VIA University College

DNP-S18

Consuming Web Services

How to issue HTTP requests to REST services

Web Services and REST

What is a Web API and why use it?

Asynchronous Programming

A look at asynchronous programming and the Task-based Asynchronous Pattern in .NET

Creating a REST Client

How to build a client for a web service using asynchronous programming

Exercises

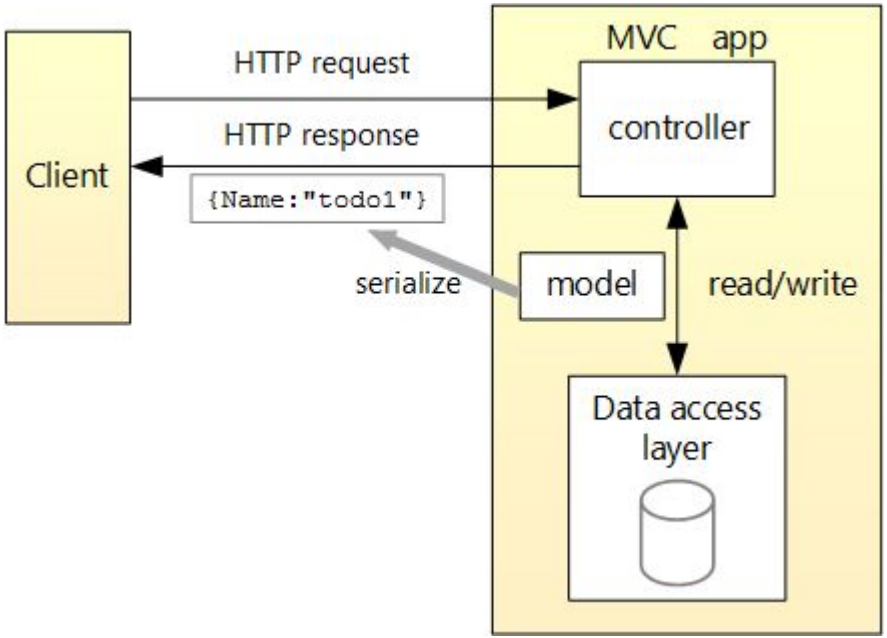
Create your own REST client!

What is a Web API?

A programmatic interface comprising a number of publicly exposed HTTP endpoints to a defined request-response message system, typically expressed in JSON

"**A**pplication **P**rogramming **I**nterface"

