

# Technosis Training

Technosis are proud to offer customized on-site training courses in C# 3.0 and the .NET Framework. We specialize in intermediate- and advanced-level courses tailored to suit your individual needs.

You'll notice that our courses don't have numbers. These are not standard Microsoft classes and are not keyed to any exams—instead, they're designed to pick up where the standard courses finish. We teach leading-edge technology, tricks and best practices from top professionals in the field, emphasizing underlying concepts so that you won't need to memorize a lot of detail. Yet we keep the focus practical and aimed at making you highly productive in the real world.

At Technosis, we teach the very latest technologies, currently C# 3.0, LINQ and the CLR & core .NET framework. And we won't teach you what you already know: tell us where you're at, and we'll tailor a course to your situation. Our material is unique: we are not affiliated with any software organization and are able to give you fresh independent perspectives.

We also believe that people learn best through lively and interesting sessions. There are no PowerPoint presentations with slide after slide of bullet points: all courses feature creative visuals and plenty of practical coding demonstrations.

Courses can range from hours to weeks; from small groups to a large class. And with all courses, we can include consultancy with the training—providing a complete package where we oversee the successful integration of the technologies we teach into your project.



## Instructor Profile: Joe Albahari

Joseph Albahari has 17 years' experience in senior development roles and is author of [C# 3.0 in a Nutshell](#). He has a keen interest in LINQ, and has answered [numerous queries](#) on the online LINQ forums. He has also written [LINQPad](#), the popular utility for interactively querying databases in LINQ, and was a [speaker at the JAOO Conferences](#) in 2008.

# Courses

## C# 3.0 and LINQ

Despite LINQ's immense benefits, a vast number of people fall into one of two categories:

- Still using C# 2.0
- Using C# 3.0, but using LINQ superficially or incorrectly

The number of clumsy or malformed LINQ queries we've seen is astounding! A lot of people are struggling with this technology—look at the [LINQ Forums](#). Many of the questions arise, though, not because LINQ is excessively complex or nuanced—but because people haven't grasped the underlying principles upon which the technology is based, relying instead on a self-taught top-down approach.

By learning the fundamental concepts in a bottom-up fashion, most of the problems just vanish. Unlike with SQL, LINQ can all be understood almost entirely through a set of core principles. This course covers those core principles in a non-superficial way. At the end of this course, you'll understand LINQ queries not as embedded SQL, but as *functional programming islands*.

Just like building good software, we don't tackle the task of learning LINQ in one complex block. Instead, we've carefully separated the process of learning LINQ into *layers*—each layer building gradually on the previous. In the process, we'll reinvent many of LINQ's standard query operators, completely demystifying them.

By mastering LINQ, you'll enjoy the full benefits of this technology: improved signal-to-noise ratio in your code, future-proofing for parallelism, and the simplicity of a single querying language for XML, in-memory collections, database tables—even SharePoint documents. Equally valuable is the additional general knowledge of C# and modern coding patterns that you'll gain through learning LINQ thoroughly.

Here are some of the course highlights:

- The C# 2.0 building blocks: iterators, anonymous methods & closures
- Sequences and higher-order functions
- Lambda expressions & how they cut noise through type inference
- Extension methods—why they exist and how (not) to abuse them
- How lazy evaluation works, and how it decouples construction from execution
- Strategies for composing query operators
- Query syntax and its killer scenarios: why this syntax exists
- Anonymous types; type identity & their limitations
- The standard query operators
- Implicit typing and best practices
- Using LINQ in daily life
- How to use LINQ to Objects with Framework 2.0
- Leveraging the LINQ to XML DOM (and why you'll never want to instantiate an XmlDocument again)
- PLINQ: How the functional model pays when it comes to parallelism
- Expression trees and IQueryable
- The basics of LINQ to SQL
- Other new features of C# 3.0

All attendees receive complementary copies of [C# 3.0 in a Nutshell](#). The material can be structured as a two- or three-day course (depending on the desired level of detail and participants' existing knowledge) and can be combined with *Thinking in LINQ to SQL*.

