MVC CRUD Lab

Demo 1: Add Product

Open the **ProductRepository.cs** file and implement the CreateEmpty() method to create a blank product.

```
public Product CreateEmpty() {
  return new Product {
    IsActive = true,
    Name = string.Empty,
    ProductNumber = string.Empty,
    Color = "Black",
    StandardCost = 0,
    ListPrice = 0,
    Size = string.Empty,
    Weight = null,
    SellStartDate = DateTime.Now,
    SellEndDate = null,
    DiscontinuedDate = null,
    };
}
```

Open the **ProductViewModel.cs** file and add a new method

```
#region CreateEmptyProduct Method
public void CreateEmptyProduct()
{
  IsAdding = true;
  // Set Selected Product to an empty product
  SelectedProduct = Repository.CreateEmpty();
}
#endregion
```

Open the **ProductMaintenanceController.cs** file and add the AddProduct() method

```
[HttpGet]
public IActionResult ProductAdd()
{
  // Create view model passing in repository
  ProductViewModel vm = new(_repo, _colorRepo)
  {
    IsDetailVisible = true
  };

  // Call method to create an empty product to add
  vm.CreateEmptyProduct();

  // Call method to load colors
  vm.LoadColors();

  return View("ProductMaintenance", vm);
}
```

Open the **\Views\ProductMaintenance_Search.cshtml** file

Create an Add button BEFORE the **Search** button.

```
<a class="btn btn-secondary"
    asp-action="ProductAdd">
  Add
</a>
```

Try it Out

Run the application and click on the Add button to make sure it displays the detail page with empty data

NOTE: The Save button does NOT yet work.

Demo 2: Save Product

Open the **ProductRepository.cs** file and locate the Insert() method and make it look like the following:

```
public Product Insert(Product entity)
{
  // Add new entity to Products DbSet
  _DbContext.Products.Add(entity);

  // Save changes in database
  _DbContext.SaveChanges();

  return entity;
}
```

Locate the Update() method and make it look like the following:

```
public Product Update(Product entity)
{
  // Update entity in Products DbSet
  _DbContext.Products.Update(entity);

  // Save changes in database
  _DbContext.SaveChanges();

  return entity;
}
```

Open **ProductViewModel.cs** and add a Save() method

```
#region Save Method
public virtual bool Save()
{
  if (IsAdding) {
    // Adding a new product
    Repository.Insert(SelectedProduct);
  }
  else {
    // Editing an existing product
    Repository.Update(SelectedProduct);
  }

  return true;
}
#endregion
```

Open the **ProductMaintenanceController.cs** file and modify the [HttpPost] method.

```
[HttpPost]
public IActionResult ProductMaintenance(ProductViewModel vm)
{
  vm.Repository = _repo;
  vm.ColorRepository = _colorRepo;

  if (ModelState.IsValid) {
    // Save the Product
    vm.Save();

    // Redirect back to product list
    return RedirectToAction("ProductMaintenance");
  }
  else {
    vm.LoadColors();
    vm.IsDetailVisible = true;

    return View(vm);
  }
}
```

Try it Out

Run the app and try editing and adding products

Demo 3: Delete

Open **_List.cshtml** and add a new table header at the end of the table columns

```
<th>Delete</th>
```

Add a new table detail using @Html.ActionLink. There is currently no way to use onclick when using tag helpers.

```
<td>
  @Html.ActionLink("Delete", "ProductDelete",
      new { id = item.ProductID },
      new { onclick = "return confirm('Delete this Product?');",
            @class="btn btn-danger" })
</td>
```

Open the **ProductRepository.cs** file and implement the Delete() method

```
public bool Delete(int id)
{
  // Locate the entity to delete in the Products DbSet
  _DbContext.Products.Remove(_DbContext.Products.Find(id));

  // Save changes in database
  _DbContext.SaveChanges();

  return true;
}
```

Open **ProductViewModel.cs** and add a DeleteProduct() method

```
#region DeleteProduct Method
public bool DeleteProduct(int id)
{
  Repository.Delete(id);

  return true;
}
#endregion
```

Open the **ProductMaintenanceController.cs** file and add a ProductDelete() method

```
[HttpGet]
public IActionResult ProductDelete(int id)
{
  // Create view model passing in repository
  ProductViewModel vm = new(_repo);

  // Call method to delete a product
  vm.DeleteProduct(id);

  return RedirectToAction("ProductMaintenance");
}
```

Try it Out

Run the app, add a new product, then delete it

Demo 4: Total Records Displayed

Let's now display the TotalRows on the page.

Open the **_Search.cshtml** file

Modify the card-footer to look like the following.

```
<div class="card-footer bg-primary text-light">
  <div class="row">
    <div class="col-9">
      <a class="btn btn-secondary"
          asp-action="ProductAdd">
        Add
      </a>
      <button formnovalidate="formnovalidate" class="btn btn-
success">Search</button>
      <a class="btn btn-primary" asp-action="ProductMaintenance">
        Reset
      </a>
    </div>
    <div class="col-3">
      <span class="mt-1 float-end">Total Records:
@Model.TotalRows</span>
    </div>
  </div>
</div>
```

Try it Out

Run the app and view the total rows