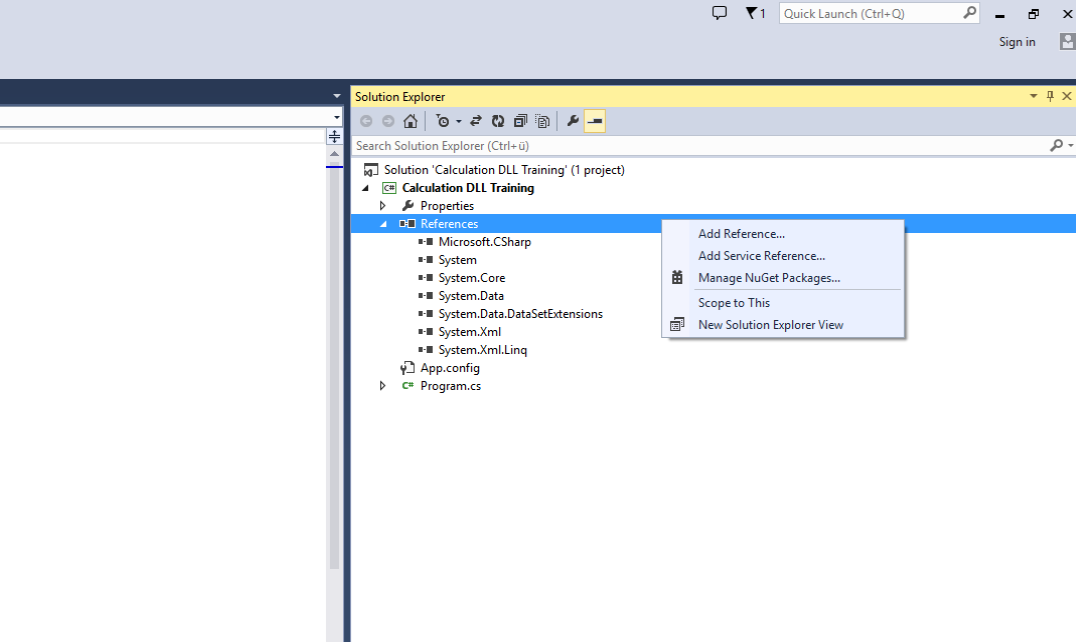
# Berechnungs DLL

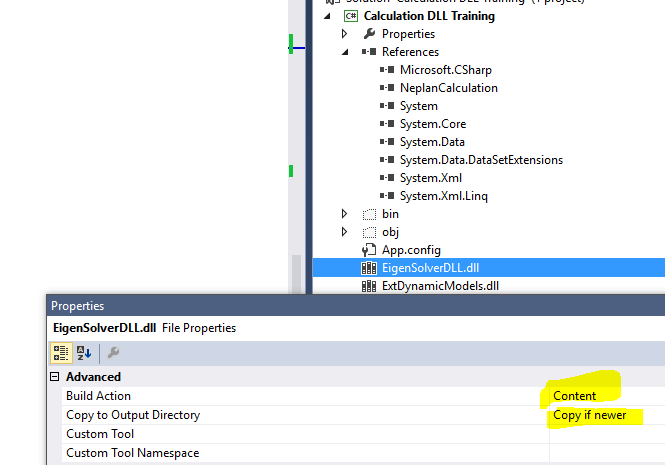
## Projekt Erstellen

* Öffnen Sie Microsoft Visual Studio und erstellen Sie eins neues “Console Application“-Projekt. Das Projekt sollte im Ordner “Übung“ gespeichert werden. Benennen Sie das Projekt mit “Calculation DLL Training“.
* Machen Sie Rechtklick auf die “References“ im “Solution Explorer“ und selektieren Sie die Option “Add Reference“.

