# Lab ToDoApp – Step3 Modeling

Übungsdauer : 60 Minuten

## Overview

In dieser Übung wird das Business Model für die ToDo App entworfen und als C# Klassen implementiert. Zusätzlich wird eine Repository Klasse für das Erstellen, Ändern, Löschen und Speichern von ToDo´s implementiert.

## Ziel

* Klassendiagramm
* OO-Design und Implementierung

## Schritte

In diesem Teil wird ein Model für die ToDo-App angelegt und mit einer Basis Implementierung versehen

1. **Öffnen sie die Solution**
2. **Klassendiegramm erstellen und ToDo Modelle anlegen**
   * Klicken sie auf das Projekt „ToDoApp.Business“ mit der rechten Maustaste und wählen das Menü „Add/New Folder“ und legen einen Ordner „Models“ an
   * Legen sie in den Ordner nun ein neues Klassendiagramm an
   * Fügen sie die Klasse „ToDoCategory“ mit folgenden Properties an
     + int ID
     + string Title
   * Fügen sie die Klasse „ToDo“ mit folgenden Properties an
     + int ID
     + string Title
     + DateTime? Date
     + List<ToDoCategory> Categories
     + string Body
3. **Implementierung**
   * Verschieben sie die Code-Datei in den richtigen Ordner und ändern sie den Namespace entsprechend ab
   * Ersetzen sie den Dummy-Code des Klassendiagramms mit folgenden Schablone:

public int ID { get; set; }

* + Schreiben sie für die zwei Klassen jeweils einen geeigneten Konstruktor:

public ToDoCategory()

{

Title = "Neue Kategorie";

}

## Schritte

Um mit dem Model arbeiten zu können wird nun in dem Bussines Projekt ein ToDo-Repository angelegt, um ToDo-Objekte zu verwalten.

1. **Anlegen einer Class**
   * Klicken sie mit der rechten Maustaste auf das Projekt „ToDoApp.Business“
   * Wählen sie die Vorlage „Class“ aus und geben sie den Namen „ToDoRepository“ an
   * Klicken sie auf Ok
   * Kennzeichnen sie die Klasse mit „public“
2. **Logik implementieren**
   * Zuerst soll das Repository eine Eigenschaft erhalten, in der alle ToDo´s gespeichert sind:

private List<ToDo> \_Data;

* + Dann soll das Repository die CRUD Situationen abbilden und wir implementieren zuerst alle Methoden-Rümpfe:

public List<ToDo> GetAll()

{

}

public ToDo Create()

{

}

public void Update(ToDo myToDo)

{

}

public void Delete(ToDo myToDo)

{

}

* + Nun Fügen sie die Implementierung für ein Create ein:

ToDo myResult = new ToDo("Neue Aufgabe");

myResult.ID = 1;

if( \_Data.Count>0)

myResult.ID = \_Data.Max(x => x.ID)+1;

\_Data.Add(myResult);

return myResult;

* + Nun Fügen sie die Implementierung für ein Update ein:

for (int i = 0; i < \_Data.Count; i++)

{

if( \_Data[i].ID==myToDo.ID)

{

\_Data[i] = myToDo;

return;

}

}

* + Nun Fügen sie die Implementierung für ein Delete ein:

\_Data.Remove(myToDo);

* + Und zum Schluss noch die Abfrage Methodik:

return \_Data.ToList();

Implementieren sie noch einen Konstruktor für die ordentliche Instanziierung des Repositories:

public ToDoRepository()

{

\_Data = new List<ToDo>();

}

## Schritte

Die Business – Klassen werden nun wieder in dem Unittest Projekt überprüft.

1. **Öffnen sie die Solution**
2. **Anlegen eines Test-Klasse**
   * Klicken sie im Projekt „ToDoApp.Unittests“ auf „Add/New Item“
   * Klappen sie die Kategorie „Visual C#/Test“ auf die Vorlage „Unit Test“ aus
   * Geben sie ihre Testklasse den Namen „ToDoTest“
   * Klicken sie auf Ok
   * Ändern sie den Methodennamen „TestMethode1“ in „ToDo\_ID\_Testcase1“
   * Fügen sie für jede Eigenschaft/Methode eine Testmethode ein
   * Wiederholen sie diesen Vorgang für das Modell „ToDoCategory“
   * Wiederholen sie diesen Vorgang für das „ToDoRepository“
3. **Testcode ausführen**
   * Öffnen sie den Test-Explorer über den Menüpunkt „Test/Window“
   * Klicken sie auf „Run All“
   * Ändern sie ihre Implementierung und testen erneut