Hands-on lab

Lab: Entity Framework

December 2015

Exercise 1: The Star Wars API

Visit https://swapi.co and familiarize yourself with this API.

Exercise 2: Build a Console application to clone the Star Wars movies

Step 1 - Build the EF model

Create a new Solution called *StarWarsUniverseClone* and add a Class library called *StarwarsUniverse.Model*. Make sure your project targets the latest .NET version (4.6)

1. Add Entity Framework and JSON.net libraries to both projects (via Nuget)
   ```
   install-package newtonsoft.json
   install-package entityframework
   ```

   > Note: newtonsoft.json stand for JSON.NET, a hugely popular library for serializing objects from/to JSON strings.

2. Create a the following classes in the model project:

   C#

   ```
   using Newtonsoft.Json;
   using System;

   namespace StarWarsUniverse.Model
   {
       public abstract class SWResource
       {
           public DateTime Created { get; set; }
           public DateTime Edited { get; set; }

           [JsonProperty(PropertyName = "url")]
           public string ResourceUri { get; set; }
       }
   }
   ```

   > Note: [JsonProperty(PropertyName = "url")] maps the json attribute *url* to the property *ResourceUri*

3. Add another class

   C#

   ```
   using Newtonsoft.Json;
   using System;

   namespace StarWarsUniverse.Model
   ```

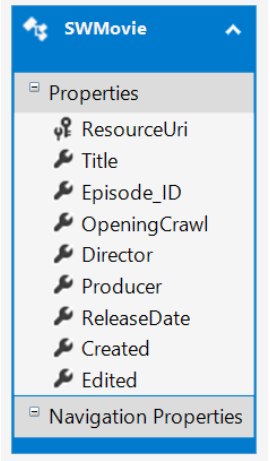
```
{
    public class SWMovie : SWResource
    {
        public string Title { get; set; }

        public int Episode_ID { get; set; }

        [JsonProperty(PropertyName = "opening_crawl")]
        public string OpeningCrawl { get; set; }

        public string Director { get; set; }

        public string Producer { get; set; }

        [JsonProperty(PropertyName = "release_date")]
        public DateTime ReleaseDate { get; set; }
    }
}
```

4. Create a folder called *DataLayer* in the Console project and create a class called *StarWarsContext*, defining a *DbSet* of *SWMovie* objects. Furthermore, define ResourceUri to be PK using Fluent API. The database created should be named *StarWarsDB*.

C# (incomplete)

```
public class StarWarsContext : DbContext
{
    public StarWarsContext() : base("StarWarsDB")
    { }
...
}
```

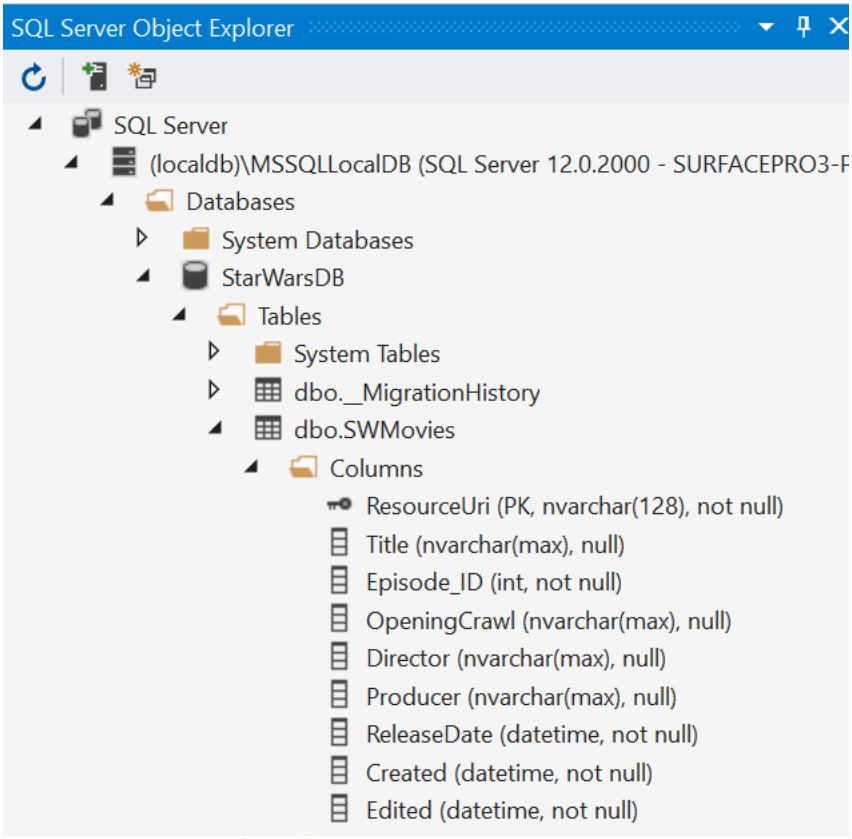
5. Verify your model using EF Powertools. It should look something like this:



