



Title: RESTful API Development
Module: ASP.NET Core
Presented by: Narasimha Rao T

RESTful API Development in ASP.NET Core

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

What is RESTful API?

1. Introduction to REST Principles and Conventions

- **REST (Representational State Transfer):**
An architectural style for building scalable, stateless services using HTTP.
- **Key Principles:**
 - **Statelessness:** Each request contains all necessary information.
 - **Client-Server Separation:** Client (UI) is separate from server (API).
 - **Uniform Interface:** Standardized endpoints and operations.
 - **Resource-Based:** Everything is treated as a resource (e.g., `/students`, `/orders`).

- **Conventions:**

- **Nouns, not verbs:** `/api/products` instead of `/api/getProducts`
- **Use of HTTP Methods:**
 - `GET` → Retrieve
 - `POST` → Create
 - `PUT` → Update (replace)
 - `PATCH` → Update (partial)
 - `DELETE` → Remove
- **Status Codes:** Use standardized responses (200, 404, 500, etc.).

2. Difference: MVC Application vs Web API Application

| Feature | MVC Application | Web API Application |
|---------------|---|--|
| Purpose | Build web apps with views (UI + logic). | Build RESTful services for data exchange. |
| Response Type | Returns HTML (Views, Razor pages). | Returns JSON or XML (default JSON). |
| Controllers | <code>Controller</code> (inherits from <code>Controller</code>). | <code>ApiController</code> (inherits from <code>ControllerBase</code>). |
| Consumers | Browsers (for web pages). | Any client (mobile, web, IoT). |
| Routing | Conventional routing (MVC pattern). | Attribute routing (<code>[Route("api/[controller]")]</code>). |

3. API Controllers vs MVC Controllers

- MVC Controller:
 - Inherits from `Controller`.
 - Used for serving Views + Data.
 - Example:

```
public class HomeController : Controller
{
    public IActionResult Index() => View();
}
```

API Controller:

- Inherits from `ControllerBase` .
- Decorated with `[ApiController]` attribute.
- Focused on data (JSON/XML) responses only.
- Example:

```
[ApiController]
[Route("api/[controller]")]
public class ProductsController : ControllerBase
{
    [HttpGet]
    public IEnumerable<string> Get() => new string[] { "Apple", "Banana" };
}
```


4. HTTP Verbs and Status Codes

- Common HTTP Verbs:
 - GET → Retrieve resource
 - POST → Create resource
 - PUT → Replace entire resource
 - PATCH → Update part of resource
 - DELETE → Remove resource

- Common HTTP Status Codes:
 - 200 OK → Successful request
 - 201 Created → New resource created
 - 400 Bad Request → Invalid input
 - 401 Unauthorized → Authentication required
 - 403 Forbidden → Access denied
 - 404 Not Found → Resource not found
 - 500 Internal Server Error → Server-side issue

5. Returning Data using IActionResult, ActionResult<T>

- **IActionResult**: Flexible return type supporting different responses.

Example:

```
public IActionResult GetProduct(int id)
{
    if (id <= 0) return BadRequest();
    return Ok(new { Id = id, Name = "Laptop" });
}
```

- **ActionResult<T>**: Combines `IActionResult` with a strong type.

Example:

```
public ActionResult<Product> GetProduct(int id)
{
    if (id <= 0) return NotFound();
    return new Product { Id = id, Name = "Laptop" };
}
```

6. Async Action Methods

- **Why Async?** Improves scalability, especially with I/O operations (DB calls, API calls).
- Example:

```
[HttpGet("{id}")]
public async Task<ActionResult<Product>> GetProduct(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product == null) return NotFound();
    return product;
}
```

CRUD Operations in API

7. CRUD Action Methods (Example: ProductsController)

Prepare Controller and Inject the context class

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

    public ProductsController(AppDbContext context) => _context = context;
}
```

Read Operation

```
[HttpGet] // GET: api/products
public async Task<IEnumerable<Product>> GetAll()
{
    return await _context.Products.ToListAsync();
}

[HttpGet("{id}")] // GET: api/products/5
public async Task<ActionResult<Product>> GetById(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product == null) return NotFound();
    return product;
}
}
```


Create Operation

```
[HttpPost] // POST: api/products
public async Task<ActionResult<Product>> Create(Product product)
{
    _context.Products.Add(product);
    await _context.SaveChangesAsync();
    return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);
}
```

Update Operation

```
[HttpPut("{id}")] // PUT: api/products/5
public async Task<IActionResult> Update(int id, Product product)
{
    if (id != product.Id) return BadRequest();
    _context.Entry(product).State = EntityState.Modified;
    await _context.SaveChangesAsync();
    return NoContent();
}
```

Delete Operation

```
[HttpDelete("{id}")] // DELETE: api/products/5
public async Task<IActionResult> Delete(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product == null) return NotFound();
    _context.Products.Remove(product);
    await _context.SaveChangesAsync();
    return NoContent();
}
```

Q & A

Narasimha Rao T

tnrao.trainer@gmail.com