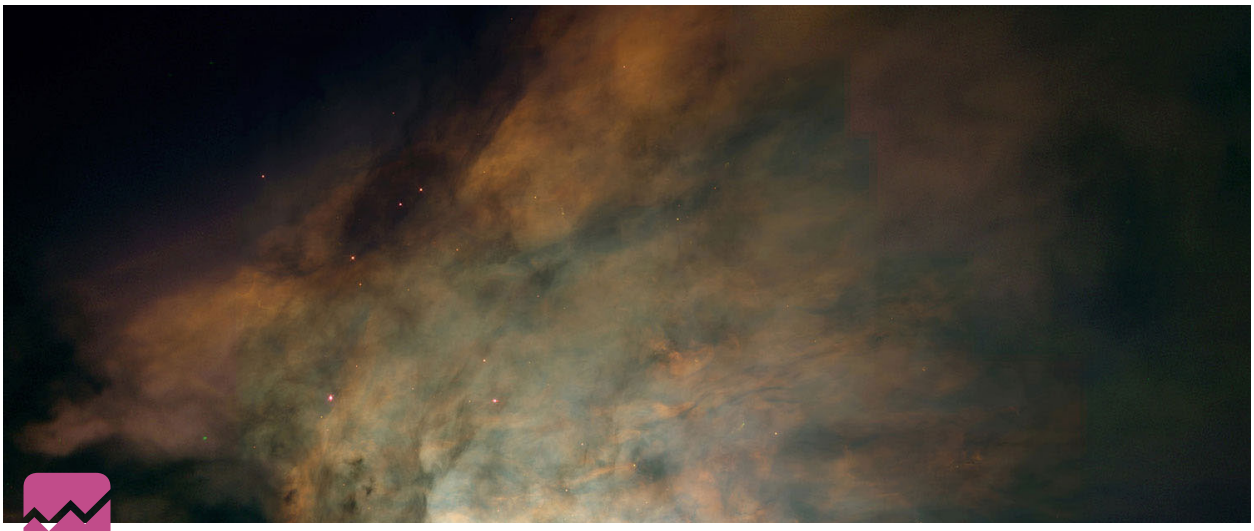

Setting Up the Databases in Visual Studio for Dwellingly | AI

Setting Up the Databases in Visual Studio for Dwellingly | AI

This guide provides detailed instructions to set up the databases for the Dwellingly | AI application, including configuring Entity Framework Core, setting up migrations, and initializing the databases with seed data.

Step-by-Step Guide

Step 1: Define the Database Context and Models

1. **Create the Database Context**:
   - Add a new class `NexHomeAgentDbContext.cs` in the `Dwellingly.Data` project:

```
csharpCopy code
using Microsoft.EntityFrameworkCore;
using NexHomeAgent.Models;

namespace NexHomeAgent.Data
{
```

```
    public class NexHomeAgentDbContext : DbContext
    {
        public NexHomeAgentDbContext(DbContextOptions<N
exHomeAgentDbContext> options)
            : base(options)
        {
        }

        public DbSet<User> Users { get; set; }
        public DbSet<Property> Properties { get; set; }
        public DbSet<Favorite> Favorites { get; set; }
        public DbSet<PropertyImage> PropertyImages { ge
t; set; }
        public DbSet<PropertyHistory> PropertyHistories
{ get; set; }
    }
}
```

2. **Define the Models**:
   - Add the following model classes in the `Dwellingly.Data` project:
     - `User.cs`:

```
csharpCopy code
namespace NexHomeAgent.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

- `Property.cs`:

```
csharpCopy code
namespace NexHomeAgent.Models
{
    public class Property
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public decimal Price { get; set; }
        public int Bedrooms { get; set; }
    }
}
```

- `Favorite.cs`:

```
csharpCopy code
namespace NexHomeAgent.Models
{
    public class Favorite
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public int PropertyId { get; set; }
    }
}
```

- `PropertyImage.cs`:

```
csharpCopy code
namespace NexHomeAgent.Models
{
```

```
    public class PropertyImage
    {
        public int Id { get; set; }
        public int PropertyId { get; set; }
        public string ImageUrl { get; set; }
    }
}
```

- `PropertyHistory.cs`:

```
csharpCopy code
namespace NexHomeAgent.Models
{
    public class PropertyHistory
    {
        public int Id { get; set; }
        public int PropertyId { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; }
    }
}
```

Step 2: Configure the Database Context in the API Project

1. **Configure the Connection String**:

- Add the connection string to `appsettings.json` in the `Dwellingly.API` project:

```
jsonCopy code
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocald
b;Database=DwellinglyDB;Trusted_Connection=True;Multipl
eActiveResultSets=true"
```

