# WEB524

## WEB PROGRAMMING ON WINDOWS

**WEEK 4 – LECTURE 2**

DEBUGGING AN ASP.NET WEB APP

# Resources

- Today we will discuss Debugging.

- Getting Started with Debugging in Visual Studio 2017
  - Skim this document.
  - Ensure you have a clear understanding of the section labelled “Debug your running code”.  You will be tested on this material.
- Using Breakpoints
- Autos and Locals Windows
- Navigating through Code with the Debugger

# Debugging Introduction

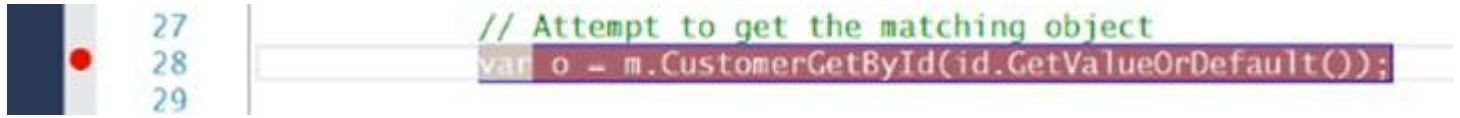
- Visual studio has a window (aka pane) called the “Error List”.
  - It shows a list of errors and warnings.
  - Errors will usually prevent you from compiling (building) the project.
  - Warnings typically do not prevent compilation but should be fixed.
  - You may double-click on an error or a warning to view the line of code that caused the message to appear.
  - You may right-click on an error or a warning for more details about the error.

# Debugging an ASP.NET MVC Web App

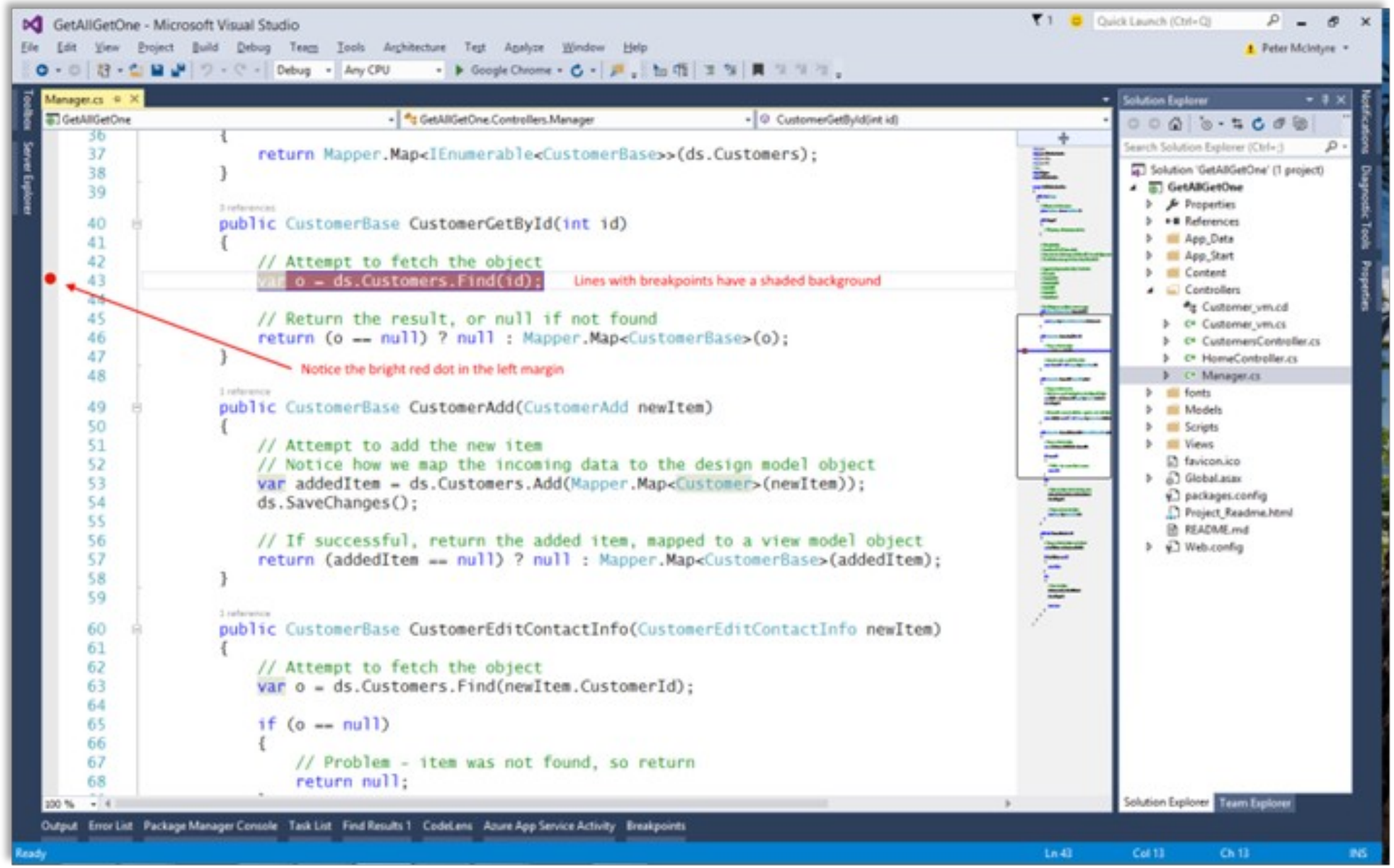
- Web apps tend behave different than a full fledged app.
- Many times they will compile correctly and show zero errors in the "Errors" pane, but when you run the web app you may receive an error in the browser.
  - These errors are known as "run-time" errors.
- Detailed error information is displayed in the browser.
  - Many times you can read the error information along with the stack trace to determine the offending line of code.
  - Sometimes errors are cryptic and not as easy to find.  You may ask your professor for help if you fall in to this situation.

