SPX FLOW

Challenge:

A client wants to build an API that allows users to submit reviews on their products. The client wants the users to submit their comments and a star rating in the review. A sample of the products data is provided in the “products.xml” file.

Solution:

Technologies used:

.Net(C#)

Entity frame work

Web API

Angular JS

HTML 5

Model classes:

These are data entities to map with data tables

Here I have created two classes:

1. Product 2.Prodcut review

Product:

Under product we have Product id, Product title, Shor description, Brand and Date Published

Product review:

Here we are binding the review data based on product id.

The fields which I have enclosed are Review ID,Rating, comment and User information

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace SPXFlowWebApiTest.Models
{
    public class Product
    {
        [Key]
        public long Id { get; set; }
```

```
        [Required]
        public string Title { get; set; }

        public string ShortDescription { get; set; }

        public BrandName Brand { get; set; }

        public List<Review> Reviews { get; set; }

        public System.DateTime DatePublished { get; set; }
    }

    public class Review
    {
        [Key]
        public long ReviewId { get; set; }

        [Required]
        public int Rating { get; set; }

        public string Comment { get; set; }

        public string User { get; set; }

        [ForeignKey("Product")]
        public long ProductId { get; set; }

        public Product Product { get; set; }
    }

    public enum BrandName
    {
        APV,
        Airpel,
        PowerTeam,
    }


}
```

Data Context

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace SPXFlowWebApiTest.Models
{
    public class SPXFlowDbContext : DbContext
    {
        public SPXFlowDbContext()
            : base("SPXFlowDbConnection")
        {
```

```
        }

        public DbSet<Product> Products { get; set; }

        public DbSet<Review> Reviews { get; set; }
    }
}
```

Data Context Initializer and seeding from XML

Seeding from and context initializing taking place structured below

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Web;
using System.Xml.Serialization;


namespace SPXFlowWebApiTest.Models
{
    public class SPXFlowDbContextInitializer<T> :
DropCreateDatabaseIfModelChanges<SPXFlowWebApiTest.Models.SPXFlowDbContext>
    {
        protected override void Seed(SPXFlowWebApiTest.Models.SPXFlowDbContext context)
        {
            var products = ReadXmlData();

            foreach (var product in products.product)
            {

                var dbproduct = new Product
                {
                    Id = product.id,
                    Title = product.title,
                    ShortDescription = product.shortDescription,
                    Brand = (BrandName)product.brand,
                    DatePublished = DateTime.Now
                };

                List<Review> lstreviews = new List<Review>();

                foreach (var review in product.reviews)
                {
                    lstreviews.Add(new Review
                    {
                        User = review.user,
                        ProductId = dbproduct.Id,
                        Comment = review.comment,
                        Rating = !string.IsNullOrEmpty(review.rating) ?
                                  int.Parse(review.rating) : 0
                    });
                }
```

```
                dbproduct.Reviews = lstreviews;

                context.Products.Add(dbproduct);

                context.SaveChanges();
            }
        }

        private products ReadXmlData()
        {
            products prdocuts = null;

            XmlSerializer serializer = new XmlSerializer(typeof(products));

            string path =
System.Web.HttpContext.Current.Server.MapPath("~/App_Data/productsData.xml");

            StreamReader reader = new StreamReader(path);
            prdocuts = (products)serializer.Deserialize(reader);
            reader.Close();

            return prdocuts;
        }
    }
}
```

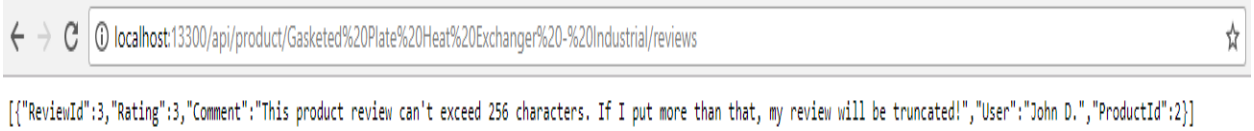
Listing of all products: GET (api/products):