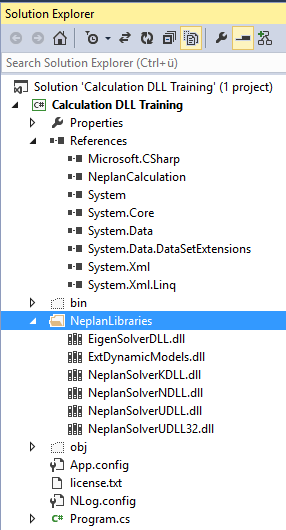


* Klicken Sie auf “Browse“ und navigieren Sie zur Berechnung-DLL.
* Machen Sie Rechtklick auf der “Calculation DLL Training“ und wählen Sie “Add existing item“.
* Selektieren Sie die “license.txt“-Datei. Machen Sie Rechtklick auf die Datei und wählen Sie “Properties“ aus. Setzen Sie die die Option “Build Action“ auf “Content“ und die Option “Copy to Output Directory“ auf “Copy if newer“.

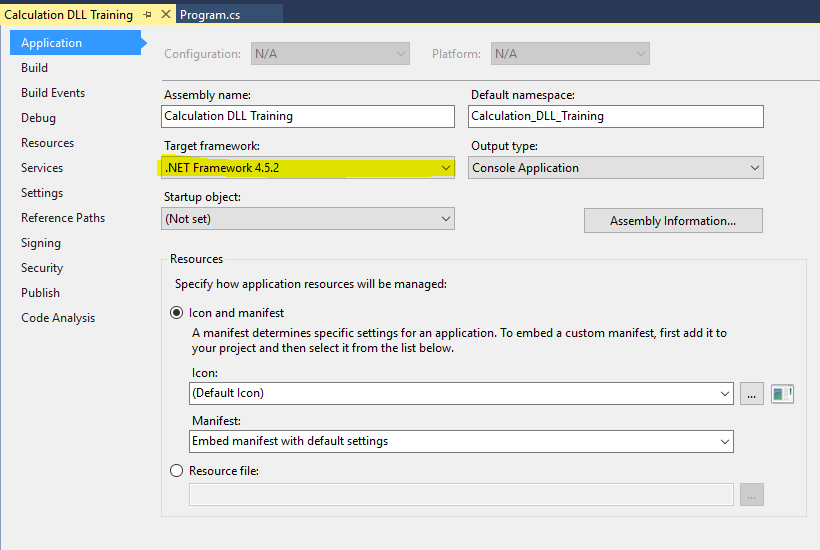
* Wiederholen Sie den Vorgang für :
  + 1. ExtDynamicModels.dll
    2. EigenSolverDLL.dll
    3. NeplanSolverKDLL.dll
    4. NeplanSolverNDLL.dll
    5. NeplanSolverUDLL.dll
    6. NeplanSolverUDLL32.dll
    7. NLog.config



* Erstellen Sie einen Ordner und fügen Sie die bearbeiteten DLLs ein. Benennen Sie den Ordner “NeplanLibraries“.



* Machen Sie Rechtklick auf “Calculation DLL Training“ und selektieren Sie “Properties“. Im Tab “Build“ deaktivieren Sie die Option “prefer 32-bit“ und wählen Sie “Any CPU“ bei “Configuration Platform“.
* Stellen Sie sicher dass Sie .Net Framework 4.5.2 (oder eine neuere Version) benützen.



## Klasse DLL

* Schliessen Sie die folgenden “Namespaces“ ein:

using System.IO;

using NeplanModel;

using BCP.Neplan.Web.Models.TechData;

using BCP.Neplan.Web.Models.External;

* Erstellen Sie eine neue Klasse mit dem Namen “DLL“ wie in 2.1 dargestellt.
* Ergänzen Sie den folgenden Code in “Main“ und starten Sie die Applikation:

DLL DLLexample = new DLL();

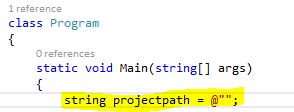
Console.WriteLine("Press any key to stop...");

Console.ReadKey();

* Kopieren Sie den Machinecode und teilen Sie diesen bitte dem Trainer mit (**Die Lizenz ist PC-abhängig!**). Sie erhalten eine Lizenz für die Berechnungs-DLL. Öffnen Sie das license.txt und ersetzen Sie die alte mit der neuen Lizenz.
* Starten Sie die Applikation neu.

