

The background of the slide features a pattern of thin, light blue wavy lines that create a sense of motion and depth. These lines are arranged in a series of concentric, slightly irregular curves that sweep across the frame from the bottom left towards the top right. The overall effect is a modern, fluid, and organic texture.

API VERSIONING IS FOR EVERYONE!



API Versioning



Versioning

- ❖ HTTP API
- ❖ gRPC Service
- ❖ GraphQL Server





The need for API versioning

- Maintain backwards compatibility
- Make changes to API without disrupting clients
- Allow API consumers to update at their own pace
- Phased introduction of breaking changes
- Reduce client-server coupling





When to version your API?

API versioning is not always
the answer

An aerial photograph of a rural landscape. A light-colored, winding road or path cuts through the scene. To the left of the road, there are large, rectangular fields of golden-brown color, likely harvested crops. To the right, there are green fields, some of which are divided into smaller sections. A small, dark, irregular shape, possibly a building or a tree, is visible near the intersection of the road. The overall scene is a typical agricultural landscape.

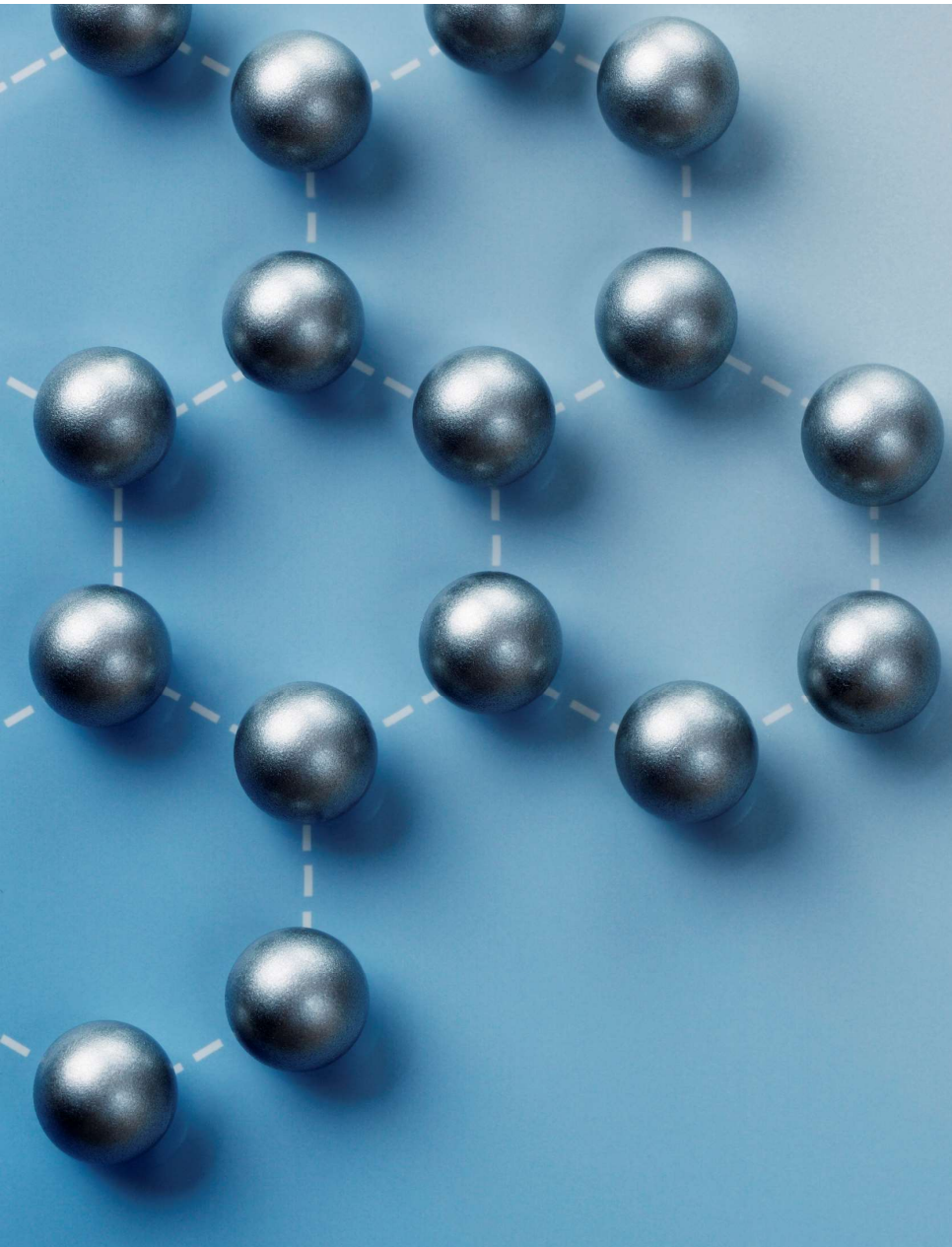
Let clients embrace or ignore changes



BREAKING CHANGES



- Changes in contract - Request, Response, Endpoints names
- Changes in application logic*
- Changes in error code or fault contracts
- Removing part of API
- Anything that violates the Principle of Least Astonishment



