WEB API

1. New generation REST Services for LOB Applications.
2. Based on ASP.NET 4.x that uses w3wp.exe engine
3. Implemented on the code-line like ASP.NET MVC
   1. ApiController class
   2. Action Methods mapped with Http Methods e.g. GET/POST/PUT/DELETE
4. WEB API can be used as Line-of-Business Adapters for Data Communications across the apps.
5. Data is exposed over http EndPoints, these endpoints can be subscribed by third-party clients

VS 2017 with latest updates

Sql Server Express edition minimum

Development using EntityFramework For Data Access Layer and Unity Application Block for Dependency Injection.

1. System.Web.dll and System.Web.Mvc.dll
   1. Runtime assemblies for WEB APIs
2. System.Net.Http and System.Net.Http.WebRequest
   1. Assemblies for Http Communication for WEB API
      1. HttpRequestMessage and HttpResponseMessage classes
3. Microsoft.AspNet.Identity.Core
   1. Identity Assembly
4. EnttyFramework,EntityFramework.SqlServer, Microsoft.AspNet.Identity.EntityFramework
   1. Assemblies for providing Identity Entities
5. DbContext
   1. Manage Db Connections
   2. Manage Model Mapping with Db Tables using DbSet<T> where T is the Model class mapped with Db Table of name T
   3. Manage Db Transactions using SaveChanges() method
6. EF Pseudo code
   1. If ctx in DbContext and T is Employee and DbSet<Employee> is employees then
      1. To Read all from DbSet
         1. Ctx.Employees.ToList()
      2. To Read Single Record based on Primary Key
         1. Ctx.Employees.Find(Primary Key)
      3. To Add Record
         1. Define Employee Instance
         2. Set its property values
         3. Ctx.Employees.Add(Employee Instance);
         4. Ctx.SaveChanges()
      4. To Update Record
         1. Search Employee by Primary Key
         2. Update Searched Property Values
         3. Ctx.SaveChanges()
      5. To Delete Record
         1. Search Employee By P.K.
         2. Ctx.Employees.Remove(Searched Record)
         3. Ctx.SaveChanges()
7. Unity.WebApi Package
   1. The UnityContainer class
      1. This is used to register all objects those are used for Dependency Injection
8. ApiController, base class for WEB API
   1. ModelState
      1. Validate the Posted (HttpPost) / Update Model (HttpPut) received from client
   2. RequestContext
      1. Represent Current HttpRequest
   3. Request
      1. Map Request to response
   4. ActionContext
      1. Map the Http Request Type from Http Request header to corresponding Method of WEB API
         1. Http Type Get 🡪 Mapped to Get() Action method from API
      2. Each Action Method Returns IHttpActionResult that map with HttpResponse
         1. Ok()
         2. NotFound()
         3. BadRequest()
         4. …..
      3. To promote WEB API as Restful Services let each action method define ResponseType for JSON data to be serialized in HttpResponse
         1. System.Web.Http.Description
            1. ResponseTypeAttribuet class to define response from Action Method
   5. ControllerContext
      1. Class that manage the Request Processing.

WEB API Programming

1. Model Validations
   1. ModelState property of ApiController class.
   2. This property map the posted data from body with the Model that is passed to Action method and validate it.
   3. Custom Model Validators
      1. Using ValidationAttribute class
         1. The IsValid(value) method, the ‘value’ is the model property value posted to the Action Method
2. Managing Model-Binders for the Http Methods
   1. Model Binders for Mapping the data from Http Request message (POST/PUT) with CLR Objects
      1. FromUrl and FromBody
         1. FromUrl 🡪 Data read from QueryString
         2. FormBody 🡪 Data read from Body
3. Action Filters
   1. Utility objects those are used to configure an additional behavior for the WEB API Request Processing. They will **Never** change or challenge the request processing instead act as a value-added object during request processing.
      1. The Namespace, System.Web.Http.Filters
         1. Exceptions
         2. Creating Custom Action Filters
   2. Filters are recommended for Global Configuration, optionally they can be used for specific controller or action method.
      1. Request Order 🡪 ActionExecuting
         1. Global
         2. Controller
         3. Action
      2. Response Order 🡪 ActionExecuted
         1. Action
         2. Controller
         3. Global
4. Attribute Based Routing
   1. Search Methods / Features for API
      1. Use QueryString or URL Parameters for infering the Header values and process them
      2. Attribute Based Routing
         1. RoutePrefixAttribute class
         2. RouteAttribute class
      3. The “MapHttpAttributeRoutes” an extension method of HttpConfiguration class to provide attribute based routing.
5. Security
   1. Basic
   2. Token