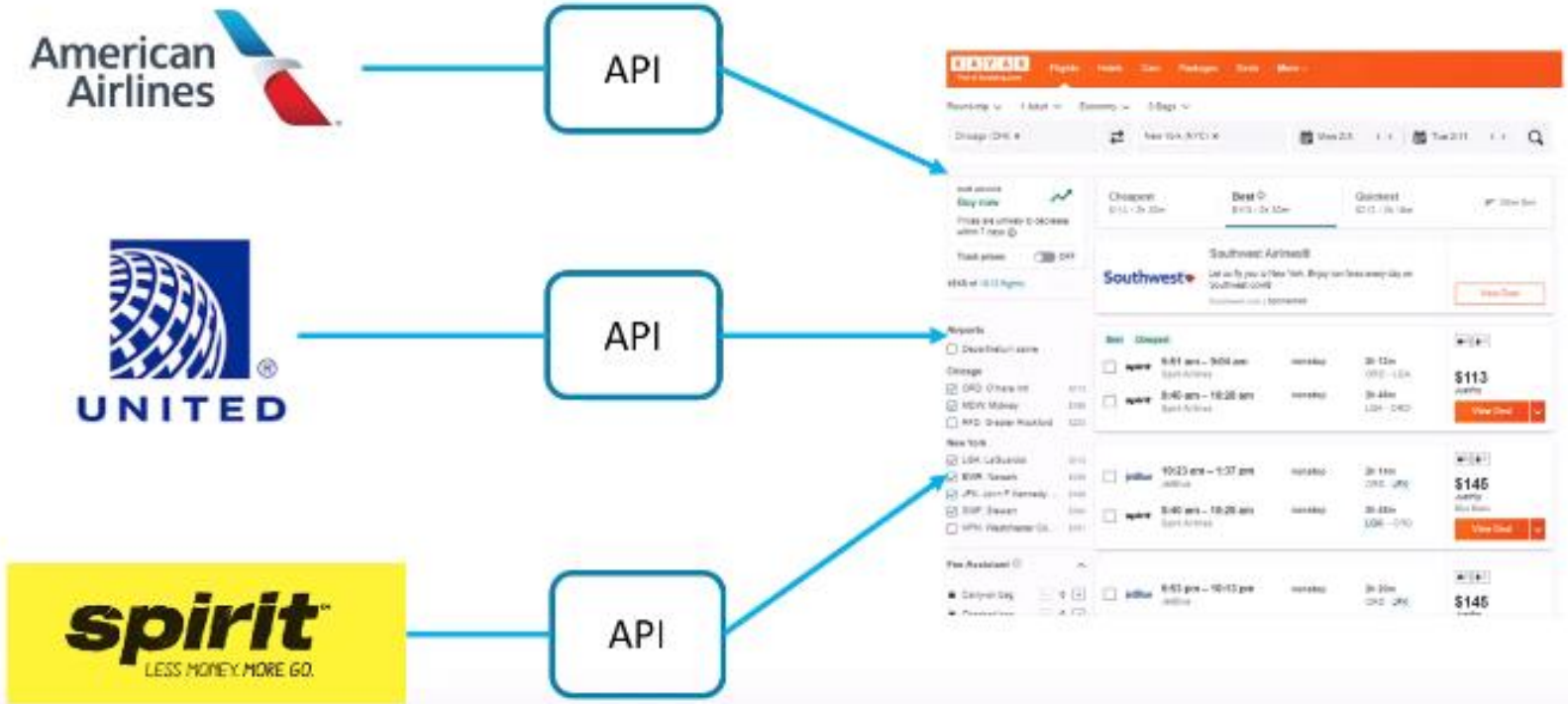
