

.NET App Dev Hands-On Lab

MVC Lab 9b – Data Services Part 2

This lab swaps out the repos for the data service. Before starting this lab, you must have completed Razor Pages/MVC Lab 9a.

All work in this lab takes place in the `AutoLot.Mvc` project.

Part 1: Update AutoLot.Mvc to use the DAL Data Service

Step 1: Change from Repos to Data Services

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

```
global using AutoLot.Services.DataServices.Dal;  
global using AutoLot.Services.DataServices.Interfaces;  
global using AutoLot.Services.DataServices.Interfaces.Base;
```

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

```
builder.Services.AddScoped<ICarDataService, CarDalDataService>();  
builder.Services.AddScoped<IMakeDataService, MakeDalDataService>();
```

Step 2: Update the BaseCrudController

- Update the primary constructor and field to declare and initialize `IDataServiceBase` instead of the `IBaseRepo`:

```
public abstract class BaseCrudController<TEntity, TController>(  
    IAppLogging<TController> appLogging, IDataServiceBase<TEntity> mainDataService) : Controller  
    where TEntity : BaseEntity, new()  
{  
    protected readonly IAppLogging<TController> AppLoggingInstance = appLogging;  
    protected readonly IDataServiceBase<TEntity> MainDataService = mainDataService;  
    //omitted for brevity  
}
```

- Update the `GetLookupValues` method to be async:

```
protected abstract Task<SelectList> GetLookupValuesAsync();
```

- Anywhere the `BaseRepoInstance` was called, change the method to async and update the code to call the `MainDataService`:

```
protected async Task<TEntity> GetOneEntityAsync(int? id)  
    => id == null ? null : await MainDataService.FindAsync(id.Value);
```

```
[HttpGet]  
public virtual async Task<IActionResult> IndexAsync()  
    => View(await MainDataService.GetAllAsync());
```

```

[HttpGet("{id?}")]
public virtual async Task<IActionResult> DetailsAsync(int? id)
{
    if (!id.HasValue)
    {
        return BadRequest();
    }
    var entity = await GetOneEntityAsync(id);
    if (entity == null)
    {
        return NotFound();
    }
    return View(entity);
}

[HttpGet]
public virtual async Task<IActionResult> CreateAsync()
{
    ViewData["LookupValues"] = await GetLookupValuesAsync();
    return View();
}

[HttpPost]
public virtual async Task<IActionResult> CreateAsync(TEntity entity)
{
    if (ModelState.IsValid)
    {
        var savedEntity = await MainDataService.AddAsync(entity);
        return RedirectToAction(nameof(DetailsAsync).RemoveAsyncSuffix(), new { savedEntity.Id });
    }
    ViewData["LookupValues"] = await GetLookupValuesAsync();
    return View(entity);
}

[HttpGet("{id?}")]
public virtual async Task<IActionResult> EditAsync(int? id)
{
    var entity = await GetOneEntityAsync(id);
    if (entity == null)
    {
        return NotFound();
    }
    ViewData["LookupValues"] = await GetLookupValuesAsync();
    return View(entity);
}

```

```

[HttpPost("{id}")]
public virtual async Task<IActionResult> EditAsync(int id, TEntity entity)
{
    if (id != entity.Id)
    {
        return BadRequest();
    }
    if (ModelState.IsValid)
    {
        await MainDataService.UpdateAsync(entity);
        return RedirectToAction(nameof(DetailsAsync).RemoveAsyncSuffix(), new {entity.Id});
    }
    ViewData["LookupValues"] = await GetLookupValuesAsync();
    return View(entity);
}

[HttpGet("{id?}")]
public virtual async Task<IActionResult> DeleteAsync(int? id)
{
    var entity = await GetOneEntityAsync(id);
    if (entity == null)
    {
        return NotFound();
    }
    return View(entity);
}

[HttpPost("{id}")]
public virtual async Task<IActionResult> DeleteAsync(int id, TEntity entity)
{
    await MainDataService.DeleteAsync(entity);
    return RedirectToAction(nameof(IndexAsync).RemoveAsyncSuffix());
}

```

Step 3: Update the CarsController

- Update the class to the following:

```
public class CarsController(IAppLogging<CarsController> logging,
    ICarDataService dataService, IMakeDataService makeDataService)
    : BaseCrudController<Car, CarsController>(logging, dataService)
{
    protected override async Task<SelectList> GetLookupValuesAsync()
        => new SelectList(await makeDataService.GetAllAsync(), nameof(Make.Id), nameof(Make.Name));
    [HttpGet("{makeId}/{makeName}")]
    public async Task<IActionResult> ByMakeAsync(int makeId, string makeName)
    {
        ViewBag.MakeName = makeName;
        return View(await ((ICarDataService)MainDataService).GetAllByMakeIdAsync(makeId));
    }
    [HttpGet]
    public IActionResult BadEndPoint() => new OkObjectResult(5);
}
```

Step 4: Update the MakesController (in the Admin Area)

- Update the class to the following:

```
[Area("Admin")]
[Route("Admin/[controller]/[action]")]
public class MakesController(
    IAppLogging<MakesController> appLogging, IMakeDataService dataService)
    : BaseCrudController<Make, MakesController>(appLogging, dataService)
{
    protected override Task<SelectList> GetLookupValuesAsync()
        => Task.FromResult<SelectList>(null);
    // GET: Admin/Makes
    [Route("/Admin")]
    [Route("/Admin/[controller]")]
    [Route("/Admin/[controller]/[action]")]
    public override async Task<IActionResult> IndexAsync() => await base.IndexAsync();
}
```

Step 5: Update the MenuViewComponent

- Update the class to the following:

```
public class MenuViewComponent(IMakeDataService dataService) : ViewComponent
{
    public async Task<IViewComponentResult> InvokeAsync()
    {
        var makes = (await dataService.GetAllAsync()).ToList();
        if (!makes.Any())
        {
            return new ContentViewComponentResult("Unable to get the makes");
        }
        return View("MenuView", makes);
    }
}
```

Step 6: Update the Custom TagHelpers

- Update the code that sets the ActionName in each of the custom TagHelpers:

```
//ItemDetailsTagHelper.cs
    ActionName = nameof(CarsController.DetailsAsync).RemoveAsyncSuffix();

//ItemEditTagHelper.cs
    ActionName = nameof(CarsController.EditAsync).RemoveAsyncSuffix();

//ItemListTagHelper.cs
    ActionName = nameof(CarsController.IndexAsync).RemoveAsyncSuffix();

//ItemCreateTagHelper.cs
    ActionName = nameof(CarsController.CreateAsync).RemoveAsyncSuffix();

//ItemDeleteTagHelper.cs
    ActionName = nameof(CarsController.DeleteAsync).RemoveAsyncSuffix();
```

Summary

This lab updated the ASP.NET Core web application using the MVC pattern to use the Data Services and completed the web application.