FHV
Vorarlberg University
of Applied Sciences

Application Integration and Security

Valmir Bekiri
Philipp Scambor

HTTP clients in .NET 8

Creating an API client with .NET 8

- If you want to consume an HTTP API then your .NET application becomes the client
- To send a HTTP request with `System.Net.Http.HttpClient` you need
  - To call the right method depending on the type of request you want to send (`PostAsync()`, `GetAsync()`, …)
  - The request URI (string) containing query parameters if necessary
  - The request content (for POST, PUT, etc.)
  - Handle exceptions because network calls are always prone to failure
- Request content and response type depend on the type of API you aim to use
  - Can anything: XML, JSON, Raw text, etc.

Consuming a JSON API in .NET

- Use `System.Text.Json.JsonSerializer` to serialise to JSON or to parse JSON response bodies.
  - Serialise a class to a JSON string: `JsonSerializer.Serialize(someObject)`
  - Parse JSON: `JsonSerializer.Deserialize<T>(jsonString)`
    - Type T is returned and mirrors the JSON structure – if the JSON and the type do not match together deserialization fails!
- Tip: You can automatically generate a class from an example JSON with your IDE
  - Create a new empty class file
  - Visual Studio: Edit > Paste Special > Paste Json as classes
  - Rider: Edit > Paste > Paste Special: Json as classes

Consuming a JSON API in .NET

- Example on the right:
  - Sending a HTTP Post with JSON content
  - Parsing the response JSON into a class
  - Using async programming as network calls might take a while

```csharp
using System.Text;
using System.Text.Json;
..

class APIClient
{

  public async Task<string?> callAPI()
  {
      var requestUri = "https://example.com/some/api";
      var requestBody = new { some = "test" };
      var jsonBody = JsonSerializer.Serialize(requestBody);
      var httpContent = new StringContent(
        jsonBody, Encoding.UTF8, "application/json");

      try
      {
          var httpClient = new HttpClient();
          var response = await httpClient.PostAsync(requestUri, httpContent);

          // throws an exception if not success status code
          response.EnsureSuccessStatusCode();

          string jsonResponse = await response.Content.ReadAsStringAsync();
          var responseObject = JsonSerializer
            .Deserialize<ExampleResponse>(jsonResponse);

          return responseObject?.example;
      }
      catch (Exception ex)
      {
          // handle exception that may occur
      }
  }

  private class ExampleResponse
  {
    public string example { get; set; }
  }
}

```

That‘s it ☺
…for now