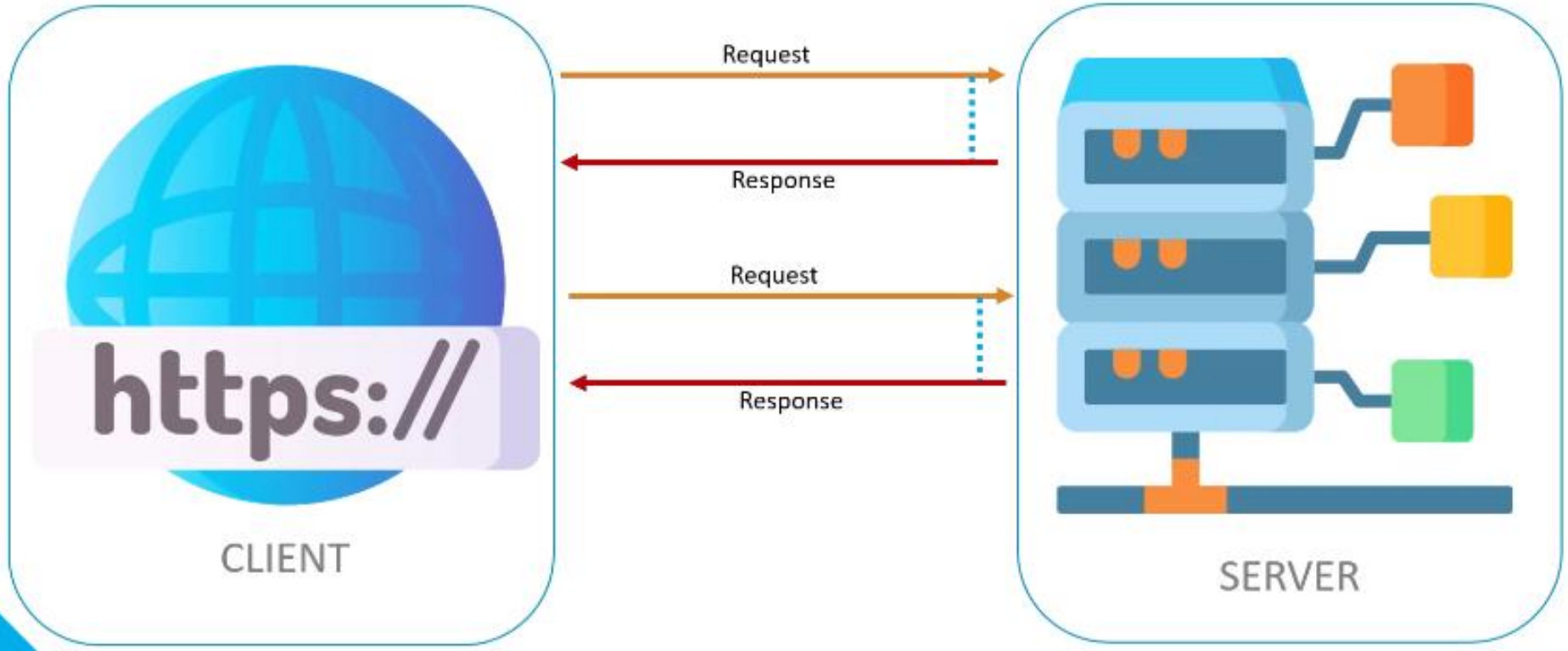
Minimal API Project

