VIA University College

DNP-S18

Exposing Web Services

How to build web APIs

Quiz: “Consuming Web Services”

Test your knowledge on last week’s topic

Web API Design

Guidelines for designing a modern web API

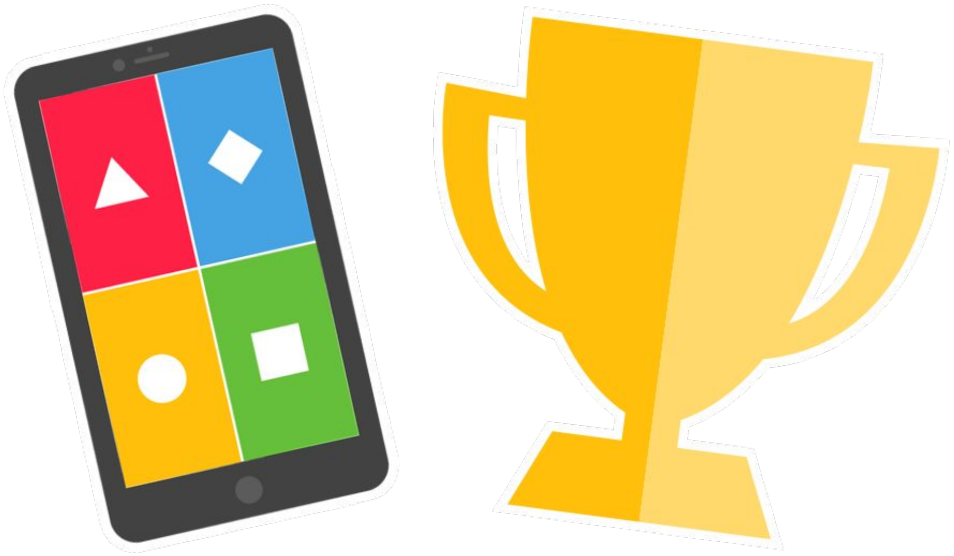
Implementing Web APIs with ASP.NET Core

How to create and use a web API

Exercises

Create your own Web API

Quiz: “Consuming Web Services”



https://kahoot.it/

NB! This quiz doesn’t really test your ability to implement software with .NET - getting a good score is in itself not proof that you have reached your learning goals.

Web API Design

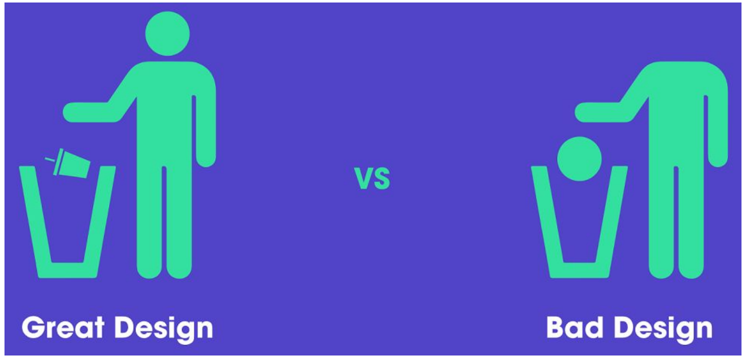
Organize the API around resources

- Base resource URIs on nouns and not verbs
  `https://adventure-works.com/orders // Good`
  `https://adventure-works.com/create-order // Avoid`

- Avoid creating APIs that simply mirror the internal structure of a database.

- A client should not be exposed to the internal implementation.

- Entities are often grouped together into collections
  `https://adventure-works.com/orders`



A well-designed web API should aim to support:

