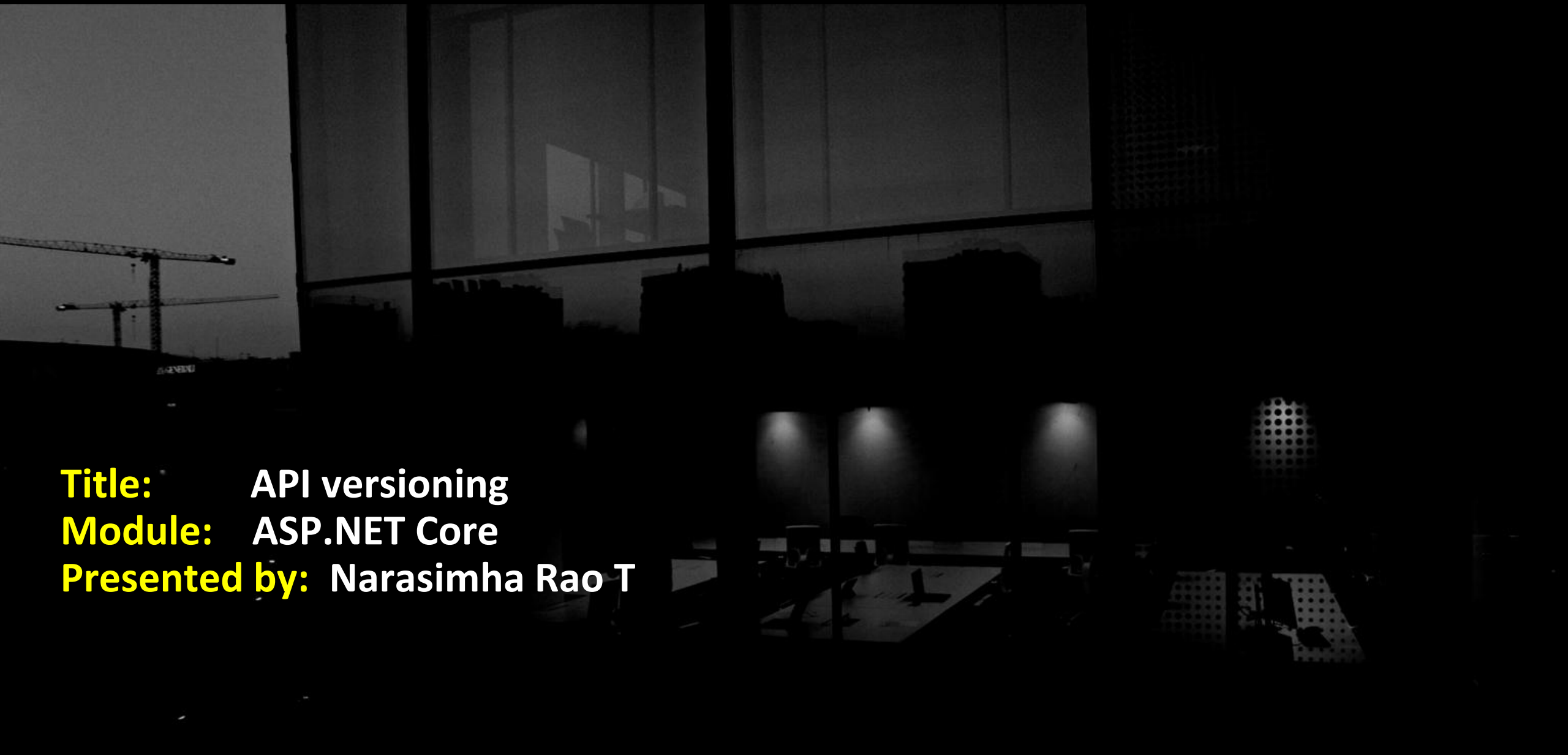
upGrad ENTERPRISE



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

upGrad ENTERPRISE

API Versioning and Testing with Swagger & Postman

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

Introduction to API Versioning

1. What is API Versioning?

- **Definition:**
  API versioning is the practice of managing and evolving an API without breaking existing client applications. It allows developers to introduce new features or modify behavior while maintaining backward compatibility.
- **Why API Versioning?**
  - Clients depend on stable APIs.
  - Applications evolve with time; breaking changes can impact consumers.
  - Supports multiple versions simultaneously.

- **Key Strategies for API Versioning:**
  - i. **URI Versioning (Path-based):**
    - Version included in the URL path.
    - Example: `/api/v1/products` or `/api/v2/products`
  - ii. **Query String Versioning:**
    - Version specified as a query parameter.
    - Example: `/api/products?api-version=1.0`

2. Using Microsoft.AspNetCore.Mvc.Versioning

ASP.NET Core provides the `Microsoft.AspNetCore.Mvc.Versioning` NuGet package to simplify versioning.

Installation:

```
Install-Package Microsoft.AspNetCore.Mvc.Versioning
Install-Package Microsoft.AspNetCore.Mvc.Versioning.ApiExplorer
```

Setup in Program.cs (or Startup.cs for older versions):

```
builder.Services.AddApiVersioning(options =>
{
    options.DefaultApiVersion = new ApiVersion(1, 0);
    options.AssumeDefaultVersionWhenUnspecified = true;
    options.ReportApiVersions = true;
    options.ApiVersionReader = ApiVersionReader.Combine(
        new QueryStringApiVersionReader("api-version"),
        new HeaderApiVersionReader("x-api-version"),
        new UrlSegmentApiVersionReader());
});
```

- `DefaultApiVersion` : Sets the default API version.
- `AssumeDefaultVersionWhenUnspecified` : Uses default version if none specified.
- `ReportApiVersions` : Adds headers to indicate supported API versions.
- `ApiVersionReader` : Determines how the version is read (query, header, URL).

3. How to Implement Versioning (Example)

Controller Implementation:

V1 Controller:

```
[ApiController]
[Route("api/v{version:apiVersion}/[controller]")]
[ApiVersion("1.0")]
public class ProductsController : ControllerBase
{
    [HttpGet]
    public IActionResult Get() => Ok(new[] { "Product A", "Product B" });
}
```

V2 Controller:

```
[ApiController]
[Route("api/v{version:apiVersion}/[controller]")]
[ApiVersion("2.0")]
public class ProductsController : ControllerBase
{
    [HttpGet]
    public IActionResult Get() => Ok(new[] { "Product A v2", "Product B v2" });
}
```

- When calling `/api/v1/products`, returns V1 result.
- When calling `/api/v2/products`, returns V2 result.

4. Testing with Swagger/OpenAPI for API Documentation

Swagger is a powerful tool for generating API documentation and testing APIs interactively.

Swashbuckle Setup:

Install the NuGet package:

```
Install-Package Swashbuckle.AspNetCore
```

5. Testing with Postman

- **Steps:**

  i. Download and install Postman.

  ii. Create a new request:

  - Method: GET
  - URL: `https://localhost:5001/api/v1/products` or `https://localhost:5001/api/v2/products`

  iii. Add versioning in:

  - Query param: `?api-version=1.0`
  - Header: `x-api-version: 2.0` (for header-based versioning)

- **Benefits of Postman:**
  - Easy testing of multiple API versions.
  - Ability to save collections for regression testing.

Conclusion:

- API versioning is essential for backward compatibility.
- ASP.NET Core makes versioning simple with `Microsoft.AspNetCore.Mvc.Versioning`.
- Swagger and Postman provide robust testing and documentation solutions.
- Always plan API versioning early to avoid breaking changes.

upGrad ENTERPRISE

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com