UNIVERSITY OF WESTMINSTER

Enterprise Application Development

Lecture 9 – Working with Web APIs (in C# and .NET)

What is an API?

- API – Application Programming Interface defines the way in which computer systems interact.

- APIs are everywhere
  - APIs in the libraries we use from our language package manager
  - APIs in the code we write ourselves

- We will focus on one special type of API that is built to be exposes over a network and used remotely by other services – **Web API**

Web API

- In Web APIs, consumers have very little control, meaning if the API builder (producer) decides to change the API then consumers will need to make adjustments to accommodate the change

- When we work with libraries, we have a local copy of a specific version and new updates, or changes do not affect our system or other consumers

- To communicate with a Web API we typically use HTTP

UNIVERSITY OF WESTMINSTER

How to design an API?

- Think about the users of the API (consumers) and what is that they should be able to do?

- Who can do what?
  - Authentication and authorization

- Which endpoints are needed and what responses should look like for each endpoint?
  - Identify input and output

UNIVERSITY OF WESTMINSTER

REST

- REST - REpresentational State Transfer is a Web architectural style
- REST describes the Web as a distributed hypermedia application whose linked resources communicate by exchanging representations of resource state
- Communication in REST is done over HTTP
  - Stateless

HTTP method	Action
POST (and PUT in creation)	Create a customer, add a meal to a menu, order goods, start a timer, save a blog post, send a message to customer service, subscribe to a service, sign a contract, open a bank account, upload a photo, share a status on a social network, and so on
GET	Read a customer, search for a French restaurant, find new friends, retrieve opened accounts for the last 3 months, download a signed contract, filter best selling books, select black-and-white photos, list friends, and so forth
PATCH/PUT	Update a customer, replace goods in an order, switch plane seat, edit an order's delivery method, change an order's currency, modify a debit card limit, temporarily block a credit card, and so on
DELETE	Delete a customer, cancel an order, close a case, terminate a process, stop a timer, and so on

UNIVERSITY OF WESTMINSTER

REST
HTTP methods & actions

UNIVERSITY OF WESTMINSTER

API request examples

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

UNIVERSITY OF WESTMINSTER

HTTP status codes

- 1xx - Informational
  - 101 - Switching Protocols
- 2xx - Success
  - 200 - OK
  - 201 - Created
- 3xx - Redirection
  - 301 - Moved Permanently
- 4xx - Client Error
  - 400 - Bad Request
  - 401 - Unauthorized
  - 403 - Forbidden
  - 404 - Not Found
- 5xx - Server Error
  - 500 - Internal Server Error
  - 501 - Not Implemented

UNIVERSITY OF WESTMINSTER

HTTP POST request

- CURL: CLI tool for testing APIs

- Request example

```
curl -X POST https://localhost:5001/api/todoitems
   -H "Content-Type: application/json"
   -d '{"name":"walk dog","isComplete":true}'
```

UNIVERSITY OF WESTMINSTER

HTTP POST response

HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Date: Tue, 07 Sep 2021 20:39:47 GMT
Location: https://localhost:5001/api/TodoItems/1
Server: Kestrel
Transfer-Encoding: chunked

```
{
  "id": 1,
  "name": "walk dog",
  "isComplete": true
}
```

UNIVERSITY OF WESTMINSTER

HTTP GET request

curl https://localhost:5001/api/todoitems/1

-H "Accept: application/json"

UNIVERSITY OF WESTMINSTER

HTTP GET response

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 07 Sep 2021 20:48:10 GMT
Server: Kestrel
Transfer-Encoding: chunked


```
{
  "id": 1,
  "name": "walk dog",
  "isComplete": true
}
```

Sync vs Async

- Sync is blocking and executes on the main (UI) thread
- Async is non blocking meaning the user can interact with the UI while they wait for the operation to be completed
- Examples
  - Long I/O operation
  - Network request to the server

UNIVERSITY OF WESTMINSTER

JSON

- JSON - JavaScript Object Notation is a data inter-change format

```
public class TodoItem
{
    public long Id { get; set; }
    public string? Name { get; set; }
    public bool IsComplete { get; set; }
}
```

```
// Instance of TodoItem
var todoItem = new TodoItem
{
    IsComplete = true,
    Name =  "walk dog"
};
```

```
{
  "id": 1,
  "name": "walk dog",
  "isComplete": true
}
```

