Controllers

Working with Controllers

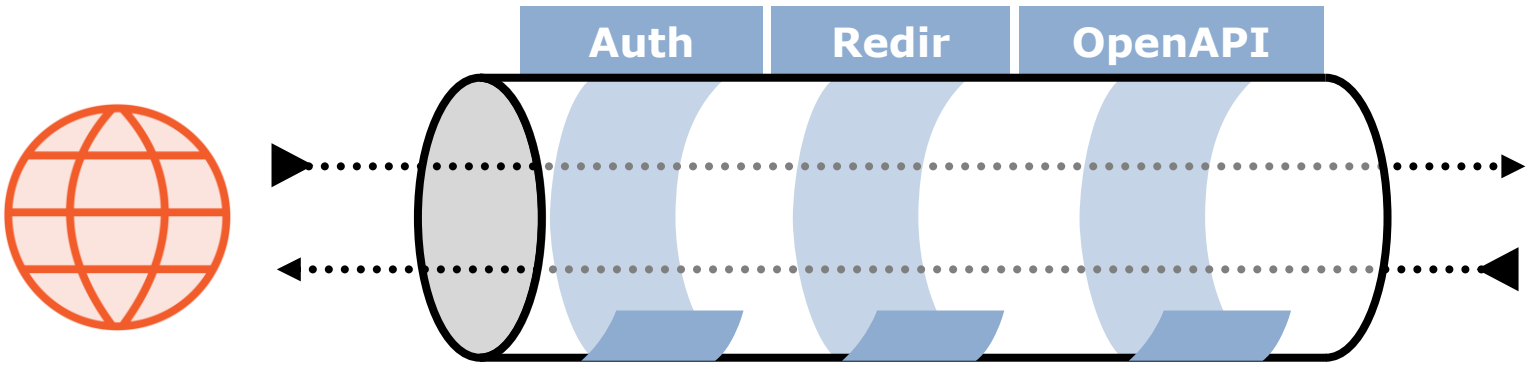
- Exploring the Web API Visual Studio template using controllers.

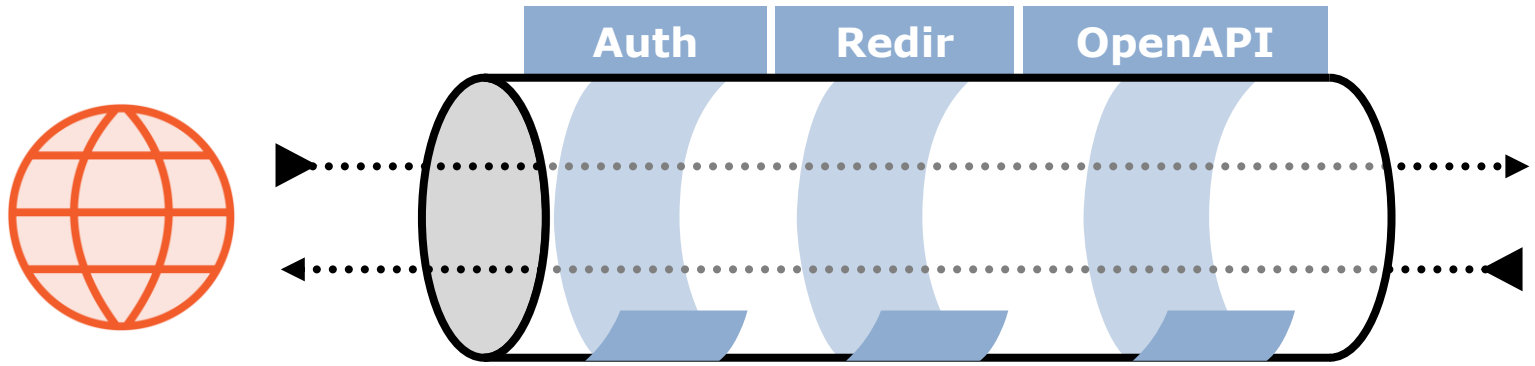
Anatomy of An ASP.NET Core Application

A common structure for all ASP.NET Core application types:

- Launchsettings.json
- Program.cs
  - Entry point
  - Builder
  - Section for Dependency injection
  - Section for pipeline