Note: installation instructions on http://bit.ly/EFPTVS2015

Step 2 - Create an Initial Migration

Create a migration called “Initial” and create the database. Run it and verify the database.



Step 3 – Seed with data

Now we will write the necessary code to retrieve all StarWars movies from the public API and store them inside our own database, while preventing duplicates.

1. Create a folder called *Services* (in the Console project) and define the following interface:

C#

```
using StarWarsUniverse.Model;
using System.Collections.Generic;

namespace StarWarsUniverse.Services
{
    interface ISWDataService
    {
        List<SWMovie> GetAllSWMovies();
        SWMovie GetSWMovieDetails(string uri);
    }
}
```

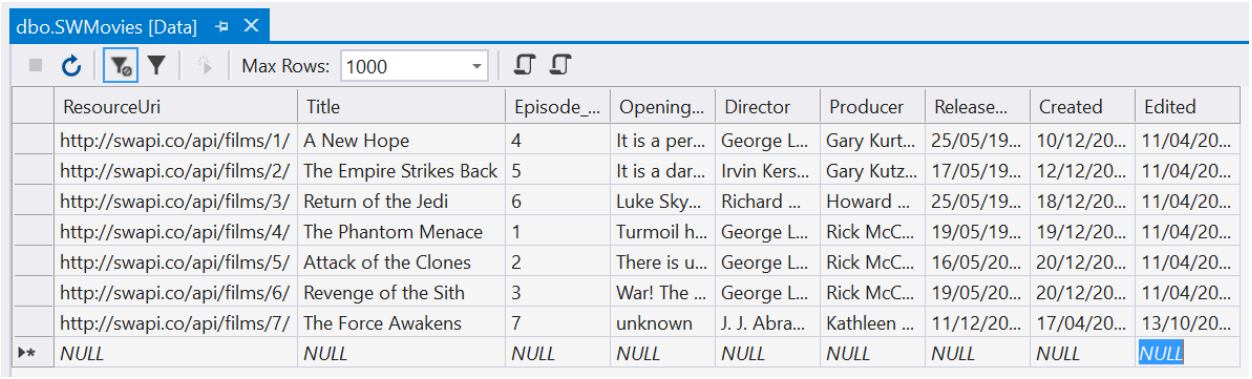
2. Now write a class called *SWDataService* that implements this interface and retrieves the data while serializing JSON to C# objects.

