HABIB UL REHMAN

01

ASP.NET CORE WEB API BEST PRACTICES

@HABIBDEVELOPER

FINCHSHIP.COM

HABIB UL REHMAN

02

Use Attribute Routing for Clear Endpoints

Employ attribute routing for Web API controllers to define clear and concise endpoint routes.

```
[Route("api/[controller]")]
public class ProductsController : ControllerBase
{
    [HttpGet("{id}")]
    public IActionResult GetProduct(int id)
    {
        // Get product logic
    }
}
```

HABIB UL REHMAN

03

DTOs for Request and Response Models

Utilize Data Transfer Objects (DTOs) to represent request and response models. This promotes clarity and separation of concerns.

```
public class CreateProductDto
{
    public string Name { get; set; }
    public decimal Price { get; set; }
}

[HttpPost]
public IActionResult CreateProduct([FromBody] CreateProductDto productDto)
{
    // Create product logic
}
```

HABIB UL REHMAN

04

AutoMapper for mapping DTOs

A mapper automates the conversion process between DTOs and domain models, making your code cleaner.

```
public class CreateProductDto
{
    public string Name { get; set; }
    public decimal Price { get; set; }
}
public class CreateProduct
{
    public string Name { get; set; }
    public decimal Price { get; set; }
}
```

```
private readonly IMapper _mapper;
public ProductController(IMapper mapper)
{
    _mapper = mapper;
}
[HttpPost]
public IActionResult CreateProduct([FromBody]CreateProductDto productDto)
{
    CreateProduct product = _mapper.Map<CreateProduct>(productDto);
    //Create product logic
}
```

HABIB UL REHMAN

05

Versioning for API Longevity

Implement API versioning to ensure backward compatibility and facilitate future updates.

```
services.AddApiVersioning(options ⇒
{
    options.DefaultApiVersion = new ApiVersion(1, 0);
    options.AssumeDefaultVersionWhenUnspecified = true;
    options.ReportApiVersions = true;
});
```

HABIB UL REHMAN

06

Use Dependency Injection for Services

Leverage Dependency Injection to inject services into controllers, promoting modularity and testability.

```public class ProductsController : ControllerBase
{
    private readonly IProductService _productService;

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

HABIB UL REHMAN

07

Handle Errors Gracefully

Implement a consistent error handling strategy using status codes and meaningful error messages.

```
1 [HttpGet("{id}")]
2 public IActionResult GetProduct(int id)
3 {
4     var product = _productService.GetProduct(id);
5
6     if (product == null)
7     {
8         return NotFound($"Product with ID {id} not found.");
9     }
10
11    return Ok(product);
12 }
```

HABIB UL REHMAN

08

Implement Caching for Performance

Apply caching strategies to enhance API performance, especially for data that doesn't change frequently.

```
[HttpGet]
[ResponseCache(Duration = 60)]
public IActionResult GetProducts()
{
    // Return products with caching
}
```

HABIB UL REHMAN

09

REPOST THIS POST IF YOU FIND IT USEFUL

@HABIBDEVELOPER

FINCHSHIP.COM