The Pipeline




Auth
Redir
OpenAPI

Swagger/OpenAPI

Creates documention for a RESTful API:

- **Discoverability**
- **Testability**
- **Swashbuckle**
- **Generates a JSON file**

- **It can only detect so much**
- **We have to help it using comments and attributes**

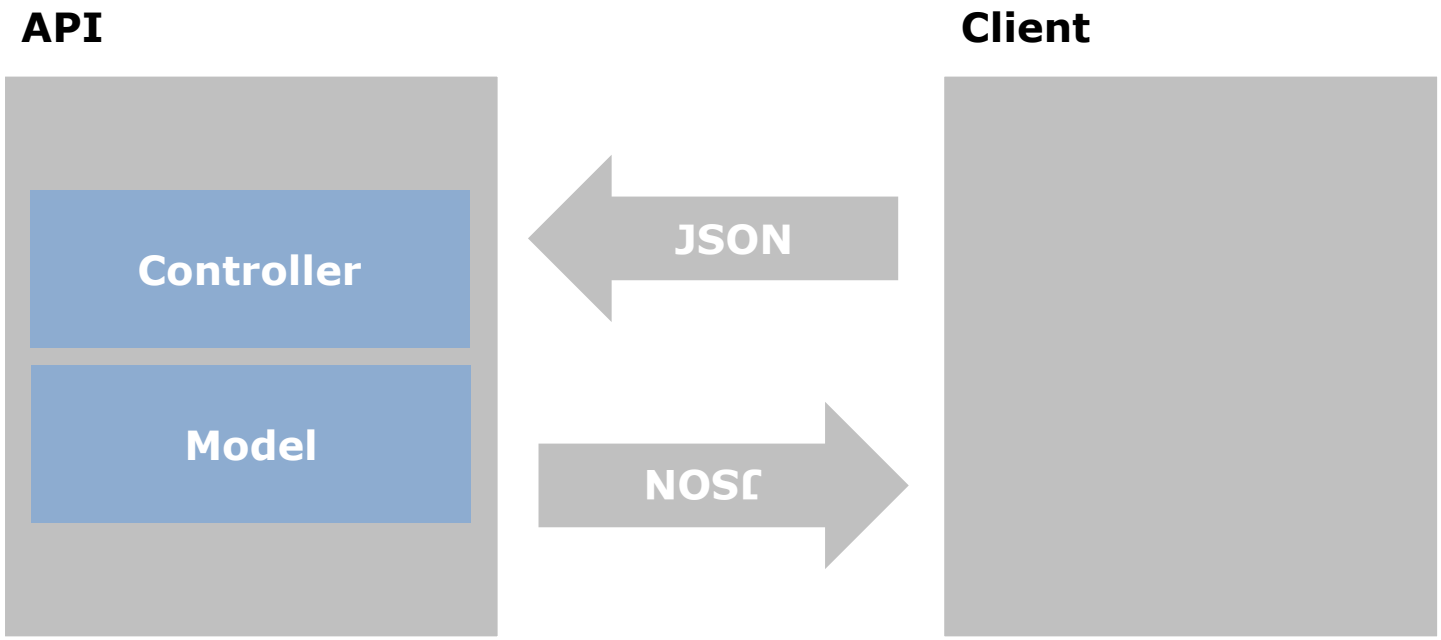
What Are Attributes?

- Use attributes to provide additional metadata about an element
- Use attributes to alter run-time behavior

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController: ControllerBase
{
   [Obsolete("This property will be removed in the next release.")]
   public string Name { get; set; }

   [HttpGet]
   public IEnumerable<Forecast> Get()
   {
            ..
   }
}
```

Controllers



The ApiController Attribute

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

Controllers: Conventions and Attributes

- When an action starts with Get, Post, Put etc. it is assumed to respond to the corresponding HTTP method.
- But Swagger/OpenAPI hates conventions.
- Also possible to use [HttpGet], [HttpPost], [HttpPut] etc.

Dependency Injection and Controllers

- Use contructor injection in the controller classes.

Status Codes and Route Parameters

- To support non-default status codes, controller actions should return an IActionResult object.

- For route parameters, use action parameters with the [FromRoute] attribute in combination with the expression in the route itself.

Lab: Controllers

- Recreate the API using controllers
- Controllers folder