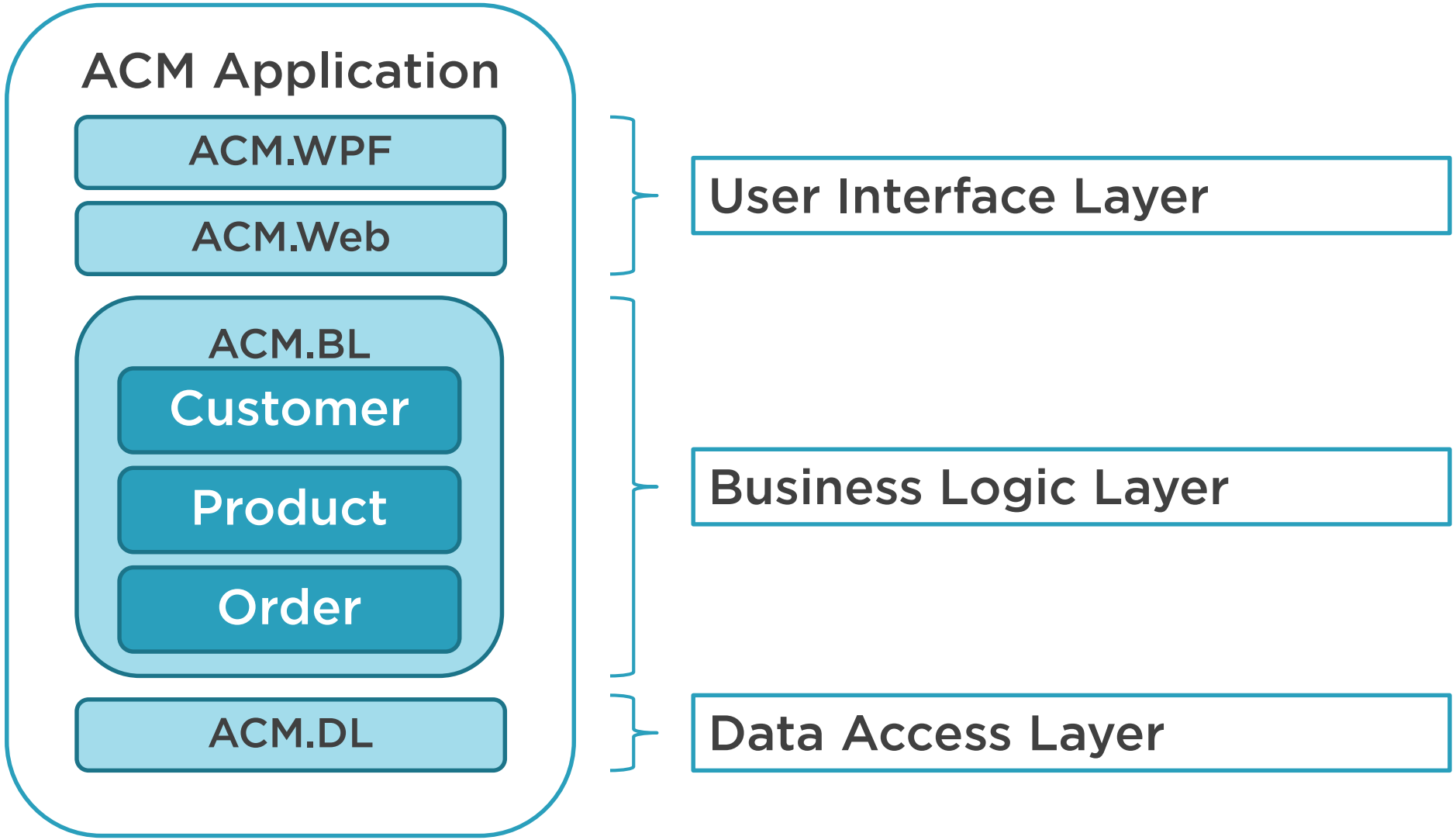
Building Reusable Components

Deborah Kurata

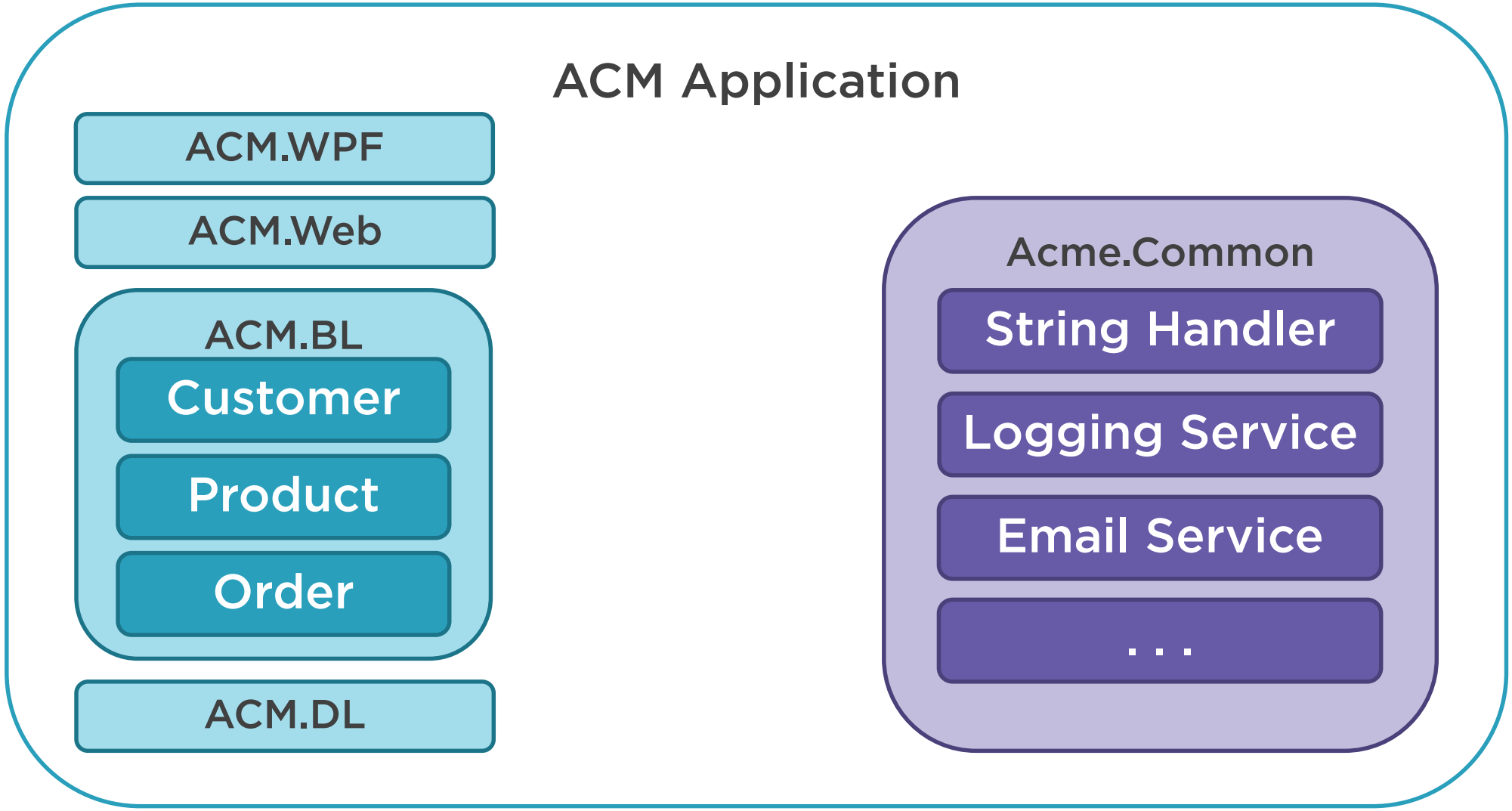
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

Components


ACM Application
ACM.WPF
ACM.Web
ACM.BL
Customer
Product
Order
ACM.DL
User Interface Layer
Business Logic Layer
Data Access Layer

