Web API, övning i att skapa en första endpoint och intro om attribute routing

I denna övningsuppgift får du testa på att bygga ett grundläggande Web API i .NET Core. Det viktiga är att förstå grundstrukturen, hur man bygger ett REST-API i .NET Core, andra viktiga saker, bl.a. Authentication, kommer vi att gå genom på kursen så småningom.

När man bygger ett REST-API brukar man oftast skapa flera olika endpoints. Vad är då en endpoint? ***En endpoint är vad man kan kalla en ”ingång” till en service (tjänst) eller en funktion i en applikation***. Endpoints är vad vi som API-utvecklare skapar för att ”visa funktionalitet utåt” för anropande applikationer. Anropande applikationer kan vara både andra applikationer som vi själva har utvecklat och som andra har utvecklat. En viktig aspekt av att arbeta med API:er är medvetenheten om att andra applikationer får åtkomst till funktionaliteten vi har utvecklat genom våra endpoints, inte på något annat sätt.

Det du ska göra i dem här uppgifterna är att skapa ett förenklat Web API med en enda endpoint: en POST-endpoint. (POST används för att skapa en resurs. Vanligtvis handlar det om att skapa en resurs mer ”permanent”, t.ex. lagra något i en databas men i denna uppgift sparar vi för enkelhetens skulle inte ned resultatet, utan vi tittar bara på det i responsen vi får tillbaka som svar på vår POST-request).

Den som anropar API:t ska kunna skicka in en valfri text och få tillbaka texten översatt till rövarspråket som response på sin request.

Info om rövarspråket: https://sv.wikipedia.org/wiki/R%C3%B6varspr%C3%A5ket

Uppgift 1

Skapa ett nytt Web API projekt i Visual Studio.

1. Öppna Visual Studio, välj **Create a new project**
2. Välj **ASP.NET Core Web Application**, klicka på knappen **Next**
3. Namnge ditt projekt: ”**RobberLanguageAPI”** och klicka på knappen **Create**
4. **I dialogen som nu visas, välj ASP.NET Core Web API.** Kontrollera att **.NET Core** och **ASP.NET Core 5.0** är dem valda inställningarna i dialogen. Klicka sedan på **Create-**knappen**.**

Notera: När grundstrukturen för Web API-projektet skapas i Visual studio(VS) kommer en del saker med default. För att visa hur ett API fungerar skapar VS automatiskt ett API i ditt

projekt: ”WeatherForeCast”. Detta ska du inte bygga vidare på, men du kan testa att köra igång applikationen redan nu genom ”IIS express”-knappen och se hur det ser ut i webbläsaren. Då öppnas API:t med ett verktyg som heter Swagger. Swagger används dels för att dokumentera API:er men kan även användas för att exponera och testa API-endpoints.

WeatherForecast-API:t har endast en endpoint: GET/WeatherForecast som hämtar aktuellt väder. Du kan testa API:t genom att klicka på knappen **Try it out** vid endpointen i webbläsaren. **Du får då tillbaka en body (data) och lite info om Responsen som kom tillbaka som svar på Requesten**.

- Stäng ner WeatherForecast-API:t och gå tillbaka till VS.

