# Project

* Site-09-Api-Ng-templateform-candeactivate-spinner-intercepter-caching
  + Copied from: Site-08-Api-Ng-interceptor-membercards-photogallery
* 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 |
| /Dtos/MemberUpdateDto.cs |  |
| /Dtos/UserClaimDto.cs |  |
| /Extensions/ClaimsPrincipalExtensions.cs |  |
|  |  |
|  |  |
|  |  |

# Resources updated

|  |  |
| --- | --- |
| MSC.Core | MSC.WebApi |
| /Mappers/Auto<apperProfiles.cs | UserController.cs |
| /BusinessLogic/UserBusinessLogic |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Clear Database

Clear the users

dotnet ef database drop

dotnet ef database update

# /MSC.Core/Dtos

## MemberUpdateDto

namespace MSC.Core.Dtos;

public class MemberUpdateDto

{

    public string Introduction { get; set; }

    public string LookingFor { get; set; }

    public string Interests { get; set; }

    public string City { get; set; }

    public string Country { get; set; }

}

## UserClaimDto.cs

using System;

namespace MSC.Core;

public class UserClaimGetDto

{

    public int Id {get; set; }

    public string UserName { get; set; }

    public Guid Guid { get; set; }

    public string DisplayName { get; set; }

    public bool HasUserName => !string.IsNullOrWhiteSpace(UserName);

    public bool HasGuid => Guid != Guid.Empty;

    public bool HasId => Id > 0;

}

# /MSC.Core/Mappers

## AutoMapperProfile.cs

Create auto mapper for mapping MemberUpdateDto to AppUser

    public AutoMapperProfiles()

    {

        Map\_AppUser\_To\_UserDto();

        Map\_Photo\_To\_PhotoDto();

        Map\_AppUser\_To\_LoggedInUserDto();

        Map\_UserRegisterDto\_To\_AppUser();

        Map\_MemberUpdateDto\_To\_AppUser();

    }

    private void Map\_MemberUpdateDto\_To\_AppUser()

    {

        CreateMap<MemberUpdateDto, AppUser>();

    }

# /MSC.Core/Extensions

## ClaimsPrincipalExtensions.cs

using System;

using System.Security.Claims;

using System.Threading.Tasks;

namespace MSC.Core.Extensions;

//check /Services/TokenService for the token that is being written

//https://www.jerriepelser.com/blog/useful-claimsprincipal-extension-methods/

public static class ClaimsPrincipalExtensions

{

    public static UserClaimGetDto GetUserClaims(this ClaimsPrincipal principal)

    {

        if(principal == null) return null;

        var claimsDto = new UserClaimGetDto()

        {

            UserName = principal.GetUserName(),

            Id = principal.GetId(),

            Guid = principal.GetGuid(),

            DisplayName = principal.GetDisplayName()

        };

        return claimsDto;

    }

    public static string GetUserName(this ClaimsPrincipal principal)

    {

        if (principal == null) return string.Empty;

        //return principal.FindFirstValue(ClaimTypes.NameIdentifier);

        var userName = principal.FindFirst(ClaimTypes.Name)?.Value;

        return userName;

    }

    public static int GetId(this ClaimsPrincipal principal)

    {

        if (principal == null) return 0;

        //return principal.FindFirstValue(ClaimTypes.NameIdentifier);

        var id = principal.FindFirst(ClaimTypes.NameIdentifier)?.Value;

        var getId = 0;

        if(!string.IsNullOrWhiteSpace(id)){

            int.TryParse(id, out getId);

        }

        return getId;

    }

    public static Guid GetGuid(this ClaimsPrincipal principal)

    {

        var getGuid = Guid.Empty;

        if (principal == null) return getGuid;

        var guid = principal.FindFirst("Guid")?.Value;

        if (string.IsNullOrWhiteSpace(guid)) return getGuid;

        try

        {

            getGuid = new Guid(guid);

        }

        catch { }

        return getGuid;

    }

    public static string GetDisplayName(this ClaimsPrincipal principal)

    {

        if (principal == null) return string.Empty;

        var displayName = principal.FindFirst("DisplayName")?.Value;

        return displayName;

    }

}

# /MSC.Core/BusinessLogic

## IUSerBusinessLogic.cs

    Task<bool> UpdateUserAsync(MemberUpdateDto memberUpdateDto, UserClaimGetDto claims);

## UserBusinessLogic.cs

    public async Task<bool> UpdateUserAsync(MemberUpdateDto memberUpdateDto, UserClaimGetDto claims)

    {

        var user = await \_userRepo.GetUserRawAsync(claims.UserName);

        if(user == null || user.Guid != claims.Guid)

            return false;

        //we have the mapped. current properties will be kept as is while other will be updated

        var updates = \_mapper.Map(memberUpdateDto, user);

        //issue update

        \_userRepo.Update(updates);

        //save updates

        if(await \_userRepo.SaveAllAsync())

            return true;

        return false;

    }

# /MSC.Web/Controllers

## UsersController.cs

    [HttpPut]

    public async Task<ActionResult> UpdateUser([FromBody] MemberUpdateDto member)

    {

        //get the user claims

        var userClaims = User.GetUserClaims();

        if(userClaims == null || (!userClaims.HasGuid || !userClaims.HasUserName))

            return BadRequest("User Issue");

        var isUpdate = await \_userBusinessLogic.UpdateUserAsync(member, userClaims);

        if(!isUpdate)

            return BadRequest("User not updated");

        return NoContent(); //204

    }