## Thinking in LINQ to SQL

LINQ to SQL has immense benefits. Our own experience is that this technology **halves the development time in writing the middle tier**—as well as creating tidier and more robust code, without adding undue complexity. But not everyone is seeing these gains. Why?

A key problem is that because LINQ queries and SQL queries look superficially alike, people tend to believe that they *are* alike. So they first mentally formulate their queries in SQL, and then *transliterate* them into LINQ. The result is clumsy queries that capture the worst of both worlds.

In this course, we address the problem by giving you a deep understanding of LINQ's underlying principles, so you will *think directly in LINQ*. And then you will find it's actually much more natural than SQL.

Here's some of what we cover:

- When to use LINQ to SQL
- LINQ's fundamental building blocks: what exactly *is* a LINQ to SQL query if you were to examine it in memory?
- Modelling queries as a pipeline of linear and shape-changing operators
- Composing queries via nesting and subquerying
- The power of SelectMany and multiple generators
- LINQ's Join operators (and why they're unnecessary in LINQ to SQL)
- How to rise above the abstraction of inner vs. outer joins
- Minimizing round-tripping
- Maximising query reusability; overcoming language limitations
- The classic "transliteration" traps, and ways to overcome SQL-based thinking
- Unusual queries: non-equi joins, composite group keys, SQL IN queries; dealing with nulls
- How to extract maximum performance out of LINQ to SQL
- Using LINQ to SQL to perform updates
- Mapping stored procedures, views and functions
- What are the practical limitations of LINQ to SQL and how do deal with these
- When to consider Microsoft's Entity Framework, instead

Of course, LINQ to SQL doesn't work in every situation, so we cover the cases where it's best to revert to old-fashioned SQL. (We also describe how it's possible to combine both technologies within a single query.)

This course typically runs over two days. We can customize it, however, to be longer or shorter.

## Practical Multithreading

Multithreading is essential in most of today's applications to create the responsive and performant applications that customers demand. And yet Microsoft's Chris Brumme claims that "humans can't reliably write free-threaded code". One could respond by arguing that this is nonsense—the truth is that *humans can't reliably write code!* This is why we employ testing teams; this is why there's been a strong move to test-driven development over the past few years.

If we accept that "free-threading makes it even harder for humans to write reliable code", then what are the special challenges, and how best can we mitigate them? This course examines this question in a very practical way, covering tried-and-tested patterns for reliability, as well as all of the essential concepts and threading constructs in the .NET Framework.

We cover in detail:

- When to use multithreading: the benefits and costs
- The mechanics of parallel execution: time-slicing, priority, pre-emption, register caching, blocking, spinning
- The plumbing: threads, thread pools, asynchronous delegates, exception handling
- Start- and end-point data transfer
- Realtime data transfer: the model of shared writable state
- Exclusive and non-exclusive locking
- The broader issue of thread safety
- Protecting collections and CLR objects from thread-unsafe access
- Signalling constructs: event wait handles vs. Pulse/Wait
- Patterns for shared asynchronous activities
- Practical producer/consumer queue examples
- Reader/writer locks and timers
- Nonblocking synchronization constructs
- When things go wrong: safely aborting a thread
- Thread safety in application servers: WCF, ASP.NET, Web Services, Remoting
- Thread safety in rich client applications: WPF, Windows Forms
- ThreadStatic and local storage
- Declarative concurrency models
- Upcoming technologies: TPL & PLINQ
- How PLINQ changes the way you think about parallelism
- How to make your applications PLINQ-ready

The course is given by an instructor with extensive practical experience in multithreading, and can be structured to run over one or two days.

For more information, or to make a booking, contact Joe Albahari

© 2008 Joseph Albahari.

<http://www.technosis.com.au/Training.aspx>