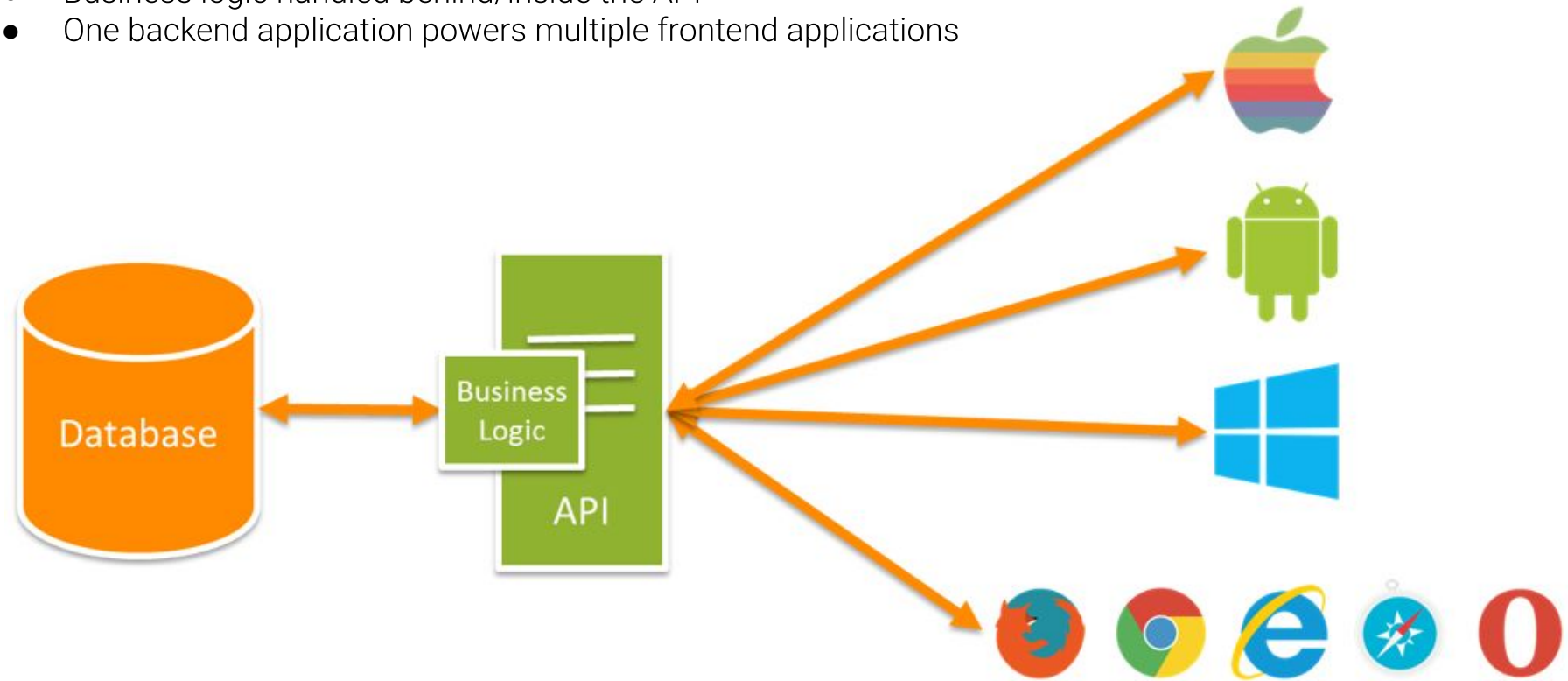
- An HTTP service
- Logic or data accessible over HTTP
- Used programmatically
- Accessible across the internet
- Also referred to as a "REST API"