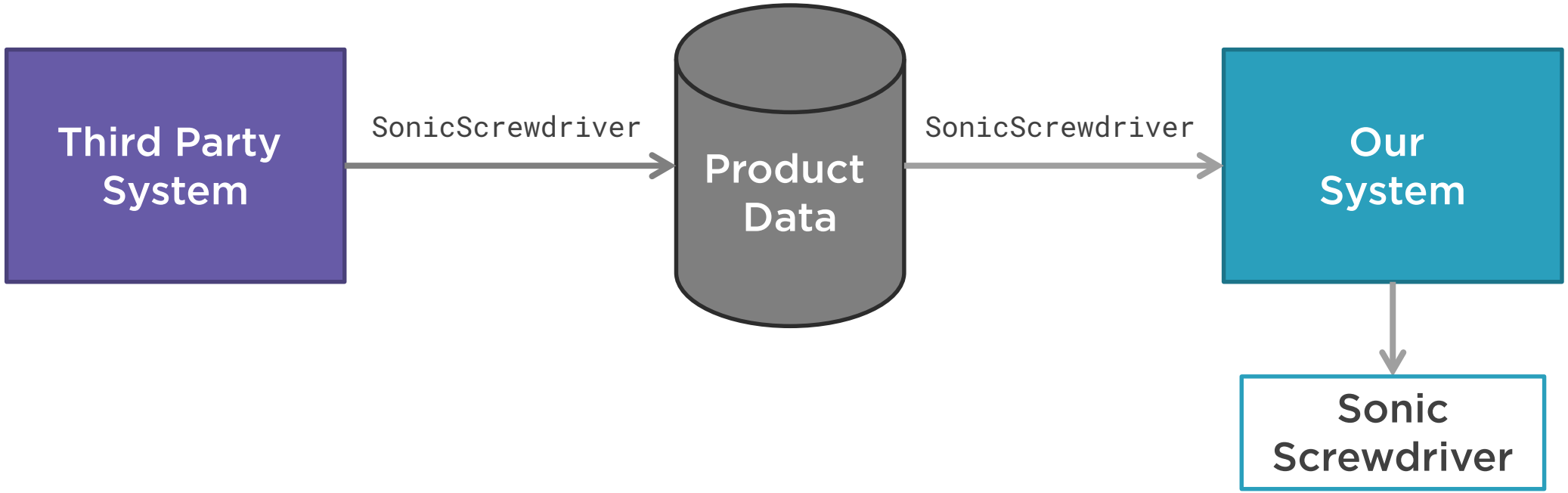
Reusable Library Component



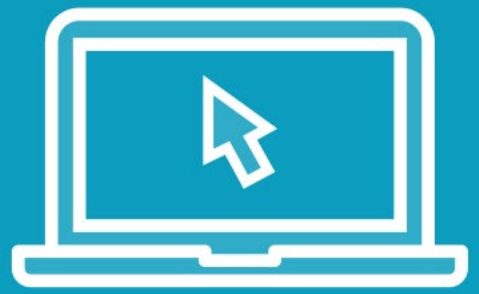
Module Outline

- Building a reusable component
- Testing a reusable component
- Using the reusable component
- Static classes
- Extension methods

Scenario

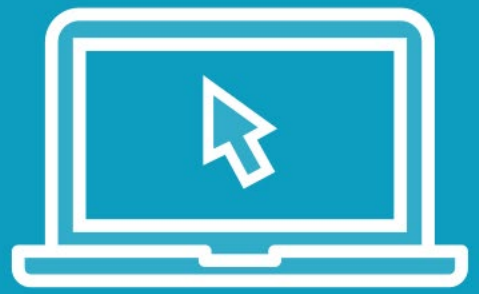

Third Party System
SonicScrewdriver
Product Data
SonicScrewdriver
Our System
Sonic Screwdriver

