Asynchronous 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: Add GetAsync() Method

Open the **IRepository.cs** file and make the interface look like the following:

```
public interface IRepository<TEntity, TSearch>
{
  // Asynchronous Methods
  Task<List<TEntity>> GetAsync();


  // Synchronous Methods
  // REST OF THE CODE HERE
}
```

Create Async Repository

Open the **CustomerRepository.cs** file and add the **partial** keyword to the class definition.

```
/// <summary>
/// Synchronous access to Customer data
/// </summary>
public partial class CustomerRepository :
IRepository<Customer, CustomerSearch>
{
  // REST OF THE CODE HERE
}
```

Right mouse-click on the RepositoryLayer folder and add a new class named **CustomerAsyncRepository.cs**.

Make this class look like the following:

```
using AdvWorksAPI.EntityLayer;
using Microsoft.EntityFrameworkCore;

namespace AdvWorksAPI.RepositoryLayer;

/// <summary>
/// Asynchronous access to Customer data asynchronously
/// </summary>
public partial class CustomerRepository
{
  #region GetAsync Method
  /// <summary>
  /// Get all Customer objects asynchronously
  /// </summary>
  /// <returns>A list of Customer objects</returns>
  public async Task<List<Customer>> GetAsync()
  {
    return await _DbContext.Customers.OrderBy(row =>
row.LastName).ToListAsync();
  }
  #endregion
}
```

Create Async Controller

Right mouse-click on the Controllers folder and add a new class named **CustomerAsyncController**.

Replace the entire contents of this new file with the following code.

```
using AdvWorksAPI.BaseClasses;
using AdvWorksAPI.EntityLayer;
using AdvWorksAPI.Interfaces;
using AdvWorksAPI.SearchClasses;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;

namespace AdvWorksAPI.Controllers;

/// <summary>
/// Asynchronous Action Methods
/// </summary>
[Route("api/[controller]")]
[ApiController]
public partial class CustomerAsyncController :
ControllerBaseAPI
{
  private readonly IRepository<Customer, CustomerSearch>
_Repo;
  private readonly AdvWorksAPIDefaults _Settings;

  public CustomerAsyncController(IRepository<Customer,
CustomerSearch> repo, ILogger<CustomerController>
logger, IOptionsMonitor<AdvWorksAPIDefaults> settings) :
base(logger)
  {
    _Repo = repo;
    _Settings = settings.CurrentValue;
  }

  #region GetAsync Method
  [HttpGet]
  [ProducesResponseType(StatusCodes.Status200OK)]
  [ProducesResponseType(StatusCodes.Status404NotFound)]

[ProducesResponseType(StatusCodes.Status500InternalServe
rError)]
  public async Task<ActionResult<IEnumerable<Customer>>>
GetAsync()
  {
    ActionResult<IEnumerable<Customer>> ret;
    List<Customer> list;
    InfoMessage = "No Customers are available.";

    try {
      // Get all data
      var task = await _Repo.GetAsync();
      // Convert data to a List<T>
      list = task.ToList();

      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}",
"GETAsync").Replace("{ClassName}", "Customer");

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

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

    return ret;
  }
  #endregion
}
```

Try it Out

Run the application and click on the **GET /api/CustomerAsync** button to see all the customer data returned.

Lab 2: Add GetAsync(id) Method

Open the **IRepository.cs** file and add a GetAsync(id) method.

```
Task<TEntity?> GetAsync(int id);
```

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

```
#region GetAsync(id) Method
public async Task<Customer?> GetAsync(int id)
{
  return await _DbContext.Customers.Where(row =>
row.CustomerID == id).FirstOrDefaultAsync();
}
#endregion
```

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

```
#region GetAsync(id) Method
[HttpGet("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public async Task<ActionResult<Customer>> GetAsync(int
id)
{
  ActionResult<Customer> ret;
  Customer? entity;

  entity = await _Repo.GetAsync(id);

  if (entity != null) {
    // Found data, return '200 OK'
    ret = StatusCode(StatusCodes.Status200OK, entity);
  }
  else {
    // Did not find data, return '404 Not Found'
    ret = StatusCode(StatusCodes.Status404NotFound,
$"Can't find Customer with a Customer Id of '{id}'.");
  }

  return ret;
}
#endregion
```

Try it Out

Run the application and click on the **GET /api/CustomerAsync/{id}** button

Enter 235 for the Customer ID

Lab 3: Add SearchAsync() Method

Open the **IRepository.cs** file and add a SearchAsync() method.

```
Task<List<TEntity>> SearchAsync(TSearch search);
```

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

```
#region SearchAsync Methods
public async Task<List<Customer>>
SearchAsync(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 await query.ToListAsync();
}
#endregion
```

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

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

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

  try {
    // Search for Data
    var task = await _Repo.SearchAsync(search);
    list = task.ToList();

    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}",
"SEARCHAsync").Replace("{ClassName}", "Customer");

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

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

  return ret;
}
#endregion
```

Try it Out

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

Enter the following values

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

Click the **Execute** button