.NET App Dev Hands-On Workshop

API Lab 3 –Controllers

This lab creates and configures the controllers for the RESTful service. Before starting this lab, you must have completed API Lab 2b.

Part 1: The BaseCrudController

Step 1: Initial File and Constructor Code

- Create a new folder named `Base` under the `Controllers` folder. Add a new class file named `BaseCrudController.cs` to the folder and update the code to the following:

```
namespace AutoLot.Api.Controllers.Base;

[ApiController]
[Route("api/[controller]")]
public abstract class BaseCrudController<TEntity, TController>(
    IAppLogging<TController> logger, IBaseRepo<TEntity> repo)
  : ControllerBase
     where TEntity : BaseEntity, new()
     where TController : class
{
  protected readonly IBaseRepo<TEntity> MainRepo = repo;
  protected readonly IAppLogging<TController> Logger = logger;
}
```

Step 2: Add the Get Methods

- There are two base methods to get records – `GetAll` and `GetOne`:

```
[HttpGet]
public ActionResult<IEnumerable<TEntity>> GetAll()
{
  return Ok(MainRepo.GetAllIgnoreQueryFilters());
}

[HttpGet("{id}")]
public ActionResult<TEntity> GetOne(int id)
{
  var entity = MainRepo.Find(id);
  if (entity == null)
  {
    return NoContent();
  }
  return Ok(entity);
}
```

Step 3: Add the Update Method

```
[HttpPut("{id}")]
public IActionResult UpdateOne(int id, TEntity entity)
{
  if (id != entity.Id)
  {
    return BadRequest();
  }
  if (!ModelState.IsValid)
  {
    return ValidationProblem(ModelState);
  }
  try
  {
    MainRepo.Update(entity);
  }
  catch (CustomException ex)
  {
    //This shows an example with the custom exception
    //Should handle more gracefully
    return BadRequest(ex);
  }
  catch (Exception ex)
  {
    //Should handle more gracefully
    return BadRequest(ex);
  }
  return Ok(entity);
}
```

Step 4: Add the Add Method

```
[HttpPost]
public ActionResult<TEntity> AddOne(TEntity entity)
{
  if (!ModelState.IsValid)
  {
    return ValidationProblem(ModelState);
  }
  try
  {
    MainRepo.Add(entity);
  }
  catch (Exception ex)
  {
    return BadRequest(ex);
  }
  return CreatedAtAction(nameof(GetOne), new {id = entity.Id}, entity);
}
```

Step 5: Add the Delete Method

```
[HttpDelete("{id}")]
public ActionResult<TEntity> DeleteOne(int id, TEntity entity)
{
  if (id != entity.Id)
  {
    return BadRequest();
  }
  try
  {
    MainRepo.Delete(entity);
  }
  catch (Exception ex)
  {
    //Should handle more gracefully
    return new BadRequestObjectResult(ex.GetBaseException()?.Message);
  }
  return Ok();
}
```

Step 6: Update the GlobalUsings

- Add the following to the `GlobalUsings.cs` file:

```
global using AutoLot.Api.Controllers.Base;
```

Part 2: Add the Entity Specific Controllers

Step 1: The Cars Controller

- Create a new class named `CarsController.cs` in the Controllers directory. Most of the functionality is handled by the base controller. Update the code to the following:

```
namespace AutoLot.Api.Controllers;
public class CarsController(IAppLogging<CarsController> logger, ICarRepo repo)
    : BaseCrudController<Car, CarsController>(logger, repo)
{
  [HttpGet("bymake/{id?}")]
  public ActionResult<IEnumerable<Car>> GetCarsByMake(int? id)
  {
    if (id.HasValue && id.Value>0)
    {
      return Ok(((ICarRepo)MainRepo).GetAllBy(id.Value));
    }
    return Ok(MainRepo.GetAllIgnoreQueryFilters());
  }
}
```

Step 2: The Remaining Controllers

- The controllers all follow the same pattern. Create the remaining controllers as shown here:

```
//DriversController
namespace AutoLot.Api.Controllers;
public class CarDriversController(IAppLogging<CarDriversController> logger, ICarDriverRepo repo)
    : BaseCrudController<CarDriver, CarDriversController>(logger, repo);
//DriversController
namespace AutoLot.Api.Controllers;
public class DriversController(IAppLogging<DriversController> logger, IDriverRepo repo)
    : BaseCrudController<Driver, DriversController>(logger, repo);
//MakesController
namespace AutoLot.Api.Controllers;
public class MakesController(IAppLogging<MakesController> logger, IMakeRepo repo)
    : BaseCrudController<Make, MakesController>(logger, repo);
//RadiosController
namespace AutoLot.Api.Controllers;
public class RadiosController(IAppLogging<RadiosController> logger, IRadioRepo repo)
    : BaseCrudController<Radio, RadiosController>(logger, repo);
```

Summary

This lab created and configured the Controllers for the service.

Next steps

In the next part of this tutorial series, you will add versioning and update Swagger to support versioning.

