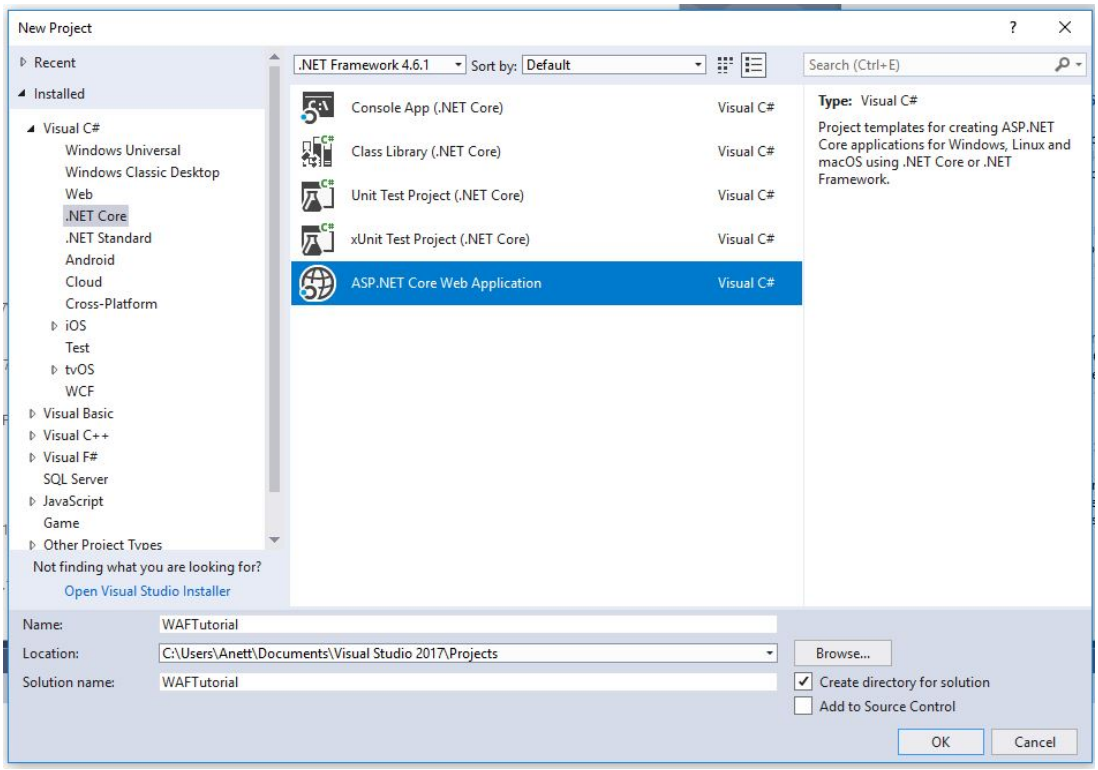
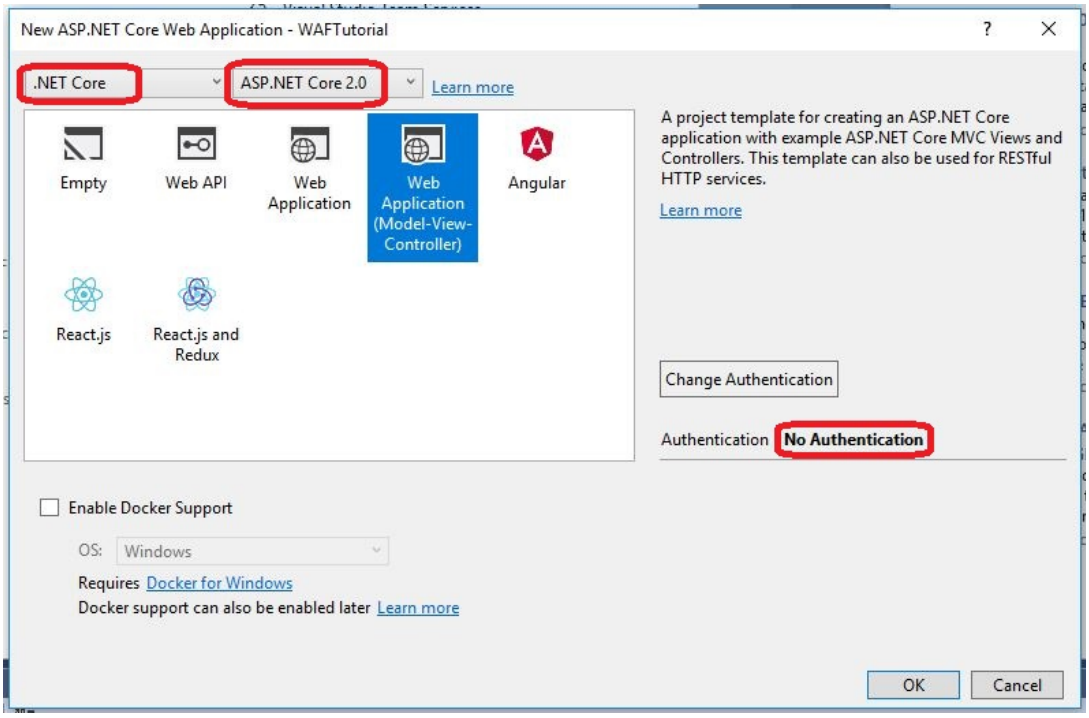
ASP.NET Core MVC Tutorial

