* Course overview
  + OData is a standard that describe best practices for building and consuming rest api
* Introduction
* Course Prerequisites
  + ASP.NET Web API
  + Entity Framework(Code First)
  + REST
* Tooling
  + Visual Studio
  + Postman
  + Browser of choice
* A Few Words on REST
  + REST is an architectural style for designing networked applications
  + REpresentational State Transfer
  + Architectural style constraints
  + REST is not a standard
* Introducing OData
  + “The Best Way to REST”
  + OData(open data protocol)
    - An open protocol to allow the creation and consumption of queryable and interoperable RESTful API’s in a simple and standard way
  + OASIS approved industry standard
  + Best practice for
    - Building
    - Consuming restful apis
  + Consistency & standardization
  + Solves common design issues
    - Naming resources, querying data
  + Uniform way to describe
    - Data
    - Data model
  + Machine-readable metadata
    - Automate interaction
    - Generic client proxies/tools
  + We’re not just building services, we’re implementing a standard
* Demo - Getting Acquainted wi…
  + @odata.context : an annotation
  + Value of annotation is the context url for the payload
    - Describes content of payload
  + Odata Uri
    - Consists of service route + resource path + query options
    - ex) <http://localhost:5810/odata> + /People(1)/VinylRecords + ?$filter=Year eq 2013
  + Download start files at ‘<https://github.com/KevinDockx/ODataV4_Course>’
* Summary
  + REST is an architectural style
  + OData: the open data protocol
    - Best practices for building and consuming RESTful API’s
    - Consistency and standardization
    - Machine-readable metadata
* Introduction
  + Entity data model and conceptual model
  + Selecting entities, entity sets and properties
  + routing
* Entity Data Model and Conce…
  + Entity data model(EDM)
    - Abstract data model
    - Describes data exposed by an OData service
  + Conceptual Model
    - Representation as entities and relationships
    - OData-CSDL implements concepts of EDM
  + Entity: instance of entity type
  + Entity type: name structured type with key
  + Complex Type: name structured type without key
  + Type Definition: name primitive type
  + Navigation Property: relationship from one entity to another
  + Entity Set: named collection of entities
* Demo - Defining an Entity Dat….
  + Add a new project with ‘ASP.NET Web Application’
    - Select an Empty template
    - Add WebAPI in the ‘Add folders & core references’ section
  + For the that project install nuget packages
    - ‘Microsoft.AspNet.OData’
    - ‘EntityFramework’
  + Add reference from the project to the other project in the solution
    - Right click on the projects ‘References’
    - Click ‘Add reference’
    - Select the project
    - Click ‘ok’
  + Build entity data model
    - Add code in the api’s ‘WebApiConfig.cs’ file
    - Comment out the current code to ‘Web API routes’
    - ‘Using System.Web.OData.Extensions’
    - public static void Register(HttpConfiguration config)
    - {
    - config.MapODataServiceRoute(“ODataRoute”, “odata”, GetEdmModel());
    - config.EnsureInitialized();
    - }
    - private static IEdmModel GetEdmModel()
    - {
    - var builder = new ODataConventionModelBuilder();
    - builder.Namespace = “AirVinyl”;
    - builder.ContainerName = “AirVinylContainer”;
    - builder.EntitySet<Person>(“People”);
    - builder.EntitySet<VinylRecord>(“VinylRecords”);
    - return builder.GetEdmModel();
    - }
  + Set the ‘AirVinylAPI’ as the startup project
* Demo - Selecting an EntitySet
  + Add a new controller and have it inherit from OdataController
    - public class PeopleController : ODataController
    - {
    - private AirVinylDbContext \_ctx = new AirVinylDbContext();
    - public IHttpActionResult Get()
    - {
    - return Ok(\_ctx.People);
    - }
    - protected override void Dispose(bool disposing)
    - {
    - \_ctx.Dispose();
    - base.Dispose(disposing);
    - }
    - }