- 1 Remove Weather related files

HOW API WORKS?

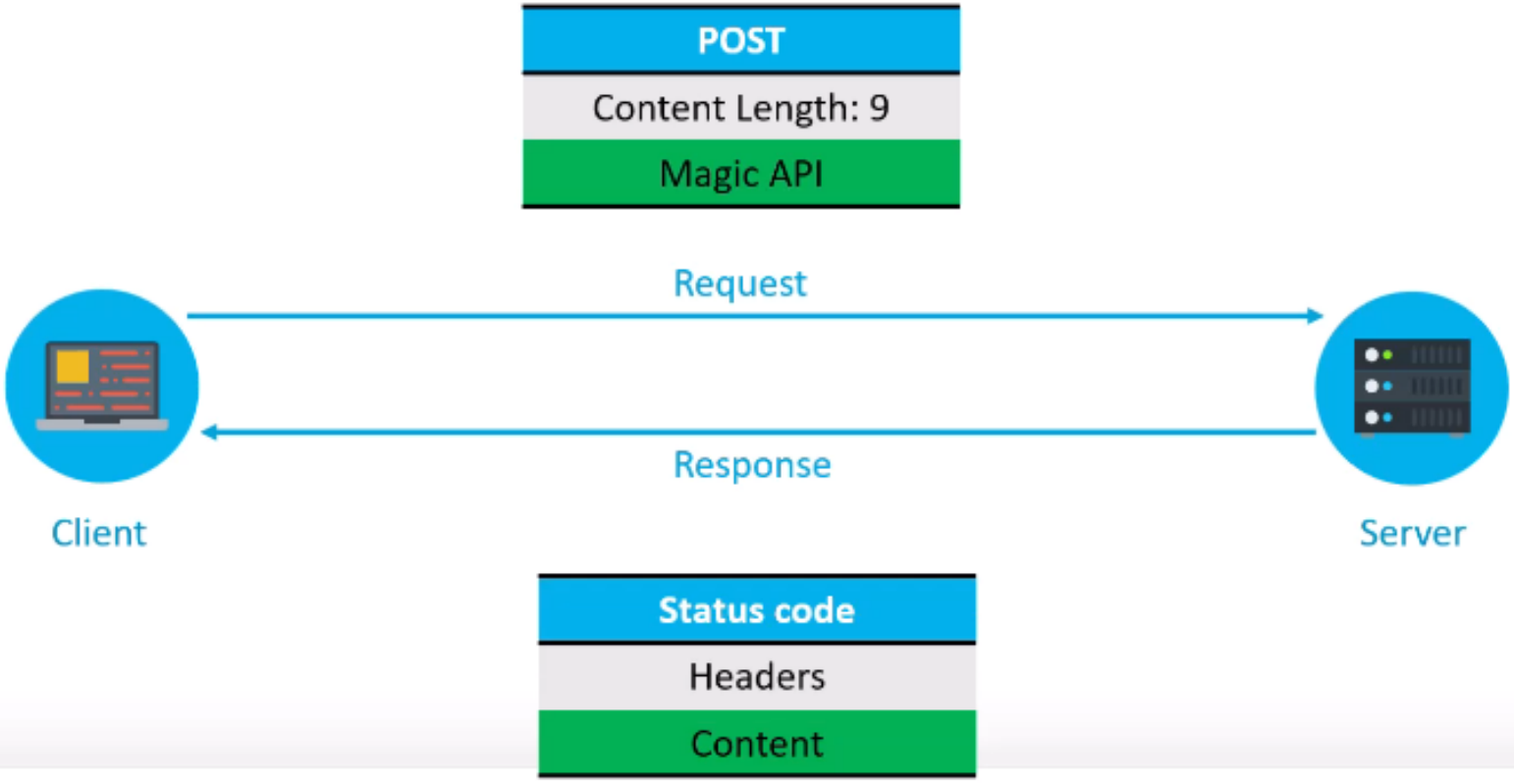


API



HOW HTTP WORKS?

HOW HTTP WORKS?



THE REQUEST OBJECT

HTTP Verbs / Actions

- **GET** : Fetches/Requests Resource
- **POST**: Creates/Inserts Resource
- **PUT** : Updates/Modifies Resource
- **PATCH** : Updates/Modifies Partial Resource
- **DELETE** : Deletes/Removes Resource
- More verbs...

THE REQUEST OBJECT

Request's Metadata

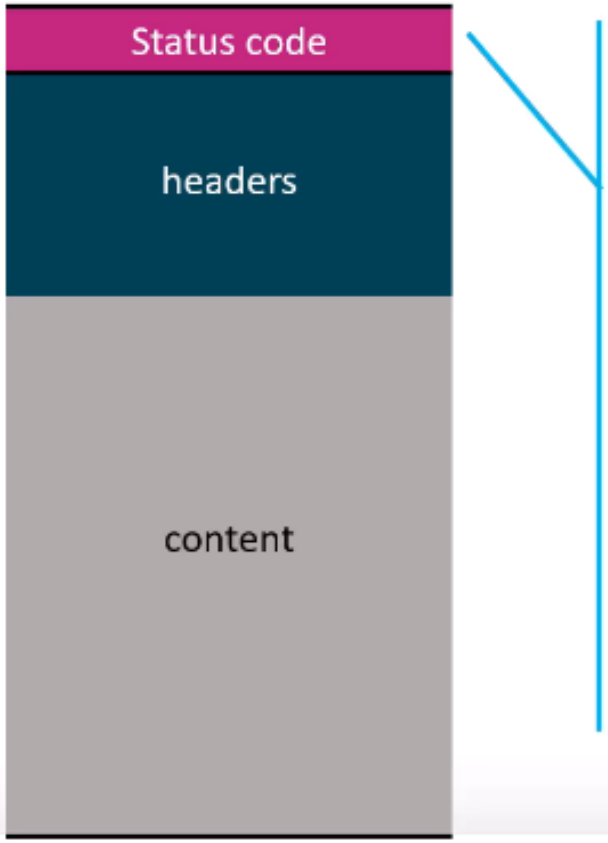
- **Content Type** : Content's Format
- **Content Length** : Size of the Content
- **Authorization** : Who is making the request
- **Accept** : What are the accepted type(s)
- More headers…

