Code Academy

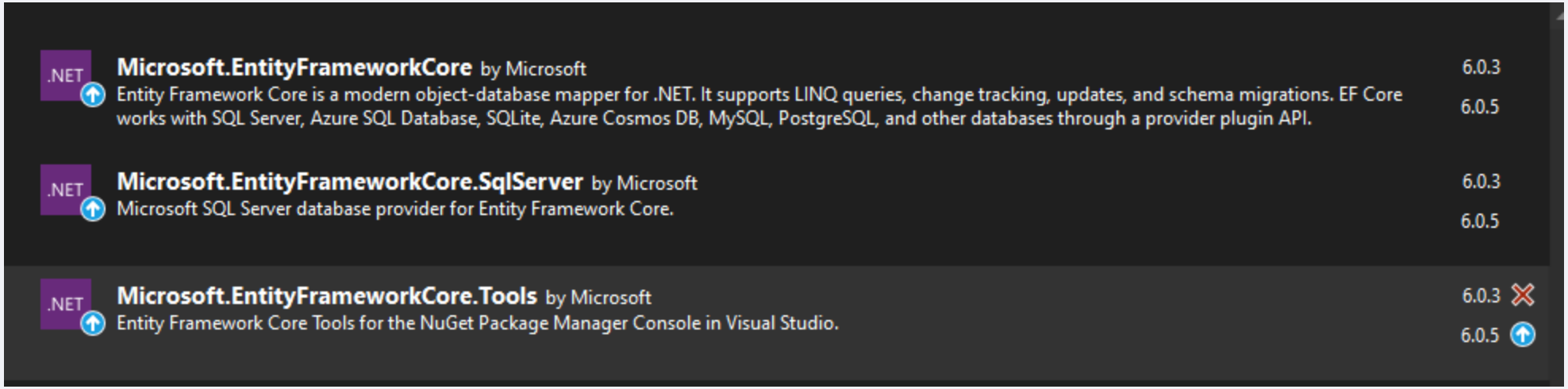
Lecturer

Rokas Slaboševičius

Connecting the database

Data

Nuggets you will need



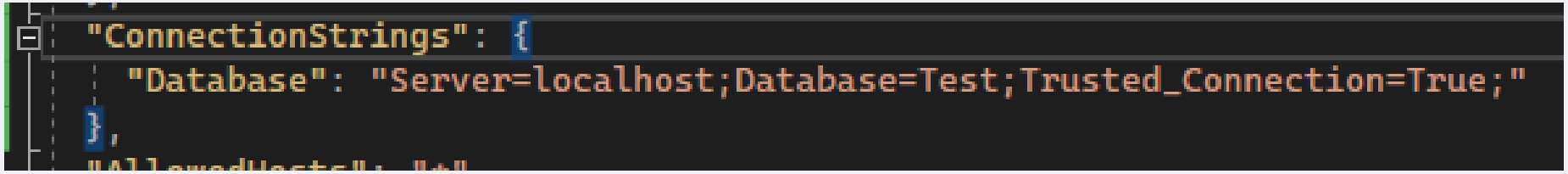
Creating a Context class

```
public class ApplicationDbContext : DbContext
{
    public DbSet<Account> Accounts { get; set; }

    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {

    }
}
```

Add ConnectionString to appsettings.development.json



Register the ApplicationDbContext class as DbContext using the connection string from appsettings

```
// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddDbContext<BookContext>(options => options.UseSqlServer(builder.Configuration.GetConnectionString("Database")));

var app = builder.Build();
```

Let's not forget migration!

Don't forget to encapsulate the context in the repository

```
public class AccountRepository : IAccountRepository
{
    private readonly ApplicationDbContext _context;

    public AccountRepository(ApplicationDbContext context)
    {
        _context = context;
    }

}
```

Register the repository service.

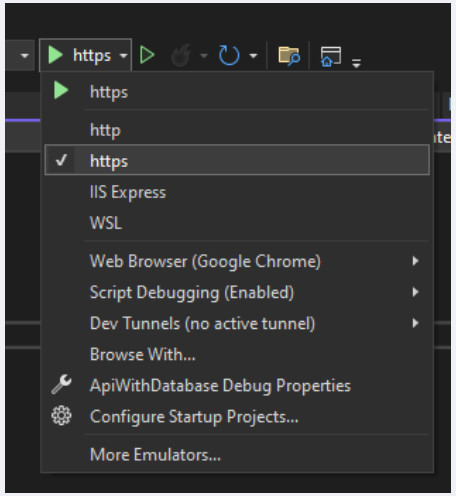
If you are hesitant about which type to use for registering a service, use Scoped.

The choice of Transient/Singleton must be well argumented.

```
services.AddScoped<IAccountRepository, AccountRepository>();
```

Logging in

The easiest way to see the ILogger service logs in the console is to run the application as an https or http instead of IIS Express.



You can configure Logger with IIS Express: https://stackoverflow.com/questions/40148660/how-to-get-a-console-output-in-asp-net-core-with-iis-express

Asp .NET core Entity framework

Logging in

We create an ILogger, Inject it and use it

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
{
    private static readonly string[] Summaries = new[]
    {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
    };

    private readonly ILogger<WeatherForecastController> _logger;

    public WeatherForecastController(ILogger<WeatherForecastController> logger)
    {
        _logger = logger;
    }

    [HttpGet]
    public void Get()
    {
        _logger.LogInformation("Information message");
        _logger.LogWarning("Warning message");
        _logger.LogError("Error message");
    }
}
```

```
C:\Users\ITWORK\Documents\CodeAcademy\DefaultTemplate\DefaultTemplate\bin\Debug\net5.0\DefaultTemplate.exe
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\ITWORK\Documents\CodeAcademy\DefaultTemplate\DefaultTemplate
info: DefaultTemplate.Controllers.WeatherForecastController[0]
      Information message
warn: DefaultTemplate.Controllers.WeatherForecastController[0]
      Warning message
fail: DefaultTemplate.Controllers.WeatherForecastController[0]
      Error message
```

Task 1

- Connect the database to the application you wrote yesterday.
- Try using ILogger to print the information you think is important.