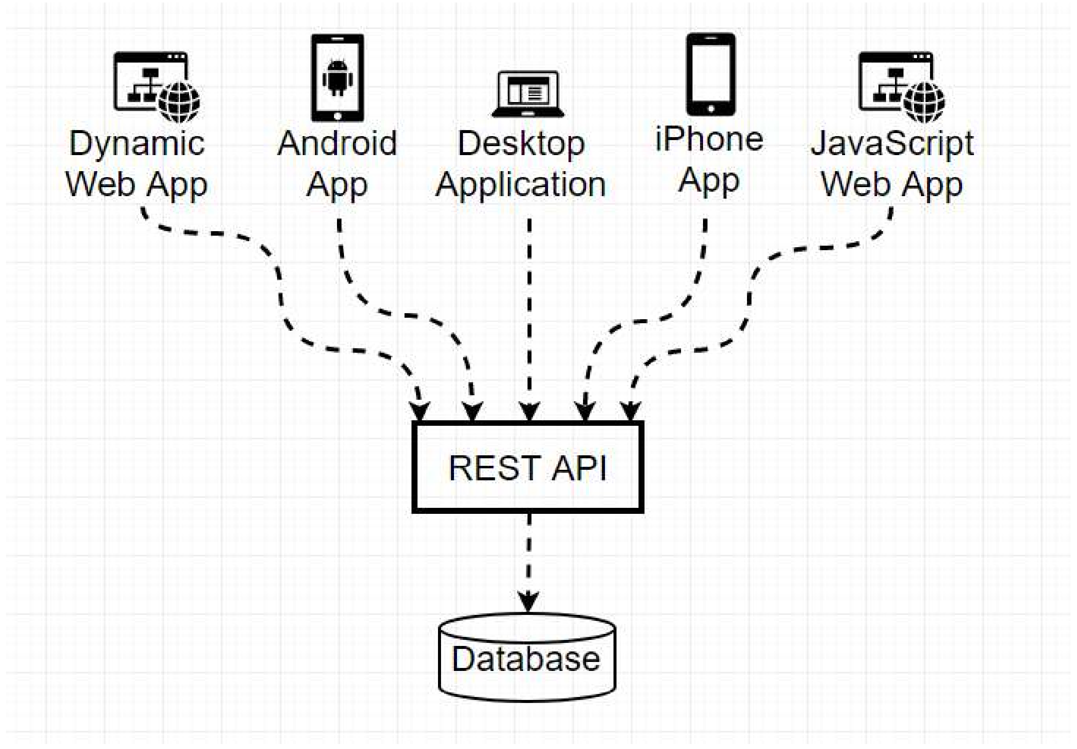
ASP.Net Core Web API

Building HTTP based Web Services

Mohamed ELshafei


Dynamic Web App
Android App
Desktop Application
iPhone App
JavaScript Web App
REST API
Database

HELLO!

What is API ?

- **Application Programing Interface.**
- API is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.
- Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc

