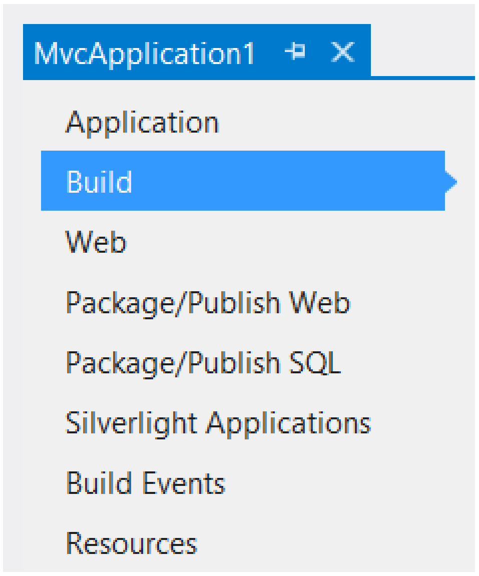
Adding API Documentation

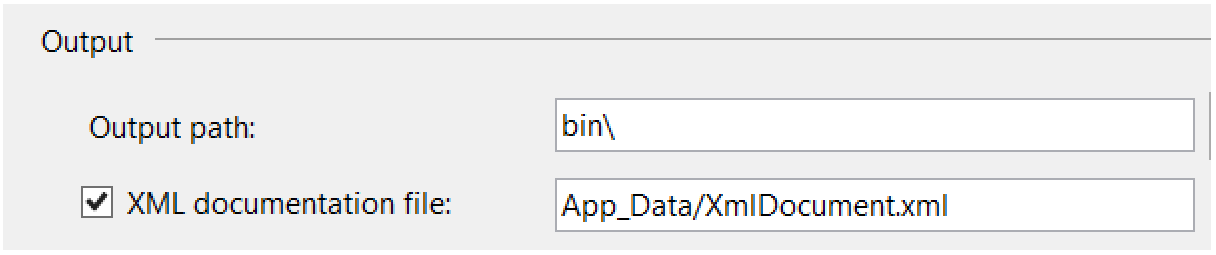
By default, the help pages have placeholder strings for documentation. You can use XML documentation comments to create the documentation. To enable this feature, open the file Areas/HelpPage/App_Start/HelpPageConfig.cs and uncomment the following line:

```
config.SetDocumentationProvider(new XmlDocumentationProvider(
    HttpContext.Current.Server.MapPath("~/App_Data/XmlDocument.xml")));
```

Now enable XML documentation. In Solution Explorer, right-click the project and select **Properties**. Select the **Build** page.



Under **Output**, check **XML documentation file**. In the edit box, type "App_Data/XmlDocument.xml".



Output path: bin\

☑ XML documentation file: App_Data/XmlDocument.xml

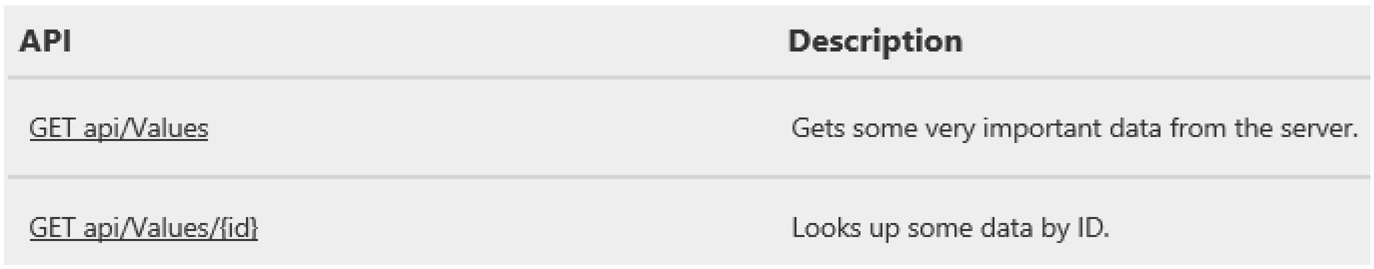
Next, open the code for the `ValuesController` API controller, which is defined in /Controllers/ValuesControler.cs. Add some documentation comments to the controller methods. For example:

```
/// <summary>
/// Gets some very important data from the server.
/// </summary>
public IEnumerable<string> Get()
{
    return new string[] { "value1", "value2" };
}

/// <summary>
/// Looks up some data by ID.
/// </summary>
/// <param name="id">The ID of the data.</param>
public string Get(int id)
{
    return "value";
}
```

Tip: If you position the caret on the line above the method and type three forward slashes, Visual Studio automatically inserts the XML elements. Then you can fill in the blanks.

Now build and run the application again, and navigate to the help pages. The documentation strings should appear in the API table.



API	Description
GET api/Values	Gets some very important data from the server.
GET api/Values/{id}	Looks up some data by ID.

The help page reads the strings from the XML file at run time. (When you deploy the application, make sure to deploy the XML file.)

Under the Hood

The help pages are built on top of the **ApiExplorer** class, which is part of the Web API framework. The **ApiExplorer** class provides provides the raw material for creating a help page. For each API, **ApiExplorer** contains an **ApiDescription** that describes the API. For this purpose, an "API" is defined as the combination of HTTP method and relative URI. For example, here are some distinct APIs:

- GET /api/Products
- GET /api/Products/{id}
- POST /api/Products

If a controller action supports multiple HTTP methods, the **ApiExplorer** treats each method as a distinct API.

To hide an API from the **ApiExplorer**, add the **ApiExplorerSettings** attribute to the action and set *IgnoreApi* to true.

```
[ApiExplorerSettings(IgnoreApi=true)]
public HttpResponseMessage Get(int id) {  }
```

You can also add this attribute to the controller, to exclude the entire controller.

The ApiExplorer class gets documentation strings from the **IDocumentationProvider** interface. As you saw earlier, the Help Pages library provides an **IDocumentationProvider** that gets documentation from XML documentation strings.

The code is located in /Areas/HelpPage/XmlDocumentationProvider.cs. You can get documentation from another source by writing your own **IDocumentationProvider**.

To wire it up, call the **SetDocumentationProvider** extension method, defined in **HelpPageConfigurationExtensions**
ApiExplorer automatically calls into the **IDocumentationProvider** interface to get documentation strings for each API. It stores them in the **Documentation** property of the **ApiDescription** and **ApiParameterDescription** objects.