**Details**

Debugging is a game of strategy. The rules are set by the computer, your debugger, and requirements or user expectations. You may enter the contest expecting a trivial opponent, only to find it like Hydra with two new problems sprouting for each one you solve. It’s a critical game because we fix bugs from the time we first check in code, and the cost of each bug tends to increase across the project lifecycle. In this talk, you’ll learn about debugging strategies that help you ask better questions in order to significantly reduce the amount of time you spend debugging. Poor performance can be a bug, and you’ll learn straightforward profiling techniques to help you understand performance problems. Core debugging principles covered in this talk apply across most languages and platforms; the demonstrations will use Visual Studio 2017. Leave this talk ready to formulate ideas into testable strategies you can use to resolve bugs faster and with more confidence.  
  
Speaker  
  
Kathleen Dollard  
Kathleen loves to code and loves to teach and talk about code. She’s written tons of articles, a book, and spoken at numerous conferences around the world. She’s on the .NET Core team at Microsoft, where she works on the .NET Core CLI and SDK and managed languages (VB.NET and C#). She’s always ready to help developers take the next step in exploring the wonderful world we call code.  
  
Agenda  
  
6:30pm - Doors open / Food by Google Cloud / Networking  
7:00pm - News and announcements  
7:10pm - Presentation  
8:30pm - Wrapped Up

Principal Program Manager

SDK, .NET Core

[kdollard@microsoft.com](mailto:kdollard@microsoft.com)

**Debugging**

What’s a bug

* Something you have to fix in existing code

What is your approach?

True regression?

Bug?

New Feature?

Code is hard to debug when **can’t read the code**

Code is hard to debug when the symptom (issue) is distant from the problem (bug) in time or space.

* Why Microsoft is investing in null exceptions in C# version 8.

Code is hard to debug when the context causing the bug is transitory

* Server, or timing problem

Code is hard to debug when someone gave you incorrect or misleading information

**Debugging Strategies**

Click on the Gear icon for the conditional breakpoint.

**Your brain** is the most important debugging tool

* But we usually turn it off.
  + Maybe we do this, maybe we do that

You should be able to talk about it or articulate it

* This is a debugging strategy

There are very hard bugs in the world

Easy bug

* Doesn’t matter what strategy

Relatively easy

Debugging Strategy - Divide and Conquer

* Go half way, figure it out, go half way, figure it out

A good debugging strategy is any strategy that you can articulate

Debugging Strategy – Scientific Method

Something happens

If there’s an exception

* You Google first / Bing lol

If your customer reports a bug

* How did the customer get the bug?
  + This is a bug in the testing

**Debugging Method – Scientific Method**

Something happens

* Can you reproduce in Visual Studio
  + If yes, reproduce, think, write test
    - 1 - Create several hypotheses
    - 2 - Select one to disprove
    - 3 - Run experiment
    - 4 - Evaluate
  + Else if – Outside production?

Some of your code is more buggy, some of your code less buggy

Increasing the test quality

Spending time to write test quality

**Create several hypotheses**

Break up of how your brain works

* Brain has a tendency to follow the idea or the bug
  + It might happen that the bug is in another code base
  + Rabbit hole

**Select one to disprove**

* Helps disrupt thinking in the rabbit hole
* In many cases it is the same tests
* You are trying to look where the bug isn’t there
  + You can prove this
* Grab the best idea and disprove it to deal with confirmation bias

Fail fast

**Evaluate**

* Go back to hypotheses
* The iteration would take to 10 – 15 mins
* Like a binary search. Cause it’s not always a pipeline

Do you select the fastest you can disapprove?

* If you are drawn to something, disapprove that

Do this example in notepad

**Case: Field is empty**

* The query isn’t returning the data
* The data isn’t in the database
* It’s not copied to the model
* The font color is the same as the background color
  + Highlight and paste it
    - You saw it
* Formatting problem between the database and the UI
* There’s a masking problem
* Data binding problem
  + Does a breakpoint
    - The data is there
      * Knocks out:
        + The query isn’t returning the data
        + The data isn’t in the database
        + Database connectivity problem
        + It was never input
* Database connectivity problem
* Not running the client code
* It was never input

You started fiddling

* Server is fine

The problem was font color

* But you did a lot of test before you came here

Did you check this? Check this again. Did you check this? Check this again.

* That’s why you need a list

Picard Tips

* follow him on Twitter

Collaborative Debugging Game

* a
* Take turns imaging the bug

Your skill keeps bug in the correct quadrant

* Quadrant IV OMG! And Hard
  + Privacy or Data Loss
    - High Value
* Quadrant I Easy and OMG!
  + Typo in the company name
* Quadrant II Easy and Oh Well
  + Ordinary Typo
    - Low Value
* Quadrant III Hard and Oh well
  + Many performance issue
    - You test that, and you (another person) test that.
    - Man power to fix the problem

**Debugging Method – Scientific Method**

Something happens

* Can you reproduce in Visual Studio
  + If yes, reproduce, think, write test
    - 1 - Create several hypotheses
    - 2 - Select one to disprove
    - 3 - Run experiment
    - 4 - Evaluate
  + Else If – Outside production?
  + No
    - Same thing?

In Microsoft, there’s a lot of tool that can grab a lot of information

**Get help with high value bugs!**

* That’s was why you practiced
* John Robbins consulting

**Light Profiling**

You have, but you don’t know what it really is.

How many is super confident in profiling.

Show a code in Visual Studio (demo session)

* Run the code
  + Showed “less than or equal 14,120ms elapsed” (14 seconds)
    - Perf tips (performance tips)

So how do you profile the 14,120ms?

In VS 2017, you don’t need console.Read to stop

In the demo source code, only 1 line of code causes the performance problem.

* In VS, look for “Performance Profiler”
  + Tick “CPU Usage”
  + Run the application again
    - Does the same thing with the Gear Icon
  + VS will enter into diagnostics
  + Uncheck “External Code” in filters
  + Go to details
  + “Call Tree”
  + “Expand Hot Path”
    - This will show System.Linq.Enumerable::Count with “Hot”

What Count() does inside Linq?

* Counts all the character?
* Replace Count() with Length
* Count() goes through the end of the string every time
* Because we are hitting the Count() a lot in a specific code block

Check the Performance Profiler again

* Performance is now down to 2 milliseconds

Usually it’s asked, is there a GC problem?

* There’s no GC problem, there could be an allocation problem
* In the Diagnostic Tool
  + If you don’t see the GC icon in the graph, there’s no GC problem

String Builder is very helpful

* This takes away the GC problem

Debugging is explicit

Profiling – you may not have seen this

* i.e. It’s not a problem in the string, there’s a problem in the array

<https://github.com/KathleenDollard/talks-debugging>

@KathleenDollard

* Debugging is a game of strategy
* Rules are set by the computer, your debugger, requirements or user expectations