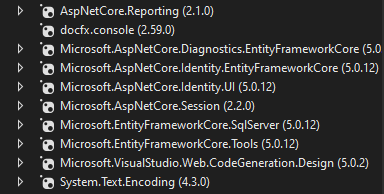
Dependencias:



**Proiektua sortu**

Proyecto ASP.Net core MVC,

Cuentas individuales

NET 5.0

Entity Framework y Sesions

AppSettings : crear nueva string de concexion, cojer la string de propierties de Microsoft SQL server.

Ejemplos: "ErronkaOndoConnection": "Server=UH-004-05\SQLEXPRESS;Database=Puntuazioa;integrated security=True"

"Persons": "Server=DESKTOP-1CJ10QG;Database=Persons;integrated security=True"

Start Up: añadir services.addDbContext y cambiar la string a el nombre de ka string en app settings.

En data crear una nueva clase, que sera la dbcontext de nuestra base de datos. Añadir Getter/Setters para la tabla de la base de datos.

Ejemplo de dbcontext:

namespace Persons.Data.Migrations

{

public class PersonsDbContext : DbContext

{

public PersonsDbContext(DbContextOptions<PersonsDbContext> options)

: base(options)

{

}

public DbSet<Person> Person { get; set; }

public DbSet<Order> Order { get; set; }

}

}

En models crear una clase por tabla de la base de datos. Como una clase de objetos con getters/setters de las columnas de la tabla.

Ejemplo:

namespace Persons.Models

{

public class Person

{

[Key]

public int Id{ get; set; }

public String LastName { get; set; }

public String FirstName { get; set; }

public int Age { get; set; }

}

}

Crear Carperta Services

Agregar una clase interfaz por cada tabla de la base de datos

Ejemplo: IPersonService

Agregar una clase normal por cada tabla de la base de datos

Ejemplo: PersonService

Rellenar la interfaz con los metodos que va ha tener.

Ejemplo de un metodo que selecciona todos las persons:

Task<IList <Person>> GetPersons();

Ejemplo de un metodo que selecciona la persona por la id:

Task<Person> GetPersonById(int id);

En la clase normal implementar la iterfaz.

public class PersonService : IPersonService

{

private readonly PersonsDbContext \_context;

public PersonService(PersonsDbContext context)

{

\_context = context;

}

public async Task<Person> GetPersonById(int id)

{

return \_context.Person.Find(id);

}

public async Task<IList<Person>> GetPersons()

{

return await \_context.Person.ToListAsync();

}

}

En startup despues de add entityFramework injectar lor servicios

Ejemplo:

services.AddScope<IPersonService, PersonService>();

Agregar un controlador de escritura y lectura,

Añadir la interfaz en el controlador.

private readonly IPersonService \_IPersonService;

clicar en un return view para generar la view y su carpeta

En el controller devolver el view con una lista de personas, y hacerlo asincrono:

public async Task<ActionResult> Index()

{

IList<Person> personLista = await \_IPersonService.GetPersons();

return View(personLista);

}

Como subir un objeto desde un servicio:

public async Task InsertOrder(Order order)

{

\_context.Order.Add(order);

\_context.SaveChanges();

}

Orderren viewComponent bat gehitu

ViewComponents deitzen den karpeta bat sortu proiektuko karpetan

Fitxategia sortu eta bete

public class OrderViewComponent : ViewComponent

{

private readonly IOrderService \_iOrderService;

public OrderViewComponent(IOrderService iOrderService)

{

\_iOrderService = iOrderService;

}

//Datuak gehitzeko da, datuak lista bat egiteko itzultzen dira.

public async Task<IViewComponentResult> InvokeAsync()

{

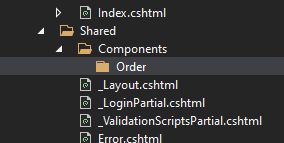
return View(await \_iOrderService.GetOrders());

}

}

Startupen zerbitzua gehitu.

*Shared* karpetan *Components* karpeta sortu, eta Components barruan order karpeta



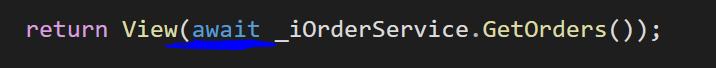
Orderren añadir vista de razorri eman eta Default fitxategia sortu, nahi duzun plantilla eta modeloarekin.

Behin bista eginda nahi duzun bistan gehitu hau jarrita:

@await Component.InvokeAsync("Order")

Order hori componentearen karpetaren izena da.

