Day 52/60 - Difference Between Throw and Throw Ex in C#

When handling exceptions in C#, understanding the difference between throw and throw ex is crucial for maintaining an accurate stack trace and debugging effectively.

Using throw (Preserves Original Stack Trace)

Re-throws the original exception without modifying the stack trace.
Recommended when logging or handling an exception before propagating it further.

Example:

```
try
{
 int result = 10 / 0; // Division by zero error
}
catch (Exception ex)
{
 Console.WriteLine("Logging error: " + ex.Message);
 throw; // Preserves stack trace
}
```

Best for: Keeping the original error source intact for debugging.

Using throw ex (Resets Stack Trace, Bad Practice)

L Resets the stack trace, making it harder to find the root cause.
Should be avoided, unless you need to modify the exception message.

Example (Not Recommended):

```
try
{
 int result = 10 / 0;
}
catch (Exception ex)
{
 Console.WriteLine("Logging error: " + ex.Message);
 throw ex; // Resets stack trace (Bad Practice)
}
```

Issue: The exception appears to originate from throw ex, losing the original source of failure.

When to Use What?

Use throw to propagate exceptions properly while preserving debugging details.
Avoid throw ex, unless you need to wrap or modify the exception before re-throwing it.

By using throw correctly, you ensure better debugging, maintainability, and accurate error tracking in your .NET applications!

#dotnet #csharp #exceptionhandling #debugging #bestpractices