SDD DEEP DIVE 2017
Software Design & Development
London, 7-9 November 2017

.NET under the hood

Andrew Clymer & Richard Blewett

This workshop is designed for the developer who wants to lift the simple veneer of the .NET platform and really understand what makes it tick. Over three intensive days we'll unpick key parts of the framework and demonstrate in detail how they work. Obtaining this core knowledge will enable you as a developer to code more effectively, and perhaps more importantly when things go wrong you'll have a bank of knowledge to help you understand why.

As well as talks, the three days will include hands-on workshops. All you will need to bring is a laptop running VS2015 or above, and a mind that's keen to learn.

Session 1: Peeling back the .NET type system

All developers know how to create types, but do you really understand the difference between reference types and value types? When should you use a value type over a reference type, what limitations exist with value types, and what benefits do you get from using a value type? In this session we will answer these questions and many more, looking at various aspects of the .NET type system including generics and nullable types.

Session 2: Plugins and AppDomains

There are several ways of making applications and frameworks extensible in terms of layering and using abstractions. However, to make applications extensible beyond compile time introduces another set of issues: how are extensions/plugins discovered? How can you restrict what a plugin is allowed to do? How can you load a new version of a plugin without restarting the application? We'll look at reflection, attributes and AppDomains, and see how they can be used to create flexible, extensible frameworks.

Session 3: Language mechanics, it's a kind of magic

In the early days of C# there was virtually a 1-1 mapping between C# and IL. Much has changed, and the developer is now encouraged simply to describe their intent and the compiler builds the appropriate code. This session will dive under the hood and show how various C# features like events, iterator methods, anonymous methods, extension methods, dynamic and many more all result in code gen by the C# compiler. These layers of abstraction initially make us more productive, but sometimes they hide those hard-to-find bugs, & can add a deluge of performance implications that once we understand how the mechanics of the features work we can quickly fix.

Session 4: LINQ, from the ground up

Most developers use LINQ on a daily basis, but do you really understand how LINQ works? Building on the knowledge from Session 3, we will re-implement LINQ for Objects from the ground up. We'll show how LINQ works, and reduce the many gotchas that developers encounter when using LINQ. Having established how LINQ to Objects works we will then move on to understanding how LINQ works with non-object stores such as relational databases. Again, with a solid knowledge of how LINQ works with these kinds of stores we can eliminate much of the head scratching that occurs when things go wrong.

Session 5: Rx, from the ground up

Reactive Framework is a library that uses the .NET IObservable interface and LINQ to create a compelling new programming model that allows you to build "event" based code with declarative LINQ statements. This session builds Rx from the ground up, allowing us to see the mechanics of how it works.

Session 6: Understanding asynchronous programming

Creating units of work that run at the same time is a key concept of asynchronous programming both for compute and IO. In this session we will look at the Task abstraction and how it can be used in various ways to create asynchronous work. Once asynchronous work is running we will look at how we can observe and cancel the asynchronous operations. Finally, we'll look at how we can write our own implementation of the Task abstraction, to allow any asynchronous behaviour to be described as a Task.

Session 7: Working safely and effectively with asynchronous programming

Asynchronous programming requires careful attention to detail since most objects are not designed with multithreaded access in mind. This session introduces the importance of synchronization using a variety of synchronization techniques from Interlocked and Monitor-based synchronization, to reader write locks to concurrent data structures, and lock free algorithms (to name just a few). Along the way, we'll discuss the merits and costs associated with each of the techniques, giving you an insight into which technique to use in which situation, with the hope that next time you want to do some form of synchronization you won't just opt for the lock statement.

Session 8: Async and Await

.NET 4 introduced the new "task" abstraction, and C# 5 takes advantage of integrating this into the language via the async and await keywords. Furthermore, the new task abstraction promotes a new way of architecting asynchronous behaviour; in this session we will explore how to take advantage of these keywords and other new types and features being exposed in the next version of .NET to deliver far simpler asynchronous Windows UIs.

Session 9: Inside the Garbage Collector

The garbage collector has been part of .NET since its inception. However, exactly how the GC works is often shrouded in mystery. Also, the fact that memory management is automated doesn't release the developer from caring about memory issues – it's just that those memory issues appear in a different guise. In this module we take the lid of the GC, look at how it works and is optimized, and then assess what this means for you when you are writing your code: things you do that can help the GC and things that cause it problems.

Session 10: Power debugging

For many developers, debugging tools start and end with Visual Studio. However, there are many problems for which Visual Studio provides very little support - particularly threading and memory management issues. WinDbg and the plugin SOS.DLL bring a new set of tools to .NET developers which can provide insights that help you solve bugs that you see during testing, but also allow you to diagnose issues occurring in production systems where the only data you can get is a crash dump file.

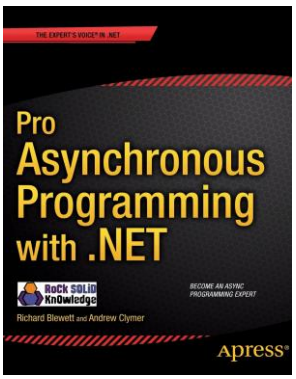
Speakers

Andrew Clymer & Richard Blewett

Andrew and Richard have been writing code for more years than they care to mention. They are two of the founders of Rock Solid Knowledge, a consultancy, development and training company based in the south west of England, specialising in the .NET framework.

They are regular conference speakers, and you will nearly always see them presenting together - one coding and one speaking - to create a fun, informative learning experience.

They co-wrote the highly acclaimed *Pro Asynchronous Programming with .NET*, published by Apress.



rocksolidknowledge.com @andrewclymer @richardblewett

BOOK YOUR PLACE NOW, & SAVE **£300** – CLICK HERE FOR PRICES