Projekt létrehozása

ASP.NET Core projektet csak Visual Studio 2017-ben tudunk létrehozni (Windows alatt).





Modell osztály hozzáadása

A modell ebben az esetben az adatbázisunk tábláinak reprezentációja. Minden táblát egy-egy osztályként jelenítünk meg a kódban.

1. Hozzunk létre egy új Class fájlt a Models mappában.

2. Minden táblának megfelelően hozzunk létre egy osztályt a táblában lévő mezőknek megfelelő propertykkel. A modell osztályok kerülhetnek ugyanazon fájlba is.

3. Az adatbázistáblában kötelezően szerepelnie kell egy ID mezőnek, ez a nézetekben automatikusan rejtve lesz. Az ID minden új sor hozzáadásával automatikusan inkrementálódik.

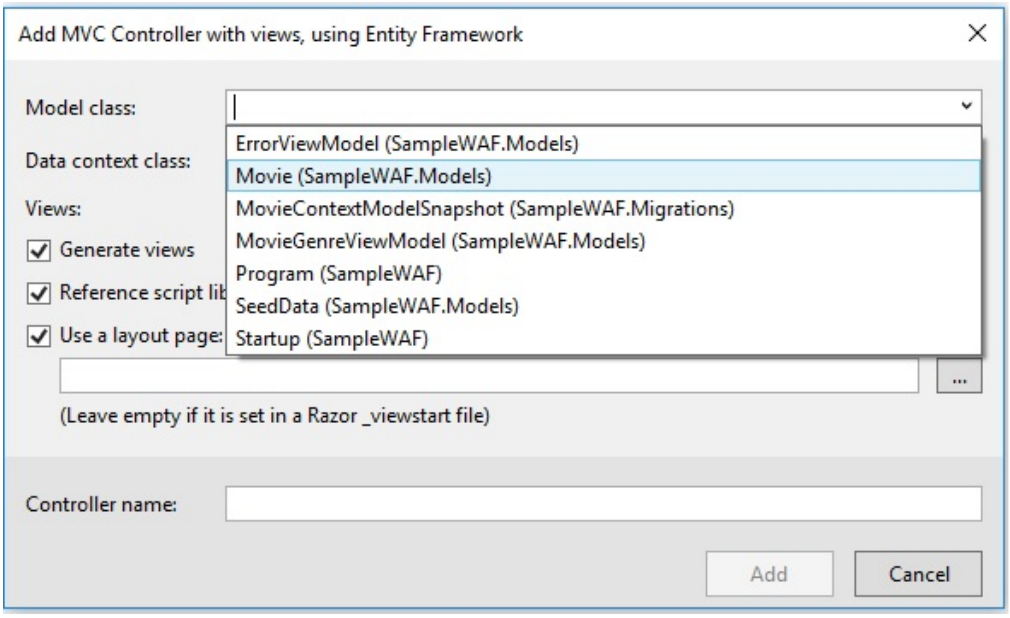
4. A mezők megjelenítését és az új sor hozzáadásakor elfogadott értékeket és formátumokat data annotation tagekkel tudjuk manipulálni. Ezekkel megadhatunk pl. dátumformátumokat, regexeket, adatskálákat stb.