The vast majority of APIs on the Internet are RESTful

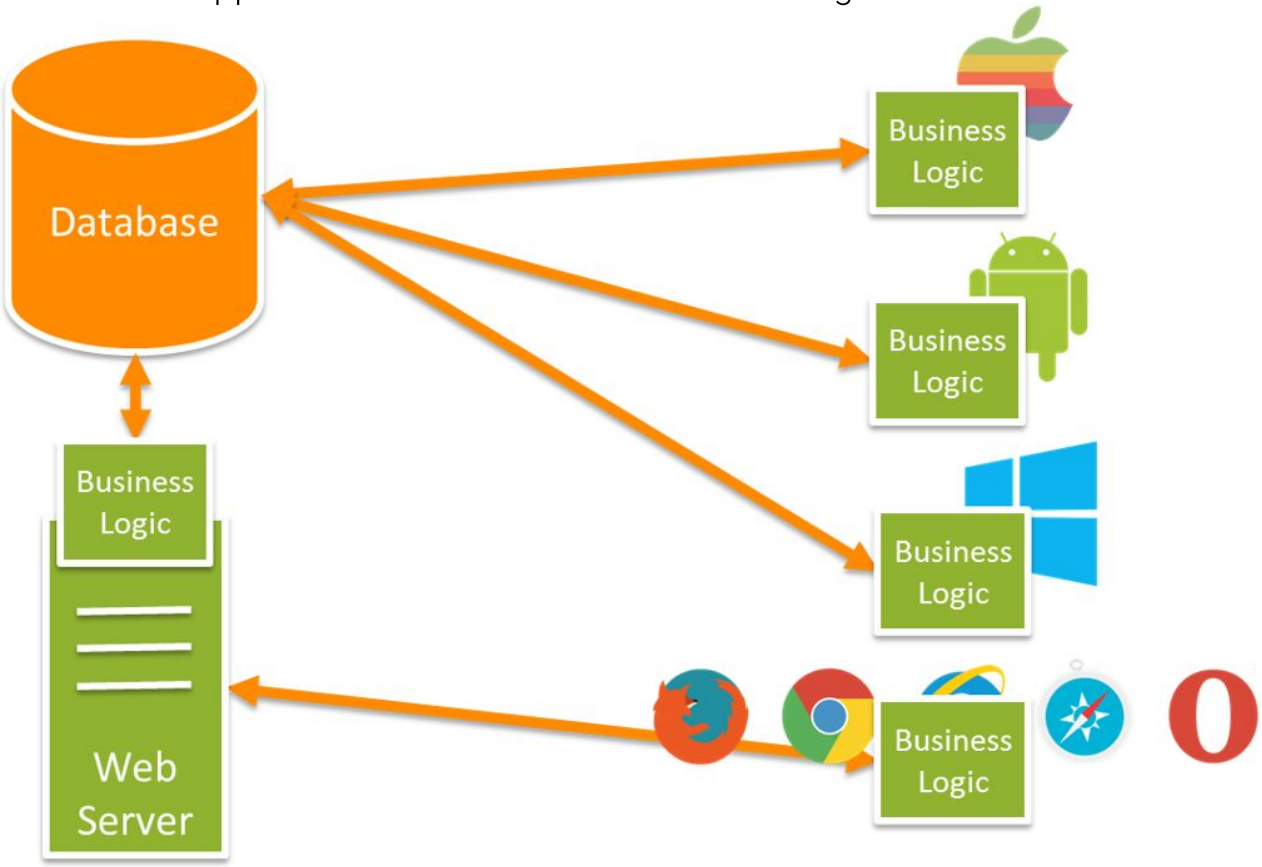
Why Create a Web API?

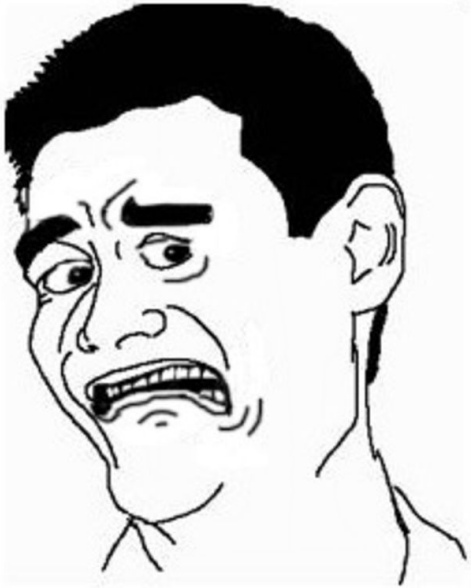
- Standard way of interacting with the business logic of your application
- Business logic handled behind/inside the API
- One backend application powers multiple frontend applications