Version Number Strategy

- SemVer – *<Major.Minor.Patch>*
- Major Versions only – *<Major>*
- Major.Minor - *<Major.Minor>*
- Date of release as the version – *<yyyy-mm-dd>*
- GroupVersion, internally interpreted as a version - *<yyyy-mm-dd>*

Keep it consistent!





HTTP API



Resources



Reuse HTTP

HTTP Verb

Url

Request Body

Response Body

Status Code

GET

/weatherforecast
/weatherforecast/{id}



200OK/404 Not
Found

POST

/weatherforecast



201 Created

PUT

/weatherforecast/{id}



200OK/204 No
Content/ 202
Accepted

DELETE

/weatherforecast/{id}



204 No
Content/404 Not
Found

Reuse HTTP for Versioning



- Version by query string
- Version by header
- Version by media type
- Version by url



```
dotnet add package Asp.Versioning.Mvc --version 7.0.0
```

```
1 namespace ApiVersioningDemo.V1;
2
3 [ApiVersion( 1.0 )]
4 [ApiController]
5 [Route( "[controller]" )]
6 public class WeatherForecastController : ControllerBase
7 {
8     [HttpGet]
9     public string Get() {
10         //code implementation
11     }
12 }
13
```

- Controllers of same name with different `[ApiVersion(v)]` attributes
- Different namespaces

```
1 namespace ApiVersioningDemo.V2;
2
3 [ApiVersion( 2.0 )]
4 [ApiController]
5 [Route( "[controller]" )]
6 public class WeatherForecastController : ControllerBase
7 {
8     [HttpGet]
9     public string Get() {
10         //code implementation
11     }
12 }
13
```

- Controllers of different names with different `[ApiVersion(v)]` attributes - `nameController`, `name#Controller`
- Same namespace

```
1 namespace ApiVersioningDemo.V1;
2
3 [ApiVersion( 2.0 )]
4 [ApiController]
5 [Route( "weatherforecast" )]
6 public class WeatherForecast2Controller : ControllerBase
7 {
8     [HttpGet]
9     public string Get() {
10         //code implementation
11     }
12 }
13
```

```
1 namespace ApiVersioningDemo.V1;
2
3 [ApiVersion( 1.0 )]
4 [ApiController]
5 [Route( "[controller]" )]
6 public class WeatherForecastController : ControllerBase
7 {
8     [HttpGet]
9     public string Get(){
10         //code implementation
11     }
12 }
```



```
1  [ApiVersion( 1.0 )]
2  [ApiVersion( 2.0 )]
3  [ApiController]
4  [Route( "[controller]" )]
5  public class WeatherForecastController : ControllerBase
6  {
7      [HttpGet]
8      public string Get() {
9          //code implementation for v1
10     }
11
12     [HttpGet, MapToApiVersion(2.0)]
13     public string GetV2(){
14         //code implementation for v2
15     }
16 }
```

Version Interleaving



```
1 .AddApiVersioning(options =>
2     {
3         options.ApiVersionReader = new MediaTypeApiVersionReader("v");
4         options.AssumeDefaultVersionWhenUnspecified = true;
5         options.ApiVersionSelector = new CurrentImplementationApiVersionSelector( options );
6     } );
```



```
1 GET weatherforecast HTTP/2
2 Host: localhost
3 Accept: application/json;v=1.0
4
```



```
1 GET weatherforecast HTTP/2
2 Host: localhost
3 Accept: application/json;v=2.0
```




```
1  
2  .AddApiVersioning(options =>  
3      {  
4          options.ApiVersionReader = new UrlSegmentApiVersionReader();  
5      } );
```

Not possible to have a default version for your API via url segment



```
1 namespace ApiVersioningDemo.V1;
2
3 [ApiVersion( 1.0 )]
4 [ApiVersion( 2.0 )]
5 [ApiController]
6 [Route( "v{version:apiVersion}/{controller}" )]
7 public class WeatherForecastController : ControllerBase
8 {
9     [HttpGet]
10    public string Get() {
11        //code implementation for v1
12    }
13
14    [HttpGet, MapToApiVersion(2.0)]
15    public string GetV2(){
16        //code implementation for v2
17    }
18 }
19
```

A serene sunset scene over a calm ocean. The sun is a bright, glowing orb on the horizon, casting a shimmering reflection across the water's surface. The sky is a mix of soft pinks, oranges, and yellows, with a few wispy clouds on the right side. The foreground shows the gentle waves of the ocean meeting a sandy beach.