C#

```
using System;
using System.Collections.Generic;
using StarWarsUniverse.Model;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace StarWarsUniverse.Services
{
    public class SWDataService : ISWDataService
    {

        public List<SWMovie> GetAllSWMovies()
        {
            string swapifilms = "http://swapi.co/api/films";
            var uri = new Uri(String.Format("{0}?format=json", swapifilms));
            var client = new HttpClient();
            var response = Task.Run(() => client.GetAsync(uri)).Result;
            response.EnsureSuccessStatusCode();
            var result = Task.Run(() =>
response.Content.ReadAsStringAsync()).Result;
            var root =
JsonConvert.DeserializeObject<RootObject<SWMovie>>(result);
            return root.results;
        }

        public SWMovie GetSWMovieDetails(string uri)
        {
            var client = new HttpClient();
            var response = Task.Run(() => client.GetAsync(uri)).Result;
            response.EnsureSuccessStatusCode();
            var result = Task.Run(() =>
response.Content.ReadAsStringAsync()).Result;
            var movie = JsonConvert.DeserializeObject<SWMovie>(result);
            return movie;
        }
    }

    class RootObject<T>
    {
        public int count { get; set; }
        public object next { get; set; }
        public object previous { get; set; }
        public List<T> results { get; set; }
```

```
    }
}
```

3. Modify the Seed-method to retrieve all movies from the public api and store these objects into the database. Run your Seed-method and verify the data. Run your Seed-method again and check there are still only 7 entries in the *SWMovies* table.



ResourceUri	Title	Episode_...	Opening...	Director	Producer	Release...	Created	Edited
http://swapi.co/api/films/1/	A New Hope	4	It is a per...	George L...	Gary Kurt...	25/05/19...	10/12/20...	11/04/20...
http://swapi.co/api/films/2/	The Empire Strikes Back	5	It is a dar...	Irvin Kers...	Gary Kutz...	17/05/19...	12/12/20...	11/04/20...
http://swapi.co/api/films/3/	Return of the Jedi	6	Luke Sky...	Richard ...	Howard ...	25/05/19...	18/12/20...	11/04/20...
http://swapi.co/api/films/4/	The Phantom Menace	1	Turmoil h...	George L...	Rick McC...	19/05/19...	19/12/20...	11/04/20...
http://swapi.co/api/films/5/	Attack of the Clones	2	There is u...	George L...	Rick McC...	16/05/20...	20/12/20...	11/04/20...
http://swapi.co/api/films/6/	Revenge of the Sith	3	War! The ...	George L...	Rick McC...	19/05/20...	20/12/20...	11/04/20...
http://swapi.co/api/films/7/	The Force Awakens	7	unknown	J. J. Abra...	Kathleen ...	11/12/20...	17/04/20...	13/10/20...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Step 4 – Retrieve the data from the database

Now write code in Program.cs that executes the following scenario:

Print all movies, including title, episode id and release date, and sort them by episode id.

Example output:

```
=== Star Wars Movies ===
Episode 1 - The Phantom Menace
        Released: 19/05/1999
Episode 2 - Attack of the Clones
        Released: 16/05/2002
Episode 3 - Revenge of the Sith
        Released: 19/05/2005
Episode 4 - A New Hope
        Released: 25/05/1977
Episode 5 - The Empire Strikes Back
        Released: 17/05/1980
Episode 6 - Return of the Jedi
        Released: 25/05/1983
Episode 7 - The Force Awakens
        Released: 11/12/2015
========================
```

Exercise 3: Add Star Wars Planets to the model and import them

Step 1 : extend the data model

Add a class *SWPlanet*

C#

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;

namespace StarWarsUniverse.Model
{
    public class SWPlanet : SWResource
    {
        public string Name { get; set; }

        [JsonProperty("rotation_period")]
        public int RotationPeriod { get; set; }

