POSTSHARP

CASE STUDY:
Siemens Audiology

"Within the code of our presentation layer (mainly ViewModels and XAML code) we saved about 15% just by using the ViewModelAspect."

Bernd Hengelein
Software Architect
Siemens Audiology

Siemens Audiology

A leading supplier of hearing devices saves from having to write INotifyPropertyChanged and creating ICommand properties needed for data binding.

Summary

When the team at Siemens Audiology, a business unit of Siemens Sector Healthcare, started development on a new WPF implementation for two of its existing hearing system software applications it chose PostSharp to deliver a decrease in boilerplate code and software defects – saving them 15% within their presentation layer code.

"We chose PostSharp, rather than alternatives like transparent proxy solutions, because we wanted to have maximum performance during runtime."

Bernd Hengelein
Software Architect
Siemens Audiology

Team to deliver new WPF implementation for existing apps

Siemens Audiology, a business unit of Siemens Sector Healthcare, develops and distributes hearing devices to hearing care professionals worldwide.

The company's development team began looking into aspect-oriented programming and PostSharp as they started development on a new WPF implementation for two of its existing hearing system software applications.

"The WPF applications that we are developing are based on the MVVM (Model-View-ViewModel) pattern for its testability and powerful data binding capabilities," says Bernd Hengelein, the software architect behind Siemens Audiology's development team. "There are some repetitive tasks involved when writing ViewModels, such as implementing INotifyPropertyChanged for each property, and creating DelegateCommand for each method on the ViewModel that you want to bind to a UI element."

Why Siemens Audiology chose PostSharp

Recognizing the importance of removing boilerplate code, the team began developing PostSharp aspects to do all the repetitive work for them.

"For standard use cases, PostSharp is very easy to use" says Bernd, "for instance, implementing a simple tracing aspect is a piece of cake. We chose PostSharp, rather than alternatives like transparent proxy solutions, because we wanted to have maximum performance during runtime."

Build and inject custom aspects into existing applications

Using PostSharp, the Siemens Hearing Instruments team built a number of aspects to cover some interesting use cases including ViewModel, Threading, and Localization.

ViewModelAspect

The most complex of all the aspects used by the team, it saves writing time for developers implementing ViewModels and increases source code readability by removing boilerplate code. While the aspect was primarily built to handle INotifyPropertyChanged , ease refactoring and avoid typos – it also saves the team from having to implement all the delegate commands in their MVVM implementation.

"Within the code of our presentation layer (mainly ViewModels and XAML code) we saved about 15% just by using the ViewModelAspect."

Bernd Hengelein
Software Architect
Siemens Audiology

"The aspect is created automatically for each public void method, and creates a delegate command together with a new property of type ICommand that can be bound to the UI," says Bernd. "This makes it very easy for us when new ViewModels are developed. We just put our ViewModelAspect on it, write the public methods – which can be bound to the UI - and all the INotifyPropertyChanged and command stuff is done instantly."

Threading Aspect

The aspect is used by the team to execute long-running business operations within a separate thread and keeps them focused on implementing relevant business logic instead of threading issues.

"We have two solutions," says Bernd, "in the first we add an attribute to a method and it's executed in a worker thread. Later we added a business thread - where all the business operations take place – so we add it to a method and the aspect takes care of routing the call into the business thread and executing it. Then, with an additional aspect we call AsyncCallback, we indicate that it's a callback method and it's routed back to the UI thread using the dispatcher in WPF."

Localization Aspect

This aspect is part of the company's localization infrastructure and relieves developers from manually having to trigger it for complex texts within ViewModels."We have to be able to change the language of our applications during runtime so when a new language is selected the UI adapts," says Bernd. "The team has two approaches to the localization requirement. For static texts, found in the XAML file, we use custom markup extensions, which are added to the text, and the localization manager takes care of the localization. Dynamic texts, which are enhanced from information in the business logic, are handled inside ViewModels. These more complex texts are built by enhancing a text snippet with the information from the business logic and, calling on the localization manager, we simply add LocalizeAspect to a string property. The aspect takes care of the string you provide with the property then goes to the localization manager, does the localization, then returns the localized string."

PostSharp reduces code duplication and adds maintenance

"Within the code of our presentation layer (mainly ViewModels and XAML code) we saved about 15% just by using the ViewModelAspect" says Bernd.

One of the most cited benefits of AOP and PostSharp is time saved in reading and writing by the removal of boilerplate code. The team at Siemens Audiology experienced the time savings first hand.

"Releasing developers from writing boilerplate or infrastructure code helps my team to complete features faster," says Bernd. "While total cost savings may be difficult to quantify, we definitely saved coding time. We were relieved from writing the code for INotifyPropertyChanged and creating ICommand properties needed for data binding. We also avoided errors/bugs which would have occurred by forgetting to raise the appropriate PropertyChanged events."

SharpCrafters s.r.o.
Namesti 14 rijna, 1307/2
150 00 Prague 5
Czech Republic

US: +1 866 576 5361
CZ: +420 270 007 790
www.postsharp.net
info@postsharp.net