Without an API

Each client app has it's own embedded business logic

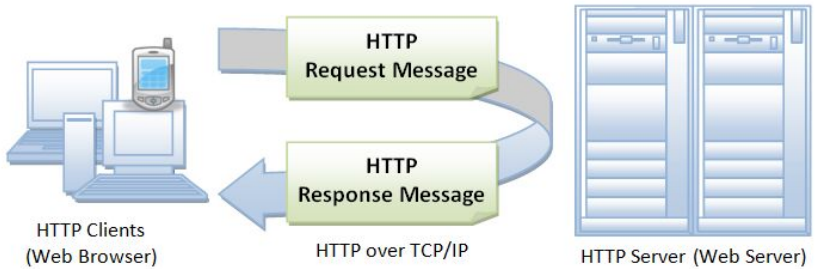




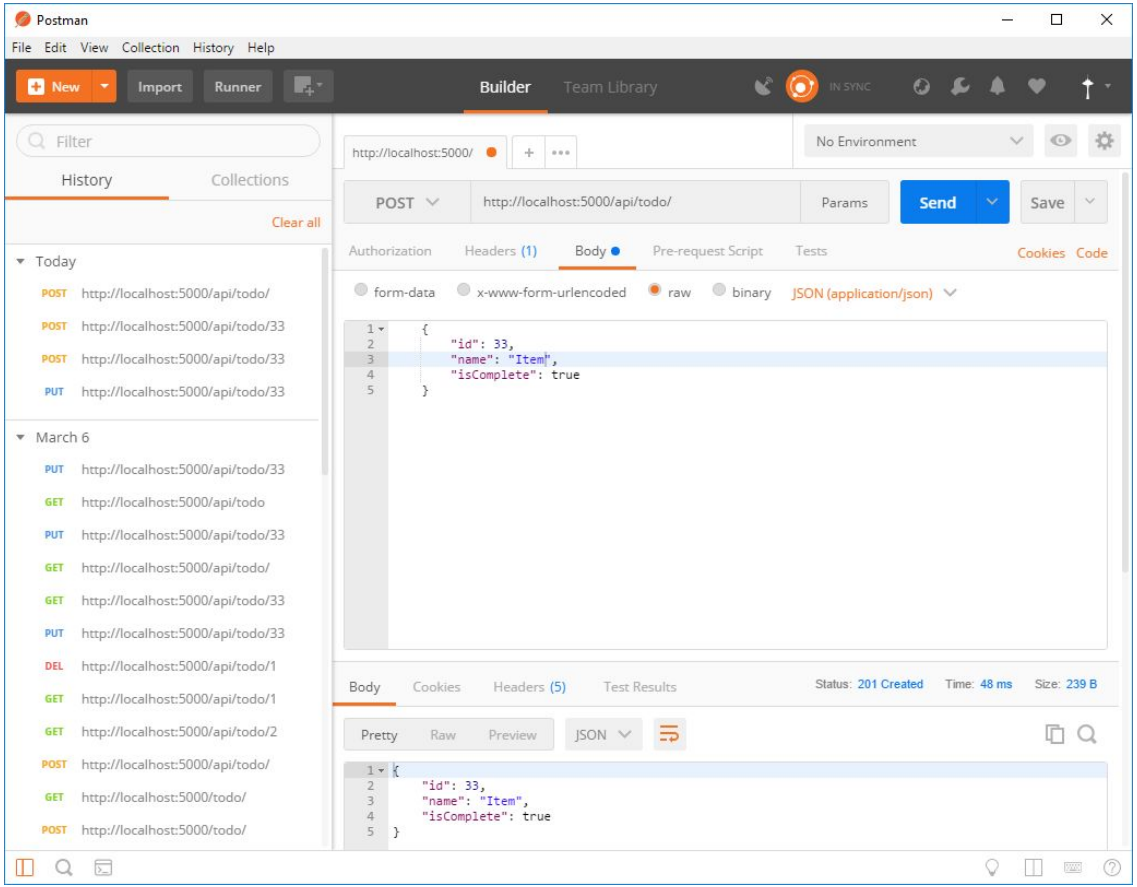
The HTTP Protocol

- **Application layer** protocol → implementation is **used**, not abstracted away
- Used to access **resources** identified by a **URI** (endpoint)
  - e.g. http://example.com/todolist?priority=important
- Access resources using standardized methods/verbs
  - **GET**, **POST**, **PUT**, **DELETE**, etc.
- **Stateless** request-reply protocol
- **Headers** - metadata about a body
- **Bodies** - resources can be **any format**!
- **Status codes** indicate the type of response
  - 200 OK, 400 Bad Request, 500 Internal Server Error