Deprecating Versions

API Versions can be marked deprecated



```
1 [ApiVersion(1.0, Deprecated = true)]
2 [ApiController]
3 [Route("[controller]")]
4
5 public class WeatherForecastController : ControllerBase
6 {
7     //code
8 }
```

Define Sunset HTTP header in the response to indicate when the API will disappear



```
1 options =>
2 {
3     options.ReportApiVersions = true;
4
5     options.Policies.Sunset(1).Effective(2024, 3, 1)
6         .Link("https://docs.api.com/policy.html?api-version=1.0")
7         .Title("API Policy")
8         .Type("text/html");
9 }
```


Remove the controller for the version



gRPC



Modern, high-performance, open-source Remote Procedure Call Framework

- 
- Google's implementation of RPC
 - Officially supported from .NET Core 3.0
 - Contract-based API development


```
syntax = "proto3";
```

```
//namespace for generated code  
option csharp_namespace = "gRPC.Basic";
```

```
//namespace of the protocol buffer messages, helps prevent namespace clashes  
package greet;
```

```
// The greeting service definition.  
service Greeter {  
    // rpc definitions  
    rpc SayHello (HelloRequest) returns (HelloReply);  
}
```

```
// The request message  
message HelloRequest {  
    string name = 1;  
}
```

```
// The response message  
message HelloReply {  
    string message = 1;  
}
```

@poornimanayar

```
// The greeting service definition.
```

```
service Greeter {
```

```
    // rpc definitions
```

```
    rpc SayHello (HelloRequest) returns (HelloReply);
```

```
}
```

Greeter.GreeterBase – abstract class

SayHello() – virtual method

```
// The request message
```

```
message HelloRequest {
```

```
    string name = 1;
```

```
}
```

HelloRequest – C# class

```
// The response message
```

```
message HelloReply {
```

```
    string message = 1;
```

```
}
```

HelloReply – C# class

```
// The greeting service definition.
```

```
service Greeter {
```

```
    // rpc definitions
```

```
    rpc SayHello (HelloRequest) returns (HelloReply);
```

```
}
```

Greeter.GreeterClient

SayHello()

```
// The request message
```

```
message HelloRequest {
```

```
    string name = 1;
```

```
}
```

HelloRequest

```
// The response message
```

```
message HelloReply {
```

```
    string message = 1;
```

```
}
```

HelloReply

```
<ItemGroup>  
    <Protobuf Include="Protos\greet.proto" GrpcServices="Server"/>  
</ItemGroup>
```

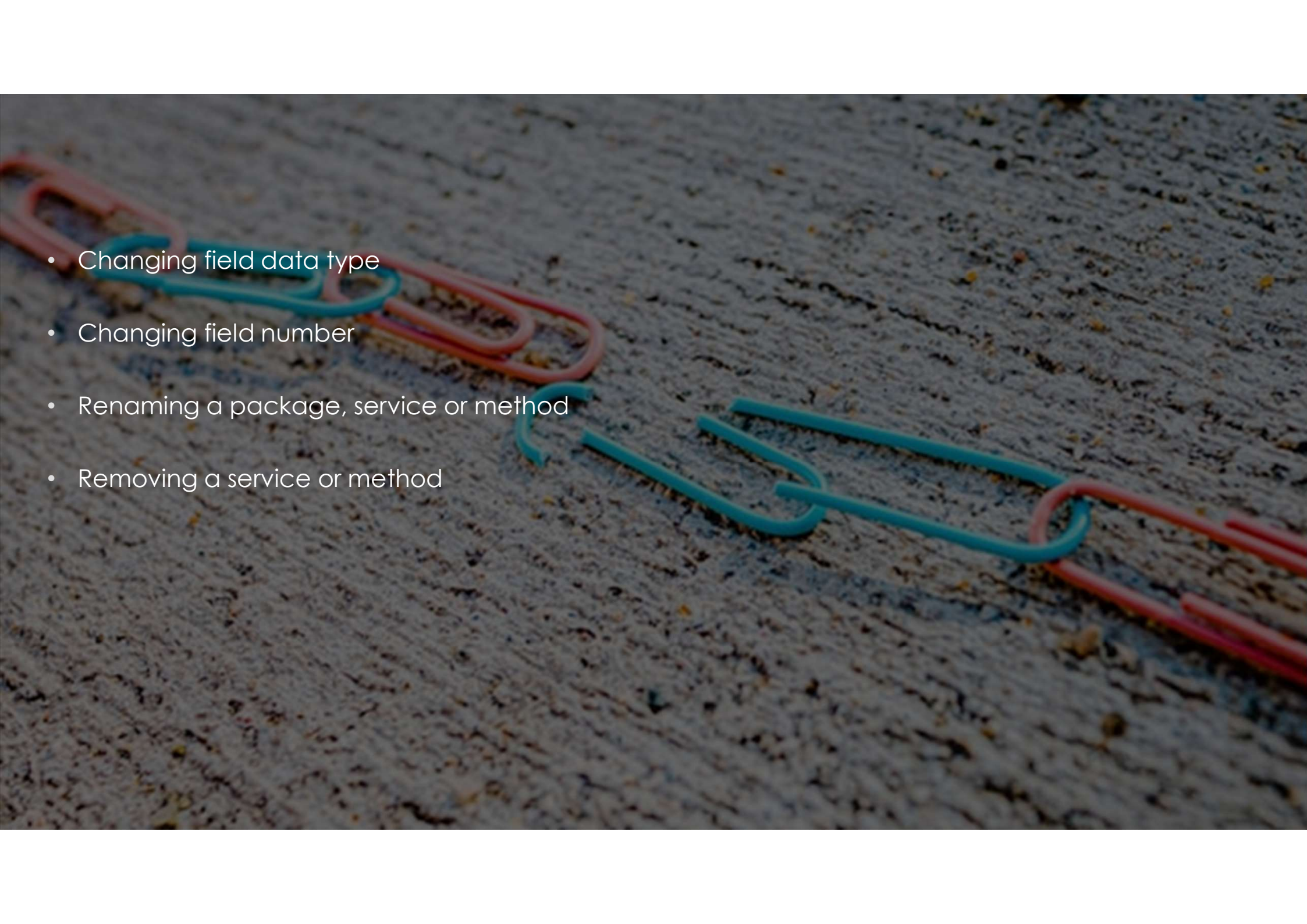
- Both(default)
- Server
- Client
- None

- Smaller and faster than JSON
- Transmitted in binary format
- Value of each field serialised against the unique identifier for that field



A white eggshell is shown cracked open, with the two halves separated. The text "BREAKING CHANGES" is overlaid in the center in a bold, black, sans-serif font. The background is a soft, light blue gradient.

BREAKING CHANGES

- 
- Changing field data type
 - Changing field number
 - Renaming a package, service or method
 - Removing a service or method

A serene sunset scene over a calm ocean. The sun is a bright, glowing orb on the horizon, casting a shimmering reflection across the water's surface. The sky is a mix of soft pinks, oranges, and yellows, with a few wispy clouds on the right side. The foreground shows the gentle waves of the ocean meeting a sandy beach.

Deprecating Versions



```
1  // The request message containing the user's name.
2  message HelloRequest {
3      // name to say hello to
4      string name = 1 [deprecated = true];
5  }
```




GraphQL

GraphQL is a query language



Supported by a runtime for fulfilling your queries

A hand is holding a Rubik's cube against a solid blue background. The cube is partially solved, with red, blue, and yellow faces visible. The text is overlaid on the image, centered horizontally and slightly above the middle vertically.

Single, smart endpoint that takes in complex queries and builds the data output

```
{
  hero {
    name
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        }
      ]
    }
  }
}
```

- Clients specify the shape of the data they need
- Server responds with the exact same data graph as the query
- Query related entities

A close-up, high-angle shot of a golden crown. The crown is ornate, featuring multiple fleur-de-lis ornaments along its top edge. It is positioned on a light-colored, textured surface, which causes it to cast a distinct, dark shadow to its left. The lighting is warm and directional, highlighting the metallic sheen of the crown and the intricate details of its design. The background is a soft, out-of-focus light color.