```
[Route("")]
        [HttpGet]
        public IEnumerable<DTO.Product> GetAllProducts()
        {
            var products = _sampledbContext.Products;
            List<DTO.Product> lstofproducts = new List<DTO.Product>();
            if (products != null && products.Count() > 0)
            {
                products.ToList().ForEach(product =>
                {
                    lstofproducts.Add(new DTO.Product
                    {
                        Title = product.Title,
                        Id = product.Id,
                        ShortDescription = product.ShortDescription,
                        DatePublished = product.DatePublished,
                        Brand = (DTO.BrandName)product.Brand
                    });
                });
            }

            return lstofproducts;
        }
```

[{"Id":1,"Title":"APV Gaulin Mono-Block","ShortDescription":"An integrated design of cylinders (3 or 5), pump valves, plunger lubrication and inlet/outlet in one block.","Brand":0,"DatePublished":"2017-11-11T12:57:13.487"},{"Id":2,"Title":"Gasketed Plate Heat Exchanger - Industrial","ShortDescription":"Gasketed Plate Heat Exchanger for Industrial Applications.","Brand":0,"DatePublished":"2017-11-11T12:57:14.05"}]

Listing all reviews for a product: Get **api/products/{productTitle}/reviews**

```
        [Route("{productTitle}/reviews")]
        [HttpGet]
        public IEnumerable<DTO.Review> GetAllReviewsofProduct(string productTitle)
        {
            var products = _sampledbContext.Products;
            List<DTO.Review> lstofproductreviews = new List<DTO.Review>();
            if (products != null && products.Count() > 0)
            {
                var product = products.ToList().SingleOrDefault(c => c.Title ==
productTitle);
                var reviews = _sampledbContext.Reviews.Where(c => c.ProductId ==
product.Id);
                if (reviews != null && reviews.Count() > 0)
                {
                    reviews.ToList().ForEach(review =>
                    {
                        lstofproductreviews.Add(new DTO.Review
                        {
                            Comment = review.Comment,
                            ProductId = review.ProductId,
                            Rating = review.Rating,
                            ReviewId = review.ReviewId,
                            User = review.User
                        });
                    });
                }
                else
                {
                    ControllerContext.Request.CreateResponse(HttpStatusCode.NotFound);
                }
            }

            return lstofproductreviews;
        }
```



ADD Review:

This is Method to Add a comment to the product

```
  [Route("addreview")]
        [HttpPost]
        public IHttpActionResult AddProduct(DTO.Review productreview)
        {
            try
            {
                if (!ModelState.IsValid)
                {
                    throw new ValidationException("Invalid User", ModelState);
                }

                _sampledbContext.Reviews.Add(new Review
                {
                    ReviewId = productreview.ReviewId,
                    User = productreview.User,
                    ProductId = productreview.ProductId,
                    Comment = productreview.Comment
                });
                _sampledbContext.SaveChanges();

                return Ok();
            }
            catch (Exception ex)
            {
                return InternalServerError(ex);
            }
        }
```

Update Review:

This is the implementation to update the review based on ID

```
[Route("updatereview/{id:int}")]
        [HttpPut]
        public IHttpActionResult UpdateProduct(int id, DTO.Review review)
        {
            try
            {
                if (!ModelState.IsValid)
                {
                    throw new ValidationException("Invalid User", ModelState);
                }

                var dbreview = _sampledbContext.Reviews.SingleOrDefault(r => r.ReviewId
== id);
                if (dbreview != null)
                {
                    dbreview.Comment = review.Comment;
                    dbreview.ProductId = review.ProductId;
                    dbreview.Rating = review.Rating;
                    dbreview.User = review.User;
                    _sampledbContext.Entry(dbreview).State =
System.Data.Entity.EntityState.Modified;
                    _sampledbContext.SaveChanges();
                }

                return Ok();
            }
            catch (Exception ex)
            {
                return InternalServerError(ex);
            }
        }
```

Delete Review:

This is the implementation to delete the review based on product id .

