ASP.NET Core Web API: Controller Deep Dive

1. What is a Controller?

A controller in ASP.NET Core Web API is a class that handles HTTP requests and produces HTTP responses. It acts as an interface between the HTTP client (browser, mobile app) and the backend services or database.

2. Basic Structure of a Controller

```
[ApiController]
[Route("api/[controller]")]
public class ProductsController : ControllerBase
{
    // Constructor Injection
    // Action Methods (GET, POST, PUT, DELETE)
}
```

3. Key Attributes and Their Purposes

[ApiController]: Enables automatic model validation and improved routing.
[Route("api/[controller]")]: Sets the base URL path.
[HttpGet], [HttpPost], [HttpPut], [HttpDelete]: Maps HTTP verbs to action methods.

4. Dependency Injection

Services are injected into controllers via constructor parameters using interfaces. This promotes testability, modularity, and separation of concerns.

5. DTO Usage

DTOs (Data Transfer Objects) are used to pass data to and from the API. They prevent exposing internal domain models directly.

6. Example Controller: ProductsController

ASP.NET Core Web API: Controller Deep Dive

```
[ApiController]
[Route("api/[controller]")]
public class ProductsController : ControllerBase
{
    private readonly IProductService _productService;

    public ProductsController(IProductService productService)
    {
        _productService = productService;
    }

    [HttpGet]
    public async Task<IActionResult> GetAll() => Ok(await _productService.GetAllProductsAsync());

    [HttpGet("{id}")]
    public async Task<IActionResult> GetById(int id)
    {
        var product = await _productService.GetProductByIdAsync(id);
        return product == null ? NotFound() : Ok(product);
    }

    [HttpPost]
    public async Task<IActionResult> Create([FromBody] CreateProductDto dto)
    {
        var newProduct = await _productService.CreateProductAsync(dto);
        return CreatedAtAction(nameof(GetById), new { id = newProduct.Id }, newProduct);
    }
}
```

7. Comparison with Basic Controller

ASP.NET Core Web API: Controller Deep Dive

A simple controller returning static data is fine for demos, but lacks scalability.

For example, HSCodeController has no DI, no service layer, and returns hardcoded values.

8. Best Practices Summary

- Keep controllers lean: delegate business logic to services.
- Use async/await for scalability.
- Validate input using model validation.
- Return proper HTTP status codes.
- Use DTOs instead of exposing domain models.
- Handle exceptions gracefully (via middleware or filters).