Intro to MEF for Silverlight

Rik Robinson
Senior Consultant
rrobinson@wintellect.com

Wintellect
Know how.

Microsoft GOLD CERTIFIED Partner

what we do

consulting ▪ training ▪ design ▪ debugging

who we are

Founded by top experts on Microsoft – Jeffrey Richter, Jeff Prosise, and John Robbins – we pull out all the stops to help our customers achieve their goals through advanced software-based consulting and training solutions.

how we do it

Consulting & Debugging

- Architecture, analysis, and design services
- Full lifecycle custom software development
- Content creation
- Project management
- Debugging & performance tuning

Training

- On-site instructor-led training
- Virtual instructor-led training
- Devscovery conferences

Design

- User Experience Design
- Visual & Content Design
- Video & Animation Production

What is MEF?

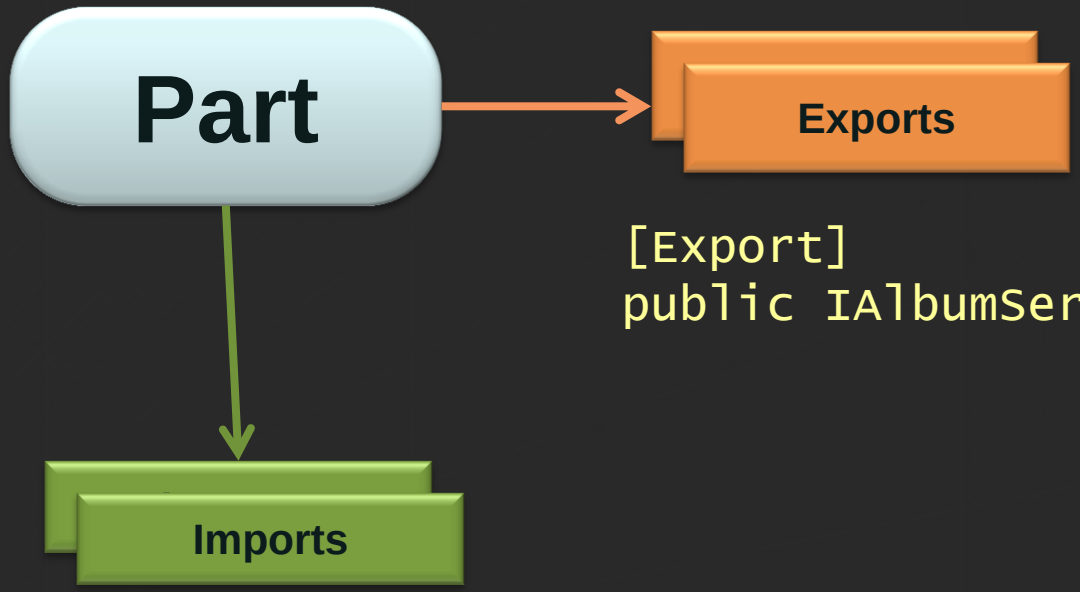
A framework that simplifies the creation of extensible applications by offering discovery and composition capabilities to load application extensions.
- CodePlex docs

An extensible framework for composing applications from a set of loosely-coupled parts discovered and evolving at run-time.
- Mike Taulty (Microsoft UK)

Benefits of MEF

- It's part of the framework!
  - System.ComponentModel.Composition
- Standardized way to expose and consume components
- Wires components together in the correct order
- Flexible discovery of components
- Metadata provides for rich querying and filtering
- Assists with lifetime management of components

Composable Parts




```
[Export]
public IAlbumService AlbumService
```

```
[Import]
public AlbumService AlbumService { get; set; }
```

Exports

Composable Parts

```
[Export]
public class DefaultInfoService
```

Properties

```
[Export(typeof(IInfoService))]
public IInfoService InfoService { get; set; }
```

Methods

```
[Export("GetInfo")]
public string GetInfo(string searchText)
```

Imports

Properties

```
[Import]
public IInfoService InfoService { get; set; }
```

Constructors

```
[ImportingConstructor]
public SomeClass(IInfoService infoService)
```

Fields

```
[Import]
private IInfoService _infoService;
```

demo

imports and exports

Imports

Optional Imports

```
[Import( AllowDefault = true )]
public IMessageSender MessageSender { get; set; }
```

Collections

```
[ImportMany]
public IEnumerable<IMessageSender> Senders { get; set; }
```

