

Re: Delegates, Generic Methods and algorithms – Functional versus Procedural

provide the delegate? Can a delegate be polymorphic as required for them to be comparable? Here, the data and action are combined and separated from the iteration.

Since generics are fairly new to C# and less powerful than C++ templates, I think it is a good time to ponder whether moving toward functional programming is advisable for C#, etc.

Thanks for your opinions,
Travis

I think most people will agree that, in general, the "readability" of code depends largely on the target audience. The same is true for any writing, not just programming. There are books for children, books for teens, books for sci-fi fans, books for computer junkies, etc. A child who is just starting to read probably isn't going to understand Stephen King. However, trying to re-write a Stephen King novel using a child's vocabulary would not only make the book exceedingly long, it would likely lose much of what makes it a great novel. As another example, I *tried* to read a philosophy book in college and failed miserably. I had to look up several words on each page, and even then I generally had no idea what the author was talking about.

I think programming is similar. Once you are familiar with the concepts (i.e. vocabulary) then suddenly the reading becomes much easier and more intuitive. If you have to look everything up, you'll have a much harder time figuring it out.

The problem, then, is when programmers have very different vocabularies. I am an "OPF oriented" programmer, and I have an incredibly difficult time working with "data set oriented" programmers. I don't use DataSets – ever. So I am not familiar with how they work and don't really care to be. I can muddle one together by reading the documentation if I have to, but I would never design an application that way. OTOH, a "data set oriented" programmer is completely lost when I start discussing the concept of an OPF. They will claim to know "OOP", but all that really means is that they know how to *use* pre-fab objects handed to them by an SDK. They have no idea how to design and implement a object framework of their own. Most programmers that I have met seem to find a "vocabulary" that they are comfortable with and stick to it. I must admit that I fall into this category. I think that there are very few who "read the dictionary" on a daily basis.

So I believe that procedural code (e.g. the "for loop") is the "lowest common denominator". It is more "readable" to a wider audience – kind of like a children's book. But it's difficult (but not impossible) and more time-consuming to create a meaningful application that way. A richer vocabulary leads to shorter and more interesting reading. But you need to be aware of your target audience. If you go off writing code using constructs that are unknown to the rest of your team then you are inviting trouble. Believe me, I've done it. And over the years I have spent as much time teaching others about OPFs as I have writing code that uses them. It almost makes me want to give up and "just load everything into a DataSet, it's so much easier".