# Breakpoints

- A breakpoint is a line marker/indicator that will stop execution before the line of code is executed. In the code editor, it appears as a bright red dot.




- The next slide shows a full screen shot with the breakpoint.


GetAllGetOne - Microsoft Visual Studio
Manager.cs
GetAllGetOne.Controllers.Manager
CustomerGetById(int id)
Lines with breakpoints have a shaded background
Notice the bright red dot in the left margin
Solution Explorer
Solution 'GetAllGetOne' (1 project)
GetAllGetOne
Properties
References
App_Data
App_Start
Content
Controllers
Models
Scripts
Views
Output
Error List
Package Manager Console
Task List
Find Results 1
CodeLens
Azure App Service Activity
Breakpoints
Ready

# Breakpoints

- There are a couple of ways to set or remove breakpoints:
  1. In the left margin (aka gutter), click the line of code that needs the breakpoint.
  2. Move the cursor to the line of code and press the F9 key.

- You can view a list of breakpoints and manage your breakpoints using the “Breakpoints” pane.
  - If the pane is not visible, click Debug > Choose Windows > Breakpoints.

# When to add Breakpoints

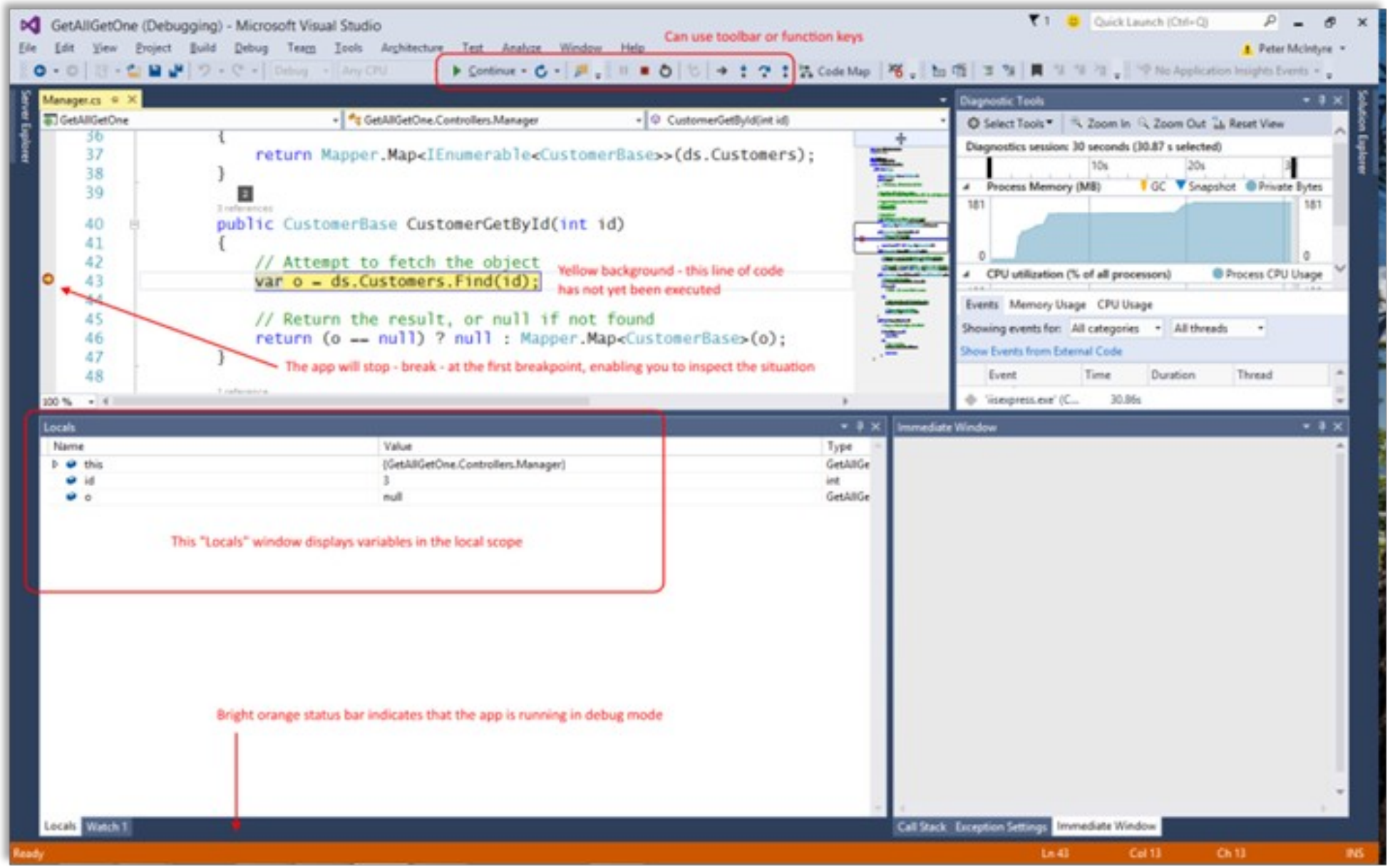
1. When you introduce new code, set a breakpoint at the beginning of the code block and step-through the code to make sure it is behaving as expected.