- **Platform independence**
- **Service evolution**

Define operations in terms of HTTP methods

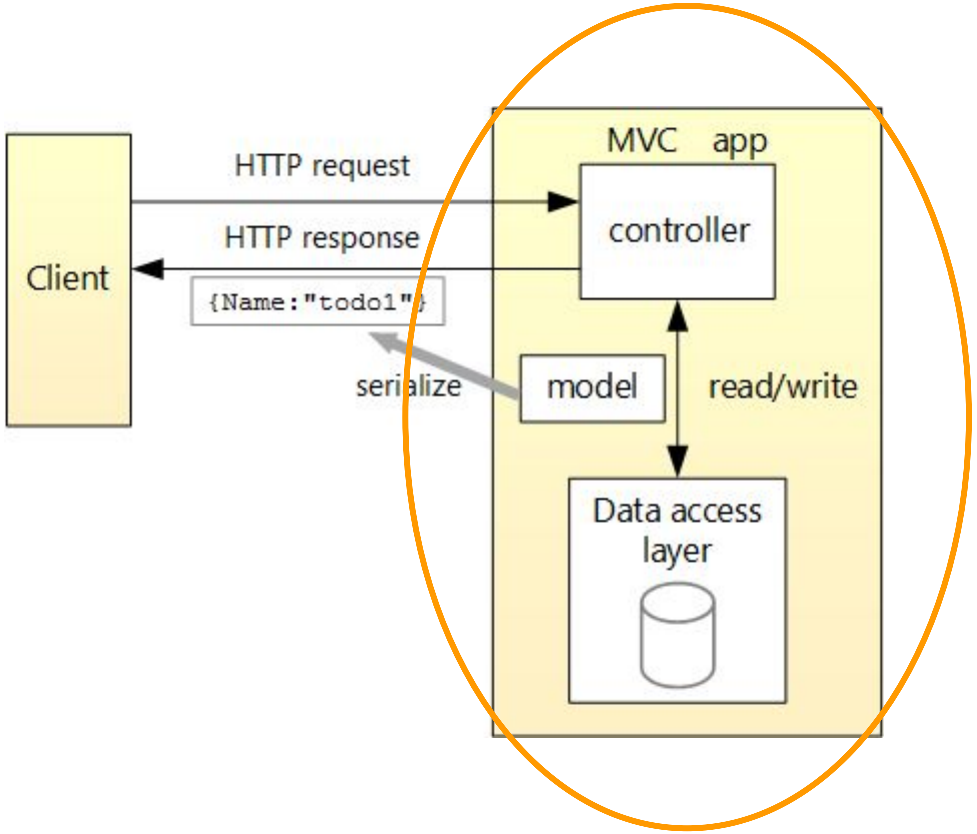
- Assign semantic meaning to a request using HTTP verbs
  I.e. GET, POST, PUT, PATCH, DELETE.

Conform to HTTP semantics

- Media types (MIME types), HTTP methods, asynchronous operations, etc..
  Refer to the specification

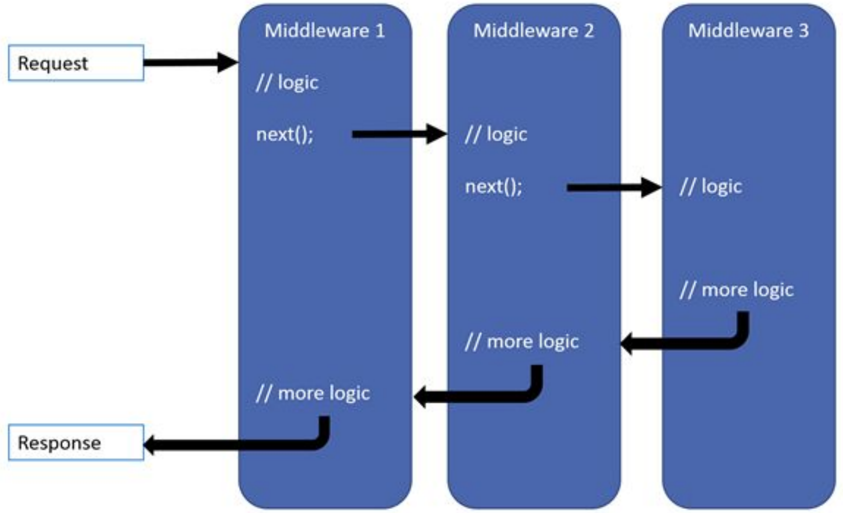
API Design

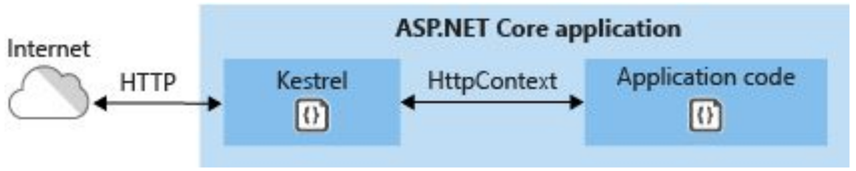
Creating a Web API



Handling HTTP Requests With ASP.NET Core

- The server (Kestrel) listens for requests
- The **middleware** pipeline is invoked for each request
- Use MVC to route requests to a **controller** and **action**
- **Responses** flow back down the middleware pipeline





Creating a Web API

- Create an ASP.NET Core project (**dotnet new web**)
- Setup MVC
- Create a class that derives from **ControllerBase**
- Implement your **action methods**

