Cheatography

Microsoft Extensibility Framework Cheat Sheet

by r.schindhelm via cheatography.com/51393/cs/14064/

Export

Export type	`[Export(typeof(IOperation))]` `class A : IOperation { }`

Export type hierarchy	`[InheritedExport(typeof(IOperation))]`
	`class A : IOperation { }`
	`class B : A { }`

Export specific contract (category)	`[Export("MyContract",`
	`typeof(IOperation))]`
	`class A : IOperation { }`

Import

0..1	`[Import(AllowDefault = true)]` `IOperation operation;`
1..1	`[Import]` `IOperation operation;`

0..*	`[ImportMany]`
	`IEnumerable<Lazy<IOperation,`
	`IOperationMetadata>> operations;`

Import specific contract	`[Import("MyContract")]` `IOperation operation;`

The import attributes work on class fields.
For the ImportMany attribute to work, the type of the field must be a collection type.

Catalogs

AggregateCatalog	Combines multiple catalogs.
ApplicationCatalog	Discovers attributed parts in the dynamic link library (DLL) and EXE files in an application's directory and path.
AssemblyCatalog	Discovers attributed parts in a managed code assembly.
DirectoryCatalog	Discovers attributed parts in the assemblies in a specified directory. Use the overload with two parameters to narrow down DLLs and EXEs.
Filtered Catalog	Represents a catalog after a filter function is applied to it.
TypeCatalog	Discovers attributed parts from a collection of types.

Boilerplate code for using 0..n catalogs:

```
var catalog = new AggregateCatalog();
catalog.Catalogs.Add(otherCatalog);
var container = new CompositionContainer(catalog);
```