

Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Co to jest DTO  
Co to jest DAO

WEB API

DTO i DAO

**DTO i DAO**

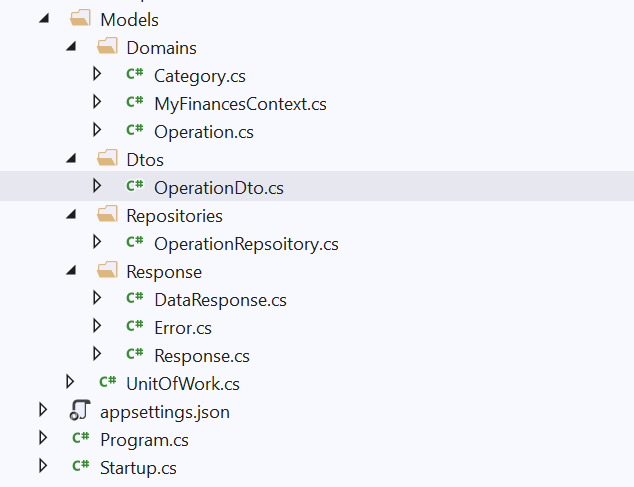
**DTO** – **Date Transfer Object** – obiekty, które służą do przesyłania danych między aplikacjami.  
Powinniśmy na zewnątrz udostępniać takie właśnie obiekty DTO, które po pierwsze mają tylko te pola, które faktycznie chcemy udostępnić, te klasy nie powinny zawierać żadnej logiki biznesowej, żadnych metod tylko właściwości.

**DAO** – **Date Access Object** – obiekty domenowe, któ®e mogą zawierać już logikę biznesową, konfigurację bazodanową

Dlatego często piszemy dodatkowo obiekty DTO i mapery pomiędzy obiektami DAO a DTO.  
Możemy użyć jakieś zewnętrznej klasy, jakiegoś automapera, a możemy też napisać własną klasę.

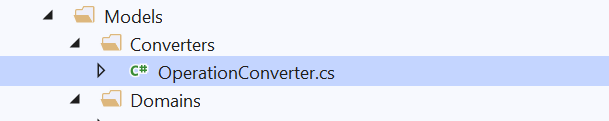
W klasach DTO zazwyczaj usuwamy pola, których nie chcemy przwsyłać np. jakieś dane wrażliwe, hasła itd. Możemy również dodać dane, których nie ma w klasach DAO np. pełen symbol faktury powstały z symbolu i numeru.

Tworzymy w folderze Models nowy folder Dtos a w nim klasę OperationDto.cs



|  |
| --- |
| namespace MyFinances.WebApi.Models.Dtos  {  public class OperationDto  {  public int Id { get; set; }  public string Name { get; set; }  public string Description { get; set; }  public decimal Value { get; set; }  public DateTime Date { get; set; }  public int CategoryId { get; set; }  }  } |

Tworzymy w folderze Models nowy folder Converters a w nim klasę OperationConverter.cs



Będzie to metoda statyczna i będziemy pisać w niej metody rozszerzające

|  |
| --- |
| namespace MyFinances.WebApi.Models.Converters  {  public static class OperationConverter  {  public static OperationDto **ToDto**(this Operation model)  {  return new OperationDto()  {  CategoryId = model.CategoryId,  Date = model.Date,  Description = model.Description,  Value = model.Value,  Id = model.Id,  Name = model.Name  };  }  public static IEnumerable<OperationDto> **ToDtos**(this IEnumerable<Operation> model)  {  if (model != null)  return Enumerable.Empty<OperationDto>();  return model.Select(x => x.ToDto());  }  public static Operation **ToDao**(this OperationDto model)  {  return new Operation()  {  CategoryId = model.CategoryId,  Date = model.Date,  Description = model.Description,  Value = model.Value,  Id = model.Id,  Name = model.Name  };  }  }  } |

Teraz pozostaje jeszcze dokonać odpowiednich zmian w kontrolerze, aby operował on  
na obiektach DTO.

|  |
| --- |
| namespace MyFinances.WebApi.Controllers  {  [Route("api/[controller]")]  [ApiController]  public class OperationController : ControllerBase  {  private readonly UnitOfWork \_unitOfWork;  public OperationController(UnitOfWork unitOfWork)  {  \_unitOfWork = unitOfWork;  }    [HttpGet]  public DataResponse<IEnumerable<OperationDto>> Get()  {  var responce = new DataResponse<IEnumerable<OperationDto>>();  try  {  responce.Data = \_unitOfWork.Operation.Get().ToDtos();  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add( new Error(ex.Source,ex.Message));  }    return responce;  }  [HttpGet("{id}")]  public DataResponse<OperationDto> Get(int id)  {  var responce = new DataResponse<OperationDto>();  try  {  responce.Data = \_unitOfWork.Operation.Get(id).ToDto();  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add(new Error(ex.Source, ex.Message));  }  return responce;  }  [HttpPost]  public DataResponse<int> Add(OperationDto operation)  {  var responce = new DataResponse<int>();  try  {  \_unitOfWork.Operation.Add(operation.ToDao());  \_unitOfWork.Complete();  // tutaj jest już znane id dodanego rekordu  responce.Data = operation.Id;  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add(new Error(ex.Source, ex.Message));  }  return responce;  }  [HttpPut]  public Response Update(OperationDto operation)  {  var responce = new Response();  try  {  \_unitOfWork.Operation.Update(operation.ToDao());  \_unitOfWork.Complete();  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add(new Error(ex.Source, ex.Message));  }  return responce;  }  [HttpDelete("{id}")]  public Response Delete(int id)  {  var responce = new Response();  try  {  \_unitOfWork.Operation.Delete(id);  \_unitOfWork.Complete();  }  catch (Exception ex)  {  // logowanie do pliku ...  responce.Errors.Add(new Error(ex.Source, ex.Message));  }  return responce;  }  }  } |