.NET Core Web APIs.

Class 2 - Hey! Give me a decent API

Who's this weird guy?

ANGEL GARCIA

Cloud and Mobile Developer @hMobile

@_AngelGarcia13

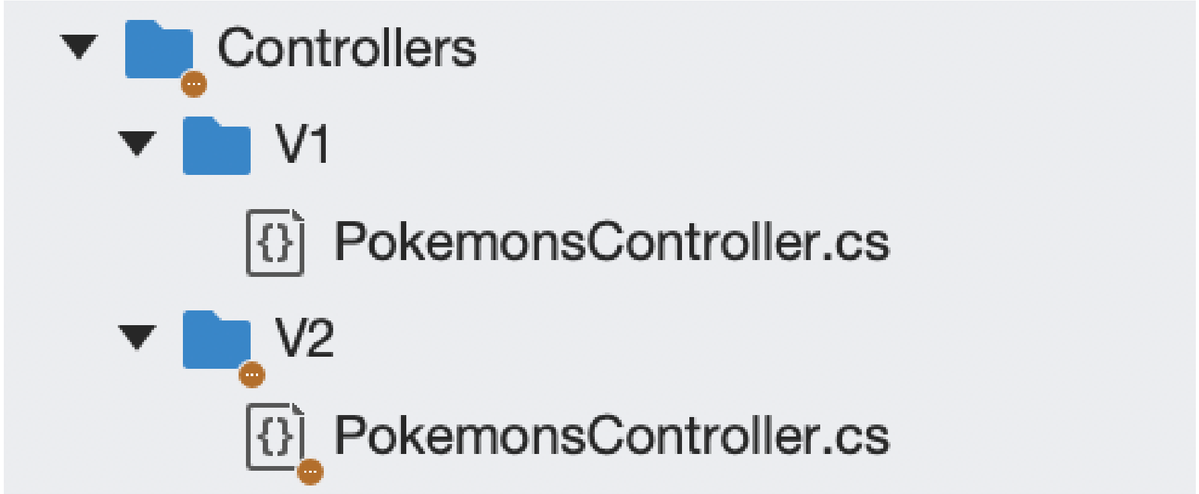
Class 2

- VERSIONING OUR API.
- SORTING/SEARCHING AND PAGINATING RESULTS.
- CACHING.
- CONTENT NEGOTIATION.

VERSIONING OUR API

API versioning in ASP.NET Core is a technique by which different clients can get different implementations of the same Controller based on the request or the URL.

Using Nuget Package Microsoft.AspNetCore.Mvc.Versioning we can start configuring the versioning options.



SEARCHING

```
// GET: api/Pokemons
[HttpGet]
public IEnumerable<Pokemon> GetPokemons(string searchString)
{
    //If we were pointing to a DB using EF this should be an IQueryable
    var pokemonsQueryResult = PokemonsMockDatabase.GetPokemons().AsQueryable();
    //Searching (non case-sensitive)
    if (!String.IsNullOrEmpty(searchString))
    {
        pokemonsQueryResult = pokemonsQueryResult.Where(s => s.Name.ToUpper().Contains(searchString.ToUpper())
                              || s.Code.ToUpper().Contains(searchString.ToUpper()));
    }
    return pokemonsQueryResult;
}
```

SORTING

```
//Sorting
pokemonsQueryResult = sortBy switch
{
    "name" => pokemonsQueryResult.OrderBy(s => s.Name),
    "name_desc" => pokemonsQueryResult.OrderByDescending(s => s.Name),
    "code" => pokemonsQueryResult.OrderBy(s => s.Code),
    "code_desc" => pokemonsQueryResult.OrderByDescending(s => s.Code),
    _ => pokemonsQueryResult.OrderBy(s => s.Code),
};
```

PAGINATING (1/2)

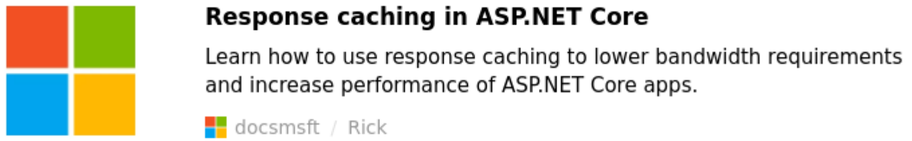
```
namespace PokeAPI.Helpers
{
    public class PaginatedList<T> : List<T>
    {
        public int PageIndex { get; private set; }
        public int TotalPages { get; private set; }

        public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)
        {
            PageIndex = pageIndex;
            TotalPages = (int)Math.Ceiling(count / (double)pageSize);
            this.AddRange(items);
        }

        public bool HasPreviousPage
        {
            get { return (PageIndex > 1); }
        }

        public bool HasNextPage
        {
            get { return (PageIndex < TotalPages); }
        }

        public static PaginatedList<T> Create(
            IQueryable<T> source, int pageIndex, int pageSize)
        {
            var count = source.Count();//With EF .CountAsync()
            var items = source.Skip((pageIndex - 1) * pageSize)
                .Take(pageSize).ToList();//With EF .ToListAsync()
            return new PaginatedList<T>(items, count, pageIndex, pageSize);
        }
    }
}
```

PAGINATING (2/2)

```
//Paging
int pageSize = 10;
var paginatedResult = PaginatedList<Pokemon>.Create(
    pokemonsQueryResult, pageIndex ?? 1, pageSize);
return paginatedResult;
```

CACHING RESPONSES

Response caching reduces the number of requests a client or proxy makes to a web server. Response caching also reduces the amount of work the web server performs to generate a response. Response caching is controlled by headers that specify how you want client, proxy, and middleware to cache responses.



CONTENT NEGOTIATION

Content negotiation occurs when the client specifies an Accept header. The default format used by ASP.NET Core is JSON.

XML formatters implemented using XmlSerializer are configured by calling AddXmlSerializerFormatters:

C#

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers()
        .AddXmlSerializerFormatters();
}
```

To restrict the response formats, apply the [Produces] filter. Like most Filters, [Produces] can be applied at the action, controller, or global scope:

C#

```
[ApiController]
[Route("[controller]")]
[Produces("application/json")]
public class WeatherForecastController : ControllerBase
{
```