Demo



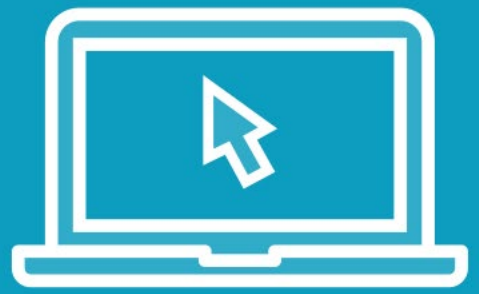
Building a reusable component

Demo



Testing a reusable component

Demo



Using a reusable component

Normal Class

Creating the method

```
public class StringHandler
{
  public string InsertSpaces(string source)
  { ...
  }
}
```

Using the method

```
var stringHandler = new StringHandler();

return stringHandler.InsertSpaces(productName);
```

Static Class

A class that cannot be instantiated

Access members using the class name

Members must also be static

Often used as a container for utility methods

Normal vs. Static Class

Creating the method

```
public class StringHandler
{
  public string InsertSpaces(string source)
  { ...
  }
}
```

Using the method

```
var stringHandler = new StringHandler();

return stringHandler.InsertSpaces(productName);
```

Creating the method

```
public static class StringHandler
{
  public static string InsertSpaces(string source)
  { ...
  }
}
```

Using the method

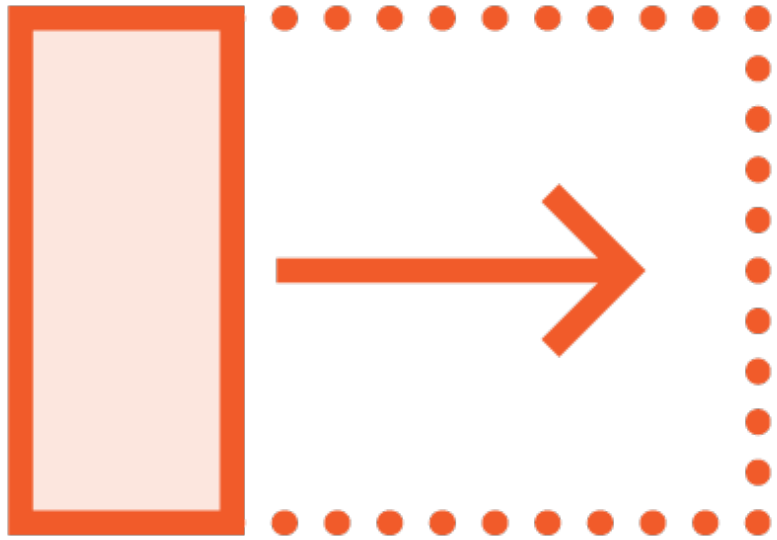
```
return StringHandler.InsertSpaces(productName);
```

Instances => Not Static

Joe Smith
Joe@aol.com
123 Main St.

Jessica Jones
Jessica@aol.com
123 First St.