2. If you have implemented a complicated behaviour, set breakpoint(s) and inspect the values of the variables.

3. In a Manager class, set a breakpoint when you are making changes in the data store. *For example, the "add new" and "edit existing" use cases.* This enables you to inspect and compare values in view model objects with design model objects.

# Starting an app with the Debugger

- Previously, you used CTRL+F5 to run your app.
- From now on, start the app with the debug environment by pressing F5 (don't hold CTRL).
- The app will run as usual.
  - If an exception is thrown, the debugger will pause the program and highlight the problematic line of code.
  - If a breakpoint is hit, the debugger will pause the program and highlight the line of code that will execute next.
  - Visual Studio will typically receive focus when it pauses debugging.

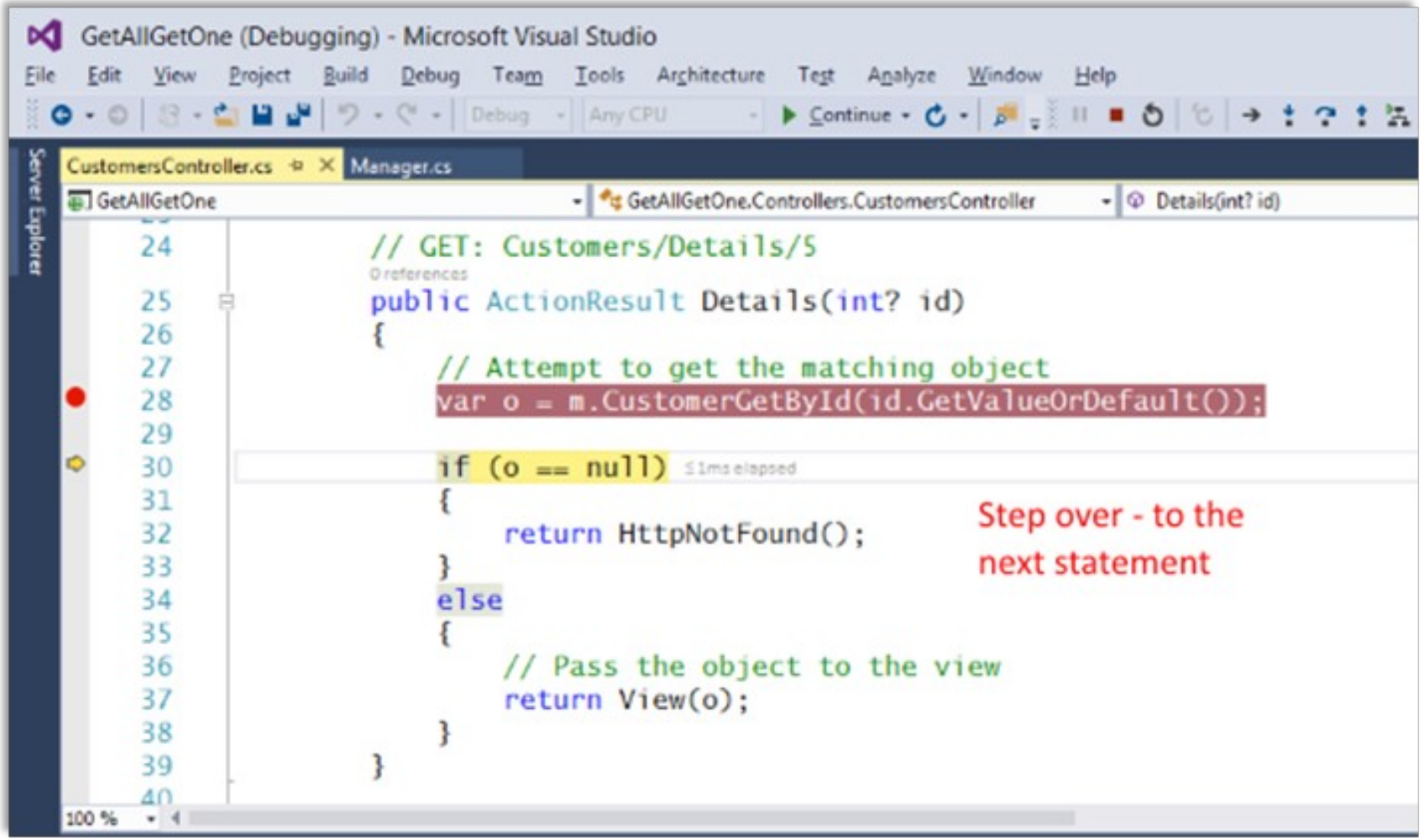

GetAllGetOne (Debugging) - Microsoft Visual Studio
Can use toolbar or function keys
Manager.cs
GetAllGetOne.Controllers.Manager
CustomerGetById(int id)
return Mapper.Map<IEnumerable<CustomerBase>>(ds.Customers);
public CustomerBase CustomerGetById(int id)
// Attempt to fetch the object
var o = ds.Customers.Find(id);
// Return the result, or null if not found
return (o == null) ? null : Mapper.Map<CustomerBase>(o);
Yellow background - this line of code has not yet been executed
The app will stop - break - at the first breakpoint, enabling you to inspect the situation
Diagnostic Tools
Diagnostics session: 30 seconds (30.87 s selected)
Process Memory (MB)
CPU utilization (% of all processors)
Events
Memory Usage
CPU Usage
Locals
Name
Value
Type
this
{GetAllGetOne.Controllers.Manager}
id
3
o
null
This "Locals" window displays variables in the local scope
Immediate Window
Bright orange status bar indicates that the app is running in debug mode
Locals
Watch 1
Call Stack
Exception Settings
Immediate Window
Ready

# Pause …

- The next line of code to be executed has a yellow background.
- At this point in time, you can:
  - Continue execution (with F5 or the “Continue” toolbar button)
  - Stop execution (with Shift+F5 or the “Stop” toolbar button)
  - Inspect the value of the variables in the local scope
  - Execute the line of code (Step Over)
  - Or, if the line of code runs a method that YOU wrote, you can jump to the beginning of that method (Step Into)

