
📘 record in C# (DTO Return Types)

1. What is a record?

A **record** is a reference type (like a class) designed primarily for **data**. It provides:

- **Value-based equality** (compares contents, not reference)
- Very concise syntax for DTOs
- Easy creation of immutable (read-only) models

Example DTO:

public record LoginRequest(string Username, string Password);

This single line creates:

- properties Username, Password
- a constructor that sets them
- value equality
- ToString() for debugging

---

2. Why record is used for DTOs (Minimal APIs)

DTOs are usually:

- simple data containers
- not heavy on behavior/methods
- passed around and serialized to JSON

record fits well because it is concise and safe.

---

3. record vs class (DTO perspective)

Feature	class	record
Primary use	Behavior + data	Data
Syntax	Longer	Short
Equality	Reference equality by default	Value equality by default
Immutability	Manual	Common by design
Good for DTOs	Yes	Yes (often better)

---

4. Two common forms of record

4.1 Positional record (most common in DTOs)

public record StudentDto(int Id, string Name, int Age);

Properties are **init-only** by default (immutable style).

Usage:

var dto = new StudentDto(1, "Ravi", 20);

4.2 Record with property block (more familiar style)

public record StudentDto

{

public int Id { get; init; }

public string Name { get; init; } = "";

public int Age { get; init; }

}

This looks similar to a class but still keeps record behavior.

---

5. Immutability and init

Records commonly use init setters:

- value is set during object creation
- after that, it cannot be changed

Example:

public record LoginResponse

{

public string Token { get; init; } = "";

public DateTime ExpiresUtc { get; init; }

}

---

6. with expression (record feature)

Records allow easy copying with modifications:

var s1 = new StudentDto(1, "Ravi", 20);

var s2 = s1 with { Age = 21 };

with creates a new object without mutating the old one.

---

7. JSON Serialization

ASP.NET Core serializes records to JSON the same way it does classes.

Example return:

app.MapGet("/login-sample", () =>

Results.Ok(new LoginResponse("token123", DateTime.UtcNow.AddMinutes(60), "admin", "Admin")));

---

8. Requirements

record was introduced in **C# 9** (.NET 5 era).
It is fully supported in .NET 6/7/8 minimal APIs.

---

9. Summary

- record is a modern type optimized for DTO-like data.
- It is concise and supports value equality.
- Positional records are a clean fit for request/response DTOs in Minimal APIs.
- Classes are still valid; record is an alternative with data-first benefits.