Dictionari error… moduko errore bat ematen badizu, saiatu viewcomponentean datuak itzultzen dituen metodoan await jartzen.



Datubasean order erregistruak gehitzeko controller berri bat sortuko dugu “OrderController”, eta gero view bat create plantillarekin.

Controller barruan index funtzio berri bat gehitu horrela:

private readonly IOrderService \_IOrderService;

public OrderController(IOrderService iOrderService)

{

\_IOrderService = iOrderService;

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Index(Order o)

{

await \_IOrderService.InsertOrder(o);

return View();

}

View-ean formularioa automatikoki sortu delako ez dugu aldaketarik egin behar aurreko funtzioari deitzeko.

**SESSIONS Autentifikazioa eta [Authorize]**

Startup.cs barruan ConfigureServices metodo barruan kodigo hau gehitu:

*services.AddDistributedMemoryCache();*

*services.AddSession();*

eta configure metodo barru hasieran hau gehitu:

*app.UseSession();*

Orain Authorice erabiltzea nahi duzun view-ean *[Authorize]* tag-a jarri index funtzio gainean adb:

// GET: OrderController

[Authorize]

public ActionResult Index()

{

return View();

}

Tag-a Post-ean jartzen baduzu inserta egin baino lehenago logeatzeko eskatuko dizu.

**ViewBag**

ViewBag bat controllerran gehitu eta indexetik deitu ahal diozu.

Controllerrean:

ViewBag.izena = "pepe";

View-an:

@ViewBag.izena

**Konektatuta dagoen erabiltzailea jakiteko**

Gehitu hau view batean:

*@User.Identity.Name*

ViewBag batean pasatzeko:

Controllerrean:

*ViewBag.user = HttpContext.User.Identity.Name;*

View-an:

*@ViewBag.user*

Formulario bateko submit botoian edo antzeko zerbaitzen hau jarri behar da datua ondo bidaltzeko.

asp-controller="Komentario" asp-action="Index" asp-route-partidaId="@item.partidaId"

**ViewModel**

Proiektuan ViewModels karpeta sortu

Karpeta horretan klase berri bat sortu OrderViewModel adb.

ViewModelean atributuak gehitu adb:

public class OrderViewModel

{

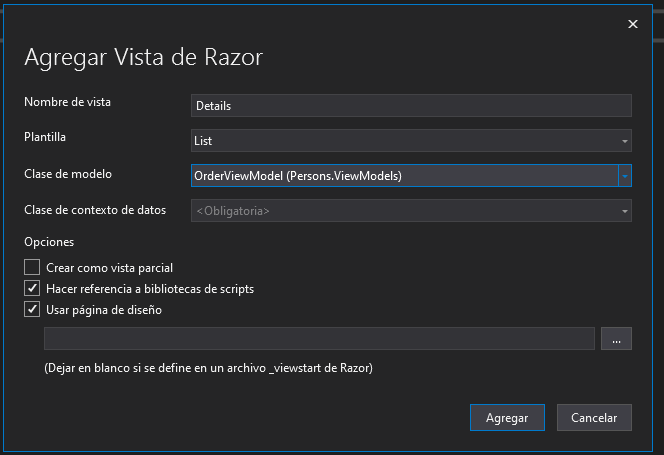
public IList<Order> Orders { get; set; }

public Person Person { get; set; }

}

Nahi dugun controllerreko metodoan gure viewmodela sortu eta baloreak emango dizkiogu.

Eta returnean klik eginda view berri bat sortu viewmodel berria eribiliz.



**REST API**

Service berri batzuk sortu

public interface IWineService

{

Task<IList<Person>> GetPersons();

}

public class WineService : IWineService

{

private Uri rutaTodos = new Uri("https://localhost:44367/person");

public async Task<IList<Person>> GetPersons()

{

List<Person> personList = new List<Person>();

using (var httpClient = new HttpClient())

{

using (var response = await httpClient.GetAsync(rutaTodos))

{

string apiResponse = await response.Content.ReadAsStringAsync();

personList = JsonConvert.DeserializeObject<List<Person>>(apiResponse);

}

}

return personList;

}

}

**UPDATE**

public async Task InkestaAldatu(Inkesta inkesta) { var inkestaEginda = await \_superNaaahiDbContext.Inkesta.SingleOrDefaultAsync(i => i.Korreoa == inkesta.Korreoa); inkestaEginda.Balorazioa = inkesta.Balorazioa; inkestaEginda.Hobetzeko = inkesta.Hobetzeko; inkestaEginda.Gomendatu = inkesta.Gomendatu; \_superNaaahiDbContext.SaveChanges(); }