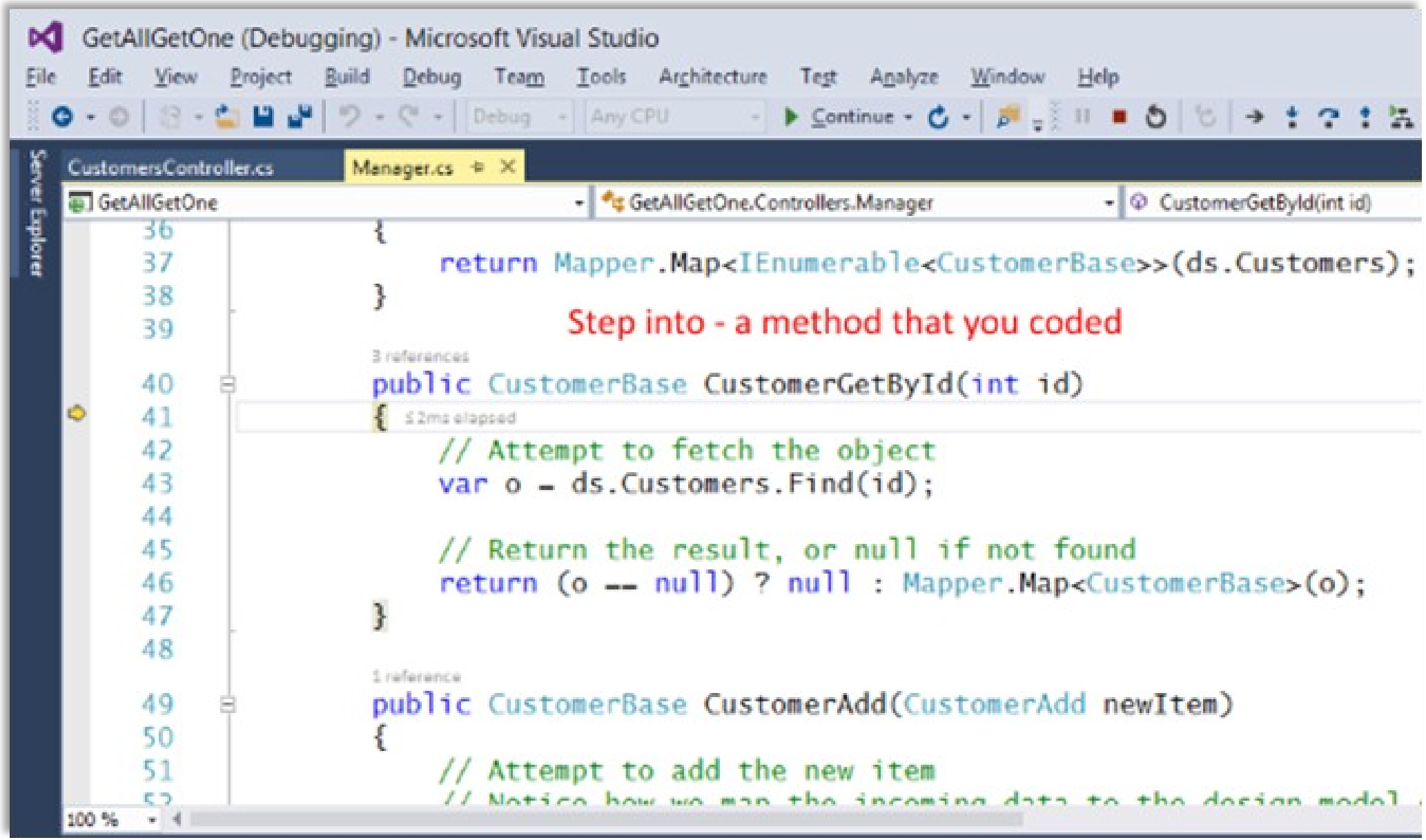
# Step Over

- “Stepping” is the process of executing code, one line at a time.
- Execute the line of code by pressing the F10 function key (or by using the debug toolbar).
- In the following image, stepping over will move the pointer (yellow line) from line 28 to line 30.


GetAllGetOne (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Debug
Any CPU
Continue
Server Explorer
CustomersController.cs
Manager.cs
GetAllGetOne
GetAllGetOne.Controllers.CustomersController
Details(int? id)
24 // GET: Customers/Details/5
0 references
25 public ActionResult Details(int? id)
26 {
27 // Attempt to get the matching object
28 var o = m.CustomerGetById(id.GetValueOrDefault());
29
30 if (o == null) ≤1ms elapsed
31 {
32 return HttpNotFound();
33 }
34 else
35 {
36 // Pass the object to the view
37 return View(o);
38 }
39 }
40
100 %
Step over - to the next statement

# Step Into

- If the current line of code has a method call to a method that YOU wrote, you can jump to the beginning of that method.
- Press F11 to “step into” a method.
- By default, the debugger will only step into methods, not properties. This behaviour is configurable.
- In the following image, stepping into will move the pointer from line 28 in the CustomersController.cs source code file to line 41 in the Manager.cs source code file.