JSON

- Serialization - the process of converting an object into a string
  - Necessary when sending data accross the network
- Deserialization - the reverse process, converting a string back into an object
  - What we get back from an API is typically a JSON string
  - We need an object so we can access specific properties and show them up on the UI

API Example

```
[Route("api/[controller]")]
[ApiController]
public class TodoItemsController : ControllerBase
{
    private readonly TodoContext _context;

    public TodoItemsController(TodoContext context)
    {
        _context = context;
    }
}
```

UNIVERSITY OF WESTMINSTER

API Example - Model vs DTO

```
// Model
public class TodoItem
{
    public long Id { get; set; }
    public string? Name { get; set; }
    public bool IsComplete { get; set; }
    public string? Secret { get; set; }
}
```

```
// DTO
public class TodoItemDTO
{
    public long Id { get; set; }
    public string? Name { get; set; }
    public bool IsComplete { get; set; }
}
```

UNIVERSITY OF WESTMINSTER

API Example - GET all items

```
// GET: api/TodoItems
[HttpGet]
public async Task<ActionResult<IEnumerable<TodoItemDTO>>> GetTodoItems()
{
    return await _context.TodoItems
        .Select(x => ItemToDTO(x))
        .ToListAsync();
}
```

UNIVERSITY OF WESTMINSTER

API Example - Model to DTO

```
private static TodoItemDTO ItemToDTO(TodoItem todoItem)
{
    return new TodoItemDTO
    {
        Id = todoItem.Id,
        Name = todoItem.Name,
        IsComplete = todoItem.IsComplete
    };
}
```

UNIVERSITY OF WESTMINSTER

API Example - GET a single item

```
// GET: api/TodoItems/5
[HttpGet("{id}")]
public async Task<ActionResult<TodoItemDTO>> GetTodoItem(long id)
{
    var todoItem = await _context.TodoItems.FindAsync(id);

    if (todoItem == null)
    {
        return NotFound();
    }

    return ItemToDTO(todoItem);
}
```

UNIVERSITY OF WESTMINSTER

API Example - POST

```
// POST: api/TodoItems
[HttpPost]
public async Task<ActionResult<TodoItemDTO>> CreateTodoItem(TodoItemDTO todoItemDTO)
{
    var todoItem = new TodoItem
    {
        IsComplete = todoItemDTO.IsComplete,
        Name = todoItemDTO.Name
    };

    _context.TodoItems.Add(todoItem);
    await _context.SaveChangesAsync();

    return CreatedAtAction(
        nameof(GetTodoItem),
        new { id = todoItem.Id },
        ItemToDTO(todoItem));
}
```

UNIVERSITY OF WESTMINSTER

API Example - PUT

```
// PUT: api/TodoItems/5
[HttpPut("{id}")]
public async Task<IActionResult> UpdateTodoItem(long id, TodoItemDTO todoItemDTO)
{
    if (id != todoItemDTO.Id)
        return BadRequest();

    var todoItem = await _context.TodoItems.FindAsync(id);
    if (todoItem == null)
        return NotFound();

    todoItem.Name = todoItemDTO.Name;
    todoItem.IsComplete = todoItemDTO.IsComplete;
```

```
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if(!TodoItemExists(id))
            return NotFound();
    }

    return NoContent();
}

private bool TodoItemExists(long id)
{
    return _context.TodoItems.Any(e => e.Id == id);
}
```

UNIVERSITY OF WESTMINSTER

API Example - DELETE

```
// DELETE: api/TodoItems/5
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteTodoItem(long id)
{
    var todoItem = await _context.TodoItems.FindAsync(id);

    if (todoItem == null)
    {
        return NotFound();
    }

    _context.TodoItems.Remove(todoItem);
    await _context.SaveChangesAsync();

    return NoContent();
}
```

