

## Asynchronous Programming in C#

### Audience Course Asynchronous Programming in C#

The course Asynchronous Programming in C# course is designed for developers who want to learn how to implement asynchronous code with `async` `await` in .NET.

### Prerequisites Course Asynchronous Programming in C#

Good knowledge of C# or a comparable language such as Java is required to participate in this course.

### Realization Training Asynchronous Programming in C#

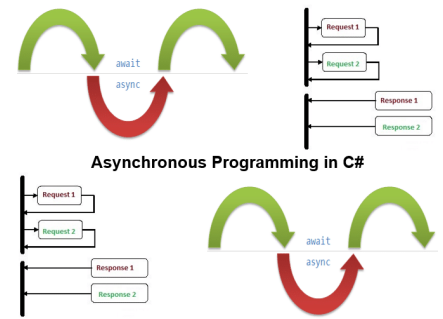
The concepts are explained using presentations and demos in Visual Studio.NET. There is ample opportunity to practice. The course times are from 9.30 to 16.30.

### Certification Course Asynchronous Programming in C#

After successfully completing the course the participants will receive a certificate Asynchronous Programming in C#.

Duration: 2 days

Price: € 1499

[Open Schedule](#)


## Content Course Asynchronous Programming in C#

The course Asynchronous Programming in C# focuses on how to use C# and the .NET Framework to write asynchronous code with the `async` and `await` mechanism.

### Intro Asynchronous Programming

The course Asynchronous Programming in C# starts with a discussion of the differences between synchronous and asynchronous code. Covered are the disadvantages of blocking behavior and the advantages of parallelism and concurrency with threads and tasks.

### Synchronous versus Asynchronous

It is explained how in synchronous code a thread that issues a blocking call waits for the result. Asynchronous code does not wait while another thread executes the call. The original thread is notified via a callback or other mechanism when the result is ready.

### Async Programming in .NET

Next the implementation of asynchronous code in the .NET Framework and .NET Core is treated. Various patterns for writing asynchronous code are discussed. The benefits of using .NET Core are covered and asynchronous algorithms are explained.

### Async Await

Then the `async` `await` mechanism is discussed. It is explained how a method preceded by the `async` keyword becomes an asynchronous method. In the body of the method the keyword `await` can then be used to wait for the result of an asynchronous call.

### Synchronization

The prevention of data corruption by means of synchronization primitives such as locks, mutexes and semaphores is also on the program of the course Asynchronous Programming in C#. Attention is paid to race conditions and deadlock as well.

### Exception Handling

Furthermore exception handling in an asynchronous environment is a topic of the course. The focus is set on faulted tasks and disposable objects. Finally asynchronous calls to services are discussed and the interaction between handling in the frontend and the backend.

## Modules Course Asynchronous Programming in C#

<b>Module 1 : Async Intro</b>	<b>Module 2 : Async in .NET</b>	<b>Module 3 : Async Await</b>
Synchronous Code Blocking Behavior Asynchronous Code Callbacks Completion Events Threads and Tasks Parallelism and Concurrency IO Bound Tasks CPU Bound Tasks Long Running Tasks Background Workers	Async in .NET IAsyncResult Asynchronous Patterns Event Based Pattern Task Based Pattern Async .NET Core .NET Core Benefits Asynchronous Algorithms Thread Pools Thread Pool Starvation Memory Consumption	Async Keyword Async Method Await Keyword Suspending Execution Yielding Control Awaitable Tasks ConfigureAwait GetAwaiter Task Completion Task Composition Task Object
<b>Module 4 : Synchronization</b>	<b>Module 5 : Exceptions</b>	<b>Module 6 : Advanced Topics</b>
Race Conditions Deadlock Need for Synchronization Thread Safe Code Lock Objects Mutexes Semaphores Timing and Synchronization	Exception Handling Asynchronous Exceptions Throwing Exceptions Task.Exception Property Faulted Tasks Catching Exceptions Disposable Objects AggregateException	Async Services Async Request Ajax Calls Async Frontend Async Backend Await Tasks Efficiently WhenAll WhenAny