✅ HTTP Status Codes in ASP.NET Core REST API

HTTP status codes tell the client whether the API request was **successful**, **failed**, or **requires attention**.

ASP.NET Core provides built-in helper methods like:

- `Ok()`
- `Created()`
- `NotFound()`
- `BadRequest()`
- `NoContent()`
- `StatusCode()`

📌 Status Code Categories

1xx – Informational

Rarely used in APIs.

2xx – Success Codes

These tell the client: **Operation successful**.

Status Code	Meaning	When to Use
200 OK	Request succeeded	GET, PUT success
201 Created	New resource created	POST success
204 No Content	Successful but no response body	DELETE, PUT (no response content)

3xx – Redirection Codes

Not common in REST APIs.

4xx – Client Error Codes

Client sent wrong data.

Status Code	Meaning	When to Use
400 Bad Request	Invalid input	Request model fails
401 Unauthorized	User not logged in	Missing/invalid token
403 Forbidden	User logged in but no permission	Role-based blocking
404 Not Found	Data not found	Wrong ID
409 Conflict	Resource conflict	Duplicate email, username exists

---

5xx – Server Error Codes

Server failed to process.

Status Code	Meaning
500 Internal Server Error	Unexpected exception
502 Bad Gateway	Reverse proxy issue
503 Service Unavailable	API down or overloaded

🔥 ASP.NET Core Examples for Each Status Code

✅ 200 OK (GET)

```
[HttpGet("{id}")]
public IActionResult GetCustomer(int id)
{
    var customer = _repo.GetCustomer(id);
    if (customer == null)
        return NotFound();

    return Ok(customer); // 200 OK
}
```

✅ 201 Created (POST)

Use when a new record is created.

```
[HttpPost]
public IActionResult CreateCustomer(CustomerDto model)
{
    var customer = _repo.Add(model);

    return CreatedAtAction(nameof(GetCustomer),
        new { id = customer.Id }, customer);  // 201 Created
}
```

✅ 204 No Content (DELETE)

```
[HttpDelete("{id}")]
public IActionResult DeleteCustomer(int id)
{
    var isDeleted = _repo.Delete(id);

    if (!isDeleted)
        return NotFound();

    return NoContent(); // 204
}
```

❌ 400 Bad Request (Invalid input)

```
[HttpPost]
public IActionResult CreateAccount(AccountDto dto)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState); // 400 Bad Request

    ...
}
```

❌ 401 Unauthorized (Token missing)

Automatically returned by `[Authorize]` attribute when no token is provided.

```
[Authorize]
[HttpGet("balance")]
public IActionResult GetBalance()
{
    return Ok(5000);
}
```

❌ 403 Forbidden (Access denied)

User is authenticated but not allowed.

```
[Authorize(Roles = "Admin")]
[HttpGet("all-users")]
public IActionResult GetAllUsers()
{
    return Ok(_repo.Users);
}
```

If user is not an Admin → returns **403 Forbidden**.

❌ 404 Not Found

```
[HttpGet("{accNo}")]
public IActionResult GetAccount(int accNo)
{
    var account = _repo.Get(accNo);

    if (account == null)
        return NotFound(); // 404

    return Ok(account);
}
```

❌ 409 Conflict (Duplicate Entry)

```
[HttpPost]
public IActionResult RegisterUser(UserDto dto)
{
    if (_userRepo.EmailExists(dto.Email))
        return Conflict("Email already registered"); // 409

    ...
}
```

❌ 500 Internal Server Error

Use for unexpected exceptions (handled via middleware).

```
catch (Exception ex)
{
    return StatusCode(500, "Something went wrong"); // 500
}
```

Typically handled by **UseExceptionHandler** or **custom middleware**.

🎯 Status Code Mapping Table (Quick Revision)

Action	Success Status	Error Status
GET resource	200 OK	404 Not Found
POST create	201 Created	400 Bad Request, 409 Conflict
PUT update	200 OK / 204	400, 404
DELETE	204 No Content	404
Auth protected GET	200 OK	401, 403

📘 Best Practices for APIs

✔ Return correct status codes

Clients (Angular/React/Mobile apps) depend on them.

✔ Always validate incoming models

Return `BadRequest(ModelState).`

✔ Use exception-handling middleware

So API never crashes with unhandled 500s.

✔ Consistent response format

Example:

```
{
  "status": 400,
  "message": "Invalid email format"
}
```