

microsoft net developer quick reference guide

Printable PDF

Microsoft .NET Developer Quick Reference Guide

Welcome to this comprehensive Microsoft .NET Developer Quick Reference Guide. Whether you're a seasoned developer or just starting your journey with the .NET platform, this guide aims to provide you with essential insights, best practices, and quick tips to enhance your productivity and understanding of .NET development. Covering core concepts, tools, frameworks, and coding practices, this reference will serve as a handy resource to navigate the .NET ecosystem efficiently.

Understanding the .NET Platform

What is .NET?

The Microsoft .NET platform is a versatile, cross-platform framework designed for building various types of applications, including desktop, web, mobile, cloud, gaming, IoT, and AI solutions. It provides a rich set of libraries, languages, and tools to streamline the development process.

Key Components of .NET

- **.NET Runtime (CLR):** Executes .NET applications, manages memory, and handles security.
- **.NET Libraries:** Base Class Library (BCL) offering pre-built classes and functions.
- **Languages:** Primarily C, F, and Visual Basic, all compiled into Intermediate Language (IL).
- **SDKs & Tools:** Command-line interface (CLI), Visual Studio, Visual Studio Code, and other IDEs.

Core .NET Technologies

.NET Core vs. .NET Framework vs. .NET 5/6/7+

Understanding the differences among these platforms is crucial:

- **.NET Framework:** Windows-only, mature, ideal for existing desktop and web apps.
- **.NET Core:** Cross-platform, open-source, optimized for modern cloud applications.
- **.NET 5/6/7+:** Unified platform combining features of .NET Core and .NET Framework, with

ongoing enhancements.

Choosing the Right Framework

Consider the following:

- For Windows desktop applications: .NET Framework (WPF, WinForms)
- For cross-platform web and cloud apps: .NET 6 or later (.NET Core-based)
- For microservices and containerized environments: .NET 6+

Development Environment Setup

Installing the .NET SDK

- Download the latest SDK from the official [.NET website](https://dotnet.microsoft.com/download).
- Follow platform-specific instructions for Windows, macOS, or Linux.
- Verify installation with the command: ``dotnet --version``.

Choosing the Right IDE

- Visual Studio (Windows and Mac): Full-featured IDE with advanced debugging, IntelliSense, and GUI designers.
- Visual Studio Code: Lightweight, cross-platform editor with extensions for C and .NET.
- JetBrains Rider: Commercial IDE supporting cross-platform development.

Creating Your First Project

Use the CLI:

```
dotnet new console -n MyFirstApp  
cd MyFirstApp
```

```
dotnet run
```

Key .NET Development Concepts

Languages

- C: The primary language for .NET development, known for its simplicity and power.
- F: Functional programming language suited for data processing and complex algorithms.
- Visual Basic: Used mainly in legacy applications.

Project Types

- Console Applications: Command-line tools.
- Class Libraries: Reusable components.
- Web Applications: ASP.NET Core MVC, Razor Pages, Blazor.
- Desktop Apps: WPF, WinForms.
- Mobile Apps: Xamarin, MAUI.
- Microservices & APIs: ASP.NET Core Web API.

NuGet Package Manager

- Used to manage third-party libraries and dependencies.
- To add a package:

`dotnet add package [PackageName]`

- To restore packages:

`dotnet restore`

ASP.NET Core for Web Development

Overview

ASP.NET Core is a high-performance, cross-platform framework for building modern web applications and APIs.

Key Features

- Middleware pipeline for request processing
- Razor Pages for page-centric development
- Blazor for WebAssembly-based client apps
- Entity Framework Core for ORM

Building a Web API

Steps:

