POSTSHARP

CASE STUDY:
Mitchell International

"PostSharp was especially good when we needed to transfer data between applications. We would have had to write a lot of code if not for PostSharp and now we don't even think about it. It made life a lot easier now from a development point of view."

Kshitij Deshmukh Rensfield
Software Development Manager
Mitchell International

Mitchell International

North America's leading provider of property & casualty claims management solutions saves from having to write tens of thousands of lines of code with PostSharp.

Summary

When the team at Mitchell International was asked to deliver a new suite of client-facing applications, one of its greatest challenges was to keep code repetition to a minimum. After evaluating a number of solutions, the development team at Mitchell chose PostSharp to deliver a significant decrease in boilerplate code and software defects – saving them from having to write tens of thousands of lines of code.

Engineers to deliver a new suite of client-facing software

As North America's leading provider of property & casualty claims management solutions, Mitchell International processes more than 50 million transactions annually for over 300 insurance companies and claims payers, and over 30,000 car collision repair shops.

The development team at Mitchell was tasked with delivering a new suite of client-facing software packages to:

1. simplify the company's claims management processes; and
2. accelerate the company's collision repair processes.

The new suite of applications, to be developed using the latest .NET technologies, had to be combined with disparate applications that have been in the marketplace for many years. This integration added a lot of complexity to the project.

Integration requires a lot of repetitive code

At the start of the project the development manager and principal engineer assessed Mitchell's existing client-facing software and discovered a mix of old Windows-based applications written in a variety of different technologies. "Visual C++, Visual Basic and WinForms just to name a few," says Kshitij Deshmukh, Mitchell's Software Development Manager.

The biggest challenge involved a huge C++ application that served as the main engine and revenue generator for their business:

- the application was 20 year old; and
- almost 2 million lines of code.

Rather than confront the daunting task of rewriting the application from the ground up, the team chose to break it into smaller components and tackle each of them separately. Continuing to use the C++ application as the main engine, the team set about rewriting all of its peripheral pieces one-by-one using .NET 4.0 and Windows Presentation Foundation as part of the new suite, visually embedding the old application in the center of the new one.

"As part of that process, we found that some of the existing patterns had a lot of repetition in them," says Kshitij, "especially in MVVM where you have to bind to all of the controls in XAML. There was a lot of code that you had to just continually keep writing all the time."

"PostSharp was especially good when we needed to transfer data between applications. We would have had to write a lot of code if not for PostSharp and now we don't even think about it. It made life a lot easier now from a development point of view."

Kshitij Deshmukh
Software Development Manager
Mitchell International

Why Mitchell chose PostSharp

The senior team began looking into run-time solutions to handle repetitive code. "We tried Spring.NET and CastleWindsor," says Kshitij, "but we soon discovered that there was a lot of run-time expense to incur with these frameworks. That's when we started looking at compile-time solutions."

Having previously tried PostSharp years earlier, Kshitij and his team evaluated version 2.1 and gave it high marks for:

- Visual Studio and MSBuild integration;
- high run-time performance;
- compile-time performance improvements over earlier versions.

Having chosen PostSharp to handle repetitive code in their new suite of software, it was now up to Tony Rensfield, Principal Engineer at Mitchell, to determine how to best implement the framework into the project.

"In the beginning of project, Kshitij and I were both writing code and did a lot of the up-front work for many of the aspects that we needed, including some small proof of concepts and their implementations," says Tony. The proof of concepts served as templates for developers coming in to the project so that, after just a few questions, they could understand how the aspects were applied and go about their day-to-day work of rewriting the peripherals.

Build and inject custom aspects into applications

Additionally to a few standard aspects, the team built several aspects to facilitate integration between the new and the old application:

- Activity Logging
- Exception Handling
- Performance Counter
- Thread Dispatching
- Data Sync

Thread Dispatching – A custom aspect was built for pushing threads and calls back onto the GUI thread. "We have events that come from the old application and the new application, and they talk back and forth." says Tony. "Depending upon when and on what thread those events come in, you might end up making calls that need to be pushed back up to the GUI thread. We have an aspect for that, so if you have a method that might need that, you basically apply that aspect, and it keeps you from having to write all the code to invoke or push it back to the dispatcher."

Activity Logging Aspect – A subset of logging, this custom aspect was built and applied to very specific methods so, when a milestone step is taken in the new application, the team is notified and has the option to push the logs up to their servers or display specific information or instructions to the end user.

Data Sync Aspect – A custom aspect was built to assure the applications stay synchronized with each other. "Approximately 30% of properties in the new applications are exact replications of the properties that exist in the old application", says Tony, "so we wrap these properties in an aspect that basically transfers the changes to the other application through an API. As the user is changing those properties live, these changes get transferred so that the two applications stay synchronized."

"The fact that none of our 18 developers have ever had to write an exception handler or even one logging line in this project is a huge win for us. Using aspects prevents a lot of user error and in that way PostSharp has been amazing for us."

Anthony Rensfield
Principal Engineer
Mitchell International

PostSharp reduces tens of thousands of lines of code

The team at Mitchell is pleased with the results from using PostSharp:

- reduced boilerplate code
- code is more readable
- code is easier to maintain

"It has reduced thousands of lines of code," says Kshitij. "You apply an aspect and it takes care of everything without having to mess around with each and every property or method. Now we can just look at business logic without actually having to worry about all the rest of the bookkeeping that needs to happen around it."

Kshitij and Tony are also pleased with how PostSharp makes enforcing good architecture practices easy. "You enforce a certain methodology so that you don't get team members doing things they're not supposed to and doing so helps to clean up everything and keep it that way." says Tony. "PostSharp has saved us from writing so many lines of code that it's hard to even quantify. The fact that none of our developers have ever had to write an exception handler or even one logging line in this project is a huge win for us. Using aspects prevents a lot of user error and in that way PostSharp has been amazing for us."

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