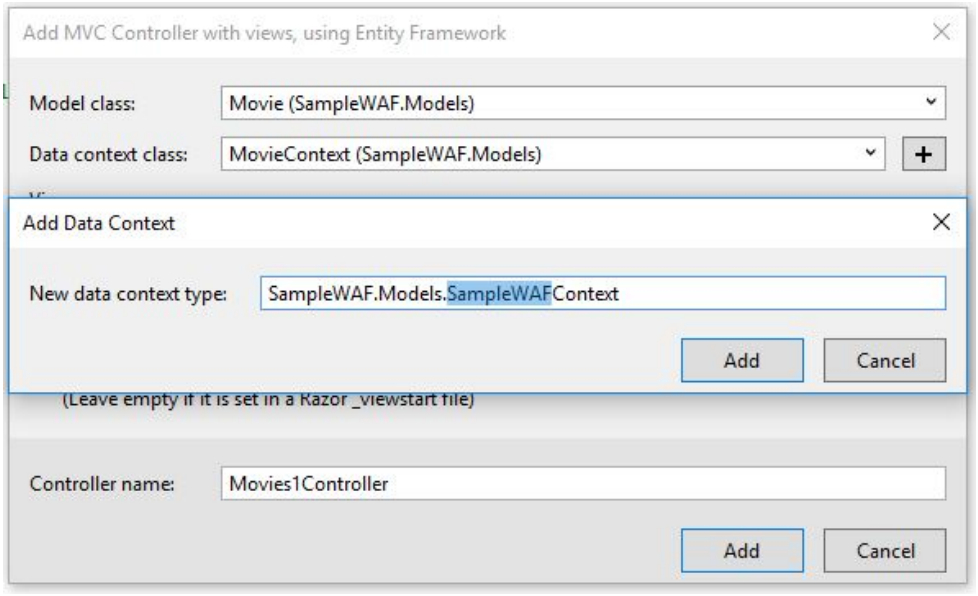
```
8 references
public class Movie
{
    7 references | 0 exceptions
    public int ID { get; set; }           // oblig
    5 references | 0 exceptions
    public string Title { get; set; }

    [Display(Name = "Release Date")]      // data
    [DataType(DataType.Date)]
    4 references | 0 exceptions
    public DateTime ReleaseDate { get; set; }
    7 references | 0 exceptions
    public string Genre { get; set; }
}
```

Adatbázis-kontextus létrehozása

Az adatbázis-kontextus csatlakozik az adatbázishoz és párosítja annak mezőit a modellben található propertykkel. Kontextust generálni és kézzel készíteni is lehet.

1. Generálás controller osztállyal (és nézetekkel) együtt: Solution explorer -> Controllers -> Add- > Controller... -> MVC Controller with views...
Válasszuk ki a modell osztályt, amihez a controllert generáljuk, majd kattintsunk a Data context class melletti "+" jelre. A VS automatikusan ad egy nevet a kontextusnak. Az így elkészített kontextust a VS a Data mappába teszi.
Fontos: ezzel a módszerrel a teljes modellhez elkészül az adatbázis-kontextus, így új controller hozzáadásakor már csak ki kell választanunk a generált kontextust.





2. Mechanikus módszer: a kontextus kerülhet új fájlba a Models mappán belül, de kerülhet a modell osztályokkal közös fájlba is. A kontextusnak a DbContext osztályból kell származnia, és a konstruktoron kívül az adatbázis minden táblájának szerepelnie kell benne egy DbSet formájában.

```
10 references
public class MovieContext : DbContext              // any database
{
    1 reference | 0 exceptions
    public MovieContext(DbContextOptions<MovieContext> options)
        : base(options){}

    10 references | 0 exceptions
    public DbSet<SampleWAF.Models.Movie> Movie { get; set; }     /
}
```

Adatbázis létrehozása code-first megközelítéssel

A code-first megközelítés lényege, hogy először a kódban létrehozzuk az adatbázisnak megfelelő modellt, majd ebből generáljuk a konkrét adatbázist.

