Reflektion - Uppfinnaren2.0 - Johannes Jakobsson

Ursprung

Applikationen Uppfinnaren är uppbyggd i ASP.NET och enligt designmönstret MVC .

I Ursprungsmodellen implementerade jag under **Model-**mappen tre klasser, Product, IProductRepository och MockProductRepository. Product innehåller genomgående variabler för vad en produkt/uppfinning ska innehålla. MockProduktRepository är en mockup av olika produkter och ärver interfacet IProductRepository's egenskaper och ser till att de metoder som behövs för att kunna hämta rätt information implementeras i MockProduktRepositoryn.

Att uppdatera models-mappen i Uppfinnaren2.0 med AppDbContext och tillhörande delar för att få InMemoryDatabasen att fungera var lite klurigt. Men annars var det mesta relativt enkelt då Gill gör väldigt bra förklaringar i pluralsightkursen. Utan de förklaringarna hade det däremot varit svårt att få samma förståelse för projektets helhet. Det känns komplext och svårt att lära ut i just det här ämnet på ett annat sätt än detta.

Under **Controllers**-mappen i Uppfinnaren1.0 implementerade jag en ny controller – ProductController. Den tar in informationen om produkterna från Models-mapparna och tillsätter de värdena i en ViewModel vars klassnamn är ProductListViewModel, när värdena är tillsatta kan den informationen presenteras så som vyn(Views)är skapad att presentera sidan. Det här sker om/när användaren väljer att klicka på knappen Produkter, eftersom en controller reagerar på användarinteraktion.

Att uppdatera ProductControllers tillsammans med Views i Uppfinnaren 2.0 för att få applikationen mer dynamiskt var det svåraste i den här uppgiften tyckte jag. Det tog lång tid att klura ut hur allt fungerar, jag fick kolla många gånger på Gills material för att få ihop allt och förstå kopplingen.

Under **Views**-mappen i Uppfinnaren1.0 skapade jag en del nytt. Jag skapade en ny mapp: Product tillsammans med filen/vyn List som visas när användaren väljer meny-knappen produkter. Under mappen Shared och i filen _Layout läggs det till en knapp på rätt plats tillsammans med rätt "asp-controller och asp-action" för att användaren ska kunna ta sig till rätt URL och få rätt vy till sig vid knapptrycket. Under mappen Home lades det till en fil med namnet Contact för att visa en statisk kontaktsida, även för denna vy behövdes det läggas till en knapp, i samma fil _Layout som nämns ovan görs detta.

I filen Startup.cs i Uppfinnaren1.0 har jag implementerat IProductRepository interfacet tillsammans med MockProductRepository under ConfigureServices. För att kunna använda de klassernas "tjänster".

Tack vare Gills bra utbildningsmaterial har det gått relativt smärtfritt att lösa den här uppgiften. Man har fått många bra tips och tricks, dock är många rader kod väldigt likt Gills lösningar. Men det har förhoppningsvis gett mig den förståelse jag behöver för att lösa nästa uppgift på ett bra sätt.

Resultat

Det som i det stora hela har lagts till är kopplingen till en InMemoryDataBase och fler produkter/uppfinningar, produkter kan delas in i kategorier beroende på typ, fler sidor är tillagda beroende på kategori, detta har gjorts dynamiskt så att konceptet DRY följs i applikationen.

I Models är den största förändringen tillägg av AppDbContext och databas kopplingen, mockupsen används inte längre förutom i enhetstesten och kategorier med tillhörande klasser och funktionalitet har lagts till för att kopplas samman med produkterna. Med de nya klasserna så blir det mycket lättare att sortera ut produkter med hjälp av categoryId som har lagts till i produkterna och metoden GetProductsByCategory. Då kan man hämta alla produkter som går under samma kategori.

I ProductControllern har det istället för att returnera en vy med alla produkter statiskt så returneras en vy beroende på en in-parameter "category" som innehåller kategorinamnet, med hjälp av det kan vi returnera rätt vy beroende på vilken knapp användaren trycker på. Det blir enklare att underhålla med enbart en ViewResult-Metod än flera olika statiska, jag byggde det med flera olika statiska ViewResult-metoder från början. Funktionaliteten för vilket värde "category" ska tillsättas vid knapptryck ligger under Components-mapparna, detta har jag rett ut med hjälp av Gills video utan att kanske greppa helheten helt och hållet, även om det känns som att jag börjar förstå det mesta. Har dock svårt att sätta ord på det för stunden och det skulle bli mycket för mig att försöka förklara det i text.

Det som underlättar vidareutveckling och underhåll mest är att applikationen är dynamisk. Så om uppfinnaren skulle höra av sig och vilja lägga till ytterligare en kategori med fler produkter så kan man bara lägga till dessa i databasen och en ny sida skapas automatiskt för den nya kategorin. Samma om en kategori eller produkt skulle behöva tas bort så går det lika enkelt att göra. Det man kanske bör göra precis som Gill gör är att lägga dessa kategorier i en dropdown-meny istället för lite mer tydlighet. Men stylingen var inte huvudfokus för mig i denna uppgift.

Enhetstester

FilterProductsByCategoryMockData - kontrollerar att metoderna(GetCategoryByName & GetProductsByCategory) för att sortera produkter efter vilken kategori fungerar som den ska. Mycket av det dynamiska i applikationen bygger på att den här funktionaliteten fungerar. Man hade kunnat göra två test där man testar metoderna var för sig men valde att ha dem i samma test.

FilterProductsByCategoryDataBase - kontrollerar samma funktionalitet som testet ovan fast mot databasen istället, plus att den istället för att kontrollera antalet produkter i nuvarande hämtade kategori, kontrollerar den mot databasen att antalet kategorier(5) stämmer överens med hur många som ska finnas. Kan vara bra att testa då det styr mycket av dynamiken på sidan, exempelvis hur många olika menyflikar tillsammans med vyer som ska visas på sidan.

GetAllCategories - kontrollerar att funktionaliteten för att hämta alla kategorier fungerar som den ska. Detta test är viktigt för att kunna skriva ut alla kategorier på exempelvis första sidans meny.

GetAllProducts - kontrollerar att funktionaliteten för att hämta alla produkter fungerar som den ska. Detta test är viktigt för att kunna skriva ut alla produkter på exempelvis sidan för "Alla uppfinningar".

Alla dessa metoder och funktioner behövs för att helheten i applikationen ska fortsätta vara dynamisk. De bygger på varandra. Därför är det av vikt att testa dess funktionalitet så ofta man kan vid utveckling och underhåll. Går något test inte igenom så vet man att domänlogiken någonstans har blivit rubbad på.

Repo: https://github.com/campus-varnamo/uppfinnaren-jajo21