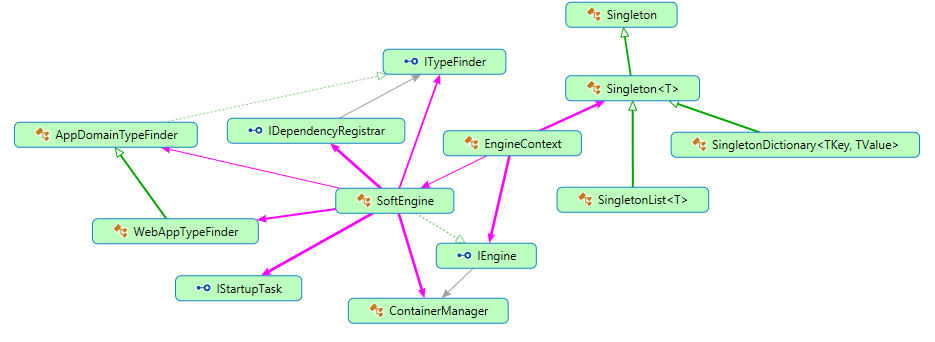
# Soft.Core.Events

## Descripcion

Establece los mecanismos basicos el manejo de codigo proveniente de archivos fuentes

## Arquitectura



### ITypeFinder

Es la interfaz que permite el manejo de los ensamblados buscandolo por tipos. Entre los metodos definidos tenemos a:

* IList<Assembly> GetAssemblies();
* IEnumerable<Type> FindClassesOfType(Type assignTypeFrom, bool onlyConcreteClasses = true);
* IEnumerable<Type> FindClassesOfType(Type assignTypeFrom, IEnumerable<Assembly> assemblies,bool onlyConcreteClasses = true);
* IEnumerable<Type> FindClassesOfType<T>(bool onlyConcreteClasses = true);
* IEnumerable<Type> FindClassesOfType<T>(IEnumerable<Assembly> assemblies, bool onlyConcreteClasses = true);

### IDependencyRegistrar

Es la interfaz que permite registrar en orden un ensamblado de tipo ItypeFinder dentro de un ContainerBuilder*:*

* void Register(ContainerBuilder builder, ITypeFinder typeFinder);
* int Order { get; }

### AppDomainTypeFinder

Es la implemtacion de de la interfaz ItypeFinder que permite hacer la identificacion de ensamblados y los carga, excluyendo a los siguiente ensamblados:

System, mscorlib, Microsoft, AjaxControlToolkit, Antlr3, Autofac, AutoMapper, Castle, ComponentArt, CppCodeProvider, DotNetOpenAuth, EntityFramework, EPPlus, FluentValidation, ImageResizer, itextsharp, log4net, MaxMind, MbUnit, MiniProfiler, Mono.Math, MvcContrib, Newtonsoft, NHibernate, nunit, Org.Mentalis, PerlRegex, QuickGraph, Recaptcha, Remotion, RestSharp, Rhino, Telerik, