# Project

* Site-04-Ng-login-reg-nav-services-templateforms-Api-MiddlewareException
  + Copied from: Site-Site-05-Ng-routing-toast-route-guard-shared-module-misc-components
* For the “MySocialConnect-API”
  + dotnet restore
  + dotnet build
  + Go to project : MSC.WebApi
    - dotnet build : to build
    - dotnet run : to run the api

# New Resources

|  |  |
| --- | --- |
| MSC.Core | MSC.WebApi |
|  |  |
|  |  |
|  |  |

# Resources updated

|  |  |
| --- | --- |
| MSC.Core | MSC.WebApi |
| Dtos/ExceptionResponseDto.cs | SampleController.cs |

# Clear Database

If you want to clear database then drop it and recreate it

dotnet ef database drop

dotnet ef database update

# Server Side – WebApi

Server side - Web Api error handling has already been implemented via [Site 04](003%20Project4%20-WebApi-ExceptionHandlingMiddleware.docx).

# Dtos/ExceptionResponseDto.cs

Adding constructor. This was initially created under [Site 04](file:///C:\00Learning\Learning-Examples\Api-Angular-React-2024\2024-Angular-Api-EF-MySocialConnect\Documents\003%20Project4%20-WebApi-ExceptionHandlingMiddleware.docx)

namespace MSC.Core.Dtos;

public class ExceptionResponseDto

{

    public ExceptionResponseDto(int statusCode, string message = null, string details = null)

    {

        StatusCode = statusCode;

        Message = message;

        Details = details;

    }

    public int StatusCode { get; set; }

    public string Message { get; set; }

    public string Details { get; set; }

}

# SampleController

We’ll use this to do error handling

## SampleLoginDto

public class SampleLoginDto

{

    [Required(ErrorMessage = "UserName is empty")]

    [MinLength(5, ErrorMessage = "UserName length must be atleast 5 chars")]

    public string UserName { get; set; }

    [Required(ErrorMessage = "Password is empty")]

    [StringLength(16, MinimumLength = 10)]

    [RegularExpression("^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?\\d)(?=.\*?[@#$&()<>]).+$", ErrorMessage = "Password must have an upper case, a lower case, a number and one special character from the set @#$&()<>")]

    public string Password { get; set; }

}

## Controller and end points

using System.ComponentModel.DataAnnotations;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Options;

using MSC.Core.BusinessLogic;

using MSC.Core.DB.Data;

using MSC.Core.DB.Entities;

using MSC.Core.Dtos.Helper;

using MSC.Core.Extensions;

namespace MSC.WebApi.Controller;

// /api/sample

public class SampleController : BaseApiController

{

    private readonly IOptions<EnvConfig> \_envConfig;

    private readonly IConfiguration \_config;

    private readonly IUserBusinessLogic \_userBal;

    private readonly DataContext \_context;

    public SampleController(IOptions<EnvConfig> envConfig, IConfiguration config,

    IUserBusinessLogic userBal, DataContext context)

    {

        \_envConfig = envConfig;

        \_config = config;

        \_userBal = userBal;

        \_context = context;

    }

    [HttpPost("sample1")]

    public ActionResult<bool> Sample1()

    {

        //IConfiguration

        var config = \_config;

        //using Ioptions

        var howdy = \_envConfig;

        return Ok(true);

    }

    [HttpPost("sample2")]

    public ActionResult<bool> Sample2()

    {

        //IConfiguration with extension

        var config = \_config;

        var tokenKey = \_config.GetTokenKey();

        //using Ioptions

        var howdy = \_envConfig;

        return Ok(true);

    }

    [Authorize]

    [HttpGet("auth")]

    public ActionResult<string> GetSecret()

    {

        return "secret text";

    }

    [HttpGet("usernotfound")]

    public ActionResult<AppUser> GetUserNotFound()

    {

        var user = \_context.Users.Find(-1);

        if(user == null)

            return NotFound();

        return user;

    }

    [HttpGet("servererror")]

    public ActionResult<string> GetServerError()

    {

        var user = \_context.Users.Find(-1);

        var userReturn = user.ToString(); //will throw null reference exception

        return userReturn;

    }

    [HttpGet("badrequest")]

    public ActionResult<string> GetBadRequest()

    {

        return BadRequest("This was not a good request");

    }

    [HttpPost("samplelogin")]

    public ActionResult<bool> SampleLogin([FromBody] SampleLoginDto sampleLoginDto)

    {

        if(sampleLoginDto == null)

            return BadRequest("Sample login dto empty");

        return true;

    }

}

public class SampleLoginDto

{

    [Required(ErrorMessage = "UserName is empty")]

    [MinLength(5, ErrorMessage = "UserName length must be atleast 5 chars")]

    public string UserName { get; set; }

    [Required(ErrorMessage = "Password is empty")]

    [StringLength(16, MinimumLength = 10)]

    [RegularExpression("^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?\\d)(?=.\*?[@#$&()<>]).+$", ErrorMessage = "Password must have an upper case, a lower case, a number and one special character from the set @#$&()<>")]

    public string Password { get; set; }

}

# Postman

Postman items available under folder Site-06