Client-focused, experience driven



Various implementations



Hot Chocolate

GraphQL.NET

SCHEMA

Object Types

*Representation of
object*

Queries

Read data

Mutations

*Insert/Update/
Delete data*

Subscriptions

Pushed Updates

Types

FIELDS

Name

Name of the field

Type

Type of the field

Resolvers

*Methods that help
fields resolve to data*

Creating Types Using Hot Chocolate



Object Type

Public properties = Fields

```
1 public class Place
2 {
3     public string Country { get; set; }
4
5     public string City { get; set; }
6 }
```

"get" = Resolver

Query

"Places" field

```
1 public class Query
2 {
3     public List<Place> GetPlaces([Service] PlaceRepository placeRepository)
4         => placeRepository.Places.ToList();
5
6 }
7
```

Resolver

Startup

Add a graphql server

Map endpoint “/graphql”

```
1  builder.Services
2      .AddSingleton<PlaceRepository>()
3      .AddGraphQLServer()
4      .AddQueryType<Query>();
5
6
7  var app = builder.Build();
8
9  //middleware for Routing, CORS...
10
11 //default endpoint /graphql
12 app.MapGraphQL();
13
14 app.Run();
```




GraphQL is version free



Multiple schemas on same server

Extend Query

“RandomPlace” field

```
1 public class NewQuery : Query
2 {
3     public Place GetRandomPlace([Service] PlaceRepository placeRepository, string place)
4     {
5         var random = placeRepository.Random.Next(0, placeRepository.Places.Count);
6         return placeRepository.Places[random];
7     }
8 }
```

Startup

Add a new server
"newgraphql"

Map a new endpoint
"/newgraphql"

```
1  builder.Services
2      .AddSingleton<PlaceRepository>()
3      .AddGraphQLServer()
4      .AddQueryType<Query>()
5      .AddGraphQLServer("newgraphql")
6      .AddQueryType<NewQuery>();
7
8
9  var app = builder.Build();
10
11  //middleware for Routing, CORS...
12
13  //default endpoint /graphql
14  app.MapGraphQL();
15
16  //new endpoint /newgraphql
17  app.MapGraphQL("/newgraphql", "newgraphql");
18
19  app.Run();
```

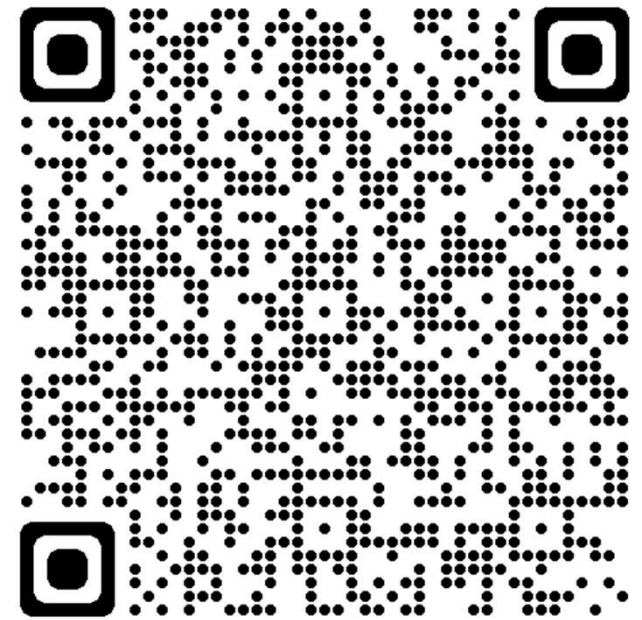


```
1  
2  [GraphQLDeprecated("Use the `capital` field instead")]  
3  public string City { get; set; }
```



Documentation

- Elements, Redoc, Postman, Swaggerhub
- GenDocu for gRPC Service
- SpectaQL, MagiDoc, DociQL, GraphDoc for GraphQL



A close-up photograph of a spiral-bound notebook. The notebook is open to a page with horizontal lines. A silver pen with a textured grip is resting on the page. The text "Key Takeaways" is overlaid in the center of the image. The spiral binding is visible on the left side.

Key Takeaways

A wooden gavel with a cylindrical head and a long handle, resting on a matching wooden block. The gavel is positioned diagonally across the frame, with its head on the left and handle extending towards the right. The background is a light, textured surface.

Not a book of rules!

Strategy & Best Practice



- Always consider a path of no/least breaking changes
- Adopt API Versioning for breaking changes
- Keep your versioning strategy consistent
- Share your API Contract
- Document! Document! Document!
- Communicate! Communicate ! Communicate!
- Sunset old API versions regularly
- Try and adopt an api versioning strategy early on in your project

API CLIENTS ARE USERS!



<https://bit.ly/version-apis>

@poornimanayar

