Implement GraphQL in ASP.NET Core Step by step

Defined GraphQL

GraphQL is an open-source, flexible query language (“QL” stands for query language) for APIs, as well as a runtime for query execution on existing data

Example Source : https://github.com/abolfazlSadeqi/Graphql_AspNetCore_Example

Steps Implement Query(Base and Read Data)

1.Install GraphQL (Core Packacge) nuget package.

2. Install GraphQL.Server.Transports.AspNetCore nuget package

3. Install GraphQL.Server.Ui.Playground(UI to work with GraphQL and do tests) nuget package

4. Install GraphQL.SystemTextJson nuget package

5. Creating GraphQL Specific Objects (Type, Query, Schema)

a) Declaring Type

used to describe the kind of item that may be retrieved through your API.

Per Entity(Example Person) and Add Fields of classes(Example Add FirstName,LastName and ID)

```
public class PersonType : ObjectGraphType<Person>
{
	public PersonType()
	{
		Field(x => x.ID, type: typeof(IdGraphType)).Description("ID");
		Field(x => x.FirstName).Description("FirstName.");
		Field(x => x.LastName).Description("LastName.");
		Field(x => x.Email).Description("Email.");

	}
}
```

b) Declaring Query (fetch the data)

```
public class PersonQuery : ObjectGraphType
{
	public PersonQuery(IUnitOfWork repository)
```

```
        {
                Field<ListGraphType<PersonType>>(
                "Persons",
                resolve: context => repository.Person.GetAll()
                );

        }
}
```

c) Declaring Schema

schema Include :Query(fetch the data) ,Mutations(Add/Edit/Delete),Subscriptions

```
public class Graphql_ExampleSchema : Schema
{
        public Graphql_ExampleSchema(IServiceProvider provider)
        : base(provider)
        {
                Query = provider.GetRequiredService<PersonQuery>();
        }
}
```

Query in graphql

```
query {

         persons {

            lastName

         }

}
```

d) Also, if you want to add a Method with a filter, you can use Query Arguments in class Query(example Class : `PersonType Method: GetById Argument:ID`)

Example:

```
public class PersonQuery : ObjectGraphType
{
        public PersonQuery(IUnitOfWork repository)
        {
        Field<ListGraphType<PersonType>>(
        "Persons",
        resolve: context => repository.Person.GetAll()
);


Field<PersonType>(
        "GetPerson",
        arguments: new QueryArguments(new
        QueryArgument<NonNullGraphType<IdGraphType>> { Name = "ID" }),
                resolve: context =>
                        {
                        var id = context.GetArgument<int>("ID");
                        return repository.Person.GetById(id);
                        }
                );
```

```
        }
}
```

Query in graphql

```
query {
  getPerson(iD: 4) {
                        lastName
                }
}
```

6. configure the Graphql service with AddGraphQL (in Program file or Startup) and Schema

```
services.AddScoped<Graphql_ExampleSchema>();
services.AddScoped<PersonQuery>();


services.AddGraphQL(x =>
{
       x.AddGraphTypes(System.Reflection.Assembly.GetAssembly(typeof(
       Graphql_ExampleSchema))).AddSystemTextJson();
});


app.UseGraphQL<Graphql_ExampleSchema>();
app.UseGraphQLPlayground();
```

Mutation

GraphQL uses mutation to allow the clients or consumers of an API to add, remove or modify data on the servers

Steps Mutation:

1.Add Type

```
public class PersonStatusAddType : InputObjectGraphType
{
       public PersonStatusAddType()
       {
       Name = "personstatusInput";
       Field<NonNullGraphType<StringGraphType>>("value");
       }

}
```

2.Add Mutation

After defining the type, you must define a QueryArgument to receive the data(Example PersonStatus) and insert it in the database

```
public class PersonStatusMutation : ObjectGraphType
{
       public PersonStatusMutation(IUnitOfWork repository)
       {
              Field<PersonStatusType>(
              "createPersonStatus",
              arguments: new QueryArguments(new
              QueryArgument<NonNullGraphType<PersonStatusAddType>>
              {
                   Name = "personstatus"
              }),
              resolve: context =>
              {
              PersonStatus personStatus = new PersonStatus();
       try
       {
              var personstatus = context.GetArgument<PersonStatus>("personstatus");
              repository.PersonStatus.Add(personstatus);
              int _code = repository.Save();
              personStatus.ID = _code;
              personStatus.Value = personstatus.Value;
       }
       catch (Exception ex)
       {
              personStatus.ID = -1;
              personStatus.Value = "";
       }

    return personStatus;

       }
);
}}
```

3)Add Class Mutation To App Schema

```
public class Graphql_ExampleSchema : Schema
{
	public Graphql_ExampleSchema(IServiceProvider provider)
	: base(provider)
	{
		Query = provider.GetRequiredService<PersonQuery>();
		Mutation = provider.GetRequiredService<PersonStatusMutation>();

	}

}
```

Query in graphql

```
mutation($personstatus:personstatusInput!){

 createPersonStatus(personstatus:$personstatus)

 {

    iD,value

 }

}
```

Variable:

```
{

  "personstatus":{ "value":"Test"}

}
```

Test and Call

Steps:

1.Install GraphQL.Client nuget package.

2.Install GraphQL.Client.Serializer.SystemTextJson nuget package

3.Write Test Query

a) Constructor class

```
public readonly GraphQLHttpClient _graphQLHttpClient;
public IntegrationTests(WebApplicationFactory<API.Startup> factory)
{
       if (_graphQLHttpClient == null)
       {
        _graphQLHttpClient = GetGraphQlApiClient( );
       }
}

public GraphQLHttpClient GetGraphQlApiClient()
{
 return new GraphQLHttpClient(Setting.UrlGraphQl, new SystemTextJsonSerializer());

}
```

b) Method Test(First, a GraphQLRequest is created, the query and variable are sent inside GraphQL , and the received result is checked)

```
//Arrange
string _Value = "test3";

var _GraphQLRequest = new GraphQLRequest
{

 Query = @"mutation CreatePS($personstatus:personstatusInput!){
  createPersonStatus(personstatus:$personstatus)
  {
iD,value
  }
}",
 OperationName = "CreatePS",
 Variables = new
 {
      personstatus = new { value = _Value }
 }};

// Assert
var response = await _graphQLHttpClient
  .SendQueryAsync<data_create>(_GraphQLRequest);

// Assert
Assert.NotNull(response.Data);

Assert.True(response.Data.createPersonStatus.iD == 1);
```