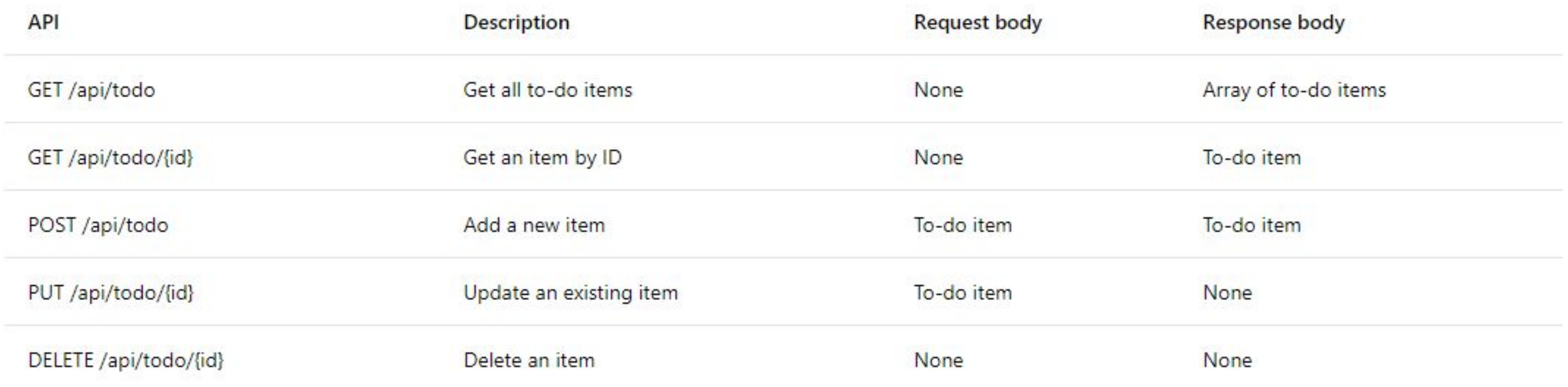
or just...

```
dotnet new webapi
```

Since .NET Core is composed, you can also create a web API starting from a basic console application

Example API Overview

A web API for managing a list of "to-do" items



API	Description	Request body	Response body
GET /api/todo	Get all to-do items	None	Array of to-do items
GET /api/todo/{id}	Get an item by ID	None	To-do item
POST /api/todo	Add a new item	To-do item	To-do item
PUT /api/todo/{id}	Update an existing item	To-do item	None
DELETE /api/todo/{id}	Delete an item	None	None

Attribute Routing

- How requests are **routed** to controller actions
- [HttpGet/Post/Put/Delete(“api/orders”)]
- Specify multiple HTTP verbs with **AcceptVerbsAttribute**
- Use **RouteAttribute** to specify no HTTP method at all
- Controller routes prepended to action routes

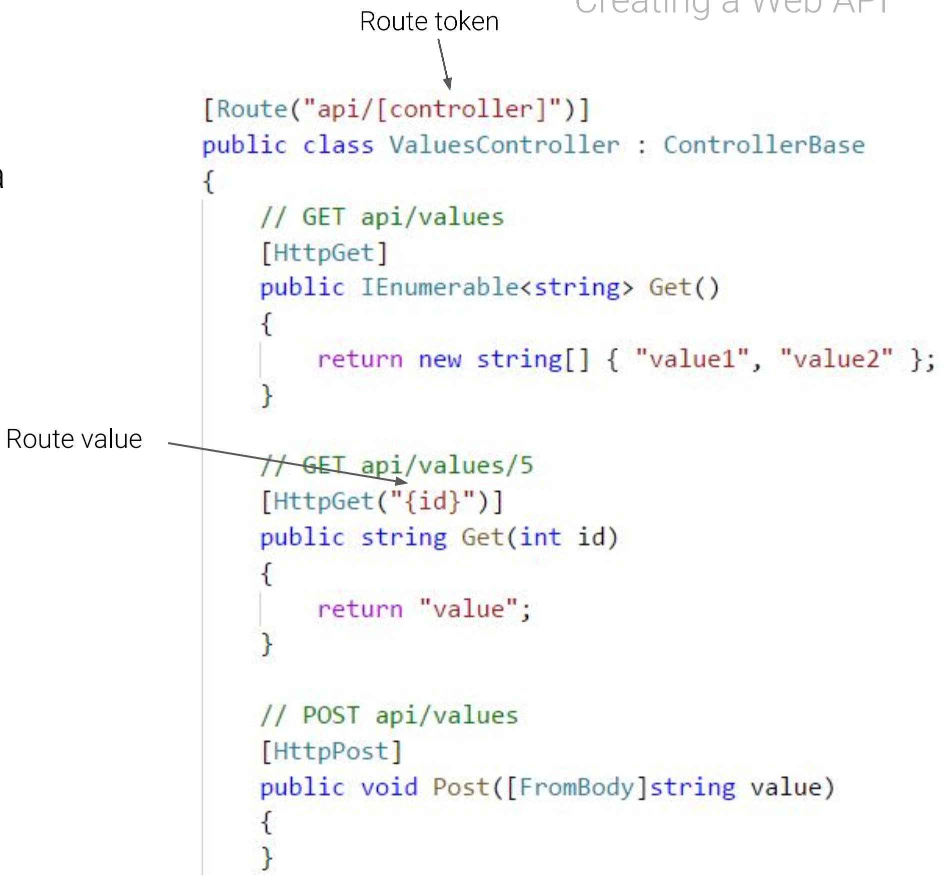
```
[Route("api/[controller]")]
public class ValuesController : ControllerBase
{
    // GET api/values
    [HttpGet]
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    [HttpGet("{id}")]
    public string Get(int id)
    {
        return "value";
    }

    // POST api/values
    [HttpPost]
    public void Post([FromBody]string value)
    {
    }
```

