Global Knowledge

Essential .NET with C# for .NET 4.5

Duration: 5 Days **Course Code: GK3296**

Overview:

Java and C++ developers usually have little problem learning the syntax of C# or VB.NET. Developing robust .NET applications requires more than just code though. Modern .NET applications rely on a complex runtime, the CLR. They are built from a mixture of source code, configuration files and XAML. Code isn't always written by hand; code generators and visual designers are used in several places to automate visual design, data access, and network communications code. Runtime features like Custom Attributes, Generics and Delegates mean .NET code is often structured very differently from similar code in C++ or Java.
Newly revised for .NET 4.5, Essential .NET will teach you to think like a .NET developer by showing you how modern applications are assembled and how the various pieces mentioned about work together to form a cohesive environment. The course begins with a thorough exploration of the "managed code" model - from "source code in Notepad" all the way through "CPU instructions executing on a CPU". You'll learn common idioms like code-behind and partial classes. You'll learn how to handle memory management issues and the "IDisposable" design pattern. You'll learn how to use C# functional idioms and LINQ to write compact, powerful, expressive, fluent code. You'll learn how to work with designers and tools to manage XAML, Code Behind, and Partial classes. You'll learn how to use Configuration Files to tweak application settings after deployment. And of course you'll take a look at the major class libraries, including the collection classes, LINQ, ADO.NET, Entity Framework, Parallel Framework Extensions, ASP.NET, XAML based UI and WCF.

Target Audience:

This course is designed specifically for reasonably experienced developers looking to transition to .NET. It is particularly aimed at C++ and Java developers, as a good portion of the course focuses on how .NET is different from those environments.

Objectives:

- Exposure to important .NET idioms, patterns, and best practices
- Use XAML, code-behind and partial classes
- Functional C# programming using Delegates, Lambda Expressions and LINQ
- Use LINQ to access objects, XML, and SQL relational data
- Write Metadata-driven code, including properties, events, and custom attributes
- Access Relational Databases and stored procedures using ADO.NET and the Entity Framework
- Build web sites with ASP.NET MVC
- Design and build web services using WCF
- Leverage the power of multi-threading and thread-safety in .NET with the parallel framework extensions

Prerequisites:

- as classes, inheritance, interfaces, and virtual methods, and some exposure to a "curlybrace"

language like C++ or Java. Developers without such background are welcome in

the course but the course moves quickly and they will likely find the pace quite

challenging.

Content:

Day 1

Platform and Architecture

The .NET runtime environment provides a set of core services to your code in terms of compilation, security and memory management. In this module we will examine the core architecture of .NET and explore the concept of "managed" execution. We will look at how code gets packed and deployed into binaries called assemblies and how this code is loaded and executed at runtime.

Types

One of the most powerful features of .NET is its comprehensive and pervasive type system. However, this type system has a number of aspects that need to be understood to work with .NET effectively. This module discusses the concepts of reference types, value types and the associated concept of boxing. It then proceeds to introduce generics - a technique for building code that can be used efficiently with many different types.

Reflection and Attributes

- The .NET runtime relies on the availability of full-fidelity type information. The runtime and its associated libraries use this type information to provide powerful services to your code. This idea, however, is not limited to code that Microsoft has written, you can use the same ideas in your own code to build rich frameworks that ca

Day 2

Delegates and Events

There are many occasions when you want to be able to pass a method to another piece of code such that they can call it at some point (the most common is to allow a component to fire events that you act upon). .NET has an abstraction to model this idea called delegates - this is the general model for how you pass methods as data. However, because the event concept is so common, .NET also has events as a first class member of the .NET type system and languages have specific keyword support for them. However, sometimes you don't specifically want to invoke one of your class's methods but rather an arbitrary block of code. To help with this idea C# introduced first

Iterators bring features from the world of functional languages into C#. These features simplify boring code and make it possible to streamline code that has traditionally been awkward in C#. This talk explains Iterator methods as the basis of the Iterator Pattern in .NET and how they are the underpinning of LINQ.

LINQ

- Language Integrated Query is an attempt to merge SQL-like constructs directly into languages like C# and VB.NET. We look at this technology including the from / select / where syntax and talk about how LINQ can be used to query your collections of objects.

Day 3

XML

XML processing is incredibly common. Whether you are working in the world of web services, storing user specific state or processing data from other systems XML is often the vehicle of choice. It is no accident that .NET has had XML processing support since its inception and that idea keeps evolving. In this module we will start by looking at the core XML streaming API and then introduce LINQ to XML as the new API for XML processing. LINQ to XML allows you to locate information with a very concise syntax. However, more that that it allows both the generation of XML and a simple way of transforming XML data into objects.

Memory and Resource Management

One of the key services the .NET runtime provides is automated memory management. In this module we will introduce how and when the garbage collector runs and what it does to reclaim your unused memory. However, memory is not the only resource that applications use: data base connections, file handles, sockets and more are all expensive resources that need management. .NET has a pattern for how non-memory resources are managed in the form of the interface IDisposable and its accompanying language support in the form of "using" blocks. The IDisposable pattern is often misunderstood so you will leave this module with a clear understanding of how it's used and when you need to provide an implementation yourself.

There are many reasons for writing asynchronous code, from performing long running actions in the background while still processing user input, to taking complex tasks and splitting them into smaller concurrent pieces of work. .NET models asynchronous work as "Tasks". Tasks can be scheduled, cancelled and coordinated. But to run highly concurrent work we also need collections that can safely be used by multiple tasks at the same time. This module examines the Task construct and the set of concurrent collections that ease the use of writing asynchronous processing on our modern inherently multi-core machines.

Thread Safety

Asynchronous programming requires careful attention to detail since unless an object has been specifically designed for concurrent access their state will get corrupted when used concurrently. This module introduces the importance of thread synchronization and looks at the different primitives that .NET provides to do this safely and efficiently.

ASP.NET

- Probably the most common type of application to build with .NET is a web application. .NET introduces a number of frameworks for building HTTP based applications and all of them sit on top of the same processing engine and deployment model. We will start this module looking at the core "HTTP Pipeline" for how requests get received and

Day 5

Windows Communication Foundation Architecture

The shift from object oriented development to component oriented development made it possible to build more loosely coupled and flexible systems where individual components evolved independently without impacting the overall system. The Windows Communication Foundation continues those practices by providing a model which is based on interfaces and contracts. This talk will introduce you to the WCF model and how to utilize it to build service-oriented and distributed applications.

anonymous delegates and then lambda expressions. This useful constructs not only allow you to show the codes intent more clearly but also open up the powerful functional pro

Iterators

Working with Databases

- Nearly all business applications have at their heart somewhere a relational database. In this module as look at the core data access API in the form of ADO.NET. We then look at Microsoft's chief Object Relational Mapper (ORM) technology - Entity Framework (EF). The idea is for framework code to take care of the mechanics of getting th

Day 4

Asynchronous Execution

Building XAML UI

Further Information:

For More information, or to book your course, please call us on 00 966 92000 9278

training@globalknowledge.com.sa

www.globalknowledge.com/en-sa/

Global Knowledge - KSA, 393 Al-Uroubah Road, Al Worood, Riyadh 3140, Saudi Arabia