Debugging

C# and .Net

Yash Lathiya

Table of Contents

Sr No.	Topic	Date	Page No.
1	Introduction		
2	Debug Points | Breakpoints		
3	Different Debug Windows		
4	Editing		
5	Conditional break points		
6	Data inspector		
7	Conditional Compilation		

Debugging

Introduction

Debugging is a crucial aspect of software development, serving as the systematic process of identifying and resolving issues or errors within a program's code.

It is a skill that developers rely on to ensure that their software functions as intended, meets requirements, and operates efficiently.

Debugging involves a careful examination of the code, the identification of unexpected behaviour, and the implementation of corrective measures to rectify problems.

Why Debug ?

- Error Identification
- Code Inspection
- Use of tools
- Logging & Tracing
- Iterative Process
- Continuous Improvement

Breakpoint

A breakpoint is a designated point in the code where the debugger pauses execution, allowing developers to inspect variables, evaluate expressions, and step through the program one line at a time.

This powerful feature provides a dynamic and interactive means to identify and resolve issues within the code.

Why breakpoints ?

- Pause Execution
- Variable Inspection
- Conditional breakpoint
- Trace Execution Flow
- Modify Code & Fly (Hot Reloading) (Editing)

Debug Windows

- Breakpoints Window:

Displays a list of all breakpoints set in your code. You can use this window to manage, enable, or disable breakpoints.

- Call Stack Window:

Shows the current call stack, allowing you to trace the sequence of method calls that led to the current point of execution. It's particularly useful for understanding the flow of our program.

- Watch Window:

Enables you to monitor the values of variables, expressions, and properties during debugging. You can add variables to watch and see their values change as you step through the code.

- Locals Window:

Similar to the Watch Window, the Locals Window displays information about local variables in the current scope during debugging.

- Immediate Window:

Allows you to interactively execute code and evaluate expressions during debugging. It's useful for testing code snippets without modifying your source files.

- Output Window:

Displays various types of output, including build output, debug output, and other messages from Visual Studio and your application.

- Error List Window:

Lists compilation errors, warnings, and messages. It's crucial for addressing issues in your code.

- Threads Window:

Displays information about the threads in your application, including their IDs and current states. It's useful for multithreaded debugging.

Conditional Breakpoint

A conditional breakpoint is a type of breakpoint in a debugger that allows you to specify a condition under which the breakpoint should be triggered.

When the specified condition evaluates to true, the program execution will pause, and you can inspect the program's state, variables, and other relevant information.

- Conditional Expression
- Hit Count
- Filter

Data Inspection

Data inspection is a crucial aspect of the debugging process in software development.

It involves examining the values of variables, expressions, and data structures during runtime to gain insights into the behaviour of a program and identify issues.

Effective data inspection helps developers understand how the data evolves and pinpoint the source of bugs or unexpected behaviours.

Conditional Compilation

Conditional Compilation allows developers to include or exclude portions of code during the compilation process based on specified conditions.

This feature is useful for creating different builds for various scenarios such as development, testing, or production environments.

It enables developers to manage different configurations and control the inclusion of specific code blocks based on defined symbols or conditions.

```
#if DEBUG
    // This code will be included only in DEBUG build
    Console.WriteLine("Debugging is enabled.");
#else
    // This code will be included in other build configurations
    Console.WriteLine("Debugging is disabled.");
#endif


#if GradeA
    // This code will be included only if GradeA is defined on the top
    Console.WriteLine("Debugging is enabled.");
#elif GradeB
    // This code will be included only if GradeB is defined on the top
    Console.WriteLine("Debugging is disabled.");
#endif
```