Extension Method



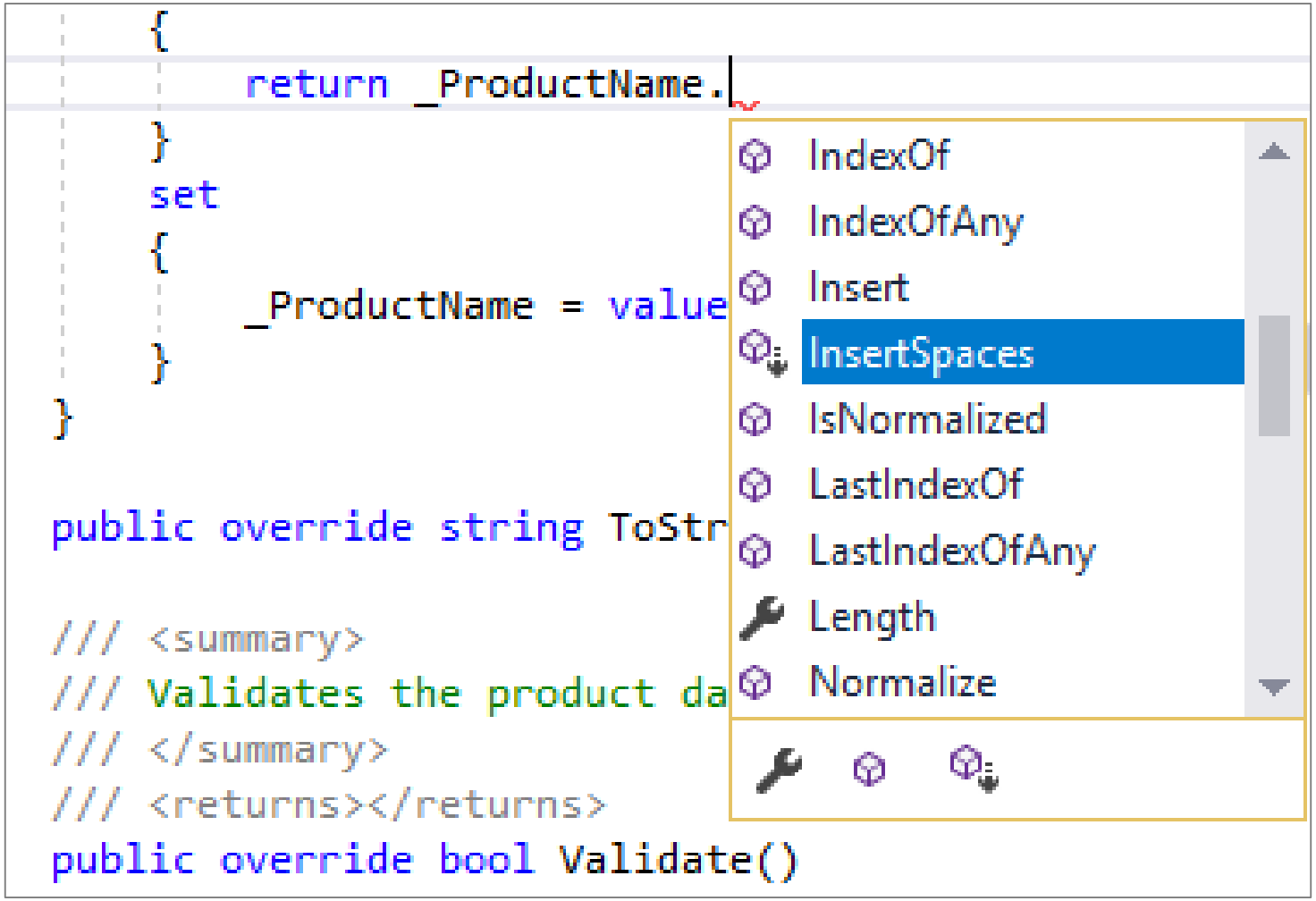
Add methods to an existing type without modifying the original type

Great for adding methods to .NET types

Extension methods appear in IntelliSense

Must be a static method in a static class

Extension Method



Extension Method

Creating the method

```
public static class StringHandler
{
  public static string InsertSpaces(string source)
  { ...
  }
}
```

Using the method

```
return StringHandler.InsertSpaces(productName);
```

Creating the method

```
public static class StringHandler
{
  public static string InsertSpaces(this string source)
  { ...
  }
}
```

Using the method

```
return _productName.InsertSpaces();
```

Static vs. Extension Method

```
public static string InsertSpaces(this string source)
```

Is the primary parameter an instance?

Does the method logically operate on that instance?

Is it desirable for the method to appear in IntelliSense for that type?

Build Reusable Components

Create a separate project for each reusable component

Build a library of general-purpose methods in the component, grouped into classes

Reuse the component

Static Class

Cannot be instantiated

Is sealed

Defined with the `static` keyword

```
public static class StringHandler
{

}
```

Use to organize utility methods

Static Method

Every method in a static class must be static

Define with the `static` keyword

```
public static string InsertSpaces(string source)
```

Access a static member with the class name

```
return StringHandler.InsertSpaces(productName);
```

Use to create reusable utility methods

Extension Method

Adds a method to an existing type without modifying the original type

Method must be static

Define by adding `this` to the first parameter

```
public static string InsertSpaces(this string source)
```

Access an extension method using the extended class instance

```
return _productName.InsertSpaces();
```

Object-Oriented Programming (OOP)