

# Base Router 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: Update Base Router Class

Open the **RouterBase.cs** file and add three more properties

```
public string InfoMessage { get; set; }  
public string ErrorLogMessage { get; set; }  
protected readonly ILogger _Logger;
```

Modify the constructor to look like the following:

```
public RouterBase(ILogger logger)  
{  
    UrlFragment = string.Empty;  
    TagName = string.Empty;  
    InfoMessage = string.Empty;  
    ErrorLogMessage = string.Empty;  
    _Logger = logger;  
}
```

## Fix up all Router Classes

Open the **CustomerRouter.cs** file

Remove the **readonly** field **\_Logger**

Modify the constructor

```
public CustomerRouter(IRepository<Customer> _repo,
    ILogger<CustomerRouter> logger) : base(logger)
{
    UrlFragment = "api/Customer";
    TagName = "Customer";
    _Repo = _repo;
}
```

Open the **ErrorRouter.cs** file and add a constructor

```
public ErrorRouter(ILogger<ErrorContext> logger) :
    base(logger)
{
}
```

Open the **LogTestRouter.cs** file

Remove the **readonly** field **\_Logger**

Modify the constructor

```
public LogTestRouter(ILogger<LogTestRouter> logger) :
    base(logger)
{
    UrlFragment = "api/LogTest";
    TagName = "LogTest";
}
```

Open the **SettingsRouter.cs** file and modify the constructor

```
public SettingsRouter(ILogger<SettingsRouter> logger) :
    base(logger)
{
    UrlFragment = "api/Settings";
    TagName = "Settings";
}
```

## Compile

Compile the application and make sure everything still works as it did before.

## Demo 2: Add Exception Handling Method

Open the **RouterBase.cs** file and add a new method to log an exception to the log and return a status code of 500 with a generic message appropriate for the caller of this API.

```
/// <summary>
/// Call this method to return a '500 Internal Server
Error' and log an exception.
/// </summary>
/// <param name="ex">An Exception object</param>
/// <param name="infoMsg">The info message to display to
the user</param>
/// <param name="errorMsg">The error message to
log</param>
/// <returns>A Status Code of 500</returns>
protected IResult HandleException(Exception ex, string
infoMsg, string errorMsg)
{
    // Set properties from parameters passed in
    InfoMessage = infoMsg;
    ErrorLogMessage = errorMsg;

    return HandleException(ex);
}

/// <summary>
/// Call this method to return a '500 Internal Server
Error' and log an exception.
/// Prior to calling this method...
///     Fill in the InfoMessage property with the value
to display to the caller.
///     Fill in the ErrorLogMessage property with the
value to place into the log file.
/// </summary>
/// <param name="ex">An Exception object</param>
/// <returns>A Status Code of 500</returns>
protected IResult HandleException(Exception ex)
{
    IResult ret;

    // Create status code with generic message
    ret = Results.Problem(InfoMessage);

    // Add Message, Source, and Stack Trace
    ErrorLogMessage += $"{Environment.NewLine}Message:
{ex.Message}";
    ErrorLogMessage += $"{Environment.NewLine}Source:
{ex.Source}";
    ErrorLogMessage += $"{Environment.NewLine}Stack Trace:
{ex.StackTrace}";

    // Log the exception
```

```
_Logger.LogError(ex, "{ErrorLogMessage}",
ErrorLogMessage);

return ret;
}
```

Open the **CustomerRouter.cs** file and modify the **Get()** method.

```
protected virtual IActionResult Get()
{
    IActionResult ret;
    List<Customer> list;
    InfoMessage = "No Customers Found.";

    try {
        // Intentionally Cause an Exception
        throw new ApplicationException("ERROR!");

        list = _Repo.Get();

        //list.Clear();
        if (list == null || list.Count == 0) {
            ret = Results.NotFound(InfoMessage);
        }
        else {
            ret = Results.Ok(list);
        }
    }
    catch (Exception ex) {
        InfoMessage = "Error in Customer API. Please Contact
the System Administrator.";

        ErrorLogMessage = "Error in CustomerRouter.Get()";

        ret = HandleException(ex);
    }

    return ret;
}
```

## Try it Out

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

Make sure still get the same exception message as before.

## Demo 3: Serialize an Object

Open the **RouterBase.cs** file and add a new property.

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

Modify the constructor

```
public RouterBase(ILogger logger)
{
    UrlFragment = string.Empty;
    TagName = string.Empty;
    InfoMessage = string.Empty;
    ErrorMessage = string.Empty;
    EntityAsJson = string.Empty;
    _Logger = logger;
}
```

Add a method named `SerializeEntity()`.

```
/// <summary>
/// Serialize an object into a JSON string
/// </summary>
/// <typeparam name="T">The type to
serialize</typeparam>
/// <param name="entity">An instance of the type</param>
/// <returns>A JSON string</returns>
protected string SerializeEntity<T>(T entity)
{
    EntityAsJson = "Nothing serialized";

    try {
        // Attempt to serialize entity
        EntityAsJson = JsonSerializer.Serialize(entity);
    }
    catch {
        // Ignore the error
    }

    return EntityAsJson;
}
```

Open the **LogTestRouter.cs** file and locate the LogCustomer() method and change the line of code shown in **bold** below.

```
protected virtual IActionResult LogCustomer()
{
    // Log a Customer Object
    Customer entity = new()
    {
        CustomerID = 999,
        FirstName = "Bruce",
        LastName = "Jones",
        Title = "Mr.",
        CompanyName = "Beach Computer Consulting",
        EmailAddress =
        "Jones.Bruce@beachcomputerconsulting.com",
        Phone = "(714) 555-5555",
        ModifiedDate = DateTime.Now
    };

    string json = base.SerializeEntity<Customer>(entity);

    _Logger.LogInformation("Customer = {json}", json);

    return Results.Ok("Check Console Window");
}
```

## Try it Out

Delete all Log\*.txt files in the **Logs** folder.

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

Check the log file to see the customer is still serialized.