Notification

```
public class SomeClass : IPartImportsSatisfiedNotification
```

Parts Lifetime

CreationPolicy

- Shared = Shared Instance (Singleton)
- NonShared = New Instance for each export request
- Any = Default

Export

```
[PartCreationPolicy( CreationPolicy.NonShared )]
[Export(typeof(IMessageSender))]
public class MessageSender : IMessageSender
```

Import

```
[Import( RequiredCreationPolicy=CreationPolicy.Shared )]
public IMessageSender MessageSender { get; set; }
```

Deferred Creation (Lazy<T>)

- MEF will defer the creation of large or resource-intensive objects marked as Lazy
- Use in cases where object may not need to be instantiated during application lifetime
- Initialization happens the first time the Lazy<T>.Value is accessed
- Use IsValueCreated to test if created
- Located in root System namespace (.NET 4.0)

ExportFactory<T>

- Use to dynamically/programmatically create imported parts
- Silverlight Only
- Full control over when the imports are instantiated
- Call CreateExport() to create a part

```
[Import]
public ExportFactory<IInfoService> InfoServiceFactory { get; set; }

// to create one
IInfoService service = InfoServiceFactory.CreateExport().Value;
```

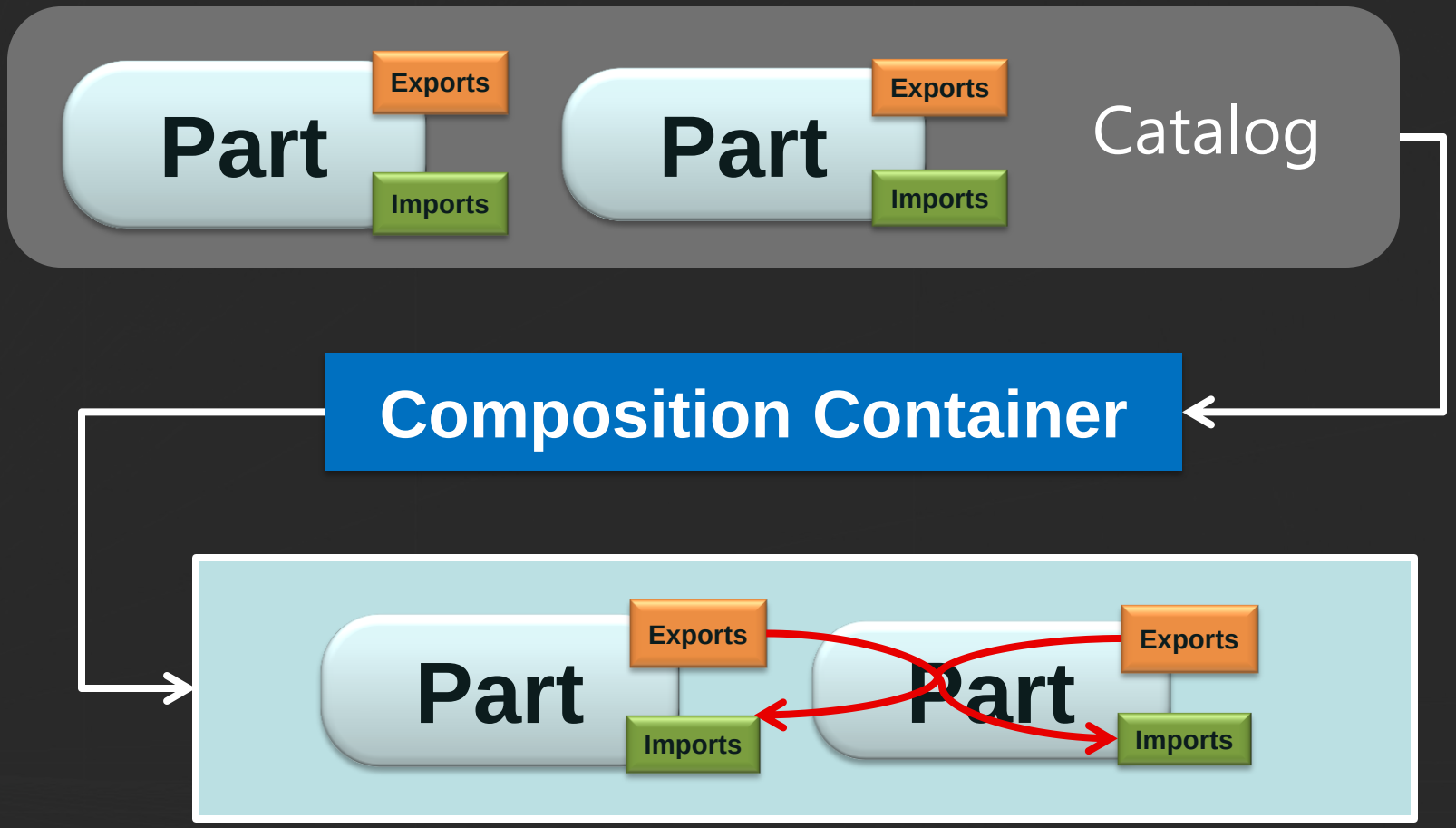
Metadata

- Add additional information about an export
- Filter exports without invoking/creating them
- Allows for plug-ins to only be generated in context
- Strongly typed
- Uses the pattern Lazy<T,TMetadata>

demo

metadata

Composition Overview



Catalogs

- Catalog keeps track of Imported and Exported Parts
- Cannot add or remove parts from catalogs
- Can add/remove catalogs from AggregateCatalog
- Create custom Catalogs by deriving from ComposablePartsCatalog

Catalogs