enables **CRUD** operations



A Test Client for a Web Service

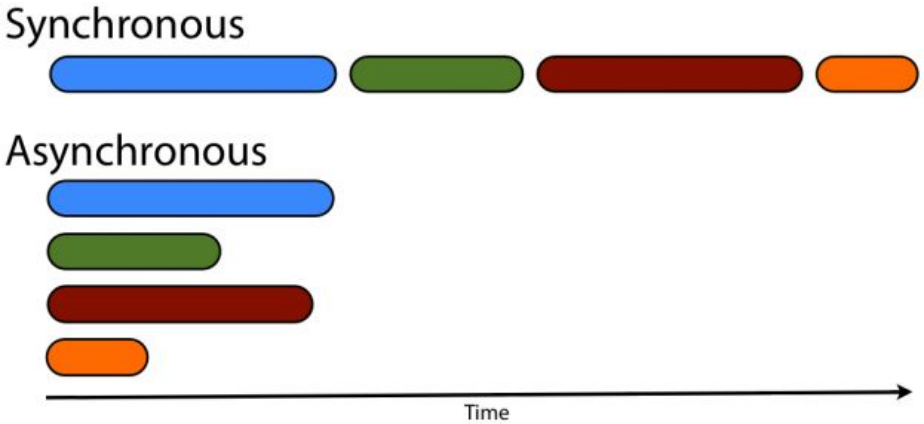


Postman used as a **client** to test a web service

Asynchronous Programming

- **Synchronous** programming execution
  - Program is executed **line by line**, one at a time
  - When a function is called, program execution has to **wait** until the function returns

- **Asynchronous** programming execution
  - A means of parallel programming
  - When a function is called, program execution **continues** to the next line, **without** waiting for the function to complete



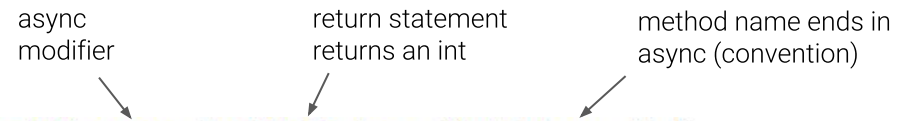
When to use?

- Accessing the web (e.g. **a Web API ;)**)
- Working with files and databases
- Working with images
- CPU-bound code (e.g. expensive calculations)