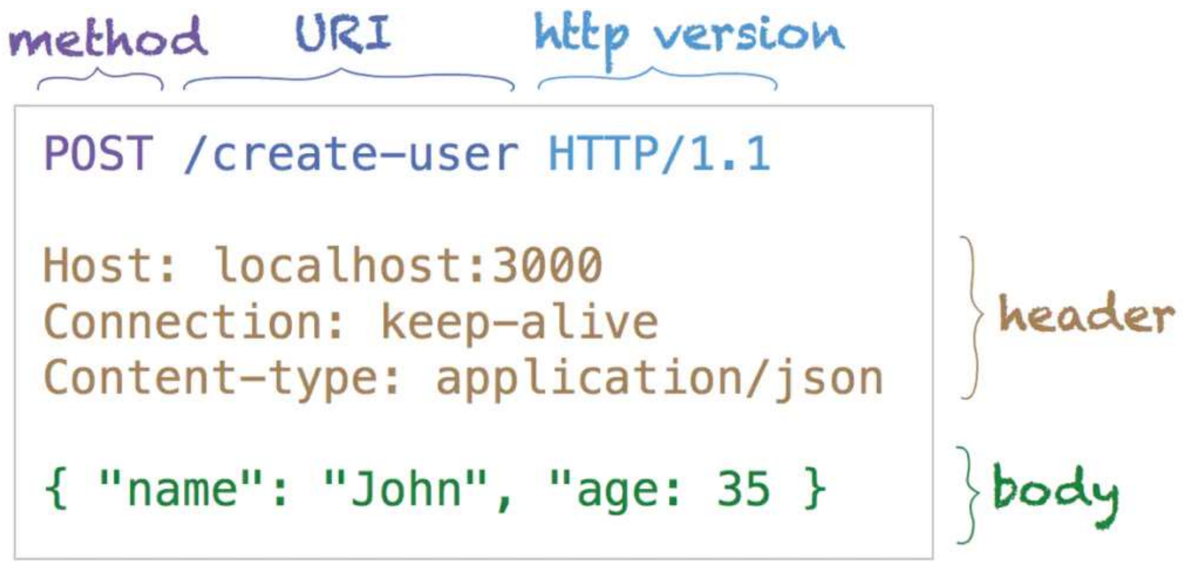
ASP.NET Web API

- **A framework** that makes it **easy** to build **HTTP services** that reach a **broad range of clients**, including browsers and mobile devices.
- **Clients:**

ASP.NET Web API Characteristics

- ASP.NET Web API is an ideal platform for building RESTful services.
- ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
- ASP.NET Web API maps HTTP verbs to method names.
- Support Data Format : JSON, XML,BSON and Custom.
- Support all clients are based on HTTP.
- Combine ASP.NET.

HTTP Request



```
POST /create-user HTTP/1.1

Host: localhost:3000
Connection: keep-alive
Content-type: application/json

{ "name": "John", "age: 35 }
```

HTTP Method

Method	Objection
GET	Retrieves information from a resource. return 200 (OK)
POST	Requests the server to create new one of entity without duplicate object. Return code 201 (Created)
PUT	Requests the server to replace the state of the target resource at the specified URI with the enclosed entity. If an existing resource exists for the current representation, it should return a 200(OK) ,204 (No Content) ,201 (Created).
DELETE	Requests the server to remove the entity located at the specified URI. Return code 200(completed) or 204 (No Content).

