Subject: Re: (re-)introducing Dependency Injection
Posted by AJM on Sun, 07 Apr 2013 10:15:22 GMT
View Forum Message <> Reply to Message

It is quite clear that you do not understand what RADICORE is. It is *NOT* a collection of libraries from which you can pick and choose, it is a fully-fledged framework which you either use in its entirety or you do not use at all. RADICORE has a single menu system and a single security system which are intertwined. You can customise the way it looks by replacing the CSS files but you cannot change the way it works. You can swap the password authentication by using either a RADIUS or an LDAP server, but you cannot change how you allow/disallow a user's access to a specific task.

There is an in-built audit logging system which you can turn off should you choose, but you cannot swap it for another, nor can you extract it to use without the rest of RADICORE.

There is an in-built workflow system which you can turn on should you choose, but you cannot swap it for another, nor can you extract it to use without the rest of RADICORE.

There is an in-built data dictionary and collection of transaction patterns which you use to generate your application components, but you cannot swap it for another, nor can you extract it to use without the rest of RADICORE.

You do not use RADICORE on its own, you use it to build your own back-end application. There are several prototype applications included in the download which you can play with and examine. Your application will have numerous database tables to maintain and numerous user transactions which implement your business logic. You can build the basic CRUD screens for each of your database tables very quickly, then all you do is add the code to process your business rules.

Although RADICORE's presentation layer is not suitable for customer-facing front-end websites (it was designed specifically for members of staff who administer the site) it's implementation of the 3 Tier Architecture means that you can easily create an alternative presentation layer using whatever tools you like, but which shares the business and data access layers already within RADICORE.

I have used RADICORE to build an order processing application called TRANSIX which is currently used by businesses as varied as custom jewellery, baby clothes and photographic prints. They all share the same back-end (although they have their own private instances of the databases) but they each have their own unique front-end websites. Because they require nothing more than a different presentation layer they are quick and easy to build.

When you build your own application using RADICORE you have instant access to all the features within RADICORE, and you are free to write your own business logic and use whatever external libraries that take your fancy.

I don't use Dependency Injection within RADICORE because it has enormous costs and zero benefits. It has benefits in only a limited set of circumstances, and those circumstances do not exist within RADICORE. I will *NEVER* refactor RADICORE to use DI, so telling me that, in your opinion, DI is a good idea is just a waste of time.