Asynchronous Programming in .NET

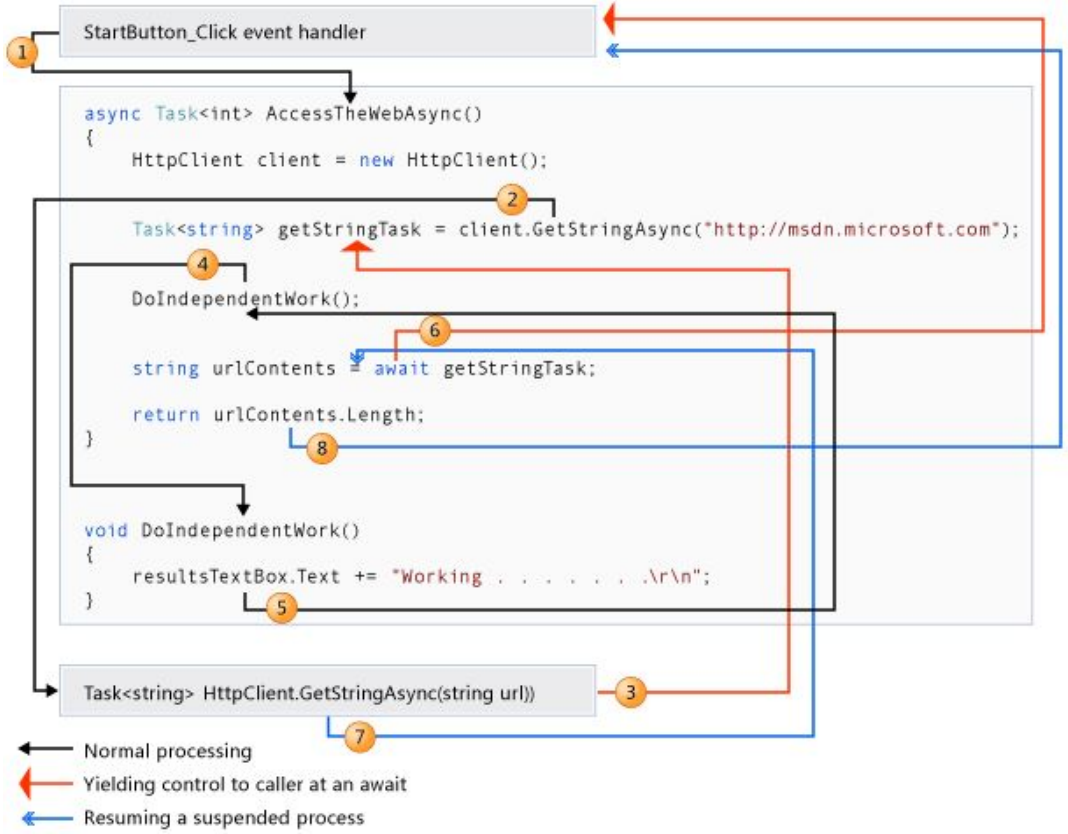
- Task-based Asynchronous Pattern (TAP)
- Task - class representing **ongoing work**
- Async - modifier **specifying that a method is asynchronous**
  - Async methods must return Task, Task<T> or void
  - Async methods normally use await at least once in body
  - Async methods are “resumed” when an “awaited” operation completes
- Await - operator applied to a task in an async method to insert a **suspension point** in the execution of the method until the awaited task completes
  - Returns control immediately to the caller
- We don’t have to juggle callbacks \o/

Asynchronous Programming Example




```
public async Task<int> AccessTheWebAsync () {

    HttpClient client = new HttpClient ();

    // When you await the task you'll get a string (urlContents).
    Task<string> getStringTask = client.GetStringAsync ("http://via.dk");

    // You can do work here that doesn't rely on the string from GetStringAsync.
    DoIndependentWork ();

    // The await operator suspends AccessTheWebAsync.
    //  - AccessTheWebAsync can't continue until getStringTask is complete.
    //  - Meanwhile, control returns to the caller of AccessTheWebAsync.
    //  - Control resumes here when getStringTask is complete.
    //  - The await operator then retrieves the string result from getStringTask.
    string urlContents = await getStringTask;

    // Any methods that are awaiting AccessTheWebAsync retrieve the length value.
    return urlContents.Length;
}
```

Asynchronous Programming Example



Asynchronous Programming Example

If no work needs to be done between calling GetStringAsync and awaiting its completion, you can simplify the code by calling and awaiting in the following single statement.

