Elev: **Gene Suelo**
Lärare: **Nevena Kicanovic**
Uppgift: **Laboration 3 - EF Core**
Ämne: **Fördjupning programmering 2025**
Skolan: **Yrkehögskola, TUC Sweden**

Besvara följande frågor:

1. Vad är syftet med att använda AsNoTracking i Entity Framework Core och hur påverkar det prestandan?

   → AsNoTracking() inaktiverar EF Core's change-tracking för en given fråga, vilket minskar overhead när vi bara läser data (ingen kostnad för att hålla koll på ändringar). För stora, read-only queries kan det ge betydande prestandaförbättring, särskilt när många entiteter laddas.

2. Hur fungerar Dependency Injection (DI) i ASP.NET Core och varför är det viktigt för att hantera DbContext?

   → I ***Program.cs*** registrerar vi tjänster mot den inbyggda IoC-containern, t.ex.
   ```
   builder.Services.AddDbContext<PubContext>(...);
   ```
   När en endpoint eller kontroller behöver PubContext så injiceras en instans med rätt livscykel (per HTTP-request). Detta hanterar automatiskt skapande och disposering av DbContext.

3. Vad är skillnaden mellan synkrona och asynkrona metoder i en webapplikation, och varför använde vi asynkrona metoder i våra endpoints?

   → Synkrona metoder blockerar den tråd som kör dem tills operationen är klar. Asynkrona (markerade async/await) returnerar en Task och frigör tråden under väntetid (t.ex. databas-I/O). I webappar ökar asynkrona endpoints skalbarheten genom att inte blockera thread pool-trådar.

4. Hur konfigurerar vi en databasanslutning i appsettings.json och hur används den i Program.cs?

   → Konfigurera databasanslutning i ***appsettings.json***
   ```
   "ConnectionStrings": {
     "PubConnection": "Data Source = (localdb)\\MSSQLLocalDB; Initial Catalog = PubDatabase"
   },
   ```
   → I ***Program.cs*** används den så här:
   ```
   builder.Services.AddDbContext<PubContext>(
       opt => opt.UseSqlServer(builder.Configuration.GetConnectionString("PubConnection"))
       .EnableSensitiveDataLogging()
       .UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking));
   ```

5. Vad är fördelen med att använda Include-metoden när vi hämtar data med relationer i Entity Framework Core?

   → Include(...) gör "eager loading" så alla relaterade objekt hämtas i samma SQL-fråga. Utan Include riskerar du N+1-problem om du istället skulle hämta relationsdata i separata anrop.

6. Hur kan vi hantera cyclic data problem när vi serialiserar objekt med relationer i JSON?

   → Objekt som refererar till varandra i cykler (t.ex. Author→Book→Author) kan ge utan hantering antingen undantag eller oändliga slingor. Lösningar är att använda DTO-objekt utan cirkulära referenser, eller konfigurera serializer att ignorera cykler.

7. Vad är syftet med att använda ReferenceHandler.IgnoreCycles i JSON-serialisering och hur påverkar det resultatet?

   → Gör att System.Text.Json hoppar över redan serialiserade objekt istället för att följa cyklerna. Du får giltig JSON utan duplikat och utan infinite loops.

8. Hur kan vi använda Swagger för att testa våra API-endpoints och vad är fördelarna med detta verktyg?

   → Med **Swashbuckle** (via AddSwaggerGen() + UseSwaggerUI()) får du ett interaktivt webb-UI där du både ser dokumentation och kan skicka live-anrop. Det underlättar snabb testning och kommunikation kring API-kontrakt.

9. Vad är skillnaden mellan att använda MapGet direkt i Program.cs och att ha logiken i separata klasser?

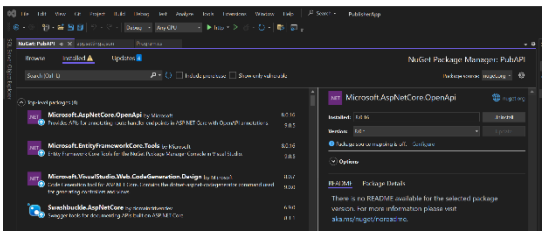
   → Direkt i Program.cs är snabbt för små exempel, men snabbt blir det rörigt om du har många endpoints.
   Separata extensions (som AuthorEndpoints, BookEndpoints) ger bättre struktur, återanvändbarhet och testbarhet.

10. Hur kan vi uppdatera paketversioner i .csproj-filen och varför är det viktigt att hålla dem uppdaterade?

    → **Via Visual Studio**

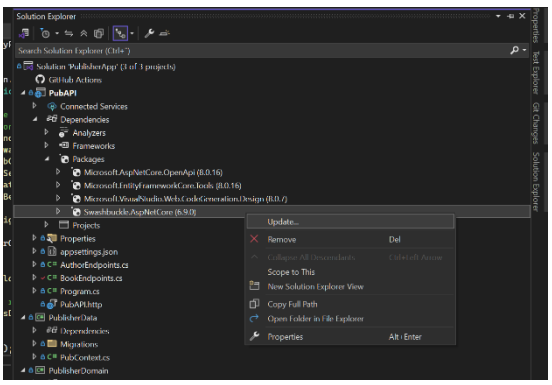
    Högerklicka på projektet → **Manage NuGet Packages...**

    Gå till fliken **Updates**, välj de paket du vill uppdatera och klicka **Update**.



    ELLER

    Dependencies → **Packages** högerklicka på paketposten och välj **Update**



    Visual Studio öppnar NuGet-pakethanteraren som är fokuserad på det paketet och visar de senaste versionerna. Välj den version du behöver, klicka på Installera och acceptera sedan licensen.

    Att hålla paket uppdaterade är viktigt för att få bugg- och säkerhetspatchar, nya funktioner och undvika inkompatibiliteter.