Example route:
http://localhost:5000/api/values/42

```
[AcceptVerbs("POST", "PUT")]
public IActionResult Add(TodoItem item)
{
    // Add or update item
    return Ok();
}
```

Route Templates

- Extract route values (e.g. “api/orders/**{id}**”)
- Route tokens (e.g. “api/**[controller]**”)
    - Specify the current controller/action/area
- Optional route values: {id**?**}
- Default route values: {id**=42**}
- Constraints: {id**:**int}



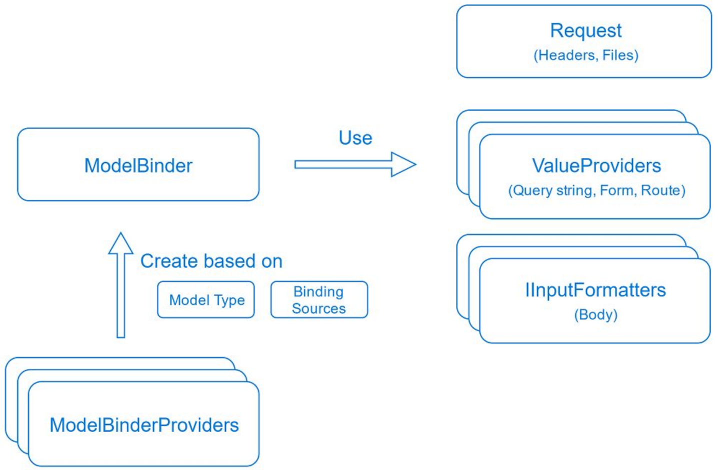
```
[Route("api/[controller]")]
public class ValuesController : ControllerBase
{
    // GET api/values
    [HttpGet]
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    [HttpGet("{id}")]
    public string Get(int id)
    {
        return "value";
    }

    // POST api/values
    [HttpPost]
    public void Post([FromBody]string value)
    {
    }
```

Model Binding

- Bind **request data** to **action parameters**
- Bind **form data**, **route values** and **query string parameters** by default
- Use **[FromBody]** to bind the request body using **formatters**
- Use **[FromRoute/Query/Form/Header]** to restrict model binding to a particular source

```
[HttpPost]
public IActionResult Create([FromBody] TodoItem item)
{
    if (item == null)
    {
        return BadRequest();
    }

    _context.TodoItems.Add(item);
    _context.SaveChanges();

    return CreatedAtRoute("GetTodo", new { id = item.Id }, item);
}
```