Uppgift 2

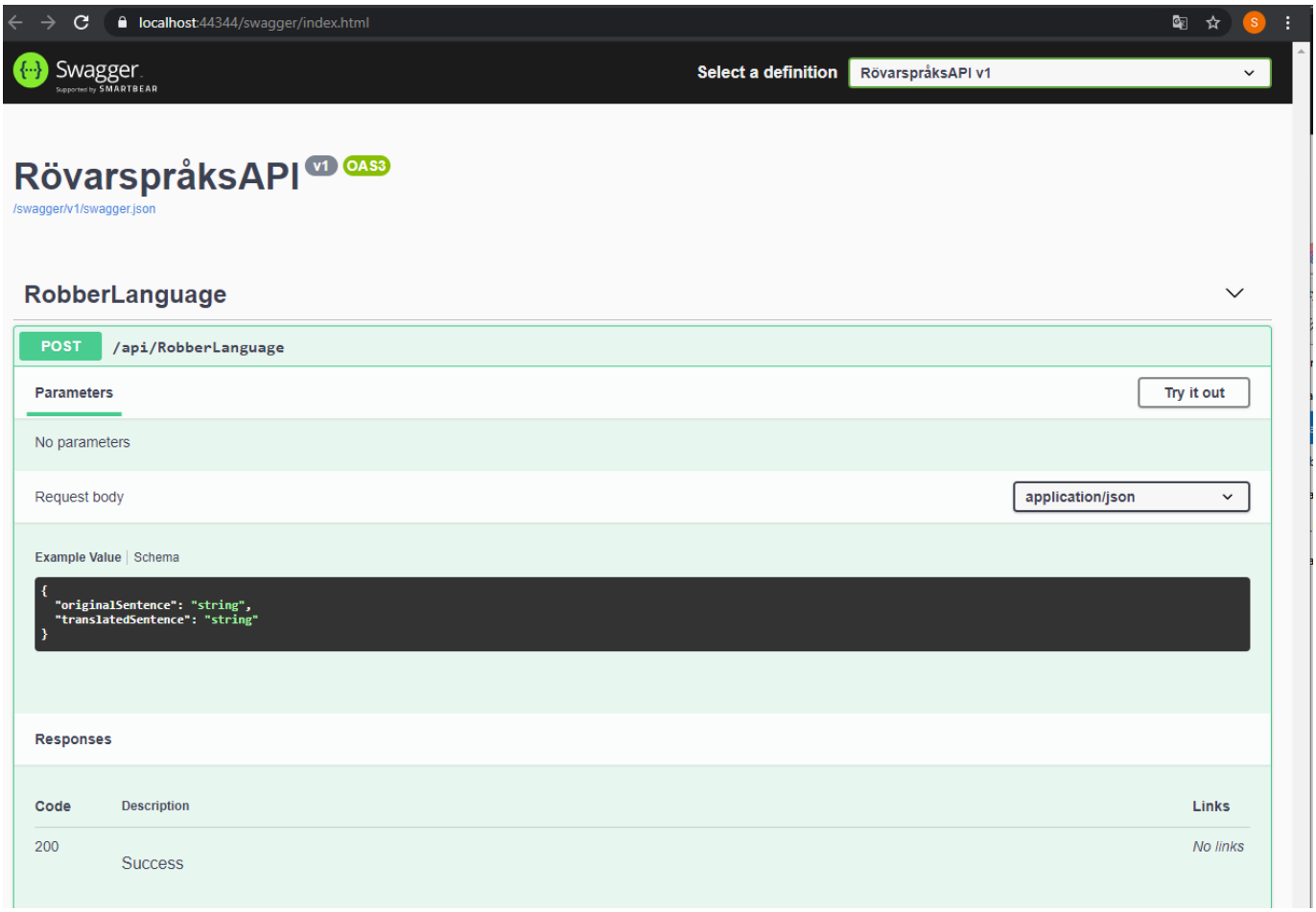
Du ska som det beskrevs ovan skapa ett förenklat API med en enda endpoint (en POST-endpoint).

1. Skapa en Modell-klass **”Translation”** i ditt projekt som kan innehålla följande properties: ”**OriginalSentence”**, ”**TranslatedSentence”**.
2. Lägg till en ny Controller genom att högerklicka på mappen **Controllers** och välj **add → Controller.** I dialogen som öppnas väljer du **API** i menyn till vänster, sedan **API Controller - Empty** och klickar på knappen **Add.** Namnge din controller **”TranslationController”** och klicka på knappen **Add.**
3. I controllern ska du nu skapa en metod som tar in en sträng och översätter den till rövarspråket (se språkregler: https://sv.wikipedia.org/wiki/R%C3%B6varspr%C3%A5ket). Metoden ska bara översätta strängen och returnera en korrekt översatt sträng i enlighet med Rövarspråkets regler.