Catalog	Description
AssemblyCatalog	All exports in a given assembly
TypeCatalog	All exports in a specific set of types
DeploymentCatalog	All exports in a dynamically loaded XAP (Silverlight only)
AggregateCatalog	Combines multiple catalogs into a single catalog
DirectoryCatalog	All exports in a given directory (not supported in Silverlight)

- **Derive from ComposablePartsCatalog to define custom Catalogs.**

CompositionContainer

- Container is responsible for composition
- Container pulls parts from a catalog and creates or returns existing part instances as requested
- Can request instances of Exports programmatically
- CompositionContainer is one implementation of an ExportProvider – you can create your own
- System.ComponentModel.Composition.Hosting

Composition Initializer

```
CompositionInitializer.SatisfyImports(this)
```

- What does this do?
  - Creates a new default container if one has not yet been created
  - Creates a default catalog (similar to AggregateCatalog)
  - Discovers all parts in currently executing assembly
  - Discovers all parts in all referenced assemblies (in main XAP file)

- CompositionInitializer is Silverlight only
- Parameter cannot be discoverable by MEF – will throw Exception if there is an Export attribute on the passed class.
- System.ComponentModel.Composition.CompositionInitialization

CompositionContainer

```
var catalog = new TypeCatalog(typeof(MyType));

var container = new CompositionContainer(catalog);

CompositionHost.Initialize(container);

CompositionInitializer.SatisfyImports(this)
```

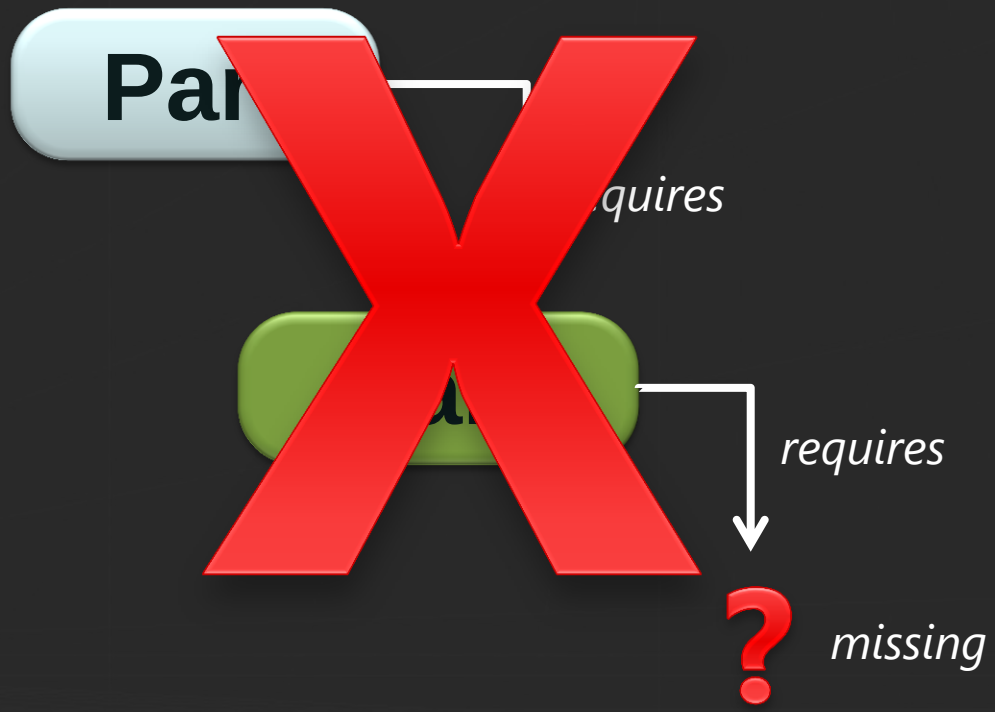
Must register the container with MEF before calling SatisfyImports() or MEF will create and use default container

