

ASP.NET MVC 2



Övningsuppgifter

Web Api och JSON

1. Du skall nu skapa ett HelloWorld projekt med ett web api. Skapa en apicontroller som du döper till HelloController. Skapa en metod som returnerar en sträng. Se till att metoden returnerar texten "Hello detta är text från ett web api". Anropa metoden från en webbläsare och se att den returnerar texten som en JSON sträng.
2. Skapa en actionmetod till i din HelloController . Låt den returnera en lista med strängar. Skapa en lista med strängar som du returnerar. Prova att anropa metoden och se resultatet i en webbläsare.
3. Skapa en modellklass i din lösning som du kallar för Person. Den skall ha 2 properties, Namn och Email.
 - a. Skapa en ny apicontroller (PersonController) med en metod som returnerar en person och ytterligare en metod som returnerar en lista med personer.
 - b. Hårdkoda in några personer och skicka med i metoderna.
 - c. Testa att anropa och se resultatet.
4. Du skall nu skapa en ny lösning. Den skall ha databaskoppling och metoderna skall ha inparametrar. Använd Northwind databasen.
 - a. Skapa en api controller för Produkter. Den skall ha en metod där man kan få tillbaka en produkt baserat på ett produktid som skickas in
 - b. Den skall även ha en metod där en lista med produkter kan returneras beroende på en sökparameter som skickas in. Denna parameter skall vara produktnamn
 - c. Testa att anropa metoderna med parametrar i url:en och kontrollera att rätt värden från databasen returneras.
 - d. Lägg på en inparameter till, unitprice, som gör urval beroende på pris på produkten. Testa att din lösning går att köra med två parametrar och ger rätt resultat.

Http Client

Här är det viktigt att du använder Postman för olika uppgifter är och lär dig hantera detta verktyg. Du skall nu till att börja med göra ett antal övningar mot ett web api som finns med olika typer av information om Star Wars filmerna. Använd Http Client för att skapa anropen. Detta web api finns under <https://swapi.co/>

1. Hämta all grundläggande data om Luke Skywalker och visa det sedan på en webbsida.
2. Hämta all data om planeten Arid (id 1) och visa på en webbsida.
3. Ta fram en lista med alla Star Wars filmer som gjorts.
 - a. Visa dessa i en lista med filmens titel, regissör (director) samt releasedatum för varje rad.
 - b. Se till att listan är sorterad i tidsordning så att den tidigaste filmen kommer först och den senaste sist i listan
4. Bygg en webbsida med en sökfunktion. Titta i SWAPI dokumentationen för att se hur en sökning kan göras. Det skall gå att söka på personer från Star Wars genom deras namn. När man söker på ett namn skall information om den personen visas. Det kan även vara flera personer beroende på hur sökningen ser ut. Finns inte någon person med det namnet skall texten "Inga personer som motsvarar sökningen" visas.
5. Du skall nu göra en valutaomvandlare liknande den som finns på forex.se. Välj ett web api som levererar aktuella valutakurser (tex <https://currencylayer.com/>) och läs in dessa när applikationen startar.

Valutaomvandlare

Jag vill: ☒ Köpa ☐ Sälja Datum: 2016-11-21

Jag köper **Jag betalar**

Euroland EUR 10 Sverige, SEK 103,86

FOREX Bank säljkurs: 10.3863 10 SEK ger dig 0,96 EUR

Aktuella kurser (räknat på 100 SEK) Senast uppdaterad: 2016-11-21 11:15.

