DAO > REPO > SERVICE > CONTROLLER

Sau khi setup thành công program.cs ta tạo controller

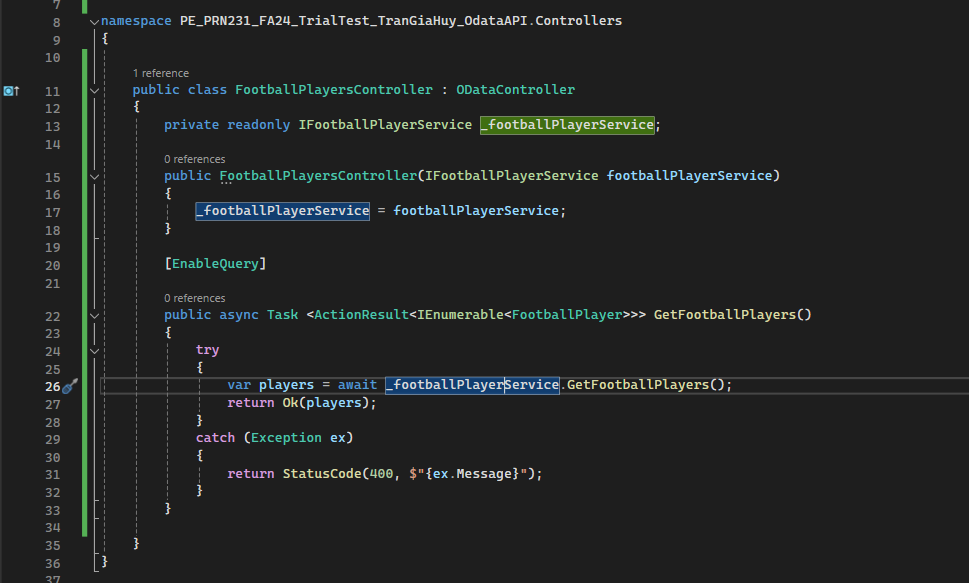
TẠO EMPTY CONTROLLER

Nhớ đặt tên controller có s+controller

Nhớ enable query

Nhớ independcy injection

Hoàn thành get all



Bước 1 dependency injection

private readonly ISoccerPlayerService \_soccerPlayerService;

public SoccerPlayersController(ISoccerPlayerService soccerPlayerService)

{

\_soccerPlayerService = soccerPlayerService;

}

Bước 2 copy code này get all

[EnableQuery]

public async Task <ActionResult<IEnumerable<FootballPlayer>>> GetFootballPlayers()

{

try

{

var players = await \_footballPlayerService.GetFootballPlayers();

return Ok(players);

}

catch (Exception ex)

{

return StatusCode(400, $"{ex.Message}");

}

}

getbyId

[HttpGet("/api/FootballPlayers/{id}")]

public async Task<ActionResult<FootballPlayer>> GetFootBallPlayer(string id)

{

try

{

var footballplayer = await \_footballPlayerService.GetFootballPlayer(id);

return Ok(footballplayer);

}

catch (Exception ex)

{

return StatusCode(400, $"{ex.Message}");

}

}

Add

[HttpPost("/api/FootballPlayers")]

public async Task<ActionResult<FootballPlayer>> AddPlayer([FromBody] FootballPlayer player)

{

if (player == null)

{

return BadRequest("Player data cannot be null");

}

try

{

var newPlayer = await \_footballPlayerService.AddFootballPlayer(player);

return Ok(newPlayer);

}

catch (Exception ex)

{

return BadRequest(new { message = ex.Message });

}

}

Update

[HttpPut("/api/FootballPlayers/{id}")]

public async Task<ActionResult<FootballPlayer>> UpdatePlayer(string id, [FromBody] FootballPlayer player)

{

try

{

var updatedPlayer = await \_footballPlayerService.UpdateFootballPlayer(player);

return Ok(updatedPlayer);

}

catch (Exception ex)

{

return StatusCode(400, $"{ex.Message}");

}

}

Delete

[HttpDelete("/api/FootballPlayers/{id}")]

public async Task<ActionResult<FootballPlayer>> DeletePlayer(string id)

{

try

{

var detletedPlayer = await \_footballPlayerService.DeleteFootballPlayer(id);

return Ok(detletedPlayer);

}

catch (Exception ex)

{

return StatusCode(400, $"{ex.Message}");

}

}

Get all Club

[HttpGet("/api/FootballClubs")]

public async Task<ActionResult<List<FootballClub>>> GetClub()

{

try

{

var clubs = await \_footballPlayerService.GetFootballClubs();

return Ok(clubs);

}

catch (Exception ex)

{

return StatusCode(400, $"{ex.Message}");

}

}

Tạo 1 controller mới để login

Vẫn sử dụng controller base

Nhớ vào program thêm

modelBuilder.EntitySet<Pet>("Pets");

modelBuilder.EntitySet<PetOwnerAccount>("PetOwnerAccounts");

dependency injection vào

A screen shot of a computer program

Description automatically generated

Copy hàm login Nhớ tạo 2 cái DTO 1 cái request 1 cái response

Thay lại các role id và password là xong

[HttpPost("Login")]

public async Task<IActionResult> Login([FromBody] LoginRequestDTO loginDTO)

{

var account = await \_petOwnerAccountService.Login(loginDTO.Email, loginDTO.Password);

if (account == null)

{

return Unauthorized("Invalid email or password.");

}

//Generate JWT Token

IConfiguration configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", true, true).Build();

var claims = new List<Claim>

{

new Claim(ClaimTypes.Email, account.OwnerEmail),

new Claim("Role", account.OwnerRole.ToString()),

new Claim("AccountId", account.OwnerId.ToString()),

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["JWT:SecretKey"]));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var preparedToken = new JwtSecurityToken(

issuer: configuration["JWT:Issuer"],

audience: configuration["JWT:Audience"],

claims: claims,

expires: DateTime.Now.AddMinutes(30),

signingCredentials: creds);

var token = new JwtSecurityTokenHandler().WriteToken(preparedToken);

var role = account.OwnerRole.ToString(); //0:Admin 1:Staff 2:Manager

var accountId = account.OwnerId.ToString();

return Ok(new LoginResponseDTO

{

Role = role,

Token = token,

AccountId = accountId

});

}