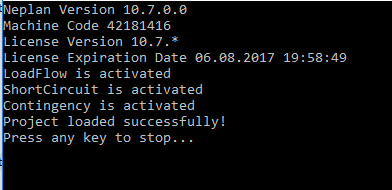
## Klasse „Program“

* Ersetzen Sie die Klasse „Program“ mit dem Code in Kapitel 2.2
* Füllen Sie den “projectpath“ für das Projekt “Neplan Beispiel.nep360“ aus



## ImportProject

* Ersetzen Sie die Funktion “ImportProject“ mit dem Code in 2.3 und starten Sie die Applikation. Das Projekt wird erfolgreich geladen.



## Lastfluss-Berechnung

* Ersetzen Sie die “RunLoadFlow“ mit dem Code in 2.4 und starten Sie die Applikation. Die Berechnung wird durchgeführt und die Verluste werden dargestellt.

## Elementeergebnisse

* Ersetzen Sie die Funktion “GetElementResult“ mit dem Code in 2.5 und starten Sie die Applikation neu. Nach der Berechnung wird die Wirkleistung der Netzeinspeisung geholt und angegeben.

## Schalterzustand Änderung

* Benützen Sie die Kode in 2.6 um die eine Last des Beispiels abzuschalten.

## Attributeänderung

* Zunächst ändern Sie die Produktion der PV-Installation in dem Sie den Code in 2.7 benützen. Die Produktion soll auf 40 kW angepasst werden.
* Starten Sie die Applikation und überprüfen Sie die Richtung des Lastflusses bei der Netzeinspeisung.

# ANHANG

#### Code der Klasse DLL

public class DLL

{

public NeplanInterface nepInterface;

public DLL()

{

NeplanInterface.SetLicensePath(Directory.GetCurrentDirectory()); //gets the license path

NeplanInterface.SetNeplanLibrariesPath(Directory.GetCurrentDirectory() + @"\NeplanLibraries"); //gets the additional dlls

Console.WriteLine("Neplan Version " + NeplanInterface.GetProductVersion()); //shows the Neplan Version

Console.WriteLine("Machine Code " + NeplanInterface.getMachineCode()); //calculates the machine code which is useful for the license

nepInterface = NeplanInterface.CreateInterface(); //creates a new instance of the calculation dll class

if (nepInterface == null)

{

Console.WriteLine("Error in activating the dll. License might be missing!");

return;

}

else

{

Console.WriteLine("License Version " + nepInterface.getLicenseVersion()); //shows the license version

Console.WriteLine("License Expiration Date " + nepInterface.GetLicenseExpiringDate().ToString()); //shows the expiration day of the license

foreach (string module in nepInterface.getModulesEnabled())

Console.WriteLine(module + " is activated"); //shows which modules are activated with the given license

}

}

}

#### Code der Klasse Program

class Program

{

bool CalcOk = true;

static void Main(string[] args)

{

string projectpath = @"";

Program example = new Program();

DLL DLLinstance = new DLL();

if (DLLinstance.nepInterface != null && example.CheckProject(projectpath)) //Checks if the interface was created and if the Neplan project exists

{

example.ImportProject(DLLinstance.nepInterface, projectpath); //loads the project on the memory

example.RunLoadFlow(DLLinstance.nepInterface); //runs a load flow

example.GetElementResult(DLLinstance.nepInterface); //gets an element result

example.OpenSwitch(DLLinstance.nepInterface); //open a switch

example.ChangePsetting(DLLinstance.nepInterface); //sets P of a 1-port element

example.RepeatLoadFLowandResults(DLLinstance.nepInterface); //repeats the process

}

Console.WriteLine("Press any key to stop...");

Console.ReadKey();

}

internal bool CheckProject(string projectpath)

{

if (string.IsNullOrEmpty(projectpath) || !File.Exists(projectpath))

{

Console.WriteLine("Project does not exist!");

return false;

}

var ext = Path.GetExtension(projectpath);

if (ext != ".nep360")

{

Console.WriteLine("Project is not a .nep360 file!");

return false;

}

return true;

}

internal void ImportProject(NeplanInterface nepInterface, string projectpath)

{

}

internal void RunLoadFlow(NeplanInterface nepInterface)

{

}

internal void GetElementResult(NeplanInterface nepInterface)

{

}

internal void OpenSwitch(NeplanInterface nepInterface)

{

}

internal void ChangePsetting(NeplanInterface nepInterface)

{

}

internal void RepeatLoadFLowandResults(NeplanInterface nepInterface)