9,63 EUR 10,17 USD 8,22 GBP 72,40 DKK 33,06 TRY

6. Du skall nu använda ett API som finns på <https://food2fork.com> . Det innehåller recept och ingredienser till maträtter. Gör en lösning där det går att söka på recept baserat på en ingrediens som sökparameter. Lägg även till möjlighet att sortera resultatet och mer funktionalitet tex paginering av sökresultatet

7. Bygg ihop flera saker så det blir en Star Wars applikation där man först kan söka och sedan klicka vidare för att se detaljer om filmer, personer och sedan det som finns kopplat till dessa
8. Hitta ett annat API som du kan testa att använda

<http://apikatalogen.se/>

<https://jsonplaceholder.typicode.com/>

Azure introduktion

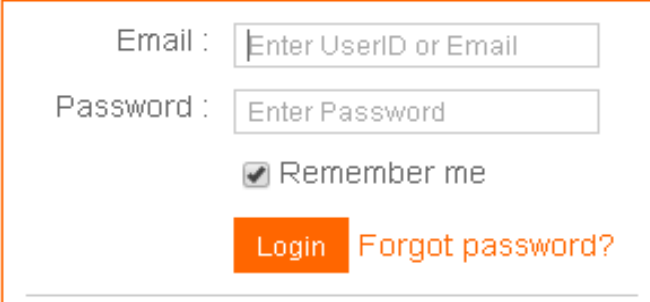
Här att det viktigt att du testat att använda olika verktyg så att du känner dig bekväm tex Cloud Explorer, Management Studio för databaser och även Azure Portalen.

1. Du skall nu deploya ditt Hello World projekt (där du skapade ditt första web api) till Azure. Gör detta direkt från Visual Studio men skapa webbappen i Azure först innan du publicerar. Testa att använda url:en till Azure och kontrollera att det fungerar.
2. Skapa en databas i Azure som innehåller en tabell Person med properties Namn och Email. Skapa ett eget web api med CRUD metoder mot den databasen dvs det skall gå att lägga till, ta bort, uppdatera och visa en person. Deploya ditt web api till Azure och se till att du får rätt connectionsträng dvs att den pekar på rätt databas. Testa sedan via Postman att alla metoder fungerar som de skall.
3. Utgå från den Blogg lösning som du gjorde på första MVC kursen (eller bygg en ny blogg) . Deploya databasen till Azure. Skapa sedan ett web api för att arbeta med mot databasen som du också deployar till Azure. Bygg presentationsdelen som en MVC lösning som kommunicerar mot web api:et via httpclient. Det skall gå att skapa, ta bort, uppdatera och söka på blogg inlägg.
4. Gör samma sak med Tomasos Pizzeria dvs bygg om den till 2 projekt ett som innehåller web api:er och ett som hanterar presentationen. Deploya till Azure.
5. Om du hinner. Läs igenom denna länken <https://docs.microsoft.com/sv-se/azure/application-insights/app-insights-overview> . Testa att använda dig av Insights.

Identity

Syftet med denna del av kursen är att repetera Identity och träna på att bygga lösningar med inloggning och behörigheter. Vi arbetade med detta på förra MVC kursen, även om alla inte gjorde de uppgifterna eftersom det var VG krav. Nu är det ett krav att alla ska kunna detta för att få G.

1. Skapa en lösning med ett enkelt inloggningsformulär. Skapa en lösning via Identity där man verifierar en användare med uppgifter från formuläret mot en Identity databas.



The image shows a login form with the following elements:

- Email :
- Password :
- ☒ Remember me
- [Forgot password?](#)

2. En vanlig situation när man har lösningar med inloggning är att utseendet och innehållet på webbsidorna anpassas för den användare som loggats in. Vad du skall göra är en applikation med inloggning som visar nyheter. De skall visas olika nyheter beroende på vilken användare som loggar in.
 - a. Skapa en lösning där det går att lägga upp användare som kan kopplas till olika roller. Varje roll skall vara kopplad till en geografisk region.
 - b. Skapa en databas med en News tabell och tre kolumner – Headline, Text, Region. Här skall nyheter kunna lagras och den skall kunna kopplas till en geografiskt region
 - c. Skapa en MVC applikation. Den skall först bestå av ett inloggningsformulär. När användaren har loggat in skall en lista med nyheter visas. Användaren skall bara se nyheter som är kopplade till den geografiska region som den tillhör.
 - d. Se till att alla metoder kräver inloggning så att användare inte kommer åt nyheter utan att logga in.
3. Du skall nu bygga ut din lösning och lägga till en ny typ av användare, Admin .
 - a. När en användare loggar in som admin skall den kunna lägga in nya nyheter , redigera befintliga nyheter samt ta bort en nyhet.
 - b. Här skall inte de nyheter som visas begränsas utan en Admin skall kunna se en lista med alla nyheter.
 - c. Se till att alla metoder kräver inloggning så att vanliga användare inte kommer åt det som bara är avsett för Admin användare.