1. Ha megvan a modell, generáljunk vagy készítsünk hozzá egy adatbázis-kontextust (ld. előző pont).

2. Adjunk connection stringet az appsettings.json fájlhoz. A Database attribútum után megadhatjuk az adatbázis nevét. Fontos: a stringet a kontextus nevével kell összekapcsolni.

```
},
"ConnectionStrings": {
  "MovieContext": "Server=(localdb)\\mssqllocaldb;Database=MovieContext;Trusted_Connection=True;MultipleActiveResultSets=true"
}
```

3. Regisztráljuk az adatbázist a Startup osztály ConfigurationServices metódusában:

   a. Használhatunk SQLite-ot (cross-platform), ekkor a .db fájl a projekt mappájában fog megjelenni.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    // the connection string should be added in appsettings.json

    // this line should create a .db file out of the model that can be opened and modified with SQLite
    // use this for Linux or Mac
    services.AddDbContext<MovieContext>(options => options.UseSqlite("Data Source=Movie.db"));

}
```

   b. Használhatunk MSSQL-t, ekkor a .mdf fájl a C:\Users\<user> mappában lesz megtalálható. A VS-n belül a View -> SQL Server Object Explorer alatt tekinthetjük meg az adatbázist.

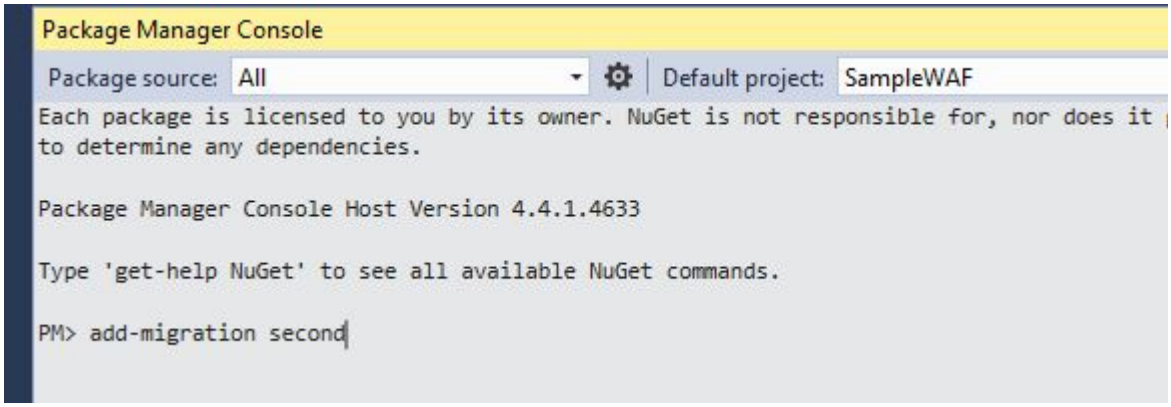
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    // the connection string should be added in appsettings.json

    // this method uses MSSQL
    // the database is accessible in View -> SQL Server Object Explorer after the first migration
    services.AddDbContext<MovieContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("MovieContext")));
}
```

4. Nyissuk meg a Package Manager Console-t (View -> NuGet Package Manager -> Package Manager Console), és készítsük el az első migrációt:

   a. add-migration <migration name> (az első migráció neve konvenció szerint initial)

   b. update-database

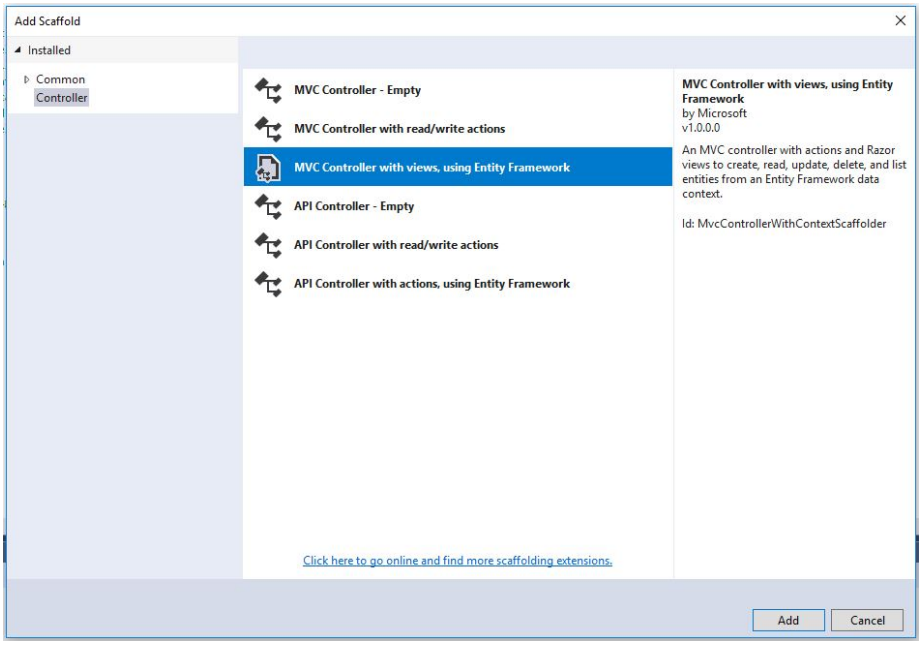
Minden alkalommal, amikor megváltoztatjuk a modellt, új migrációt kell létrehoznunk, hogy szinkronban tartsuk a modellt és az adatbázis szerkezetét.