        [JsonProperty("orbital_period")]
        public int OrbitalPeriod { get; set; }

        public int Diameter { get; set; }

        public string Climate { get; set; }

        public string Gravity { get; set; }

        public string Terrain { get; set; }

        [JsonProperty("surface_water")]
        public string SurfaceWater { get; set; }

        public string Population { get; set; }

        [JsonIgnore]
        public virtual List<SWMovie> Films { get; set; }

        [JsonProperty(PropertyName = "films")]
        public List<string> FilmUris { get; set; }
    }
}
```

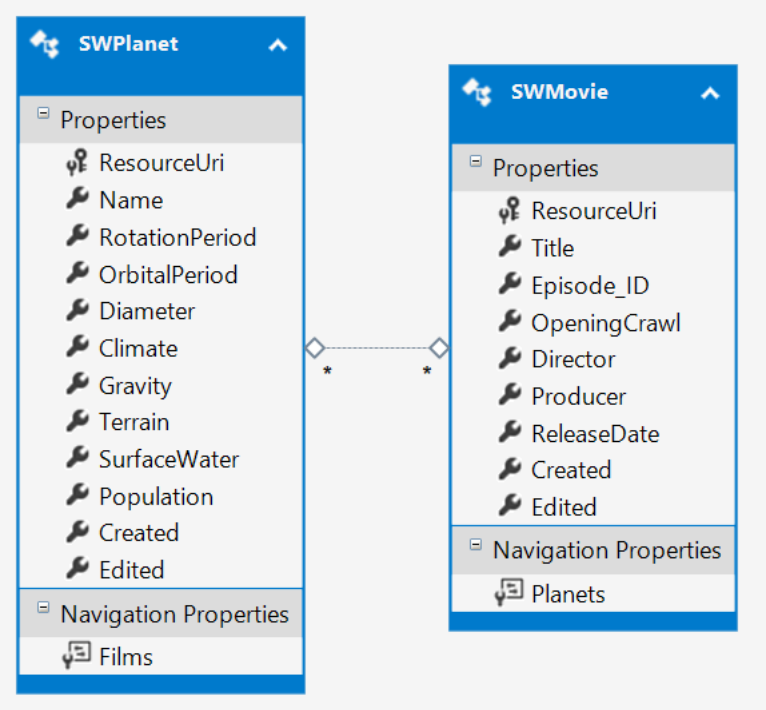
Add the following properties to SWMovie

C#

```
[JsonIgnore]
public virtual List<SWPlanet> Planets { get; set; }

[JsonProperty(PropertyName = "planets")]
public List<string> PlanetUris { get; set; }
```

Note: the api “links” together objects using its URL. We keep these uri but we also implement lists of POCO’s.

After model validation, you should get the following diagram:



Step 2 : create second migration

Create a second migration called *AddPlanets* that adds a second table and new columns to the database. Run the migration and verify the database structure.

Step 3 : modify seed method

Augment the seed method with the following scenario:

- Retrieve all movies and add or update them (this was already present)
- Retrieve all planets
- Assign the planets to the movies (and vice versa)

Note: if you do this correctly, the table *SWPlanetSWMovie* will contain data…

Step 4 : Modify Program.cs

Now modify the program that with every movie, also the planets are listed.

Example output:

```
=== Star Wars Movies ===
Episode 1 - The Phantom Menace
   Released: 19/05/1999
   [( Naboo ) ( Coruscant ) ]
Episode 2 - Attack of the Clones
   Released: 16/05/2002
   [( Kamino ) ( Geonosis ) ( Naboo ) ( Coruscant ) ]
Episode 3 - Revenge of the Sith
   Released: 19/05/2005
   [( Alderaan ) ( Dagobah ) ( Naboo ) ( Coruscant ) ]
Episode 4 - A New Hope
   Released: 25/05/1977
   [( Alderaan ) ( Yavin IV ) ]
Episode 5 - The Empire Strikes Back
   Released: 17/05/1980
   [( Hoth ) ( Dagobah ) ( Bespin ) ]
Episode 6 - Return of the Jedi
   Released: 25/05/1983
   [( Dagobah ) ( Endor ) ( Naboo ) ( Coruscant ) ]
Episode 7 - The Force Awakens
   Released: 11/12/2015
   []
========================
```

Exercise 4 (optional): Add all Star Wars entities to the database

Now add all the remaining entities to the database and add a migration per entity:

- People
- StarShips
- Vehicles
- Species