# Lab ToDoApp – Step6 Modularisierung

Übungsdauer : 60 Minuten

## Overview

In dieser Übung soll die ToDo-App über ein Repository Interface erweitert werden, um die Anwendung per Konfiguration mit zwei unterschiedlichen Repositories (File, DB) betreiben zu können. Der Umschalt-Mechanismus soll dazu mit einer losen Kopplung erfolgen.

## Ziel

* Interfaces
* Reflection

## Schritte

In diesem ersten Schritt wird ein Interface für die Repositorien erstellt und angewendet.

1. **Öffnen sie die Solution**
2. **Interface erstellen**
   * Klicken sie auf das Projekt „ToDoApp.Business“ mit der rechten Maustaste und wählen das Menü „Add/New Item“
   * Wählen sie die Vorlage „Interface“ aus und geben sie der Interface den Namen „IToDoRepository“
   * Kennzeichnen sie das Interface mit „public“
   * Fügen sie die Signaturen aus der Klasse „ToDoRepositoryEDM“ an
3. **Interface anwenden**
   * Klicken sie im Projekt „ToDoApp.Business“ die Datei ToDoRepository an
   * Schreiben sie die Verwendung des Interfaces an die Klasse:

public class ToDoRepository: IToDoRepository

{

* + Da die Klasse in der GetAll() Methode ein etwas anderen Rückgabe Wert hat, muss die Funktion an das Interface angepasst werden:

public IQueryable<ToDo> GetAll()

{

return \_Data.ToList().AsQueryable();

}

* + Wiederholen sie die Schritte mit der Klasse „ToDoRepositoryEDM“

## Schritte

In diesem Schritt wird nun in dem ViewModel eine dynamische Umschaltung der Repositories eingebaut. Dazu wird in der App.config eine Option hinzugefügt, um das entsprechende ToDoRepository anzugeben

1. **Öffnen sie die Solution**
2. **Konfiguration erstellen**
   * Klicken sie in dem Projekt „ToDoApp.Client“ auf die Datei „App.config“
   * Tragen sie in dem Bereich Configuration folgende Einstellung ein:

<appSettings>

<add key="Repository" value="ToDoApp.Business,ToDoApp.Business.ToDoRepository" />

</appSettings>

* + Fügen sie in das Projekt die Reference auf „System.Configuration“ ein
  + Öffnen sie die Datei „MainViewModel.cs“
  + Ersetzten sie das Erstellen des Repository durch eine Factory Klasse, die in Schritt 6 erstellen:

\_ToDoRepository = RepositoryFactory.Create(ConfigurationManager.AppSettings["Repository"]);

* + Änderen sie den Datentyp der privaten Variable \_ToDoRepository durch das Interface

1. **Factory Klasse für Repositories erstellen**
   * Klicken sie in dem Projekt „ToDoApp.Client“ und erstellen dort eine neue Klasse „RepositoryFactory“
   * Kennzeichnen Sie die Klasse mit static
   * Fügen sie eine statische Methode „Create“ ein:

public static IToDoRepository Create(string myClassname)

{

}

* + Nun kann mit Hilfe des .NET Reflector Klasse Activator ein Objekt erstellen:

string[] myToken = myClassname.Split(',');

Assembly myAssembly = Assembly.LoadFile(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, myToken[0]+".dll"));

var myResult = myAssembly.CreateInstance(myToken[1]);

return myResult as IToDoRepository;

* + Testen sie ihre Applikation mit beiden Repositories