Day 21 - Angular & Asp in 21 days Handouts

Status

Day 21: Filtering Lists

To pass filter parameters to the server you shouldn't pass them in the body as a payload.

Since you send an HttpGet request for listing items, pass the filters as a concatenated query string.

- You should also set the `FromQuery` attribute to the parameters of the action in the controller.
- Configure the Swagger to describe the parameters in the camel case format before regenerating the API client services.

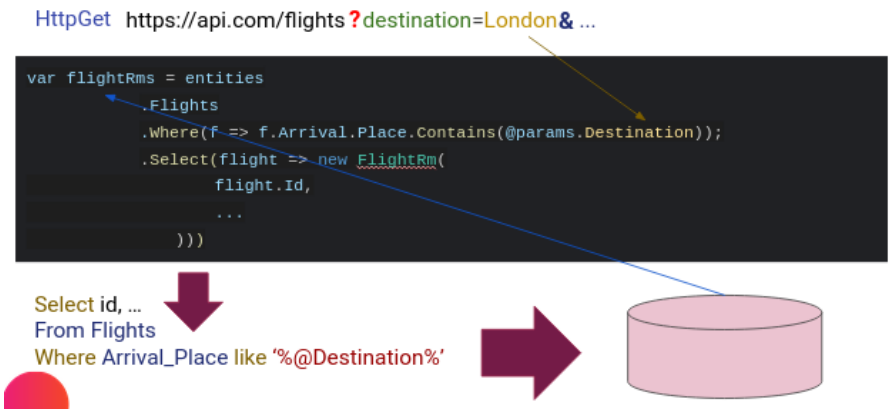
```
builder.Services.AddSwaggerGen(c =>
{
    c.DescribeAllParametersInCamelCase();
    ...
});
```

Linq

Linq is an abstract language. The abstractness enables developers to reuse their knowledge to create queries that can be executed on different types of data sources.

The syntax is almost the same whether you want to transform the items of a list or create a query and run it on a database.

Linq to entities creates a query using an Abstract Syntax Tree and passes it to the database to filter a table. It finally translates the query result back to objects.



- Use the contains extension method to partially match strings

```
"Tutuorials EU".Contains("EU");
```

- Hover over method names and learn their signature and what they do from the hints.
  - The `Where` method returns `IQueryable` for example. It's important because you may want to decide which filters to apply at run time.

```
IQueryable<Flight> flights = entities.Flights;

if (!string.IsNullOrWhiteSpace(@params.Destination))
    flights = flights.Where(f => f.Arrival.Place.Contains(@params.Destination));

if (!string.IsNullOrWhiteSpace(@params.Source))
    flights = flights.Where(f => f.Departure.Place.Contains(@params.Source));

if (@params.FromDate != null)
    flights = flights.Where(f => f.Departure.Time >= @params.FromDate.Value.Date);//Start of the day

if (@params.ToDate != null)
    flights = flights.Where(f => f.Departure.Time >= @params.ToDate.Value.Date.AddDays(1).AddTicks(-1)); //End of the day

if (@params.NumberOfPassengers != 0 && @params.NumberOfPassengers != null)
    flights = flights.Where(f => f.RemainingNumberOfSeats >= @params.NumberOfPassengers);
else
    flights = flights.Where(f => f.RemainingNumberOfSeats >= 1);// Filter out the fully booked flights

return new(flights.Select(flight => new FlightRm(
              flight.Id,
              flight.Airline,
              flight.Price,
              new TimePlaceRm(flight.Departure.Place.ToString(), flight.Departure.Time),
              new TimePlaceRm(flight.Arrival.Place.ToString(), flight.Arrival.Time),
              flight.RemainingNumberOfSeats
          )));
```