# KPU Lab MEF

## Kommentarer til løsningsforslag

Der er 2 alternative løsningsforslag.

### Version 1

Denne løsning består af projekterne ExtensibleApp, PlugIn1 og PlugInn2.

Dette er den simpleste, og ligner C++ versionen mest.

I applikationen laves en property til dependency injection af eventuelle pluginns:

[ImportMany(typeof(IDLLcontract))]

public IEnumerable<IDLLcontract> PlugInns { get; set; }

Og inden run kaldes, så “kompones” applicationen dvs. eventuelle pluginns loades:

// Make a directoryCatalog of plugins found in the Extensions folder

var catalog = new DirectoryCatalog(@".\Extensions");

// Make an instance of a CompositionContainer

var container = new CompositionContainer(catalog,   
 CompositionOptions.DisableSilentRejection);

// Finally call composeparts to connect the Import with Export.

container.ComposeParts(this);

I plugins exporters plugin-klassen som en implementring af IDLLcontract:

[Export (typeof(IDLLcontract))]

public class PlugInn1 : IDLLcontract

### Version 2

Denne løsning består af projekterne \*\_V2.

Det som adskiller denne løsning fra den første, er at funktionen Init ikke bruges. I stedet bruges MEF til at lave en shared instans af AppUtil, som så importeres af både applikationen og pluginns.

I application:

[Export (typeof(IAppUtil))]

[PartCreationPolicy(CreationPolicy.Shared)]

public class AppUtilImp : IAppUtil

[Import(typeof(IAppUtil))]  
AppUtilImp Util { get; set; }

I plugins:

[Import (typeof(IAppUtil))]

IAppUtil appUtil {get; set;}