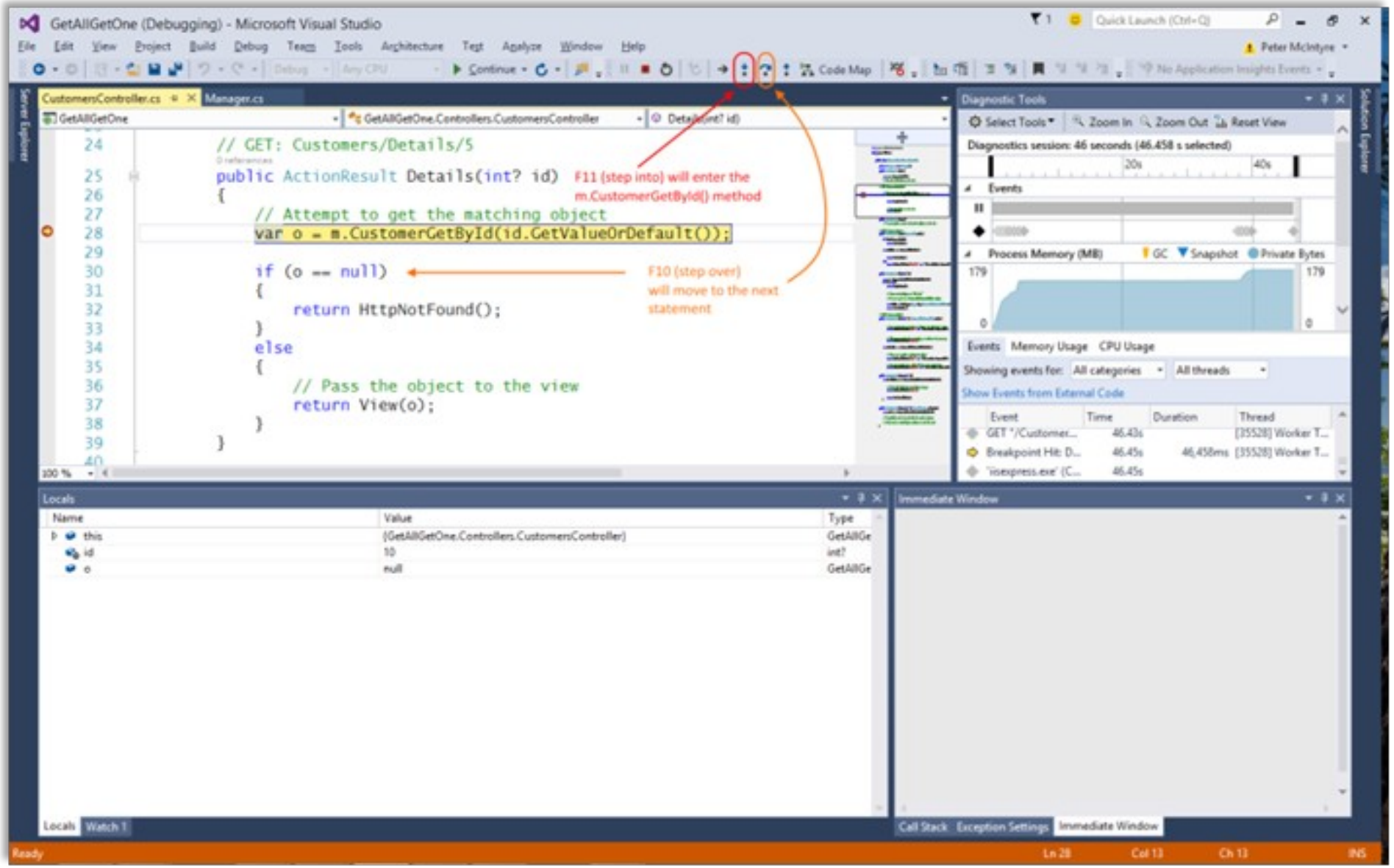

GetAllGetOne (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Debug
Any CPU
Continue
Server Explorer
CustomersController.cs
Manager.cs
GetAllGetOne
GetAllGetOne.Controllers.Manager
CustomerGetById(int id)
36 {
37 return Mapper.Map<IEnumerable<CustomerBase>>(ds.Customers);
38 }
39
Step into - a method that you coded
3 references
40 public CustomerBase CustomerGetById(int id)
41 {
42 // Attempt to fetch the object
43 var o = ds.Customers.Find(id);
44
45 // Return the result, or null if not found
46 return (o == null) ? null : Mapper.Map<CustomerBase>(o);
47 }
48
1 reference
49 public CustomerBase CustomerAdd(CustomerAdd newItem)
50 {
51 // Attempt to add the new item
100 %

# Step Out

- While in the method, you can continue to F10 “step over” or F11 “step into”, if appropriate.