Identity OAUTH

1. Bygg ut uppgift 1 från föregående avsnitt. Inloggningsformuläret skall nu även fungera med inloggning mot Facebook.
2. Fortsätt med uppgift 1. Skapa även möjlighet att logga in med Google och Twitter också. Vilka skillnader upplever du mellan de olika sätten att logga in?
3. Om du hinner testa fler providers för att verifiera inloggningar.

Arkitektur och best practices

1. Denna uppgift handlar om hur man skall kunna skapa tunna controllers och skicka business logik i en MVC lösning.
 - a. Skapa ett web api projekt. Lägg in EF så att det kan kommunicera med Northwind databasen.
 - b. Skapa en egen klass för business logik. Skapa en GetProducts metod i klassen. Web API:et skall ha en metod som tar emot ett produktnamn som en sökning skall kunna göras på.
 - c. Skapa en klass som du kallar för ProductReturn. Låt den innehålla properties Name, Price, Comment.
 - d. Din logik metod skall hämta alla produkter i databasen som motsvarar sökvillkoret och lägga över värdena i en lista av ProductReturn klassen.
 - e. Logiken är att om unitsinstock är lika med 0 eller om unitsinstock minus unitsonorder är mindre eller lika med 0 skall Comment vara "Not in stock". Skicka tillbaka listan från logik skiktet och låt sedan kontrollern returnera den.
2. Du skall nu bygga ut din lösning med ett repository. Detta skall vara i samma projekt. Se till att all kod som kommunicerar med databasen isoleras i ditt repository. Bygg på med fler metoder för att hantera produkter och även andra saker som finns i databasen.
3. Bygg på med ett nytt ASP.NET MVC projekt i din lösning. Detta projekt skall innehålla applikationen som pratar med web api:et. Du skall få till en fullständig SOA lösning med web api och alla skikt. Bygg på med mer funktionalitet mot databasen tex en sökfunktion för produkter och funktionalitet för att hantera ordrar. Det är viktigt att du följer den arkitektur som nu satts upp.

4. Du har nu skapat skiktning med logiska skikt. Du skall nu skapa samma lösning fast med fysiska skikt dvs varje skikt skall ligga i ett eget projekt. Testa att bryta ut business skiktet och repositoryt och lägg dessa i egna projekt.
5. Bygg om Tomasos pizzeria så att den lösningen får en skiktad arkitektur. Lägg in generell hantering av meddelanden och felhantering för hela applikationen.

Unit testing

1. Skapa en metod som tar emot två tal och dividerar dessa med varandra och returnerar svaret. Skriv ett enhetstest för denna metod och kör den i Visual Studio. Testa sedan att köra den flera gånger efter varandra med olika inparametrar. Använd xUnit.
2. Skapa en enkel miniräknare med de fyra räknesätten. Skriv enhetstester för att testa att alla metoder räknar rätt.
3. Tidigare i en uppgift skapade du en valutakonverterare. Skapa ett enhetstest som kontrollerar att den räknar rätt. Testa att köra metoderna flera gånger samtidigt med olika värden.
4. Skapa enhetstester för Tomasos pizzeria lösningen.
5. Testa de andra ramverken, MSTest och NUnit och skriv några tester med dessa.

ASP NET Webforms – Razor pages

1. Öppna ett webforms projekt. Lägg ut en label på sidan och sätt värdet på labeln till "Hello world" när sidan öppnas via kod.
2. Testa att bygga Blogg lösningen med Webforms istället. Det skall gå att söka på blogg inlägg och spara, uppdatera och ta bort ett inlägg.
3. Testa att bygga Blogg lösningen med Razor pages. Välj själv hur mycket funktionalitet som du vill ta med, det viktiga är att du testat att använda denna teknik också.