HTTP Methods

Operation	HTTP Method
Create	POST
Read	GET
Update	PUT
Delete	DELETE

ASP.NET Web API

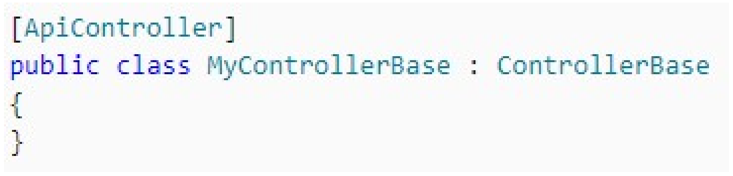
Uses common concepts from ASP.NET MVC

- Controllers
- Routing
- Model Binding
- Model Validation
- Security

ApiController attribute

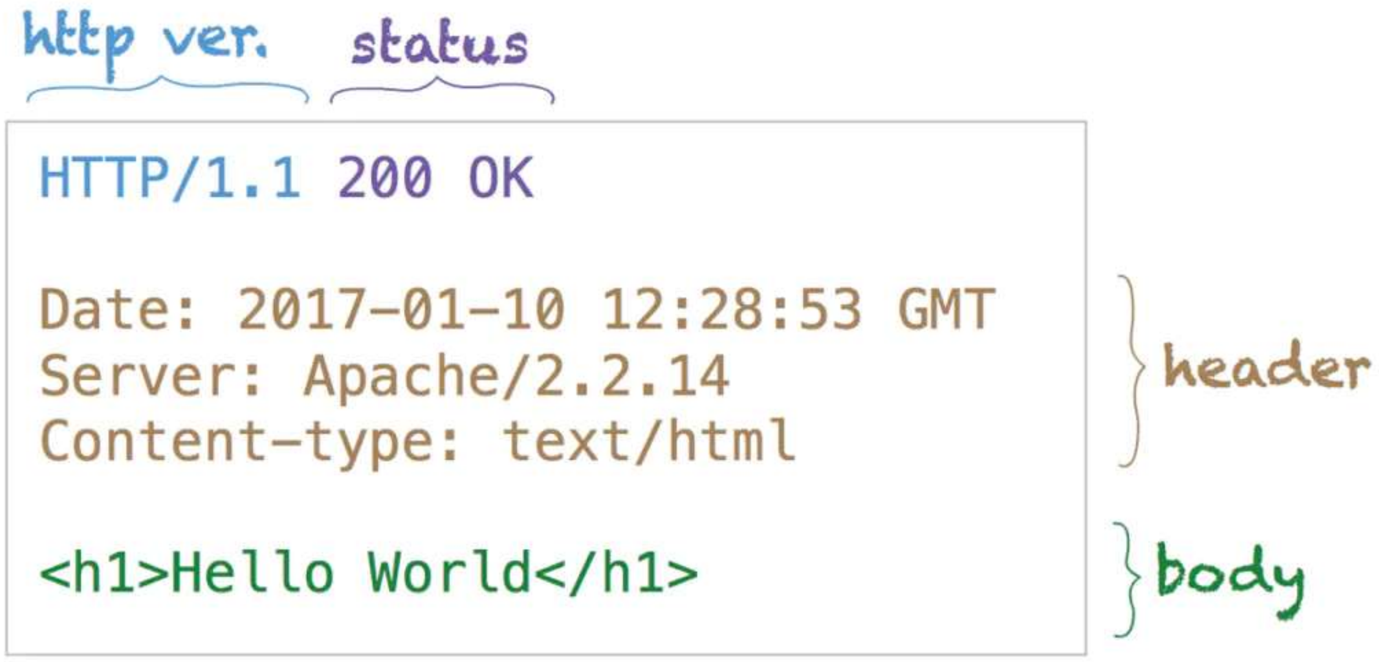
The [ApiController] attribute can be applied to a controller class to enable the following opinionated, API-specific behaviors:

- Attribute routing requirement
- Automatic HTTP 400 responses
- Binding source parameter inference
- Multipart/form-data request inference
- Problem details for error status codes



```
[ApiController]
public class MyControllerBase : ControllerBase
{
}
```

HTTP Response



```
HTTP/1.1 200 OK

Date: 2017-01-10 12:28:53 GMT
Server: Apache/2.2.14
Content-type: text/html

<h1>Hello World</h1>
```

HTTP Status Codes

Identify the type of request success and failures

- 2xx – Successful Response
  - 201 - Created
- 3xx – Redirect Response
  - 301 – Moved Permanently
- 4xx – Client Error Response
  - 401 – Unauthorized
- 5xx – Service Error Response
  - 503 – Service Unavailable

HTTP Status Codes

- ❑ 201 Created
- ❑ 200 Success - ok
- ❑ 204 Success but No Content
- ❑ 401 Not authorized
- ❑ 404 Does not exist
- ❑ 400 Bad Request
- ❑ 500 Server Error

Generate OpenAPI documents

- The OpenAPI specification is a programming language-agnostic standard for documenting HTTP APIs.
- This standard is supported in ASP.NET Core apps through a combination of built-in APIs and open-source libraries.
- There are three key aspects to OpenAPI integration in an application:
  - Generating information about the endpoints in the app.
  - Gathering the information into a format that matches the OpenAPI schema.
  - Exposing the generated OpenAPI document via a visual UI or a serialized file.

```
https://localhost:<port>/openapi/v1.json
```

