Table Sorting Lab

Lab 1: Modify ViewModelBase Class

Open the **\BaseClasses\ViewModelBase.cs** file in the **AdvWorks.Common** project and add a new using statement

```
using System.Text.Json;
```

Add some new properties

```
/// <summary>
/// Get/Set the current field to sort on
/// </summary>
public string SortExpression { get; set; }

/// <summary>
/// Get/Set the previous field that was sorted on
/// </summary>
public string SortExpressionPrevious { get; set; }

/// <summary>
/// Get/Set the current sort direction ("asc" or "desc")
/// </summary>
public string SortDirection { get; set; }

/// <summary>
/// Get/Set the last search
/// </summary>
public string SearchAsJson { get; set; }
```

Modify the Init() method

```
public virtual void Init() {
  SortExpression = string.Empty;
  SortExpressionPrevious = string.Empty;
  SortDirection = "asc";
}
```

Add three new methods

```
#region StoreSearchAsJson Method
public virtual void StoreSearchAsJson<T>(T value) {
  SearchAsJson = JsonSerializer.Serialize(value);
}
#endregion

#region RestoreSearchFromJson Method
public virtual T RestoreSearchFromJson<T>(string value) {
  return JsonSerializer.Deserialize<T>(value);
}
#endregion

#region SetSortProperties Method
protected virtual string SetSortProperties() {
  string ret;

  // See if sort expression is same as previous one
  if (SortExpression == SortExpressionPrevious) {
    ret = (SortExpression + (SortDirection == "asc" ? "_desc" :
"_asc")).ToLower();
    SortDirection = SortDirection == "asc" ? "desc" : "asc";
  }
  else {
    ret = SortExpression.ToLower() + "_asc";
    SortDirection = "asc";
  }

  // Set Previous Expression
  SortExpressionPrevious = SortExpression;

  return ret;
}
#endregion
```

Lab 2: Add a SearchBase Class

Go to the **AdvWorks.Common\BaseClasses** folder

Right mouse-click on this folder and add a new class named **SearchBase**

Add the following code into this file

```
#nullable disable

namespace AdvWorks.Common {
  /// <summary>
  /// All search classes should inherit from this class
  /// </summary>
  public class SearchBase {
    public SearchBase() {
    }

    /// <summary>
    /// Get/Set the complete sort expression such as lastname_asc,
firstname_desc, etc.
    /// </summary>
    public string SortExpression { get; set; }
  }
}
```

Open the **ProductSearch.cs** file in the **AdvWorks.EntityLayer** project and make the ProductSearch class inherit from the SearchBase class.

```
public class ProductSearch : SearchBase {
  // REST OF THE CODE HERE
}
```

Open the **ColorSearch.cs** file in the **AdvWorks.EntityLayer** project and make the ColorSearch class inherit from the SearchBase class.

```
public class ColorSearch : SearchBase {
  // REST OF THE CODE HERE
}
```

Lab 3: Modify Product View Model

Open the **ProductViewModel.cs** file in the **AdvWorks.ViewModelLayer** project and modify the Init() method to look like the following:

```
public override void Init() {
  base.Init();

  SelectedProduct = new();
  SearchEntity = new();
  SearchEntity.SortExpression = "Name";
  SortExpression = "Name";
}
```

Add a new method named RestoreSearchFromJson()

```
#region RestoreSearchFromJson Method
public void RestoreSearchFromJson() {
  if (!string.IsNullOrEmpty(SearchAsJson)) {
    SearchEntity =
base.RestoreSearchFromJson<ProductSearch>(SearchAsJson);
  }
}
#endregion
```

Modify the Search() method to look like the following:

```
public virtual void Search() {
  IsDetailVisible = false;

  // Store Search Data
  base.StoreSearchAsJson<ProductSearch>(SearchEntity);

  // Set Sort Property
  SearchEntity.SortExpression = base.SetSortProperties();

  if (Repository == null) {
    throw new ApplicationException("Must set the Repository
property.");
  }
  else {
    Products = Repository.Search(SearchEntity).ToList();
  }

  if (Products != null) {
    TotalRows = Products.Count;
  }
}
```

Lab 4: Modify the Product Repository

Open the **ProductRepository.cs** file in the **AdvWorks.DataLayer** project and modify the AddOrderByClause() method.

```
public IQueryable<Product> AddOrderByClause(IQueryable<Product>
query, ProductSearch search) {
  // Determine how to sort the data
  switch (search.SortExpression.ToLower()) {
    case "name_asc":
      query = query.OrderBy(x => x.Name);
      break;
    case "name_desc":
      query = query.OrderByDescending(x => x.Name);
      break;
    case "productnumber_asc":
      query = query.OrderBy(x => x.ProductNumber);
      break;
    case "productnumber_desc":
      query = query.OrderByDescending(x => x.ProductNumber);
      break;
    case "standardcost_asc":
      query = query.OrderBy(x => x.StandardCost);
      break;
    case "standardcost_desc":
      query = query.OrderByDescending(x => x.StandardCost);
      break;
    case "listprice_asc":
      query = query.OrderBy(x => x.ListPrice);
      break;
    case "listprice_desc":
      query = query.OrderByDescending(x => x.ListPrice);
      break;
  }

  return query;
}
```

Lab 5: Modify Product Controller

Open the **ProductController.cs** file in the **AdvWorks** project and add a new method to help with sorting

```
[HttpGet]
public IActionResult SortPage(ProductViewModel vm) {
  // Assign Repository to View Model
  vm.Repository = _repo;

  // Restore Search Data
  vm.RestoreSearchFromJson();

  // Call method to sort data
  vm.Search();

  // Reset model state to force new variables to be written
  ModelState.Clear();

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

Lab 6: Modify HTML

Right mouse-click on the **\Views\Shared** folder and create a new view named **_SortHiddenFields.cshtml**. Add the following code into this view

```
@model AdvWorks.Common.ViewModelBase

<input type="hidden" asp-for="SortExpression" />
<input type="hidden" asp-for="SortExpressionPrevious" />
<input type="hidden" asp-for="SortDirection" />
<input type="hidden" asp-for="SearchAsJson" />
```

Open the **\ProductMaintenance_List.cshtml** file and wrap a <form> element around the <table>

```
<form method="get" asp-action="SortPage">
  <partial name="_SortHiddenFields" />
  <table class="table table-bordered table-hover table-striped">
    // REST OF THE HTML HERE
</form>
```

Expand the **wwwroot\js** folder and open the **site.js** file and add a method

```
function sortOnHeaderClick(sortField) {
  $("#SortExpression").val(sortField);

  $("form").submit();
}
```

Go back to the _List.cshtml file and modify the <thead> so it looks like the following:

```
<thead>
  <tr>
    <th>Actions</th>
    <th>
      <a href="#" onclick="sortOnHeaderClick('Name');">
        Product Name
      </a>
    </th>
    <th>
      <a href="#" onclick="sortOnHeaderClick('ProductNumber');">
        Product Number
      </a>
    </th>
    <th class="text-right">
      <a href="#" onclick="sortOnHeaderClick('StandardCost');">
        Cost
      </a>
    </th>
    <th class="text-right">
      <a href="#" onclick="sortOnHeaderClick('ListPrice');">
        Price
      </a>
    </th>
    <th>Delete</th>
  </tr>
</thead>
```

Try it Out

Run the application and navigate to the Product Maintenance page

Click on any of the headers to watch the data sort

Perform a search and see that the sort still works with the search

This is because the Search data is stored in a hidden field