{

}

}

#### Code der ImportProject

internal void ImportProject(NeplanInterface nepInterface, string projectpath)

{

if (!CalcOk)

return;

ImportReturnInfo info = nepInterface.ImportNep360Project(projectpath); //import the project

if (info != null && info.ReturnValue == 0)

Console.WriteLine("Project loaded successfully!");

else

{

Console.WriteLine("Project could not be loaded!");

CalcOk = false;

}

}

#### Code der Lastflussberechnung

internal void RunLoadFlow(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

AnalysisReturnInfo info = nepInterface.RunLoadFlow(); //run the load flow with the default calculation parameters

if (info != null && info.ReturnInfo == 1)

{

Console.WriteLine("Load Flow run successfully!");

foreach (var str in info.LogList)

Console.WriteLine(" " + str);

}

else

{

Console.WriteLine("Could not run Load Flow!");

CalcOk = false;

}

}

#### Code Elementeergebnisse holen

internal void GetElementResult(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

TechElement techelem = nepInterface.GetElementByName("Netz"); //get the element (and its technical data) from the project

if (techelem == null)

{

Console.WriteLine("Netz does not exist in the project!");

CalcOk = false;

return;

}

double P = 0.0;

int portNumber = 0; //0 is the first port connection of an element in Neplan

if (nepInterface.GetResultValueOfElement("LoadFlow", techelem.Id, "SymLFPortRes.P", portNumber, out P)) //gets the power import for the network

Console.WriteLine(string.Format("The network is supplied with: {0:0.00} kW", -P \* 1000));

else

{

Console.WriteLine("Could not retrieve the result!");

CalcOk = false;

}

}

#### Code für die Änderung des Schalterzustandes

internal void OpenSwitch(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

string elementname = "Verbraucher A";

if (!nepInterface.OpenConnection(elementname, 0)) //Opens the switch of the 1-port element

{

Console.WriteLine("Cannot disconnect " + elementname);

CalcOk = false;

}

else

Console.WriteLine(string.Format("{0} diconnected", elementname));

}

#### Code Änderung der Attribute

internal void ChangePsetting(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

string elementname = "PV Installation";

string power = "40";

var techelement = nepInterface.GetElementByName(elementname);

if (techelement != null)

{

((ACDisperseGenerator)techelement).Pset = 40;

Console.WriteLine(string.Format("Output of {0} set to {1}kW", elementname, power)); //Change the P setting of the PV installation

}

else

{

Console.WriteLine("Could not change output of " + elementname);

CalcOk = false;

}

#region Alternatively

//string str = nepInterface.ChangeMemberOfElement(elementname, "Pset", power);

//if(string.IsNullOrEmpty(str) || str != "Member changed")

//{

// Console.WriteLine("Could not change output of " + elementname);

// CalcOk = false;

//}

//else

// Console.WriteLine(string.Format("Output of {0} set to {1}kW", elementname, power));

#endregion

}

#### Ganzer Code (Program.cs)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using NeplanModel;

using BCP.Neplan.Web.Models.TechData;

using BCP.Neplan.Web.Models.External;

namespace Calculation\_DLL\_Training

{

class Program

{

bool CalcOk = true;

static void Main(string[] args)

{

string projectpath = @"";

Program example = new Program();

DLL DLLinstance = new DLL();

if (DLLinstance.nepInterface != null && example.CheckProject(projectpath)) //Checks if the interface was created and if the Neplan project exists

{

example.ImportProject(DLLinstance.nepInterface, projectpath); //loads the project on the memory

example.RunLoadFlow(DLLinstance.nepInterface); //runs a load flow

example.GetElementResult(DLLinstance.nepInterface); //gets an element result

example.OpenSwitch(DLLinstance.nepInterface); //open a switch

example.ChangePsetting(DLLinstance.nepInterface); //sets P of a 1-port element

example.RepeatLoadFLowandResults(DLLinstance.nepInterface); //repeats the process

}

Console.WriteLine("Press any key to stop...");

Console.ReadKey();

}

internal bool CheckProject(string projectpath)

{

if (string.IsNullOrEmpty(projectpath) || !File.Exists(projectpath))

{

Console.WriteLine("Project does not exist!");

return false;