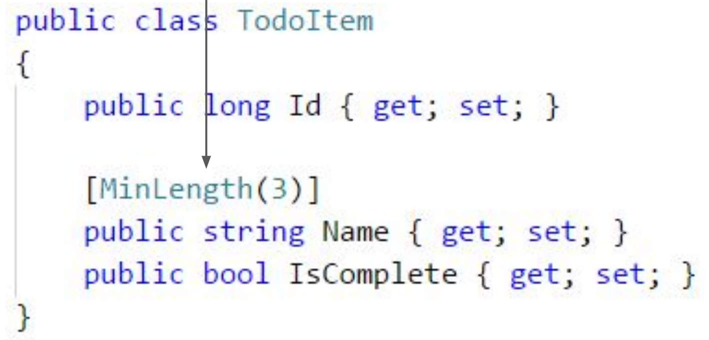
value based on
request body



Model Validation

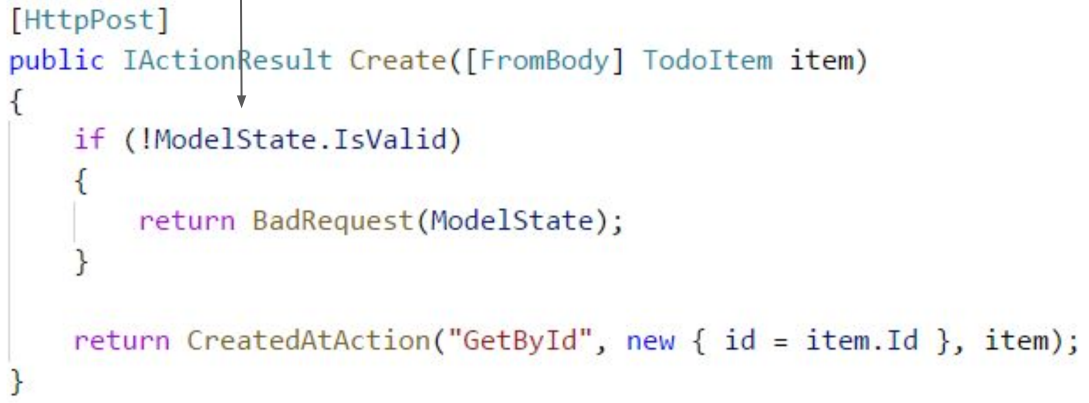
- Use data annotations and check ModelState.IsValid

Data annotations are applied to the model



```
public class TodoItem
{
    public long Id { get; set; }

    [MinLength(3)]
    public string Name { get; set; }
    public bool IsComplete { get; set; }
}
```

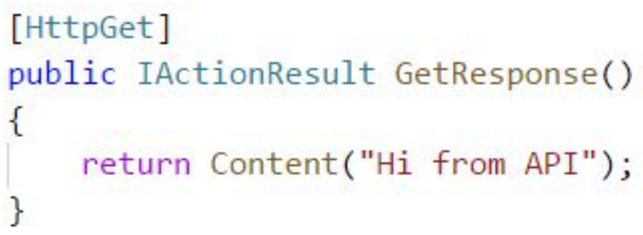
ModelState is checked in the controller



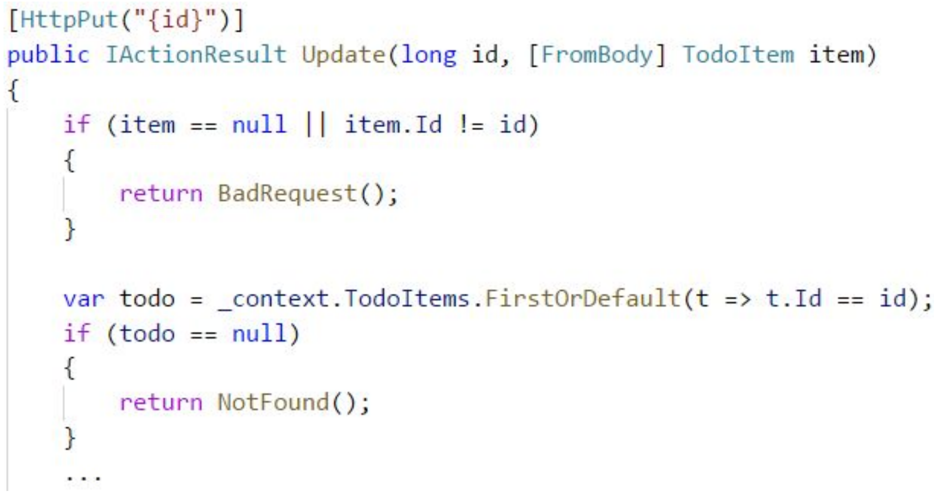
```
[HttpPost]
public IActionResult Create([FromBody] TodoItem item)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    return CreatedAtAction("GetById", new { id = item.Id }, item);
}
```