THE REQUEST OBJECT

Request's Content

- HTML, CSS, XML, JSON
- Information for the request.
- Blobs etc.

THE RESPONSE OBJECT



Status Codes for Operation Result

- **100-199:** Informational
- **200-299:** Success
  - 200 – OK
  - 201 – Created
  - 204 – No Content
- **300-399:** Redirection
- **400-499:** Client Errors
  - 400 – Bad Request
  - 404 – Not Found
  - 409 – Conflict
- **500-599:** Server Errors
  - 500 – Internal Server Error

THE RESPONSE OBJECT

Response's Metadata

- **Content Type** : Content's Format
- **Content Length** : Size of the Content
- **Expires** : When is this invalid
- More headers...

HTTP Verb	API	Action	Request	Response
GET	localhost:7001/api/coupon	Get all coupons	n/a	List<Coupon>
GET	localhost:7001/api/coupon/{id}	Get one coupon	n/a	Coupon
POST	localhost:7001/api/coupon	Create coupon	Coupon	Created Coupon
PUT	localhost:7001/api/coupon/{id}	Update Coupon	Coupon	n/a
DELETE	localhost:7001/api/coupon/{id}	Delete Coupon	n/a	n/a

1. Add Controller – First Endpoint

Add Controller [right click on controller folder]

```
[Route("/api/students")]
//[Route("api/[controller]")]
[ApiController]
0 references
public class HomeController : Controller
{
    [HttpGet]
    0 references
    public List<Student> GetStudents()
    {
        return new List<Student>()
        {
            new Student(){ Id=1 , Name = "Amol"},
            new Student(){ Id=2 , Name = "Sachin"},
            new Student(){ Id=3 , Name = "Rahul"},
            new Student(){ Id=4 , Name = "Ramesh"},
            new Student(){ Id=5 , Name = "Rohit"},
        };
    }
}
```

2. Add DTO - StudentDTO

```
0 references
public class Student
{
    0 references
    public int Id { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public DateTime CreatedDate { get; set; }
}
```

```
7 references
public class StudentDTO
{
    5 references
    public int Id { get; set; }
    5 references
    public string Name { get; set; }
}
```

3. Create Student Repo

```
public class StudentData
{
    public static List<StudentDTO> stds = new List<StudentDTO>()
    {
            new StudentDTO(){ Id=1 , Name = "Amol"},
            new StudentDTO(){ Id=2 , Name = "Sachin"},
            new StudentDTO(){ Id=3 , Name = "Rahul"},
            new StudentDTO(){ Id=4 , Name = "Ramesh"},
            new StudentDTO(){ Id=5 , Name = "Rohit"},
    };
}
```

4. Find Student By Id

```
[Route("/api/students")]
//[Route("api/[controller]")]
[ApiController]
0 references
public class HomeController : Controller
{
    [HttpGet]
    0 references
    public List<StudentDTO> GetStudents()
    {

        return StudentData.stds;
    }
    [HttpGet("{id:int}")]
    0 references
    public StudentDTO GetStudent(int Id)
    {
        return StudentData.stds.FirstOrDefault(s=>s.Id==Id);
    }
}
```

5. Add Response Codes

```
0 references
public ActionResult<List<StudentDTO>> GetStudents()
{

    return Ok(StudentData.stds);
}
[HttpGet("{id:int}")]
[ProducesResponseType(200)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound )]
0 references
public ActionResult<StudentDTO> GetStudent(int Id)
{
    if(Id== 0)
    {
        return BadRequest();
    }
    var std = StudentData.stds.FirstOrDefault(s=>s.Id==Id);
    if (std == null)
    {
        return NotFound();
    }
    else
    {
        return Ok(std);
    }
}
```

