ASP.NET CORE RESTFUL SERVICES

Philip Japikse (@skimedec)
skimedec@outlook.com
CTO, Author, Teacher
Microsoft MVP, ASPInsider, PST, PSM II, PSD

https://github.com/skimedic/presentations

All slides copyright Philip Japikse http://www.skimedic.com
Phil.About()

- CTO/Chief Architect, Pintas & Mullins
- Professional Scrum Trainer (PSF, PSD)
- Speaker: http://www.skimedic.com/blog/page/Abstracts.aspx
- Microsoft MVP, ASPInsider, PST, PSM II, PSD
- Founder, Agile Conferences, Inc.
- http://www.cincydeliver.org
- President, Cincinnati .NET User’s Group
PLEASE MUTE YOUR CELL PHONES

attendees.Select(x => x.Phone.IsOn)
{
    x.Phone.EnableSilentMode();
}

attendees.where(x => x.Phone.Rings())
{
    x.Dispose();
}
DO YOU NEED TO VERSION?

- Your API is Public
- Your API needs updating
- Your API has more than one client OR you plan on adding more
- You want to plan for the future (but not gold plate)

- Clients need to count on services being stable over time
- Business needs to add new features and make changes
VERSIONING OPTIONS
**VERSION FORMATS**

- Services are versioned using the Major.Minor versioning scheme
  - 1.0, 2.0
- Services can opt for only the Major version - the “.0” is implied
  - v1 => v1.0
- Status (RC, Alpha, Beta, etc.) can be specified after the Minor version
  - 1.0-Alph
- Grouping using YYYY-MM-DD
  - 2018-06-12.1.0-RC

https://github.com/Microsoft/api-guidelines/blob/vNext/Guidelines.md
VERSIONING OPTIONS

- Embed the version after the service root
  - https://www.skimedec.com/api/v1.0/classes
- Use a query string parameter
  - https://www.skimedec.com/api/classes?api-version=1.0
- HTTP Headers (Not compliant with MS REST Guidelines)
  - api-version: 2.0
- Use Media Type (Not compliant, but generally accepted)
  - Content-Type: application/json;v=2.0 (or Accept)

https://github.com/Microsoft/api-guidelines/blob/vNext/Guidelines.md
API GUIDANCE

- Present a consistent user experience
- Guarantee stability of the REST APIs
- Do not change names or structures over time

https://github.com/Microsoft/api-guidelines/blob/vNext/Guidelines.md
VERSIONING GUIDANCE

- Be consistent with versioning mechanism (URL v. Query String)
- Indicate deprecated version(s)
- Update versions with breaking changes

https://github.com/Microsoft/api-guidelines/blob/master/Guidelines.md#12-versioning
ALL REQUIRE UPDATED VERSIONING

Breaking Changes –

- Removal or renaming APIs or API parameters
- Changes in the behavior of an existing API
- Changes in Error Codes and Fault Contracts

New Features

Anything that violates the Principle of Least Astonishment
PRINCIPLE OF LEAST ASTONISHMENT

- If a necessary feature has a high astonishment factor, it may be necessary to redesign the feature - 1984
- A component of a system should behave in a way that users expect
- For an API, function or method names intuitively match their behavior

GROUP VERSIONING

- Group Versioning is an optional feature
- Defined using the YYYY-MM-DD format
- Does not replace the Major.Minor version format

- Allows for logical grouping of API
- Developers can lookup a single version and use it across related end points

- Can cause confusion due to reusing versions
VERSIONING ASP.NET CORE WEB SERVICES
THE BARE MINIMUM

- Add Package Microsoft.AspNetCore.Mvc.Versioning
- Add call to services.AddApiVersioning in ConfigureServices (Startup.cs)
- Set option to report API versions
- Use ApiVersion Attributes on Controllers
  - ApiVersion, MapToApiVersion, ApiVersionNeutral, AdvertiseApiVersions
- [optional] Add route for URL version scheme
ADDING VERSIONING TO CONTROLLERS

Use the ApiVersion attribute to add versioning

```
//Query String and Media Type
[ApiVersion( "2.0" )]
[Route( "api/helloworld" )]
public class HelloWorld2Controller : Controller
{
    ...
}

//URL Versioning
[ApiVersion( "1.0" )]
[Route( "api/v{version:apiVersion}/[controller]" )]
public class HelloWorldController : Controller
{
    ...
}
```
VERSION INTERLEAVING

