms being constructed of multiple service layers, such as the GUI layer, data access layer, security layers etc.

Looking back at the architecture of the last twelve months of projects they are all layered into logic layers of functionality. More and more the functional layers of application are splitting themselves out into distinct divisions.

Less ColdFusion, more JQuery

With my projects there has been a trend for less and less server side technology. With the power and flexibility of client side languages like JQuery, and the availability of staff who are now experienced enough in them for projects to be guaranteed on time and spec, a lot of the functionality is being moved away from the server side and onto the clients side.

This has had the double effect of creating a richer front end layer, and a more streamlined back end layer. The majority of back end layer is now in effect just a service layer handling non client side functionality. This is ideal as with flexible development languages like ColdFusion, ASP or

PHP it is very easy for developers to litter condition logic all over the front end display layer, making for an evil mix of spaghetti code. Anything that stops this is good.

Looking at the last dozen or so software platforms I've been involved with the server side language has only been employed to handle WebService requests, business object handling requests or specifically identified secure functions that cannot be passed to the client side.

Noticing this trend has been an interesting experience for me, one I didn't actually see happening at the time, but can retrospectively identify. I'm wondering now if it will continue through 2012, or if other developers have seen similar changes to their project patterns?