```
[Route("deletereview/{id:int}")]
        [HttpDelete]
        public IHttpActionResult DeleteProduct(int id)
        {
            try
            {
                var dbreview = _sampledbContext.Reviews.SingleOrDefault(r => r.ReviewId
== id);
                if (dbreview != null)
                {
                    _sampledbContext.Reviews.Remove(dbreview);
                    _sampledbContext.SaveChanges();
                }

                return Ok();
            }
```

```
            catch (Exception ex)
            {
                return InternalServerError(ex);
            }
        }
```

DATA MIGRATION:

```
namespace SPXFlowWebApiTest.Migrations
{
    using System;
    using System.Data.Entity.Migrations;

    public partial class Initial : DbMigration
    {
        public override void Up()
        {
            CreateTable(
                "dbo.Products",
                c => new
                    {
                        Id = c.Long(nullable: false, identity: true),
                        Title = c.String(nullable: false),
                        ShortDescription = c.String(),
                        Brand = c.Int(nullable: false),
                        DatePublished = c.DateTime(nullable: false),
                    })
                .PrimaryKey(t => t.Id);

            CreateTable(
                "dbo.Reviews",
                c => new
                    {
                        ReviewId = c.Long(nullable: false, identity: true),
                        Rating = c.Int(nullable: false),
                        Comment = c.String(),
                        User = c.String(),
                        ProductId = c.Long(nullable: false),
                    })
                .PrimaryKey(t => t.ReviewId)
                .ForeignKey("dbo.Products", t => t.ProductId, cascadeDelete: true)
                .Index(t => t.ProductId);

        }

        public override void Down()
        {
            DropForeignKey("dbo.Reviews", "ProductId", "dbo.Products");
            DropIndex("dbo.Reviews", new[] { "ProductId" });
            DropTable("dbo.Reviews");
            DropTable("dbo.Products");
        }
    }
}
```

Migration Configuration:

```
namespace SPXFlowWebApiTest.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;
    using SPXFlowWebApiTest.Models;
    using System.Xml.Serialization;
    using System.IO;
    using System.Collections.Generic;

    internal sealed class Configuration :
DbMigrationsConfiguration<SPXFlowWebApiTest.Models.SPXFlowDbContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = false;
        }


    }
}
```

Seeding config in Global.ASAX:

```
using SPXFlowWebApiTest.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace SPXFlowWebApiTest
{
    public class WebApiApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            GlobalConfiguration.Configure(WebApiConfig.Register);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);

GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();
```

```
            Database.SetInitializer<SPXFlowDbContext>(new
SPXFlowDbContextInitializer<SPXFlowDbContext>());
        }


    }
}
```

DTO Objects:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Runtime.Serialization;
using System.Web;

namespace SPXFlowWebApiTest.DTO
{
    [DataContract]
    public class Product
    {
        [DataMember]
        public long Id { get; set; }

        [DataMember]
        public string Title { get; set; }

        [DataMember]
        public string ShortDescription { get; set; }

        [DataMember]
        public BrandName Brand { get; set; }

        [DataMember]
        public System.DateTime DatePublished { get; set; }
    }

    [DataContract]
    public class Review
    {
        [DataMember]
        public long ReviewId { get; set; }

        [Required(ErrorMessage = "Please Give Rating")]
        [DataMember]
        public int Rating { get; set; }

        [DataMember]
        public string Comment { get; set; }

        [DataMember]
        public string User { get; set; }

        [Required(ErrorMessage = "Please Specify Product Id")]
```

```
        [DataMember]
        public long ProductId { get; set; }

    }

    public enum BrandName
    {
        [EnumMember]
        APV,
        [EnumMember]
        Airpel,
        [EnumMember]
        PowerTeam,
    }

}
```

Listing the Product:

Due to tight schedule unable to finish UI part