Use the ApiVersion/MapToApiVersion attributes to add versioning

```csharp
[ApiVersion( "2.0" )]
[ApiVersion( "3.0" )]
[Route( "api/v{version:apiVersion}/helloworld" )]
public class HelloWorld2Controller : Controller
{
    [HttpGet]
    public string Get() => "Hello world v2!";

    [HttpGet, MapToApiVersion( "3.0" )]
    public string GetV3() => "Hello world v3!";
}
```
DEPRECATED VERSIONS

Add Deprecated to the ApiVersion attribute

```csharp
[ApiVersion( "2.0" )]
[ApiVersion( "1.0", Deprecated = true )]
[Route( "api/[controller]" )]
public class HelloWorldController : Controller
{
    //omitted
}
```
VERSION NEUTRALITY

Use the ApiVersionNeutral attribute to expose an endpoint to all versions

```csharp
[ApiVersionNeutral]
[Route("api/v{version:apiVersion}/[controller]/[action]")] public class HealthController : Controller {
    [HttpGet]
    public string Ping() => "Ok";
}
```
REQUESTS AND VERSION INFORMATION

- Request Formats (querystring, header, URL):
  - QueryString (?api-version=1.0)
  - URL (api/v1.0/[Controller])
  - Media Type (Content-Type: application/json;v=2.0) || Accept
  - Header – must configure manually with ApiVersionReader
  - Note: All are customizable

- Getting Version requested:
  - HttpContext.GetRequestedApiVersion
  - Model Binding supported in 3.0+
GETTING THE REQUESTED VERSION INFORMATION

🚀 Use the GetRequestedApiVersion or ModelBinding to return the requested version information

```csharp
[ApiVersion("1.0")]
[ApiVersion("2.0")]
[Route("api/v{version:apiVersion}/[controller]/[action]"室]
public class DifferentVersionsController : Controller
{
    [HttpGet]
    public string RequestedApiVersion() =>
        JsonConvert.SerializeObject(HttpContext.GetRequestedApiVersion());

    [HttpGet] //3.0+
    public string Get(ApiVersion apiVersion)
        => "Controller = {GetType().Name}
        nVersion = {apiVersion}";
}
```
API VERSIONING OPTIONS
SETTING DEFAULT VERSION FOR REQUESTS

AssumeDefaultVersionWhenUnspecified
- Used when adding versioning to an existing API
- Returned version is configured with ApiVersionSelector

ApiVersionSelector defines the behavior for unspecified requests.
- Default – the configured default
- Constant – always selects the specified version
- Current/Lowest Implementation – greatest/lowest version number
DEFAULT VERSIONS FOR APICONTROLLERS

- DefaultApiVersion – Versions Controllers w/o ApiVersion attribute
- Configured default value is 1.0
- Can be set to another value

- Also used with AssumeDefaultVersionWhenUnspecified when Default is the ApiVersionSelector
The **IApiVersionReader** interface defines the behavior of how an API version is read in its raw, unparsed form from the current HTTP request.

Can update defaults for:
- **QueryStringApiVersionReader**
- **HeaderApiVersionReader**
- **MediaTypeApiVersionReader**
The ASP.NET/ASP.NET Core API Versioning project provides Swagger and Swashbuckle support.

1. Add SwaggerGen to Configure Services (Startup.cs)
2. Add Swagger and SwaggerUI to Configure (Startup.cs)
3. To add Swashbuckle, must leverage Swashbuckle Extensibility model
   - Implement an IOperationFilter and add to Swagger
4. Resolving conflicts:
   - `c.ResolveConflictingActions(apiDescriptions => apiDescriptions.First());`
SWAGGER/OPENAPI

- Swagger is a language-agnostic spec for describing REST APIs
- When donated to the OpenAPI Initiative was renamed to OpenAPI
- Provides interactive documentation, client SDK Generation, and API discoverability
- Swagger.json describes the service, SwaggerUI presents it on the web
- Options for ASP.NET Core
  - Swashbuckle.AspNetCore generates docs
  - Nswag generates docs + client code
DEFAULT IMPLEMENTATION DOESN’T SUPPORT VERSIONING

ASP.NET Core team supplies additional information on their GitHub page on how to extend Swashbuckle

Create new IOperationFilter

Create new IConfigureOptions<SwaggerGenOptions> implementation
Thank You!

Questions?

Contact Me
skimed@outlook.com
www.skimed.com/blog
www.twitter.com/skimedic


https://github.com/skimedic/presentations

https://github.com/skimedic/presentations