Use Swagger UI for local testing

- By default, the Microsoft.AspNetCore.OpenApi package doesn't ship with built-in support for visualizing or interacting with the OpenAPI document. Popular tools for visualizing or interacting with the OpenAPI document include Swagger UI :
- Install ***Swashbuckle.AspNetCore.SwaggerUi*** package.
- Enable the swagger-ui middleware

```
app.UseSwaggerUI(options => { options.SwaggerEndpoint("/openapi/v1.json", "v1"); });
```

```
https://localhost:<port>/Swagger/
```

Validation Attributes

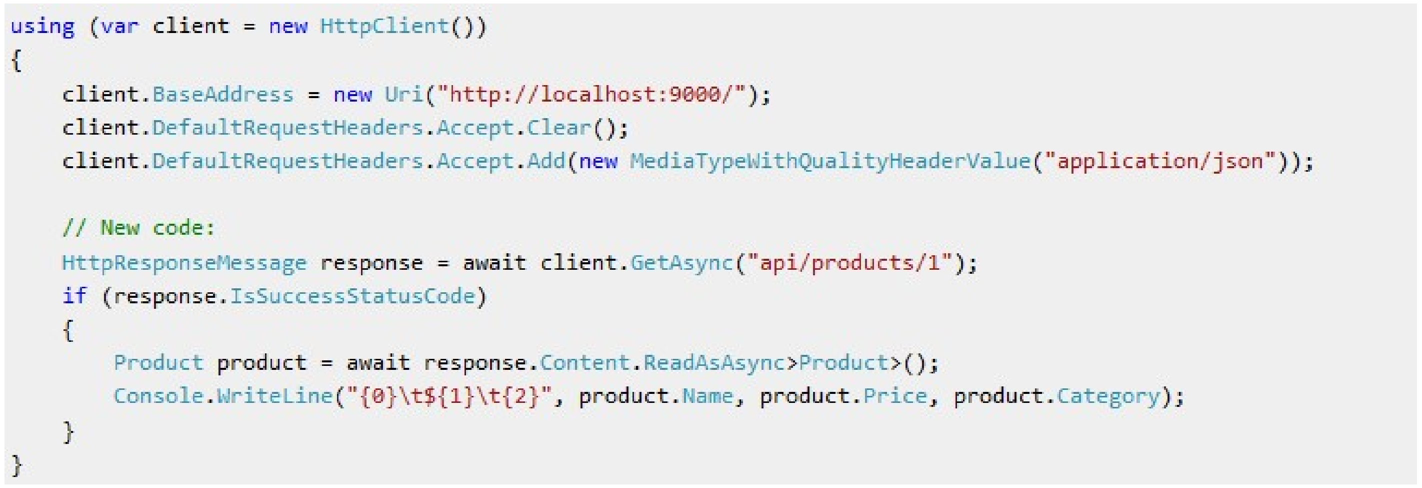
Metadata Validation Attributes:

- [Required]
- [Exclude]
- [DataType]
- [Range]
- [StringLength(60)]
- [RegularExpression]
- [AllowHtml]
- [Compare]

```
public partial class Employee
{
    internal sealed class EmployeeMetadata
    {
        [StringLength(60)]
        [RoundtripOriginal]
        public string AddressLine { get; set; }
    }
}
```

