Table of Contents

[Current Resources with Changes 2](#_Toc114684502)

[New Resources 2](#_Toc114684503)

[/Core/Dto 2](#_Toc114684504)

[UserTokenDto 2](#_Toc114684505)

[UserUpdateDto 2](#_Toc114684506)

[UserClaimGetDto 2](#_Toc114684507)

[/Core/Dto/AutoMapper/AutoMapperProfiles.cs 2](#_Toc114684508)

[/Core/Services 3](#_Toc114684509)

[TokenService 3](#_Toc114684510)

[/Controller/BaseApiController 3](#_Toc114684511)

[/Core/BusinessLogic 3](#_Toc114684512)

[IUsersBusinessLogic 3](#_Toc114684513)

[UsersBusinessLogic 3](#_Toc114684514)

[LoginAsync and RegisterAsync 3](#_Toc114684515)

[UpdateUserAsync 4](#_Toc114684516)

[Controllers 4](#_Toc114684517)

[UsersController 4](#_Toc114684518)

# Current Resources with Changes

1. /Core/Dto/UserTokenDto
2. /Core/BusinessLogic/UsersBusinessLogic
3. /Core/services/TokenService
4. /Controllers/BaseApiController
5. /Controllers/UsersController

# New Resources

1. /Core/Dto/UserClaimGetDto
2. /Core/Dto/UserUpdateDto

# /Core/Dto

## UserTokenDto

Add a new propertu to it

public Guid GuId { get; set; }

## UserUpdateDto

This dto is used to update the user information

namespace MSC.Api.Core.Dto;

public class UserUpdateDto

{

    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; }

}

## UserClaimGetDto

using System;

namespace MSC.Api.Core.Dto;

public class UserClaimGetDto

{

    public string UserName { get; set; }

    public Guid Guid { get; set; }

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

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

}

# /Core/Dto/AutoMapper/AutoMapperProfiles.cs

Create a new function to map UserUpdateDto to AppUser

    private void Map\_UserUpdate\_To\_AppUser(){

        CreateMap<UserUpdateDto, AppUser>();

    }

And call it inside the AutoMapperProfiles as well

    public AutoMapperProfiles()

    {

        Map\_AppUser\_To\_UserDto();

        Map\_Photo\_To\_PhotoDto();

        Map\_UserUpdate\_To\_AppUser();

    }

# /Core/Services

## TokenService

Add the guid to the claims as sid 😊

        var claims = new List<Claim>

        {

            new Claim(JwtRegisteredClaimNames.NameId, user.UserName),

            new Claim("Guid", user.GuId.ToString()),

            new Claim("DisplayName", user.DisplayName),

        };

Run the api again and on the Angular side logout and login

Pick up the token from the console and go to <https://jwt.io/>

Put your token and then look at it under payload: data

# /Controller/BaseApiController

Create a helper method to get the user claims

    public UserClaimGetDto GetLoggedInCalims(){

        var username = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;

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

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

        /\*

        can loop through as well

        var claims = User.Claims;

        foreach(var c in claims){

            var x = c.Value;

            var y = c.Type;

            var z = "";

        }

        \*/

        var claimsDto = new UserClaimGetDto(){

            UserName = username,

            Guid = string.IsNullOrWhiteSpace(guid) ? Guid.Empty : new Guid(guid),

            DisplayName = displayName

        };

        return claimsDto;

    }

# /Core/BusinessLogic

## IUsersBusinessLogic

Add a definition for the updating the user

Task<bool> UpdateUserAsync(UserUpdateDto userUpdateDto, UserClaimGetDto claims);

## UsersBusinessLogic

### LoginAsync and RegisterAsync

Update both login and register to return the token via the model as well

var userToken = new UserTokenDto

        {

            UserName = user.UserName,

            GuId = user.GuId,

            Token = \_tokenService.CreateToken(user)

        };

### UpdateUserAsync

    public async Task<bool> UpdateUserAsync(UserUpdateDto userUpdateDto, UserClaimGetDto claims)

    {

        var user = await \_usersRepo.GetAppUserAsync(claims.UserName);

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

            return false;

        //data from the userUpdateDto will be moved to user while the rest of the properties will be kept as is

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

        //issue update but it will not save

        \_usersRepo.Update(updates);

        //save update

        if(await \_usersRepo.SaveAllAsync())

            return true;

        return false;

    }

# Controllers

## UsersController

Create a new action which will use default route since only one PUT method is available here

    [HttpPut]

    public async Task<ActionResult> UpdateUser([FromBody] UserUpdateDto userUpdateDto)

    {

        //get the clams

        var userClaims = base.GetLoggedInCalims();

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

        {

            return BadRequest("User issue");

        }

        var isUpdate = await \_usersBl.UpdateUserAsync(userUpdateDto, userClaims);

        if (!isUpdate)

        {

            return BadRequest("User not updated");

        }

        return NoContent();

    }