```
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

2. **Configure the Database Context in `Program.cs`:**

- Update `Program.cs` in the `Dwellingly.API` project to register the `DbContext`:

```
csharpCopy code
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Serilog;
using NexHomeAgent.Data;

var builder = WebApplication.CreateBuilder(args);

// Configure Serilog
Log.Logger = new LoggerConfiguration()
    .MinimumLevel.Information()
    .WriteTo.Console()
    .WriteTo.File("logs/nexhomeagent.txt", rollingInter
val: RollingInterval.Day)
    .CreateLogger();
builder.Host.UseSerilog();

// Add services to the container
builder.Services.AddControllers();
builder.Services.AddDbContext<NexHomeAgentDbContext>(op
```

```
tions =>
    options.UseSqlServer(builder.Configuration.GetConne
ctionString("DefaultConnection")));

builder.Services.AddApplicationInsightsTelemetry(option
s =>
{
    options.InstrumentationKey = builder.Configuration
["ApplicationInsights:InstrumentationKey"];
});

var app = builder.Build();

// Configure the HTTP request pipeline
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}

app.UseHttpsRedirection();
app.UseRouting();
app.UseAuthorization();
app.UseAuthentication();

app.MapControllers();

app.Run();
```

Step 3: Add Migrations and Update the Database

1. **Add Entity Framework Tools**:
    - Install the `Microsoft.EntityFrameworkCore.Tools` package in the `Dwellingly.API` project via the NuGet Package Manager Console:

```
shCopy code
Install-Package Microsoft.EntityFrameworkCore.Tools
```

2. **Add Initial Migration**:
   - Open the Package Manager Console, select the `Dwellingly.API` project as the Default Project, and run:

```
shCopy code
Add-Migration InitialCreate
```

3. **Update the Database**:
   - Run the following command in the Package Manager Console to apply the migration and create the database:

```
shCopy code
Update-Database
```

Step 4: Seed Initial Data (Optional)

1. **Create Seed Data Class**:
   - Add a new class `DbInitializer.cs` to the `Dwellingly.API` project to seed initial data:

```
csharpCopy code
using Microsoft.Extensions.DependencyInjection;
using NexHomeAgent.Models;

public static class DbInitializer
{
    public static void Initialize(NexHomeAgentDbContext
context)
    {
```

```
        context.Database.EnsureCreated();

        // Check if users already exist
        if (context.Users.Any())
        {
            return;   // DB has been seeded
        }

        var users = new User[]
        {
            new User{Email="user1@example.com", Passwor
d="Password1"},
            new User{Email="user2@example.com", Passwor
d="Password2"},
        };

        foreach (var u in users)
        {
            context.Users.Add(u);
        }

        context.SaveChanges();

        var properties = new Property[]
        {
            new Property{Name="Property1", Address="Add
ress1", Price=100000, Bedrooms=3},
            new Property{Name="Property2", Address="Add
ress2", Price=200000, Bedrooms=4},
        };

        foreach (var p in properties)
        {
            context.Properties.Add(p);
        }
```

```
            context.SaveChanges();
        }
    }
```

2. **Update `Program.cs` to Seed Data**:
    - Update `Program.cs` to call the seed method during application startup:

```
csharpCopy code
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Serilog;
using NexHomeAgent.Data;

var builder = WebApplication.CreateBuilder(args);

// Configure Serilog
Log.Logger = new LoggerConfiguration()
    .MinimumLevel.Information()
    .WriteTo.Console()
    .WriteTo.File("logs/nexhomeagent.txt", rollingInter
val: RollingInterval.Day)
    .CreateLogger();
builder.Host.UseSerilog();

// Add services to the container
builder.Services.AddControllers();
builder.Services.AddDbContext<NexHomeAgentDbContext>(op
tions =>
    options.UseSqlServer(builder.Configuration.GetConne
ctionString("DefaultConnection")));

builder.Services.AddApplicationInsightsTelemetry(option
s =>
```

```
{
    options.InstrumentationKey = builder.Configuration
["ApplicationInsights:InstrumentationKey"];
});

var app = builder.Build();

// Seed the database
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    var context = services.GetRequiredService<NexHomeAg
entDbContext>();
    DbInitializer.Initialize(context);
}

// Configure the HTTP request pipeline
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}

app.UseHttpsRedirection();
app.UseRouting();
app.UseAuthorization();
app.UseAuthentication();

app.MapControllers();

app.Run();
```

Conclusion

Following these steps, you will set up the databases for the Dwellingly | AI application, including configuring Entity Framework Core, setting up migrations, and initializing the databases with

seed data. This ensures that your application has a robust and scalable database structure, ready for development and testing.