Day 5: - Angular & Asp in 21 days Handouts

Status

Day 5: REST API

REST stands for REpresentational State Transfer. It’s a set of formal specifications for specifying web services. We use Asp.net Web API to create REST APIs. Asp.net Web API sends and receives data in JSON format by default. JSON stands for Javascript Object Notation.

It’s a programming language agnostic textual format for data structures.

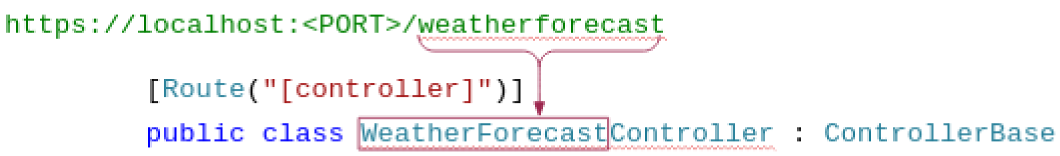
Calling a route of a REST API like `https://exampl.com/resource` may return such a JSON object:

```
{
   "key": "value",
   "number": 1,
   "anObject": {
       "internalKey": "Internal Value"
   },
   "AnArray": [
       {
           "item1Key": "1st value"
       },
       {
           "item2Key": "2nd value"
       },
       {
           "item3Key": "3rd value"
       }
   ]
}
```

ASP.Net web API enables us to create web services that conform to REST API specifications. When we run the application, the framework exposes all controllers and their actions to be accessible on HTTP.

ASP.Net web API contains a default controller named WeatherForecastController when you create a new project. The controller contains a Get action (method in a controller) by default which returns a list of items. To see the items the Get method returns call the action's address from the browser: `https://localhost:<port>/weatherforecast`

This is how the framework routes it:



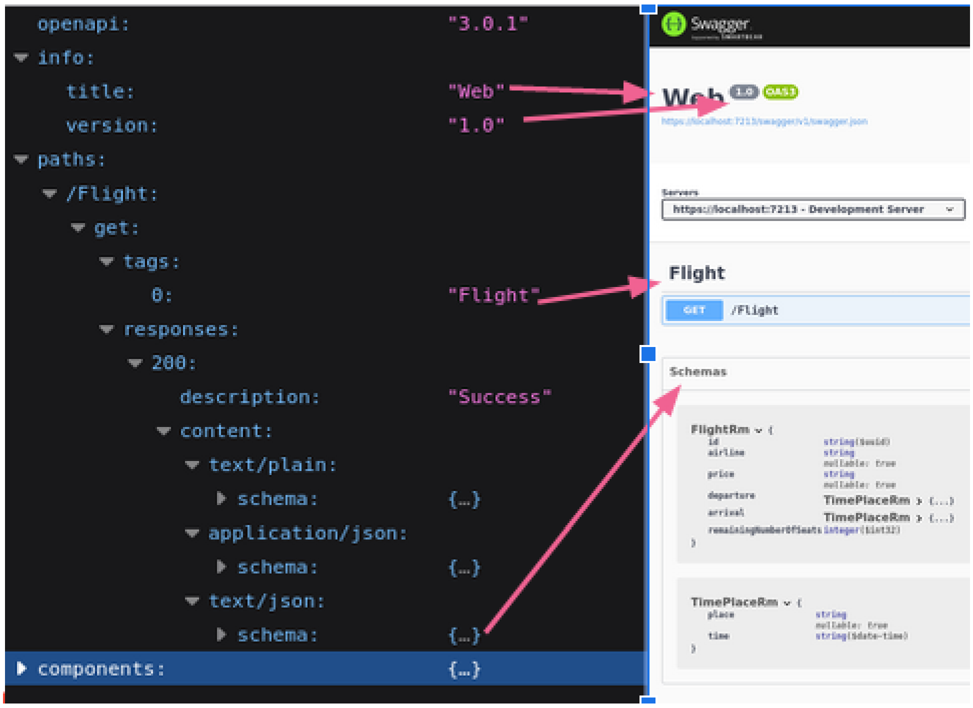
```
https://localhost:<PORT>/weatherforecast

        [Route("[controller]")]
        public class WeatherForecastController : ControllerBase
```

Open API

SwaggerUI is an auto-generated general-purpose user interface for interacting with REST APIs.

Open API Specification defines a public contract for documenting REST APIs. A small library generates the Open API spec from our Web API app and then translates the generated Open API document into an interactive user interface.



```
openapi:                             "3.0.1"
▾ info:
    title:                           "Web"
    version:                         "1.0"
▾ paths:
  ▾ /Flight:
    ▾ get:
      ▾ tags:
          0:                         "Flight"
      ▾ responses:
        ▾ 200:
            description:             "Success"
          ▾ content:
            ▾ text/plain:
              ▸ schema:              {…}
            ▾ application/json:
              ▸ schema:              {…}
            ▾ text/json:
              ▸ schema:              {…}
▸ components:                        {…}
```

Swagger
Web 1.0 OAS3
https://localhost:7213/swagger/v1/swagger.json
Servers
https://localhost:7213 - Development Server
Flight
GET /Flight
Schemas

```
FlightRm ⌄ {
   id                         string($uuid)
   airline                    string
                              nullable: true
   price                      string
                              nullable: true
   departure                  TimePlaceRm › {...}
   arrival                    TimePlaceRm › {...}
   remainingNumberOfSeats     integer($int32)
}

TimePlaceRm ⌄ {
   place                      string
                              nullable: true
   time                       string($date-time)
}
```

C# Records

Records are like classes but they are immutable.

```
public record TimePlaceRm(string Place, DateTime Time);
```

You cannot change the value of its fields once it’s created.

```
var original = new TimePlaceRm(Place: "Los Angeles", Time: DateTime.Today);
original.Place = "Berlin";x
```

However, you can use a record as a prototype to create a new one with different field values:

```
var copy = original with { Place = "Berlin"};✓
```