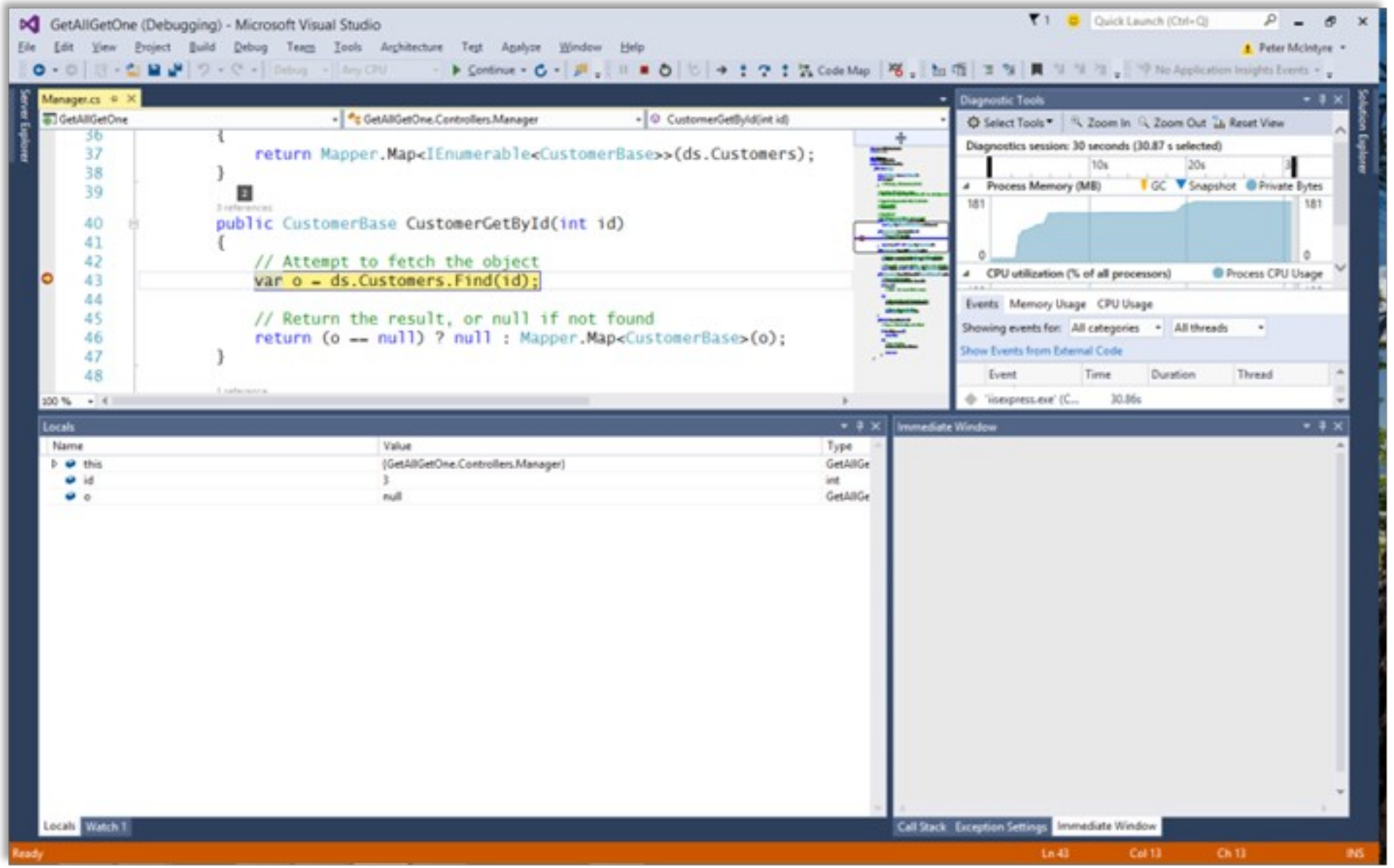
- If/when you want to leave the method you are in, you can Shift+F11 to “step out” of the method and return to the line of code that was pointed to before entering the method.

- Here’s a full-screen image that summarizes the difference between “step into” and “step over”.

