# Lab ToDoApp – Exception

Übungsdauer : 60 Minuten

## Overview

In dieser Übung wird die Fehlerbehandlung der ToDo-App angebunden. Es soll auf alle allgemeinen Fehler reagiert werden und diese detailliert Protokolliert werden. Dazu wird der vorbereitete WPF-View mit einer Klasse (MainViewModel) erweitert und dort der Fehlerbehandlungscode geschrieben.

## Ziel

* Exception-Handling

## Schritte

In diesem Teil wird zuerst eine ViewModel Klasse mit den Eigenschaften für den View und die Methoden für Add,Delete und Update implementiert

1. **Öffnen sie die Solution**
2. **ViewModel erstelle**
   * Klicken sie in dem Projekt ToDoApp.Client auf die Datei „MainViewModel.cs“
   * Fügen sie eine Eigenschaft für eine ToDo-Liste ein:

private List<ToDo> \_DataList;

public List<ToDo> DataList

{

get { return \_DataList; }

set { \_DataList = value; RaisePropertyChange("DataList"); }

}

* + Fügen sie eine Eigenschaft für ein selektiertes ToDo-Objekt ein

private ToDo \_SelectedToDo;

public ToDo SelectedToDo

{

get { return \_SelectedToDo; }

set { \_SelectedToDo = value; RaisePropertyChange("SelectedToDo"); }

}

* + Fügen sie Eigenschaften für als ICommand für Add,Delete und Update

public ICommand AddCommand { get; set; }

public ICommand DeleteCommand { get; set; }

public ICommand UpdateCommand { get; set; }

* + Für die Datenabfragen wird das ToDoRepository verwendet, legen sie dazu eine private Member Variable in der ViewModel Klasse an:

private ToDoRepository \_ToDoRepository;

1. **Logik Methoden im ViewModel implementieren**
   * Klicken sie in dem Projekt ToDoApp.Client auf die Datei „MainViewModel.cs“
   * Implementieren sie zuerst einen Konstruktor für die Klassen um die Variablen zu initialisieren

DataList = new List<ToDo>();

SelectedToDo = null;

AddCommand = new StandardCommand(Add);

DeleteCommand = new StandardCommand(Delete);

UpdateCommand = new StandardCommand(Update);

\_ToDoRepository = new ToDoRepository();

* + Erzeugen sie nun für die drei Situationen private Member Methoden in der ViewModel Klasse wie z.B.:

private void Add(object parameter)

{

}

* + Für die Add Situation wird ein neues ToDo-Objekt über das Repository erstellt und die DataList vom Repository neu gesetzt:

var myToDo = \_ToDoRepository.Create();

this.DataList = \_ToDoRepository.GetAll();

this.SelectedToDo = myToDo;

* + Implementieren sie die Delete und Update Methode nach dem gleichen Prinzip durch.
  + Testen sie die Applikation

1. **Fehlerbehandlung**
   * Klicken sie auf ToDoRepository.cs im Projekt 1ToDoApp.Bussiness
   * Werfen sie zum testen in der Create Methode eine Exception:

throw new Exception();

* + Starten sie die Applikation und prüfen sie das Verhalten bei Add
  + Nun wird die Applikation abstürzen, dazu soll ein Fehlerbehandlungscode im ViewModel eingebaut werden. Öffnen sie die Datei MainViewModel.cs
  + Fügen sie in den jeweiligen CRUD Methoden die try-catch Anweisungen ein:

try

{

…

}

catch( Exception ex )

{

}

* + In dem catch Block kann nun ihr Fehlerbhandlungscode stehen, z.b. das Protokollieren des Fehlers und eine Ausgabebox auf dem Bildschirm:

Logging.WriteLog("Fehler: " + ex.ToString());

MessageBox.Show("Fehler: " + ex.Message);

* + Testen sie ihre Anwendung und entfernen anschließen den throw Exception() Anweisung im Repository.