Unter ASP.Net 5 mit Entity Framework 7 arbeiten

Mein Ziel in diesem Tutorial ist es, das neue ASP.Net 5 oder ASP MVC 6 mit dem Entity Framework 7 auszustatten. Dazu soll eine Tabelle Personal mit einer Adresstabelle in einer n:m Beziehung stehen. Na dann… lets go.

1. **Projekt erstellen**

Unter Web finden wir in Visual Studio 2015 den Eintrag ASP.NET Web Application

Es ist egal, ob wir ein leeres Projekt, die Web API oder Web Application wählen. Dies sind nur die Vorlagen. Wichtig ist nur, einen aus den unteren ASP.NET 5 Templates zu wählen

Ich habe meine Klassen gerne kategorisiert und daher lege ich für die Datenbank Modelle einen Ordner dbModels an.

2. Entity Framework 7 installieren und Klassen anlegen

a) *Entity Framework installieren*

mit Alt + T + N + O rufen wir die Nuget Console auf und geben dort folgendes ein um die neuste EF7 zu installieren:
[crayon-67679ecbc24e9135124800/]
und dann noch die Commands:
[crayon-67679ecbc24ef195561911/]
Da das ganze noch in den Kinderschuhen ist, hat bei mir in der project.json Datei die Abhängigkeit zu den Commands gefehlt. Diese muss man evtl. manuell nachbessern. Dazu in der

project.json Datei
[crayon-67679ecbc24f0124545254/]
mit
[crayon-67679ecbc24f1087557834/]
ersetzen. Und gleichzeitig einen Blick drauf werden, ob unter dependencies EF und Commands richtig instelliert sind:
[crayon-67679ecbc24f3188689209/]
Falls man eine Änderung vornehmen musste, Project speichern, schließen und als Administrator neu aufmachen.

b) *Code First Modell erstellen*

Jetzt kann man das Abbild der Datenbank, das Modell der Personen Tabelle als Klasse in den neu angelegten Ordner erstellen:
[crayon-67679ecbc24f4297290451/]
wobei die PersNr später der PrimaryKey sein wird. Heißt eine Property Id, dann erkennt Entity Framework diese automatisch als PrimaryKey an.

c) *dbContext Klasse erstellen*

Auch EF7 arbeitet mit einer Klasse, die von DbContext erbt. Hier legen wir Grundlegende Eigenschaften zur Verbindung und den Modellen an. Ich habe diese Klasse ebenfalls in den dbModels Ordner gelegt.
[crayon-67679ecbc24f5853002985/]
Der 1.Teil mit dem Konstruktor ist da, der sagt, wenn die Datenbank nicht vorhanden ist, so soll doch bitte eine erstellt werden.

Der 2. Teil OnConfiguring: Hier gebe ich den Datenbankpfad an.

Der 3. Teil von OnModelCreating, hier kann ich unter anderem den Tabellennamen und den Namen des Schemas wählen. Ich habe das gerne, dass zusammenhängende Tabellen einen gemeinsamen Namen bekommen. Per Standard heißt das Schema immer dbo.

Der 4. Teil ist eine Property der dbContext Klasse vom Typ

DbSet, damit wir über diese später Daten lesen,einfügen,ändern und löschen können.

d) *Migration erstellen*

Nun ist es an der Zeit eine Datenbankverbindung zu testen und gleichzeitig eine erste Migration zu erstellen.

Dazu öffnen wir wieder die NuGet Console.

Da wir mit ASP.Net 5 arbeiten, gelten für uns nicht die alten Entity Framework Befehle, sondern die von DNX. Diese verlang von uns, dass wir die Befehle dort ausführen, wo sich die Datei befindet. Die EF Dateien befinden sich in der Ordnerstruktur src/[Projektname].

Navigieren kann man mit **cd Ordnername** nach vorne bzw. **cd ..** einen Satz zurück. Mit **dir** bekommt man eine Übersicht.

Konkret muss man in den Ordner **src\ProjektName** wechseln

Nun gibt man dort folgendes ein:
[crayon-67679ecbc24f7957464791/]
Dann sollte man auch ein Done. Als Bestätigung bekommen. Weiterhin sieht man nun einen Ordner Migrations und in der Datenbank ist die Datenbank angelegt worden mit der Tabelle Personal

3. Klasse Adresse erstellen, welche eine Beziehung zu der Klasse Personal hat

Jetzt erstellen wir eine weitere Klasse Adress

Eine Adresse kann nur einer Person gehören. Daher bekommt jede Adresse nur eine Person zugewiesen. Gäbe es einen Fall, wo die

Beziehungsklasse mehrere Personen enthalten kann, so würde man dies in eine ICollection<Person> packen.
[crayon-67679ecbc24f9520003068/]

4. Default Connection String

Im Punkt 2c) haben wir einen Connection String direkt eingetragen. Will man das Projekt eines Tages publizieren, könnte man ihn so nicht ändern. Deshalb müssen wir diesen Connection String auslagern. Dazu erstellt man eine neue config.json Datei mit der Vorlage von **ASP.NET Configuration File**

und schreibt dort den Connection String rein.

Weiter muss nun auch die startup.cs angepasst werden:

1. Property erstellen:
   [crayon-67679ecbc24fa854311416/]
2. Den Konstruktor anpassen:
   [crayon-67679ecbc24fb592182167/]

3. die alte Context Klasse anpassen:
   [crayon-67679ecbc24fc796601802/]

4. Damit die Datenbank erstellt und aktuallisiert werden kann, fügen wir noch folgendes ein:
   [crayon-67679ecbc24fd466765874/]