Searching Lab

Perform these labs on your own computer using Visual Studio 2022 to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Create Customer Search Class

Create a base class for all search classes to inherit from.

Right mouse-click on the BaseClasses folder and add a new class named **SearchBase**.

```
namespace AdvWorksAPI.BaseClasses;

public class SearchBase
{
  public SearchBase()
  {
    OrderBy = string.Empty;
  }

  public SearchBase(string orderBy)
  {
    OrderBy = orderBy;
  }

  public string OrderBy { get; set; }
}
```

Create Customer Search Class

Right-mouse click on the AdvWorksAPI folder and add a new folder called **SearchClasses**.

Right-mouse click on the SearchClasses folder and add a new class named **CustomerSearch**. Replace the entire contents of this new file with the following code.

```
using AdvWorksAPI.BaseClasses;

namespace AdvWorksAPI.SearchClasses;

public class CustomerSearch : SearchBase
{
  public CustomerSearch()
  {
    OrderBy = "LastName";
    FirstName = string.Empty;
    LastName = string.Empty;
    Title = string.Empty;
  }

  public string? FirstName { get; set; }
  public string? LastName { get; set; }
  public string? Title { get; set; }
}
```

Lab 2: Add Search Methods to Customer Repository Class

Open the **CustomerRepository.cs** file and add some new Search methods.

```
#region Search Methods
public List<Customer> Search(CustomerSearch search)
{
  IQueryable<Customer> query = _DbContext.Customers;

  // Add WHERE clause(s)
  query = AddWhereClause(query, search);

  // Add ORDER BY clause(s)
  query = AddOrderByClause(query, search);

  return query.ToList();
}

protected virtual IQueryable<Customer>
AddWhereClause(IQueryable<Customer> query,
CustomerSearch search)
{
  // Perform Searching
  if (!string.IsNullOrEmpty(search.FirstName)) {
    query = query.Where(row =>
row.FirstName.Contains(search.FirstName));
  }
  if (!string.IsNullOrEmpty(search.LastName)) {
    query = query.Where(row =>
row.LastName.Contains(search.LastName));
  }
  if (!string.IsNullOrEmpty(search.Title)) {
  // NOTE: Do NOT simplify this expression, or the query
will not work.
#pragma warning disable IDE0075 // Simplify conditional
expression
    // Title allows nulls, so you have to check for a
null value
    query = query.Where(row =>
string.IsNullOrEmpty(row.Title) ? true :
row.Title.StartsWith(search.Title));
#pragma warning restore IDE0075 // Simplify conditional
expression
  }

  return query;
}

protected virtual IQueryable<Customer>
AddOrderByClause(IQueryable<Customer> query,
CustomerSearch search)
```

```
{
  switch (search.OrderBy.ToLower()) {
    case "":
    case "lastname":
      query = query.OrderBy(row => row.LastName);
      break;
    case "firstname":
      query = query.OrderBy(row => row.FirstName);
      break;
    case "title":
      query = query.OrderBy(row => row.Title);
      break;
  }

  return query;
}
#endregion
```

Lab 3: Add Search Methods to IRepository Interface

Open the **IRepository.cs** file and **replace** the **entire** contents of the file with the following code.

```
namespace AdvWorksAPI.Interfaces;

public interface IRepository<TEntity, TSearch>
{
  List<TEntity> Get();
  TEntity? Get(int id);
  List<TEntity> Search(TSearch search);

  TEntity Insert(TEntity entity);
  TEntity Update(TEntity entity);
  TEntity SetValues(TEntity current, TEntity changes);
  bool Delete(int id);
}
```

Lab 4: Update all Usages of IRepository Interface

You have now just broken everywhere that you were using IRepository<Customer>.

Open the **CustomerRepository.cs** file and modify the declaration

```
public class CustomerRepository : IRepository<Customer,
CustomerSearch>
```

Open the **ServiceExtensions.cs** file and modify the AddRepositoryClasses()

```
public static void AddRepositoryClasses(this
IServiceCollection services)
{
  // Add Repository Classes
  services.AddScoped<IRepository<Customer,
CustomerSearch>, CustomerRepository>();
}
```

Open the **CustomerController.cs** file and modify the readonly field

```
private readonly IRepository<Customer, CustomerSearch>
_Repo;
```

Modify the constructor

```
public
CustomerController(IOptionsMonitor<AdvWorksAPIDefaults>
settings, IRepository<Customer, CustomerSearch> repo,
ILogger<CustomerController> logger, IConfiguration
config) : base(logger)
```

Compile the code to ensure you fixed everything.

Lab 5: Retrieve Data Using the Search Method

Let's add a search method for data based on items filled into the Customer Search class.

Open the **CustomerController.cs** file and remove all the previous "Search" methods you added.

- SearchByTitle
- SearchByFirstLast

Add a new method that looks like the following:

```
#region Search Method
[HttpGet()]
[Route("Search")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServe
rError)]
public ActionResult<IEnumerable<Customer>>
Search([FromQuery()] CustomerSearch search)
{
  ActionResult<IEnumerable<Customer>> ret;
  List<Customer> list;

  InfoMessage = "Can't find products matching the
criteria passed in.";

  try {
    // Search for Data
    list = _Repo.Search(search);

    if (list != null && list.Count > 0) {
      return StatusCode(StatusCodes.Status200OK, list);
    }
    else {
      return StatusCode(StatusCodes.Status404NotFound,
InfoMessage);
    }
  }
  catch (Exception ex) {
    InfoMessage = _Settings.InfoMessageDefault
        .Replace("{Verb}",
"SEARCH").Replace("{ClassName}", "Customer");

    ErrorLogMessage = "Error in
CustomerController.Search()";

    ret = HandleException<IEnumerable<Customer>>(ex);
  }

  return ret;
}
#endregion
```

Note that you must use the **[FromQuery()]** attribute to map the values from the values coming in on the URL line to the appropriate properties in the CustomerSearch class.

Try it Out

Run the application and click on the **GET /api/Customer/Search** button.

```
FirstName = A
LastName = B
Title = Ms
OrderBy = LastName
```

Click on the **Execute** button and you should see several records returned.