demo

catalogs

Stable Composition

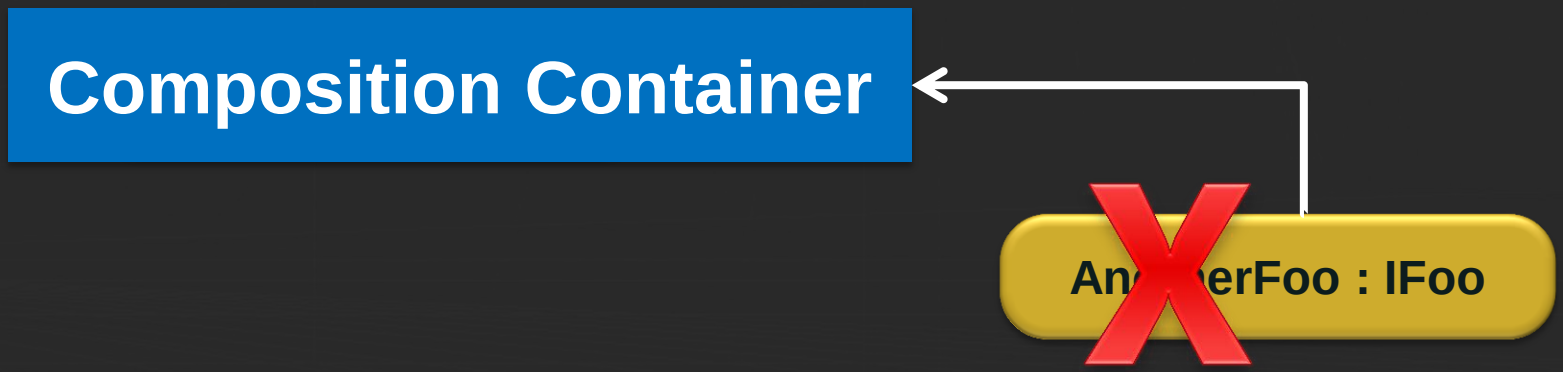
- MEF will not create parts that have unsatisfied Imports



Stable Composition

- MEF will reject changes that break current agreements

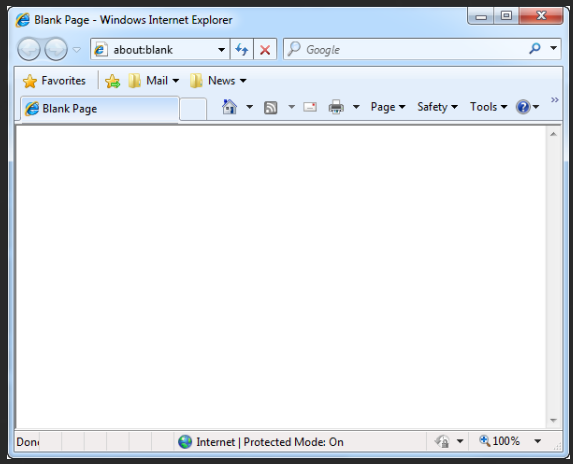
BarPart [Import<IFoo>] - *needs exactly one IFoo*



demo

Stable composition

Recomposition is key for Silverlight



Extensibility: Dynamic XAP Files

- Use an AggregateCatalog to allow multiple XAP files
- DeploymentCatalog allows asynchronous downloads for XAP files
- Use a new Silverlight Application to create satellite XAP files
- CopyLocal = false for duplicate references
- AllowRecomposition = true

demo

Recomposition and DeploymentCatalog

Resources

- MEF on CodePlex
  - mef.codeplex.com
- Glenn Block
  - blogs.msdn.com/gblock
- Mike Taulty
  - mtaulty.com
  - channel9.msdn.com/tags/learnMEFSL4
- Jeremy Likness
  - csharperimage.jeremylikness.com

Make sure to check the information on Debugging & Diagnostics in MEF

Questions?

Rik Robinson
Senior Consultant
rrobinson@wintellect.com