

Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Entity Framework podejście database first

WEB API

Operacje na bazie danych

**Operacje na bazie danych**

W tym tygodniu nie użyjemy Dependency Injection, ponieważ operujemy na małej bazie.  
Utworzymy klasę Repozytorium oraz UnitOfWork.

**Repozytorium**

Oczekujemy, że kontekst zostanie wstrzyknięty do konstruktora za pomocą DI.  
Tworzymy standardowo akcje Get(),Add(),Delete(),Update()

|  |
| --- |
| namespace MyFinances.WebApi.Models.Repositories  {  public class OperationRepsoitory  {  private readonly MyFinancesContext \_context;  public OperationRepsoitory(MyFinancesContext context)  {  \_context = context;  }    public IEnumerable<Operation> Get()  {  return \_context.Operations.ToList();  }  // nie chcemy aby był rzucony wyjątek dlatego  // nie dajemy ani Single() ani First()  public Operation Get(int id)  {  return \_context.Operations.FirstOrDefault(x => x.Id == id);  }  public void Add(Operation operation)  {  operation.Date = DateTime.Now;  \_context.Operations.Add(operation);  }  public void Delete(int id)  {  var operationToDelete = \_context.Operations.Single(x => x.Id == id);  \_context.Operations.Remove(operationToDelete);  }  public void Update(Operation operation)  {  var operationToUpdate = \_context.Operations.Single(x => x.Id == operation.Id);  operationToUpdate.Name = operation.Name;  operationToUpdate.Value = operation.Value;  operationToUpdate.CategoryId = operation.CategoryId;  }  }  } |

**UnitOfWork**

|  |
| --- |
| namespace MyFinances.WebApi.Models  {  public class UnitOfWork  {  private readonly MyFinancesContext \_context;  public UnitOfWork(MyFinancesContext context)  {  \_context = context;  Operation = new OperationRepsoitory(\_context);  }  public OperationRepsoitory Operation { get; }  public void Complete()  {  \_context.SaveChanges();  }  }  } |