Action Results

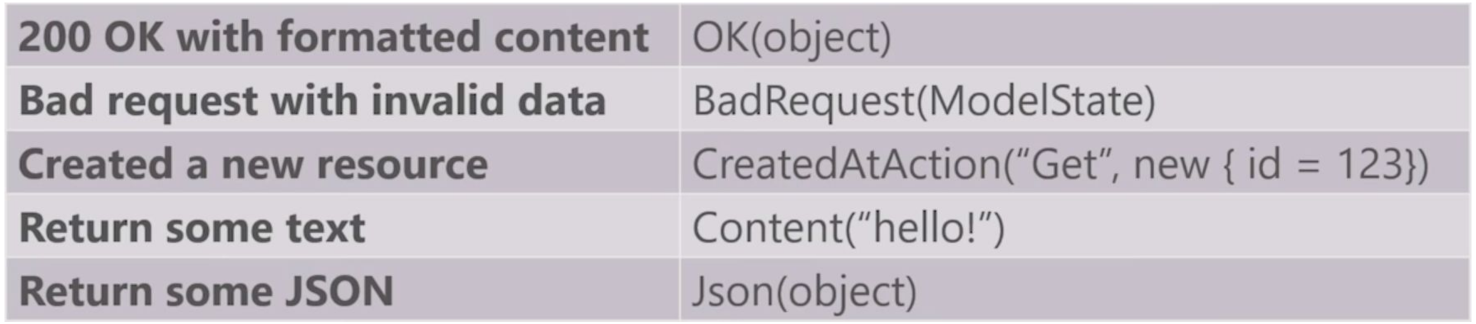
- Used to produce the **response**
- Return **IActionResult** (or Task <IActionResult>)
- Use helper extension methods on ControllerBase



```
[HttpGet]
public IActionResult GetResponse()
{
    return Content("Hi from API");
}
```



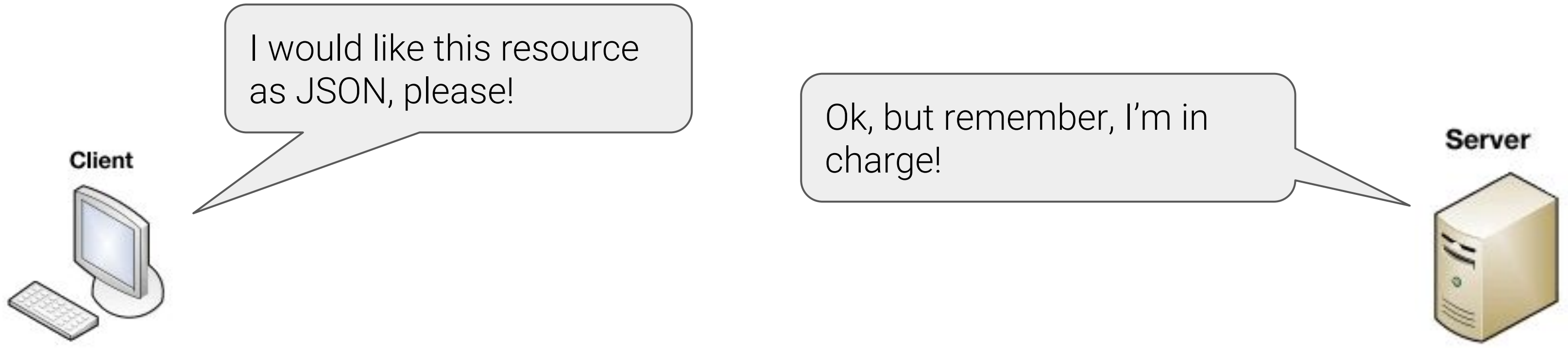
```
[HttpPut("{id}")]
public IActionResult Update(long id, [FromBody] TodoItem item)
{
    if (item == null || item.Id != id)
    {
        return BadRequest();
    }

    var todo = _context.TodoItems.FirstOrDefault(t => t.Id == id);
    if (todo == null)
    {
        return NotFound();
    }
    ...
```



200 OK with formatted content	OK(object)
Bad request with invalid data	BadRequest(ModelState)
Created a new resource	CreatedAtAction(“Get”, new { id = 123})
Return some text	Content(“hello!”)
Return some JSON	Json(object)