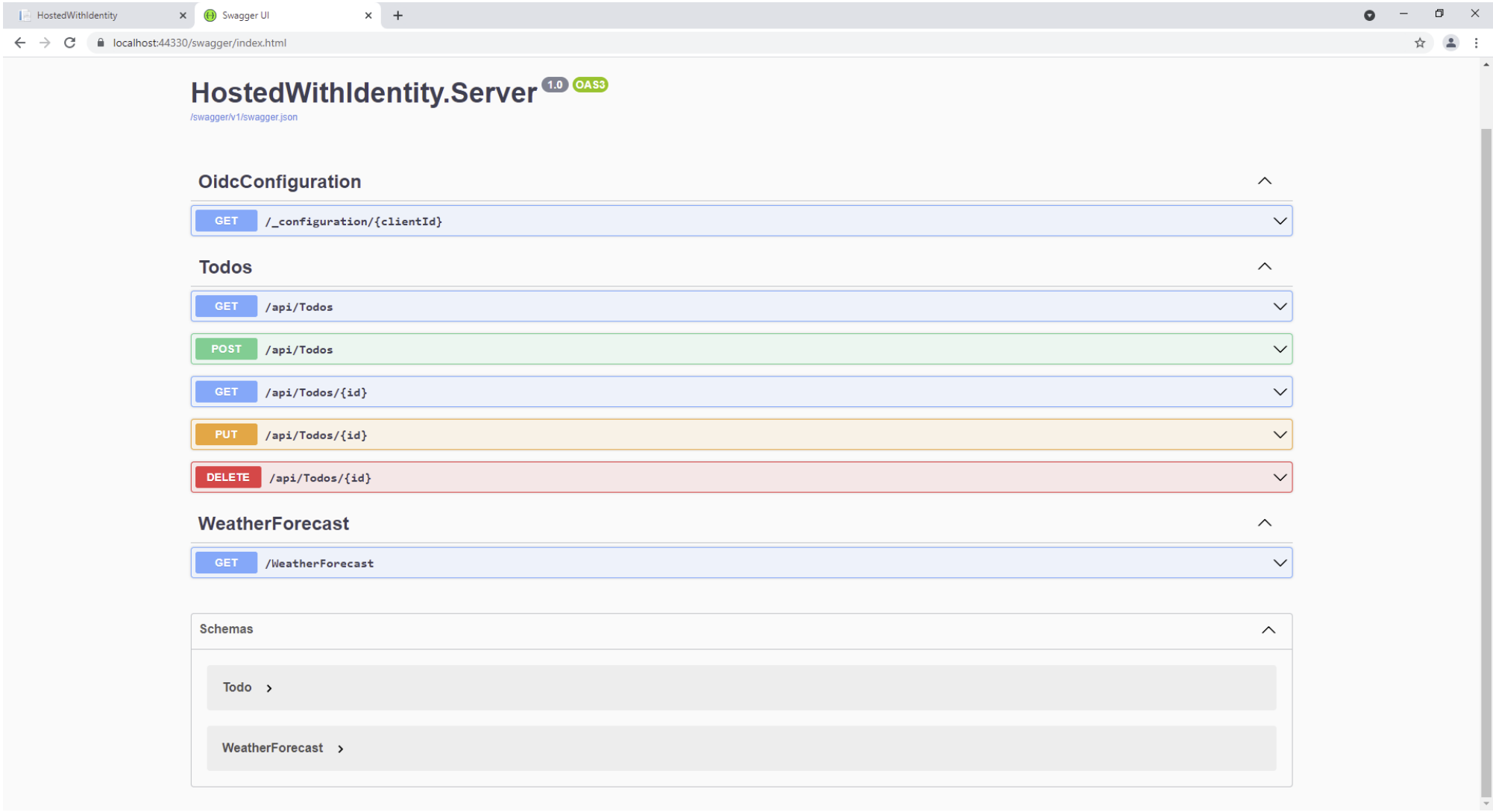
API Security

- Authentication is a process in which a user provides credentials that are then compared to those stored in a database
- Authorization the process that determines what a user is allowed to do
- JWT
  - Token based authentication and authorization
- Add [Authorize] data annotation to the controller class
  - This will make sure requests contain authentication data

UNIVERSITY OF WESTMINSTER

Open API standard

- Swagger



Resources

- Useful links
  - https://dotnet.microsoft.com/apps/aspnet/apis
  - https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-6.0&tabs=visual-studio#overview
  - https://github.com/microsoft/api-guidelines

- Books
  - The Design of Web APIs, by Arnaud Lauret
  - REST in Practice, by Jim Webber, Savas Parastatidis, Ian Robinson