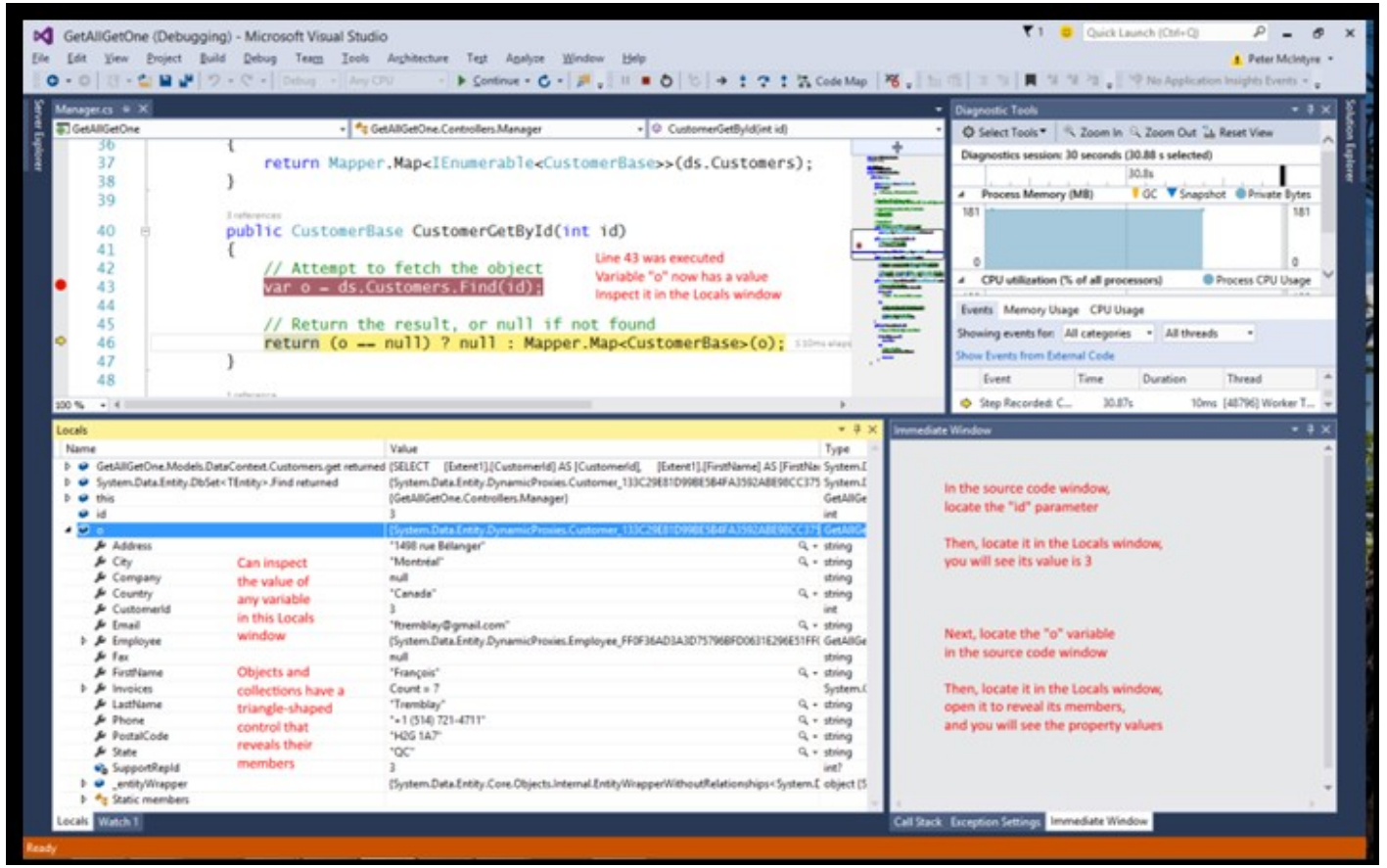

GetAllGetOne (Debugging) - Microsoft Visual Studio
CustomersController.cs
Manager.cs
GetAllGetOne.Controllers.CustomersController
24 // GET: Customers/Details/5
25 public ActionResult Details(int? id)
26 {
27 // Attempt to get the matching object
28 var o = m.CustomerGetById(id.GetValueOrDefault());
29
30 if (o == null)
31 {
32 return HttpNotFound();
33 }
34 else
35 {
36 // Pass the object to the view
37 return View(o);
38 }
39 }
F11 (step into) will enter the m.CustomerGetById() method
F10 (step over) will move to the next statement
Diagnostic Tools
Diagnostics session: 46 seconds (46.458 s selected)
Events
Process Memory (MB)
Memory Usage
CPU Usage
Locals
Name
Value
Type
this
id
o
null
Immediate Window
Call Stack
Exception Settings
Ready

# Inspecting Variables

- During a break in execution, you can inspect variable values:
  - In the code editor, hover your mouse over a variable and a popup will show its value.
  - Use the Locals window to see all of the variables in scope.
- The following image shows the Locals window – lower-left – at the first breakpoint.


GetAllGetOne (Debugging) - Microsoft Visual Studio
Manager.cs
GetAllGetOne.Controllers.Manager
CustomerGetById(int id)
36 {
37 return Mapper.Map<IEnumerable<CustomerBase>>(ds.Customers);
38 }
39
40 public CustomerBase CustomerGetById(int id)
41 {
42 // Attempt to fetch the object
43 var o = ds.Customers.Find(id);
44
45 // Return the result, or null if not found
46 return (o == null) ? null : Mapper.Map<CustomerBase>(o);
47 }
48
Diagnostic Tools
Diagnostics session: 30 seconds (30.87 s selected)
Process Memory (MB)
CPU utilization (% of all processors)
Events
Memory Usage
CPU Usage
Locals
Name
Value
Type
this
{GetAllGetOne.Controllers.Manager}
id
3
int
o
null
Immediate Window
Locals
Watch 1
Call Stack
Exception Settings
Immediate Window
Ready

# Pressing F10 *(See Locals Window)*



# Hands On

- Today's code example – DebuggingIntro
- Displays the Edit and Delete use cases but will introduce errors and to get you started with a debugging experience.

1. Download the code example.
2. Go through the example error conditions.
   - Object reference not set to an instance of an object
   - Mapping error; missing AutoMapper map
   - Validation error when saving to the data store