Walkthrough 13 - Using a RESTful Web Service

Setup

This walkthrough will create an MVC application that will consume a RESTful web service.

1. Start Visual Studio.
2. Click Create a new project.
3. Set language to **C#** and project type to **Web**.
4. Select the ASP.NET Core Web App (Model-View-Controller) template, click Next.
5. Set Project name to **Blog**.
6. Set Location to a **folder of your choosing**.
7. Ensure Place solution and project in the same directory is not selected, click Next.
8. Set version to **.NET 5.0**, unselect Configure for HTTPS, click Create.

Post.cs

1. Navigate to https://jsonplaceholder.typicode.com/ (https://jsonplaceholder.typicode.com/) . We will use this site to generate our test data. Click the **/posts** link and inspect the data.
2. Change the URL to **https://jsonplaceholder.typicode.com/posts/1** to look at a single post.
3. Select the JSON and click Ctrl+C to copy it.
4. Right-click the Models folder and add a Class named **Post.cs**.
5. Delete the Post class.

6.
   ```
namespace Blog.Models
{
    ~~public class Post~~
    ~~{~~
    ~~}~~
}
```
7. Place the cursor before the closing brace of the namespace.
8. From the Edit menu, select Paste Special / Paste JSON As Classes.

9.
   ```
namespace Blog.Models
{
    public class Rootobject
    {
        public int userId { get; set; }
        public int id { get; set; }
        public string title { get; set; }
        public string body { get; set; }
    }
}
```
10. Change the name of the class to Post.

11.
    ```
public class ~~Rootobject~~Post
{
    public int userId { get; set; }
    public int id { get; set; }
    public string title { get; set; }
    public string body { get; set; }
}
```
12. Save the file.

HomeController.cs

1. Update the Index method to be asynchronous.

2.
   ```
public async Task<IActionResult> Index()
{
    return View();
}
```
3. Add a try/catch.

```
public async Task<IActionResult> Index()
{
    try
    {

        return View();
    }
    catch (Exception)
    {

        throw;
    }
}
```

1. Add the **using System.Net.Http;** and **using System.Net.Http.Headers;** directives. Instantiate a list of posts and set up a call to the web server.
2.
```
public async Task<IActionResult> Index()
{
    try
    {
        var posts = new List<Post>();

        using (var client = new HttpClient())
        {
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
        }

        return View();
    }
    catch (Exception)
    {

        throw;
    }
}
```
3. Call the web server and get the JSON.
4.
```
public async Task<IActionResult> Index()
{
    try
    {
        var posts = new List<Post>();

        using (var client = new HttpClient())
        {
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
            var response = await client.GetAsync("https://jsonplaceholder.typicode.com/posts");

            if (response.IsSuccessStatusCode)
            {
                var json = await response.Content.ReadAsStringAsync();
            }
        }

        return View();
    }
    catch (Exception)
    {

        throw;
    }
}
```
5. Add a breakpoint to the assignment to the response variable (the web service call).
6. Press F5 to build and run in debug mode. Press F10 to step until after the json variable is assigned.
7. Hover over the json variable and click the magnifying glass to see a text representation of the JSON. Close the dialog.
8. Hover again, but click the drop-down to the right of the magnifying glass and select JSON Visualizer. Close the dialog.
9. Stop the debugger and remove the breakpoint.
10. Add the **using System.Text.Json;** directive, parse the JSON into the list of posts and return them in the view.
11.
```
public async Task<IActionResult> Index()
{
    try
    {
        var posts = new List<Post>();

        using (var client = new HttpClient())
        {
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
            var response = await client.GetAsync("https://jsonplaceholder.typicode.com/posts");

            if (response.IsSuccessStatusCode)
            {
                var json = await response.Content.ReadAsStringAsync();
                posts = JsonSerializer.Deserialize<List<Post>>(json);
            }
        }

        return View(posts);
    }
    catch (Exception)
    {

        throw;
    }
}
```
12. Handle the exception.

```
public async Task<IActionResult> Index()
{
    try
    {
        var posts = new List<Post>();

        using (var client = new HttpClient())
        {
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
            var response = await client.GetAsync("https://jsonplaceholder.typicode.org/posts");

            if (response.IsSuccessStatusCode)
            {
                var json = await response.Content.ReadAsStringAsync();
                posts = JsonSerializer.Deserialize<List<Post>>(json);
            }
        }

        return View(posts);
    }
    catch (Exception ex)
    {
        return View("Error", new ErrorViewModel { RequestId = ex.Message });
        ~~throw;~~
    }
}
```

1. Right-click in the Index method and select Add View..., select Razor View, click Add.
2. Set View name to **Index**, Template to **List**, Model class to **Post (Blog.Models)**, click Add, click Yes to replace the existing view.
3. Press Ctrl+F5 to run, all 100 posts are displayed.
4. Test the error handling. Change the URL to a site that doesn't exist.

```
...
var response = await client.GetAsync("https://jsonplaceholder.typicode.~~com~~org/posts");
...
```

1. Save the file, refresh the browser to see the error.
2. Restore the URL.

```
...
var response = await client.GetAsync("https://jsonplaceholder.typicode.~~org~~com/posts");
...
```

1. Save the file, refresh the browser to see the data again.
2. To return only the first 10 records, update the return statement.
3.
```
...
return View(posts.Take(10));
...
```
4. Save the file. Refresh the page to see only 10 posts.
5. To skip the first 25 posts, update the return statement.
6.
```
...
return View(posts.Skip(25).Take(10));
...
```
7. Save the file. Refresh the page to see a different set of 10 posts.
8. Navigate to the documentation of the web service at **https://github.com/typicode/json-server#slice** **(https://github.com/typicode/json-server#slice)** to see that we don't have to get all 100 posts, but that we can retrieve a subset.
9. Update the request to only return a subset of posts. Update the return statement to return all posts in the list.
10.
```
...
var response = await client.GetAsync("https://jsonplaceholder.typicode.com/posts?_start=35&_end=41");
...
return View(posts~~.Skip(25).Take(10)~~);
...
```
11. Save the file. Refresh the page to see the specified posts.
12. Update the request to sort the posts by title.
13.
```
...
var response = await client.GetAsync("https://jsonplaceholder.typicode.com/posts?_start=35&_end=41&_sort=title");
...
```
14. Save the file. Refresh the page to see the posts sorted by title.