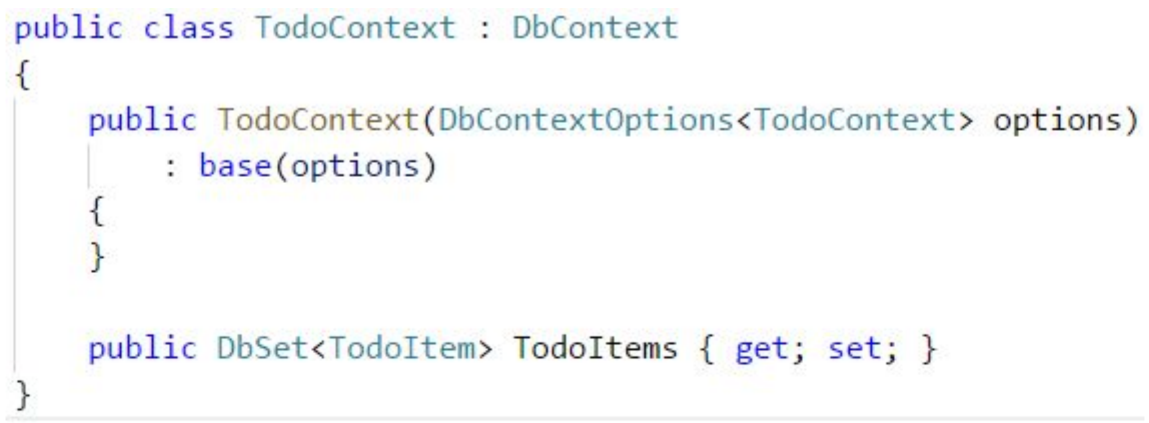
Formatting

- Separate **input** and **output** formatters
- **Configure formatters** through MVC options
- **Input** formatters handle **request** body formats
  - Don't forget [FromBody]!
- **Output** formatters handle **response** content-negotiation
- Constrain formats per action using **[Produces/Consumes]**


I would like this resource as JSON, please!
Client
Ok, but remember, I'm in charge!
Server

Data Persistence

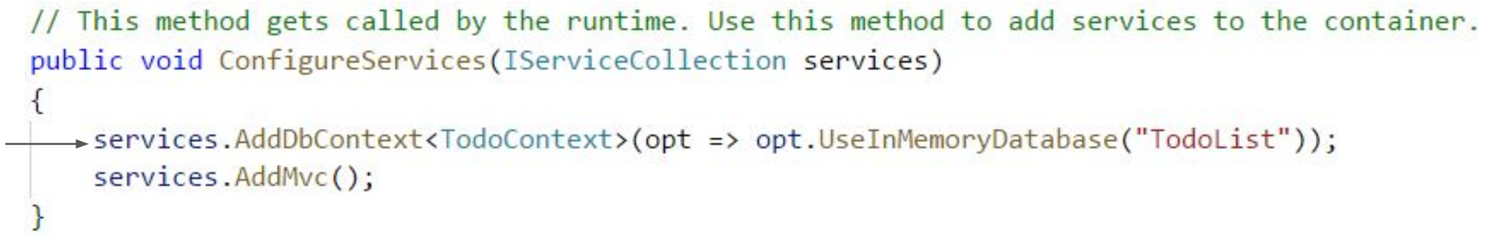
- Use **Entity Framework Core** to access a variety of data sources
- Inject your **DbContext** into your Web API controllers
- For now we use **InMemoryDatabase**...



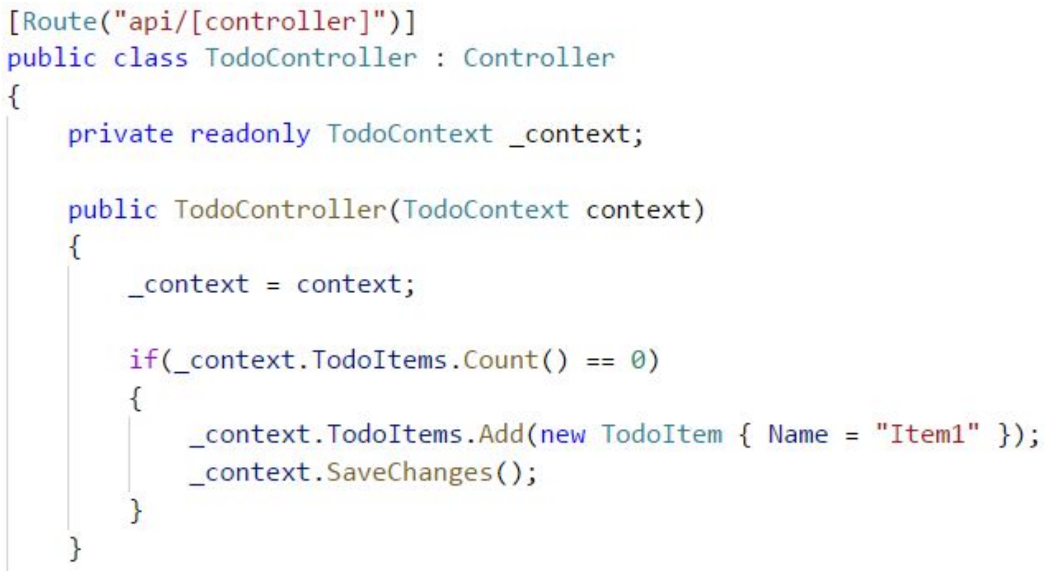
```
public class TodoContext : DbContext
{
    public TodoContext(DbContextOptions<TodoContext> options)
        : base(options)
    {
    }

    public DbSet<TodoItem> TodoItems { get; set; }
}
```

