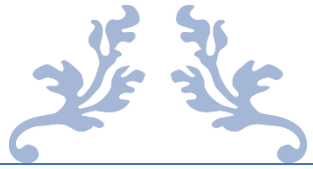

HANTVERKSBODEN 2.0

Inlämningsuppgift - Datalagring med Entity Framework

DEN 18 DECEMBER 2020

WIVIANNE GRAPENHOLT
grwi20ql@student.ju.se

Innehåll

Ursprung

Hantverksboden 1.0 har jag byggt upp från en tom mall och själv lagt till den allmängiltiga mapp- och filstruktur som gäller för ASP.NET MVC.

Models

Jag två domänklasser Handicraft och CraftMethod, de har varsitt interface IHandicraftRepository samt ICraftMethodRepository som möjliggör användning av instanser av dessa klasser i resten av applikationen utan insyn i själva klassen. Jag skapade även två MockRepository där jag implementerade en hårdkodad uppsättning hantverk och hantverksmetoder. Handicraft har fem properties: tre string properties Name, Description, Image, en int properties Price och CraftMethod. CraftMethod har två properties: string CraftMethodName och list Handicrafts.

Controllers

Här ligger applikationens enda controller class, HandicraftController. Klassen har två privata readonly fält för att hålla koll på instanser som skapats av injectorn. Den har dock två actionmetoder: List() samt Contact(). Jag följde Gills exempel och skapade en ViewModel, utan att riktigt ha kopplat nyttan med en sådan. I List() initierarar jag en ny instans av HandicraftListViewModel och skickar med hantverken i ovannämnda fält. För att slutligen returnera vyn med HandicraftListViewModel som parameter. Contact() returnerar en konfigurerad ”statisk” Razor vy.

Views

skapade jag två undermappar Handicraft och en Shared. I den förstnämnda finns två Razor view filer, List och Contact. List använder sig av HandicraftListViewModel och loppar sedan igenom alla Handicraft objekt i listan och presenterar dessa för användaren. Jag har nästan använt Gills kod för att visa pajer rakt av, mest för att få ett snyggt resultat utan att behöva lägga tid och energi på css och bootstrap. Contact är en enkel Razor vy med hårdkodad kontaktinformation. I Shared mappen finns _Layout som är den grundmall alla sidor/vyer i mitt program använder, även här är mycket ”copy & paste” från BethanysPieShop. Det finns ytterligare två filer i Views mappen nämligen: _ViewImport och _ViewStart. Den första gör det möjligt att på ett enkelt sätt använda instanser av mina domänklasser i vyerna. Och den andrar pekar helt enkelt ut _Layout för ASP.NETCore som mallen jag vill använda.

Startup.cs

I ConfigureServices() har jag lagt till stöd för MVC och registrerat mina interface och dess implementation. I Configure() är UseStaticFiles() tillagt som möjliggör användandet av statiska filer i mappen wwwroot. I UseEndpoints() har jag ändrat till MapControllerRout() för att tillåta routing med controller/action.

wwwroot

Finns tre undermappar: content, här ligger bara min egen CSS-fil. Images, som förstås innehåller de bilder jag använder och lib där jag lagt till bootstrap och jquery bara för att underlätta stylingen av vyerna.

Uppdatering

Svårast var implementeringen av SQL databas och enhetstesterna. Det var lite knepigt att fixa sorteringen av hantverk i respektive kategori (CraftMethod) och samtidigt möjliggöra nya vyer och inkludera navigering till dessa. Enklast men tråkigast var nog att flytta över all mockdata.