C# client

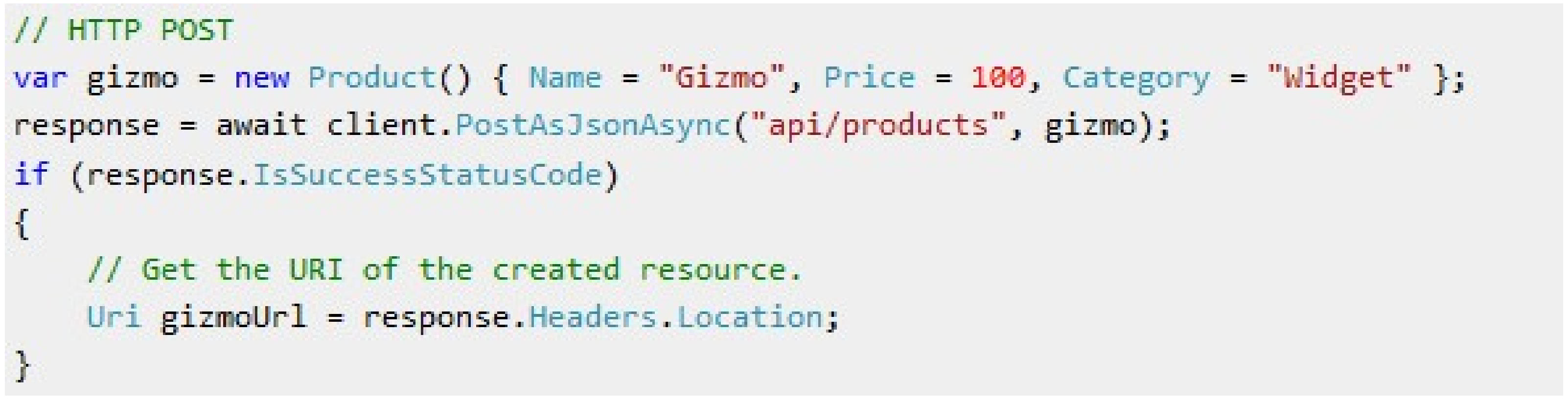
- Install-Package Microsoft.AspNet.WebApi.Client



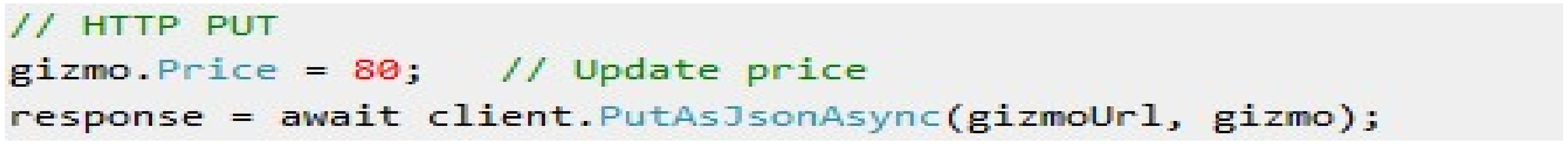
```
using (var client = new HttpClient())
{
    client.BaseAddress = new Uri("http://localhost:9000/");
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

    // New code:
    HttpResponseMessage response = await client.GetAsync("api/products/1");
    if (response.IsSuccessStatusCode)
    {
        Product product = await response.Content.ReadAsAsync>Product>();
        Console.WriteLine("{0}\t${1}\t{2}", product.Name, product.Price, product.Category);
    }
}
```

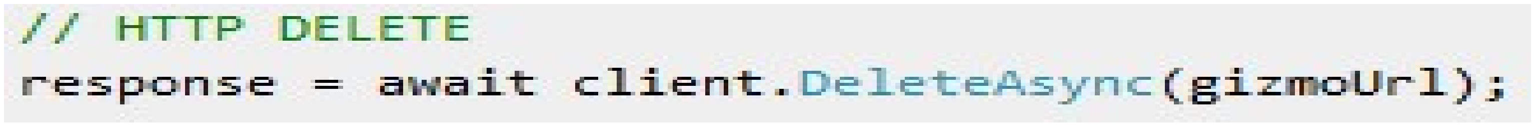
C# client



```
// HTTP POST
var gizmo = new Product() { Name = "Gizmo", Price = 100, Category = "Widget" };
response = await client.PostAsJsonAsync("api/products", gizmo);
if (response.IsSuccessStatusCode)
{
    // Get the URI of the created resource.
    Uri gizmoUrl = response.Headers.Location;
}
```



```
// HTTP PUT
gizmo.Price = 80;   // Update price
response = await client.PutAsJsonAsync(gizmoUrl, gizmo);
```



```
// HTTP DELETE
response = await client.DeleteAsync(gizmoUrl);
```

THANKS!

Any questions?