Data Persistence

In Startup.cs



```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<TodoContext>(opt => opt.UseInMemoryDatabase("TodoList"));
    services.AddMvc();
}
```



```
[Route("api/[controller]")]
public class TodoController : Controller
{
    private readonly TodoContext _context;

    public TodoController(TodoContext context)
    {
        _context = context;

        if(_context.TodoItems.Count() == 0)
        {
            _context.TodoItems.Add(new TodoItem { Name = "Item1" });
            _context.SaveChanges();
        }
    }
```

Web API - Demo

Help Pages for Your API

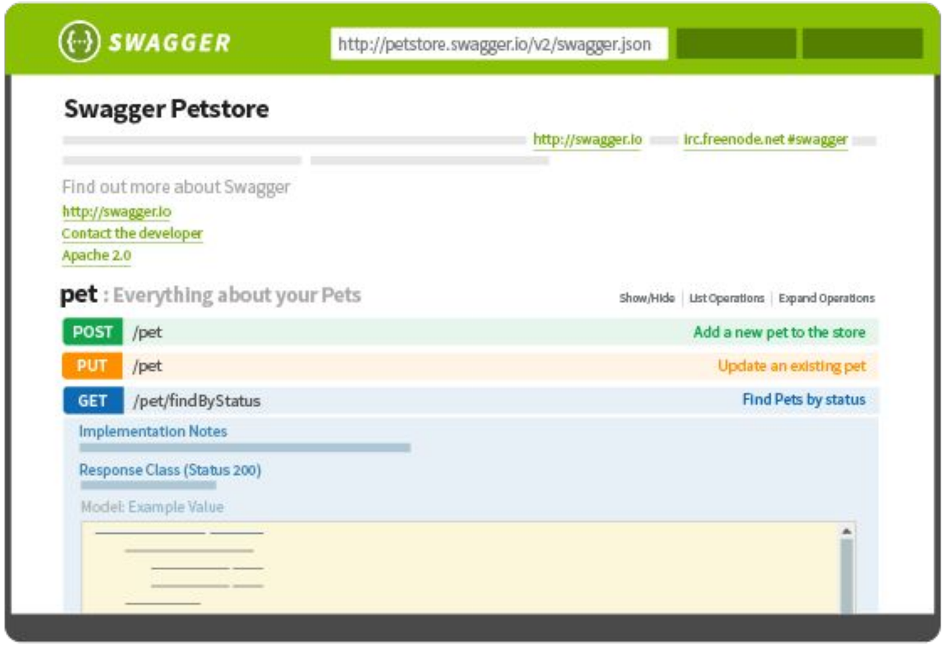
Understanding the various methods of an API can be a challenge for a developer when building a consuming app…

Use Swagger to generate documentation and help pages for your Web API

Example UI

https://swagger.io/

Using Swagger with .NET Core Web API



Swagger Petstore

http://swagger.io irc.freenode.net #swagger

Find out more about Swagger
http://swagger.io
Contact the developer
Apache 2.0

pet : Everything about your Pets

Show/Hide | List Operations | Expand Operations

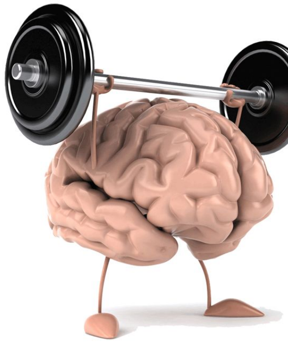
POST /pet Add a new pet to the store
PUT /pet Update an existing pet
GET /pet/findByStatus Find Pets by status

Implementation Notes

Response Class (Status 200)

Model: Example Value

Exercises



Building a Web API (video)   Build web APIs with ASP.NET Core