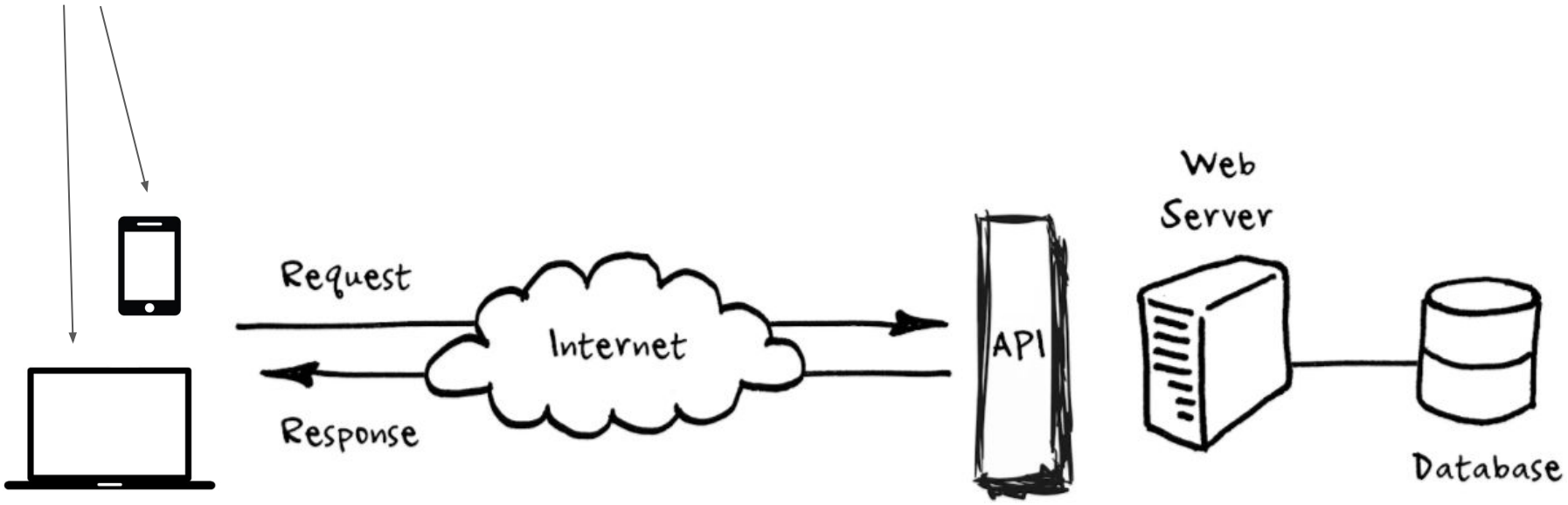
```
public async Task<int> AccessTheWebAsync () {

    HttpClient client = new HttpClient ();

    string urlContents = await client.GetStringAsync ("http://via.dk");

    return urlContents.Length;
}
```

For each blocking method in .NET, an asynchronous version exists

Using Async/await

A Client for the API

The client is whatever consumes the API



Creating a REST Client

A basic console application...

```
class Program
{
    static void Main(string[] args)
    {
        // This is a blocking operation
        // We prevent the program from terminating until we receive a result
        string s = GetData().GetAwaiter().GetResult();
        System.Console.WriteLine(s);
    }

    static async Task<string> GetData()
    {
        HttpClient client = new HttpClient();

        System.Console.WriteLine("Fetching data...");

        var str = await client.GetStringAsync("http://localhost:5000/api/");

        return str;
    }
}
```

Send a GET request to the specified Uri and return the response body as a string

HTTP Requests

How to perform a HTTP request using HttpClient

```
static async Task<string> PostData()
{
    var client = new HttpClient();

    StringContent httpContent = new StringContent(
        "{'id': 2,'name': 'Clean Room','isComplete': true}",
        Encoding.UTF8,
        "application/json"
    );

    HttpResponseMessage response = await client.PostAsync(
        "http://localhost:5000/api/todo",
        httpContent
    );

    return response.ToString();
}
```

Derived from abstract class HttpContent

GetAsync
PostAsync
PutAsync
DeleteAsync

HttpContent HttpClient

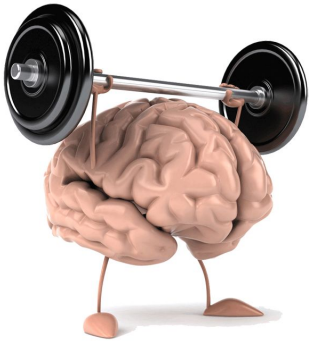
Extending HttpClient

Adding package **`Microsoft.AspNet.WebApi.Client`** allows us to skip HttpContent altogether, providing more methods on HttpClient

```
var response = await client.PostAsJsonAsync("http://localhost:5000/api/todo", new TodoItem
{
    Id = 42,
    Name = "Do DNP Exercises",
    IsComplete = false
});
```

```
dotnet add package Microsoft.Asp.Net.WebApi.Client
```

Exercises



REST Client tutorial