Controller és nézetek létrehozása

A modell minden táblájához létrehozhatunk saját controllert nézetekkel együtt.

1. Controllers mappa -> Add -> Controller... -> MVC Controller with views, using Entity Framework

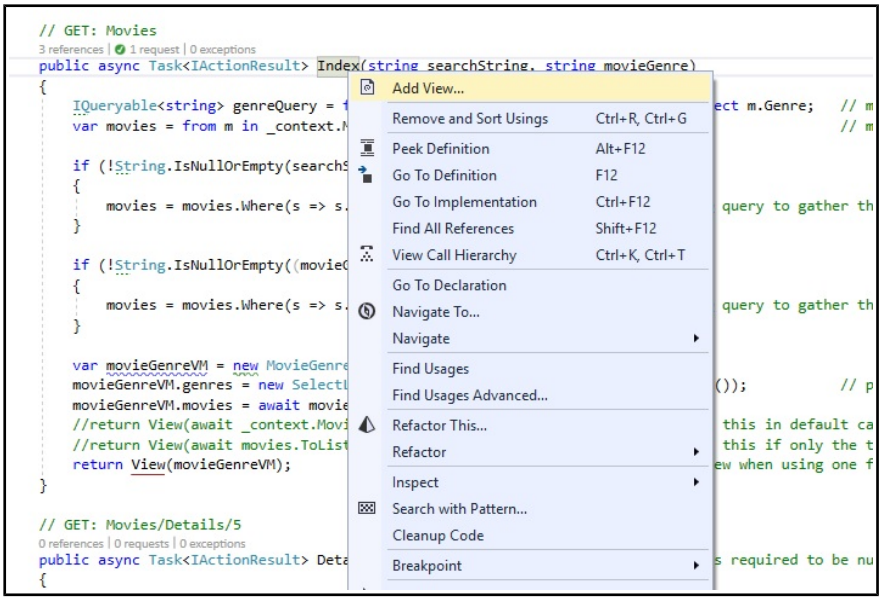


2. Válasszuk ki a modell osztályt, amihez controllert akarunk adni, majd az adatbázis-kontextust (az Adatbáis-kontextus létrehozása pont alatt látható képeknek megfelelően). Ha nem választunk ki layoutot, a nézetek az alapértelmezett Razor page elrendezést fogják megkapni.

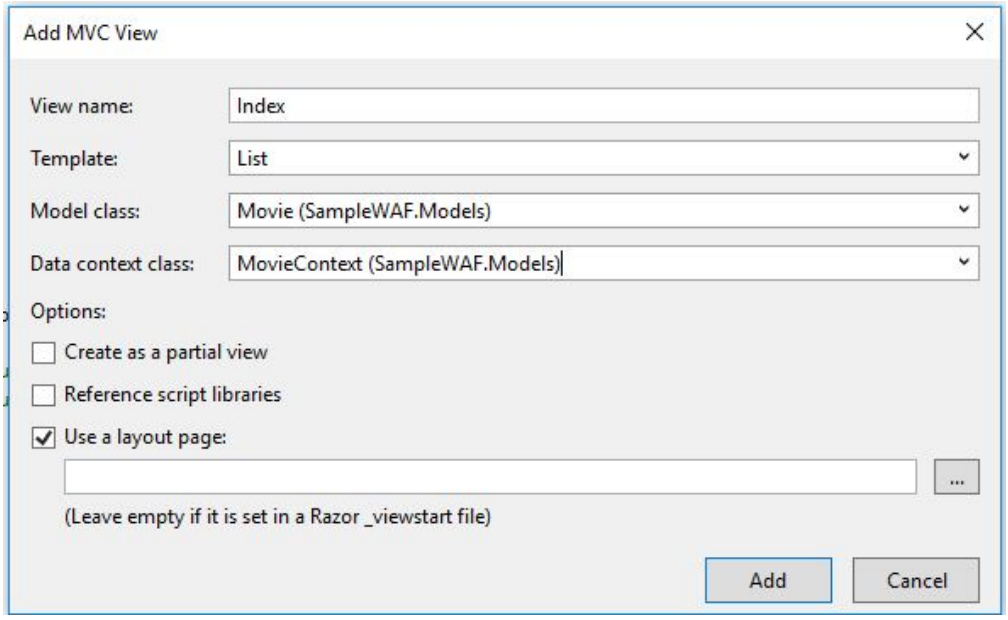
A controller <model class name>+Controller néven fog létrejönni. Tartalmazni fogja a CRUD (create, read, update, delete) műveleteknek megfelelő Create, Index, Edit, Delete, valamint a Details metódusokat, amelyek elnevezésüknek megfelelő műveleteket hajtanak végre a tábla elemein.