}

var ext = Path.GetExtension(projectpath);

if (ext != ".nep360")

{

Console.WriteLine("Project is not a .nep360 file!");

return false;

}

return true;

}

internal void ImportProject(NeplanInterface nepInterface, string projectpath)

{

if (!CalcOk)

return;

ImportReturnInfo info = nepInterface.ImportNep360Project(projectpath); //import the project

if (info != null && info.ReturnValue == 0)

Console.WriteLine("Project loaded successfully!");

else

{

Console.WriteLine("Project could not be loaded!");

CalcOk = false;

}

}

internal void RunLoadFlow(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

AnalysisReturnInfo info = nepInterface.RunLoadFlow(); //run the load flow with the default calculation parameters

if (info != null && info.ReturnInfo == 1)

{

Console.WriteLine("Load Flow run successfully!");

foreach (var str in info.LogList)

Console.WriteLine(" " + str);

}

else

{

Console.WriteLine("Could not run Load Flow!");

CalcOk = false;

}

}

internal void GetElementResult(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

TechElement techelem = nepInterface.GetElementByName("Netz"); //get the element (and its technical data) from the project

if (techelem == null)

{

Console.WriteLine("Netz does not exist in the project!");

CalcOk = false;

return;

}

double P = 0.0;

int portNumber = 0; //0 is the first port connection of an element in Neplan

if (nepInterface.GetResultValueOfElement("LoadFlow", techelem.Id, "SymLFPortRes.P", portNumber, out P)) //gets the power import for the network

Console.WriteLine(string.Format("The network is supplied with: {0:0.00} kW", -P \* 1000));

else

{

Console.WriteLine("Could not retrieve the result!");

CalcOk = false;

}

}

internal void OpenSwitch(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

string elementname = "Verbraucher A";

if (!nepInterface.OpenConnection(elementname, 0)) //Opens the switch of the 1-port element

{

Console.WriteLine("Cannot disconnect " + elementname);

CalcOk = false;

}

else

Console.WriteLine(string.Format("{0} diconnected", elementname));

}

internal void ChangePsetting(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

string elementname = "PV Installation";

string power = "40";

var techelement = nepInterface.GetElementByName(elementname);

if (techelement != null)

{

((ACDisperseGenerator)techelement).Pset = 40;

Console.WriteLine(string.Format("Output of {0} set to {1}kW", elementname, power)); //Change the P setting of the PV installation

}

else

{

Console.WriteLine("Could not change output of " + elementname);

CalcOk = false;

}

#region Alternatively

//string str = nepInterface.ChangeMemberOfElement(elementname, "Pset", power);

//if(string.IsNullOrEmpty(str) || str != "Member changed")

//{

// Console.WriteLine("Could not change output of " + elementname);

// CalcOk = false;

//}

//else

// Console.WriteLine(string.Format("Output of {0} set to {1}kW", elementname, power));

#endregion

}

internal void RepeatLoadFLowandResults(NeplanInterface nepInterface)

{

if (!CalcOk)

return;

RunLoadFlow(nepInterface);

GetElementResult(nepInterface);

}

}

public class DLL

{

public NeplanInterface nepInterface;

public DLL()

{

NeplanInterface.SetLicensePath(Directory.GetCurrentDirectory()); //gets the license path

NeplanInterface.SetNeplanLibrariesPath(Directory.GetCurrentDirectory() + @"\NeplanLibraries"); //gets the additional dlls

Console.WriteLine("Neplan Version " + NeplanInterface.GetProductVersion()); //shows the Neplan Version

Console.WriteLine("Machine Code " + NeplanInterface.getMachineCode()); //calculates the machine code which is useful for the license

nepInterface = NeplanInterface.CreateInterface(); //creates a new instance of the calculation dll class

if (nepInterface == null)

{

Console.WriteLine("Error in activating the dll. License might be missing!");

return;

}

else

{

Console.WriteLine("License Version " + nepInterface.getLicenseVersion()); //shows the license version

Console.WriteLine("License Expiration Date " + nepInterface.GetLicenseExpiringDate().ToString()); //shows the expiration day of the license

foreach (string module in nepInterface.getModulesEnabled())

Console.WriteLine(module + " is activated"); //shows which modules are activated with the given license

}

}

}

}