5. Add Post Method

```
[HttpPost]
[ProducesResponseType(200)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
public ActionResult<StudentDTO> CreateStudent([FromBody] StudentDTO student)
{
    if (student == null)
    {
        return BadRequest();
    }
    if (student.Id > 0)
    {
        return StatusCode(StatusCodes.Status500InternalServerError);
    }
    student.Id = StudentData.stds.OrderByDescending(s => s.Id).FirstOrDefault().Id + 1;
    StudentData.stds.Add(student);
    return Ok(student);
}
```

6. Add Model Validation

```
public class StudentDTO
{
    10 references
    public int Id { get; set; }

    [Required]
    [MaxLength(100)]
    5 references
    public string Name { get; set; }
}
```

```
if (!ModelState.IsValid)
{
    return BadRequest();
}
```

7. Add Custom Validation

```
if (!ModelState.IsValid)
{
    return BadRequest();
}
if ( StudentData.stds.Where(s=>s.Name == student.Name).ToList().Count() > 0 )
{
    ModelState.AddModelError("CustomError", "Student Already Exist");
    return BadRequest(ModelState);
}
```

9. Update Operation

```
[HttpPut("{Id:int}", Name = "UpdateStudent")]
[ProducesResponseType(204)]
[ProducesResponseType(404)]
[ProducesResponseType(400)]
0 references
public ActionResult UpdateStudent(int Id , [FromBody] StudentDTO dTO)
{
    if (Id == 0 || dTO==null)
    {
        return BadRequest();
    }
    var student = StudentData.stds.FirstOrDefault(s => s.Id == Id);
    if (student == null)
    {
        return NotFound();
    }

    student.Name = dTO.Name;
    return NoContent();
}
```

8. Delete Operation

```
[HttpDelete("{Id:int}",Name ="DeleteStudent")]
[ProducesResponseType(204)]
[ProducesResponseType(404)]
0 references
public ActionResult DeleteStudent ( int Id)
{
    if(Id==0)
    {
        return BadRequest();
    }
    var student = StudentData.stds.FirstOrDefault(s => s.Id == Id);
    if (student == null)
    {
        return NotFound();
    }
    StudentData.stds.Remove(student);
    return NoContent();
}
```

10. Inject Logger

```
private readonly ILogger<HomeController> logger;

0 references
public HomeController(ILogger<HomeController> logger)
{
    this.logger = logger;

}
```

10. Inject Logger

```
private readonly ILogger<HomeController> logger;

0 references
public HomeController(ILogger<HomeController> logger)
{
    this.logger = logger;

}
```

11. Add Nuget Packages

Microsoft.EntityFrameworkCore.SqlServer by aspnet, dotnetframework, EntityFramework, Mi 9.0.1
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by aspnet, dotnetframework, EntityFramework, Microso 9.0.1
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

12. Modify Student Class

```
0 references
public class Student
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    0 references
    public int Id { get; set; }
    0 references
    public string Name { get; set; }

    0 references
    public double Marks { get; set; }

    0 references
    public DateTime CreatedDate { get; set; }
    0 references
    public DateTime UpdatedDate { get; set;
}
```

13. Add Connection String in appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultSQLConnection": "Data Source=LAPTOP-FHLR08P2;Initial Catalog=dbd;Integrated Securi
  }
}
```

14. Add ConnectionString to Services [program.cs]

```
// Add services to the container.
builder.Services.AddDbContext<ApplicationDBContext>(option =>
{
    option.UseSqlServer(builder.Configuration.GetConnectionString("DefaultSQLConnection"));

});
```

14. Add DBContext class and DBSet

```
3 references
public class ApplicationDBContext:DbContext
{
    0 references
    public ApplicationDBContext(DbContextOptions<ApplicationDBContext> options) : base(options) { }
    0 references
    public DbSet<Student> Students { get; set; }
}
```

15. Apply Migrations

Add Migration <name>

Update-Database

16. Inject DBContext in Controller

```
ApplicationDBContext dbContext;

0 references
public HomeController(ApplicationDBContext dbContext)
{
    this.dbContext = dbContext;
}
```

17. Make Changes in Controller