Mivel a controllerrel együtt nézeteket is hozzáadtunk, a Views mappában a modell osztálynak megfelelő néven létrejöttek a controllerbeli metódusoknak megfelelő nézetek. Természetesen nézetek nélkül is létrehozhatunk controllert, valamint önállóan is hozzáadhatunk nézeteket:

1. A controller osztályban a metódus nevére kattintva válasszuk ki az Add View opciót.



2. A nézet alapértelmezett neve a metóduséval megegyezik (de megváltoztatható). Válasszuk ki a nézet sablonját (az Index metódusnak a List sablon felel meg), a modell osztályt és a kontextust.



Az adatbázis feltöltése kódból

Az adatbázishoz adhatunk adatokat külső program (pl. SQLite) vagy a VS segítségével, a weboldalon keresztül a Create metódussal, vagy a kódban külön erre a célra létrehozott osztály segítségével. Ehhez hozzunk létre egy osztályt a Models mappán belül, a SampleWAF projektben ez a SeedData osztály. Dependency injection használatával adjuk meg a kontextust, majd adjunk a modellbeli táblákhoz új objektumokat attribútumaikkal megadva. Az osztályt a Program osztály Main metódusában hívjuk meg. Fontos: ha az inicializáló osztály nem ellenőrzi, hogy vannak-e már adatok az adatbázisban, akkor

minden meghíváskor duplikátumokat fog hozzáadni.

```
namespace SampleWAF.Models
{
    1 reference
    public class SeedData
    {
        1 reference | 0 exceptions
        public static void Initialize(IServiceProvider serviceProvider)
        {
            using (var context = new MovieContext(
                serviceProvider.GetRequiredService<DbContextOptions<MovieContext>>()))
            {
                // Look for any movies.
                if (context.Movie.Any())
                {
                    return; // DB has been seeded
                }

                // add new lines to the database in the form of objects
                context.Movie.AddRange(
                    new Movie
                    {
                        Title = "When Harry Met Sally",
                        ReleaseDate = DateTime.Parse("1989-1-11"),
```

```
    1 reference
    public class Program
    {
        0 references | 0 exceptions
        public static void Main(string[] args)
        {
            // originally generated code
            // BuildWebHost(args).Run();

            // add all these in order to run a seeding method
            var host = BuildWebHost(args);

            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;

                try
                {
                    // Requires using MvcMovie.Models;
                    SeedData.Initialize(services);
                }
                catch (Exception ex)
                {
                    var logger = services.GetRequiredService<ILogger<Program>>();
                    logger.LogError(ex, "An error occurred seeding the DB.");
                }
            }

            host.Run();
        }
```

Az alapértelmezett elrendezés (layout) megváltoztatása

A Views -> Shared mappa _Layout nézete felelős a weboldal alapértelmezett nézetéért. Legfontosabb része az oldal felső részén megjelenített navigációs sáv, amihez hagyományos html tagekkel adhatunk linkeket, amelyek a controllerek bizonyos nézeteire mutatnak. Az új linknek meg kell adnunk a controllert (asp-controller) és a metódust (asp-action), amelyre mutatni fog, valamint a megjelenítendő nevet.

```
<!-- this is the navigation bar you can see on the top of the website -->
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li><a asp-area="" asp-controller="Home" asp-action="Index">Home</a></li>
        <li><a asp-area="" asp-controller="Home" asp-action="About">About</a></li>
        <li><a asp-area="" asp-controller="Home" asp-action="Contact">Contact</a></li>
        <li><a asp-area="" asp-controller="Movies" asp-action="Index">List of Movies</a></li>
    </ul>
</div>
```