   - För enkelhetens skull kan du skapa din översättnings-metod som en separat metod i RobberLanguageTranslationController (alltså, inte som en endpoint, utan snarare en metod som ser ut ungefär såhär: **private static string TranslateSentence**). Vanligtvis vill man inte lägga metoder som hanterar affärslogik direkt i controllern men för denna uppgift går det bra.
4. Du ska nu skapa en POST-endpoint i din controller. Den ska returnera ett **Translation**-objekt innehållandes värden för OriginalSentence och TranslatedSentence.

   I din endpoint måste du alltså anropa din **TranslateSentence**-metod som du skapade i steg 3.
5. För att testa din endpoint/anropa den kan du köra igång din applikation i Visual Studio med **IIS Express.**

Ditt API ska nu öppnas i webbläsaren genom Swagger. Det bör se ut på ett ungefär såhär i webbläsaren:



6. Du kan nu testa din endpoint genom att klicka på knappen **Try it out** i webbläsarfönstret.
7. Då öppnas en texteditor-ruta där du kan fylla i data i **request bodyn**, det vill säga datan som du vill skicka med din POST-request. För att testa din endpoint behöver du bara skicka med data för **OriginalSentence**- valfri mening som du vill ha översatt med rövarspråket.

   Genom detta har du nu definierat den **JSON-data** som du vill skicka med din **POST-request**.
8. Klicka på den stora blå knappen **Execute** under texteditor-rutan.
9. Nedanför visas nu sektionen **Responses**

   - Där kan vi se följande: vilken data vi får tillbaka (**Response Body**), som i vårt fall ska innehålla ett nytt värde för propertyn **”translatedSentence”**= den med requesten inskickade meningen översatt till rövarspråket.

   OBS! *Kontrollera i Response Body att translatedSentence nu innehåller en korrekt översättning till rövarspråket.*

- Vi får även veta att HTTP-statuskoden som vi fick tillbaka efter att ha skickat vår request är 200, vilket betyder att requesten gick genom som förväntat, den är OK.

(Vi kommer så småningom att lära oss hur vi kan returnera olika ”resultat” som svar på anropa till våra API-endpoints, och hur vi kan definiera olika HTTP-statuskoder beroende på vad resultatet av anropet till en endpoint blev. Vi lämnar dock det åt sidan så länge.

10. Du har nu skapat din första POST-endpoint, testat att anropa den genom att skicka med JSON-formaterad data i request-bodyn och kontrollerat responsen som kom tillbaka som svar på anropet. Bra jobbat!

Uppgift 3

I ett WEB API kan man justera så att olika endpoints får mer anpassade Routes- det vill säga ”adresser” kopplade till sig. Det för att t.ex. tydligare illustrera vad en viss Endpoint i API:t faktiskt gör. Genom featuren Attribute Routing i ASP.NET Core Web API kan man ställa in så att routen till en viss endpoint bättre matchar vad endpointen gör. Man kan även ändra Routen för en controller som helhet.

1. Testa att ändra Route-attributet för din controller som helhet till ”api/RobberLanguage”. (När du skapar en Web API-controller får du med ett default-värde för Route-attributet som du alltså i detta steg ska ändra).
2. Låt Route-attributet som är satt på controllern som helhet vara kvar och sätt ett passande mer specifikt Route-attribut på din POST-endpoint, till exempel: Route(”CreateTranslation”).

   - Din endpoint kommer nu att följa det här mönstret för sin Request-URL:

   https://localhost:44344/api/RobberLanguage/CreateNewTranslation

   Notera: Du kan kolla hur endpointen får olika adresser när du gör de olika stegen genom att köra programmet och titta i Swagger-dokumentationen i webbläsaren.

   Om vi skulle bygga vidare på det här API:t skulle vi sedan kunna lägga till passande routes för respektive Endpoint och/eller justera våra ovan satta routes på ett sätt som vi tycker passar.