Resultat

Models

För att möjliggöra att kunna visa en sida per hantverk, kunna sortera hantverken utifrån kategori samt visa nya alster på indexsidan så fick jag lägga till två int properties HandicraftId, CraftMethodId samt en bool property IsNewItem. IHandicraftRepository fick därför också fyra nya IEnumerable<Handicraft> properties: NewItem, Crochet, Macrame, Drawing och en metod GetHandicraftById. I HandicraftRepository implementeras dessa genom att hitta de instanser som uppfyller de olika kraven och lägger till objekten i respektive lista. Det har också tillkommit en AppDbContext som, om jag förstått rätt är en ”brygga” mellan databasen och min kod. Här fylls även SQL-databasen med seed-data via metoden OnModelCreating (jag blir dock inte riktigt klok på hur det fungerar eftersom det står att metoden inte har några referenser) när man initierar en migration och uppdaterar. Medan användandet av InMemory-databas kräver att anrop till EnsureCreated.

Controllers

Det har tillkommit en HomeContoller med en actionmetod: Index. Här returneras en view model som inkluderar att visa nya alster. HandicraftController har fått tillskott med fyra actionmetoder; Crochet, Macrame, Drawing och Details. Som är nästa led till att kunna visa anpassade vyer på enskilda produkter eller lista dem utifrån hantverksmetod. De tre första har alla varsin view model som returneras. Medan Details bara är en vy under Handcraft-mappen.

Views

Har utökats med en undermapp Home där Index vyn finns. Handicraft-mappen består nu av hela sex olika vyer. Alltså fyra nya: Details, Crochet, Macrame och Drawing de tre sistnämnda samt List och Index använder sig numera av partial vyn _HandicraftCard som är tillagd i Shared-mappen. I _Layout finns nu en navbar där jag använder tag-helpers för att generera länkar.

Startup.cs

Här har jag lagt till property och metoder för att kunna köra min applikation både på SQL och InMemory databas. Vilket också kräver en koppling till AppDbContext och en referens till connectionString i appsettings.json. Jag har naturligtvis även ändrat implementeringen av mina interfaces till ”riktiga” repositories.

Analys

Överlag känns det som att min lösning är acceptabelt strukturerad för att i framtiden kunna lägga till, ta bort eller ändra hantverk och eller hantverksmetoder. Jag inser dock nu att en lösning där de sorterade alstren som properties till CraftMethod, istället för listor under själva Handicraft skulle underlätta om utbudet skulle utökas med fler olika hantverksmetoder. Som min lösning är strukturerad nu känns nästan CraftMethod klassen överflödig, jag använder tillexempel inte ens propertyn CraftMethodId, vilket säkert hade kunnat vara användbart i filtreringen. Då hade en det behövts en ny controller class till CraftMethod. Jag misstänker att jag hade kunnat förenkla mina view models och view, så att alla kategorier använder samma, eller till och med kanske HandicraftListViewModel / List?

Enhetstester

Att skapa enhetstester till detta projekt var verkligen en utmaning! Men när jag väl fick rätt på det så anser jag att det är väl valda tester som dessutom använder InMemory-databasen. Jag skapar och använder en mock AppDbContext som jag använder som parameter för metodanropen i mina repository classes.

AddingHandicrafts och AddingCraftMethods

Dessa var först ett tester för att få lösningen med InMemory-databas och mock AppDbContext att fungera. Då använde jag en hårdkodad variabel för expected, eftersom jag visste hur många alster det fanns i listan. För att testet skall fungera även i vidareutveckling så ändrade jag den till att iterera igenom alla hantverk/metoder i repositoryt och lägga till dem i en lista. Sedan kollar jag att handicraftRepository.AllHandicrafts/ craftMeyhodRepository.AllCraftMethods inte är tomma. Sedan jämför jag antalet med antalet i den lokalt sparade listan. På detta sätt kan antalet alster/tillverkningssätt både öka och minska utan att testet misslyckas.

SortByCraftMethod

Här testar jag att filtreringen av skapelser baserat på vilken kategori den tillhör fungerar, i grunden är testet uppbyggt på samma sätt som ovan nämnda. Nackdelen med detta test är dock att jag i min domänlogik inte utnyttjar det faktum att alla CraftMethods har en id-property som hade kunnat matchas med motsvarande värde på varje enskild Handicraft instans. Viket hade gett ett test som var oberoende av antalet olika CraftMethods eller namnet på dessa.