# Lab ToDoApp – Step7 Generalisierung

Übungsdauer : 60 Minuten

## Overview

In dieser Übung wird nun unseres ToDo-Repository mit Hilfe von Generics verallgemeinert. Die Logik des Repositorien soll auch für ToDo-Kategorie oder zukünftigen Entitäten möglich sein.

## Ziel

* Generische Interfaces

## Schritte

Zuerst wird das Interface in ein generisches Interface umgebaut und auf die wesentlichen Elemente reduziert.

1. **Öffnen sie die Solution**
2. **Generisches Interface**
   * Öffnen sie in dem Projekt „ToDoApp.Business“ das Interface IToDoRepository
   * Fügen sie die Definition für die Generik an die Deklaration:

public interface IToDoRepository<T>

{

* + Zusätzlich muss an den generischen Platzhalter eine Bedingung geknüpft werden, da es min. eine Klasse sein muss und einen new() Operator haben soll:

where T:class, new()

* + Tauschen sie alle Stellen an den die Klasse ToDo verwendet wurde durch T aus z.B:

IQueryable<T> GetAll();

1. **Repositories anpassen**
   * Öffnen sie im Projekt „ToDoApp.Business“ die Datei „ToDoRepository.cs“
   * Fügen sie auch hier die Generik für das Repository an:

public class ToDoRepository: IToDoRepository<ToDo>

{

…

* + Öffnen sie im Projekt „ToDoApp.Business“ die Datei „ToDoRepositoryEDM.cs“
  + Fügen sie auch hier die Generik für das Repository an:

public class ToDoRepositoryEDM:DbContext, IToDoRepository<ToDo>, IToDoRepository<ToDoCategory>

{

…

* + Fügen sie die GetAll Situation für ToDoCategory an das Repository an:

IQueryable<ToDoCategory> IToDoRepository<ToDoCategory>.GetAll()

{

return ToDoCategories;

}

* + Fügen sie die Create Situation für ToDoCategory an das Repository an:

ToDoCategory IToDoRepository<ToDoCategory>.Create()

{

ToDoCategory myResult = new ToDoCategory() { Title = "Neue Kategorie" };

ToDoCategories.Add(myResult);

this.SaveChanges();

return myResult;

}

* + Fügen sie die Update Situation für ToDoCategory an das Repository an:

public void Update(ToDoCategory myToDoCategory)

{

this.SaveChanges();

}

* + Fügen sie die Create Situation für ToDoCategory an das Repository an:

public void Delete(ToDoCategory myToDoCategory)

{

ToDoCategories.Remove(myToDoCategory);

this.SaveChanges();

}

## Schritte

Nun wird das neue Repository angewendet

1. **Öffnen sie die Solution**
2. **Generisches Interface anwenden**
   * Öffnen sie in dem Projekt „ToDoApp.Client“ die Klasse MainViewModel
   * Ändern sie die Deklaration für das Repository und fügen noch das zweite Repository an:

private IToDoRepository<ToDo> \_ToDoRepository;

private IToDoRepository<ToDoCategory> \_ToDoCateoryRepository;

* + Öffnen sie die Datei RepositoryFactory.cs
  + Ändern sie den Rückgabewert des Repositories:

public static IToDoRepository<ToDo> Create(string myClassname)

{

…

return myResult as IToDoRepository<ToDo>;

}

* + Öffnen sie in dem Projekt „ToDoApp.Client“ die Klasse MainViewModel
  + Ergänzen sie die Initialisierung im Konstruktor für das zweite Repository:

\_ToDoCateoryRepository = \_ToDoRepository as IToDoRepository<ToDoCategory>;

* + Testen sie ihre Applikation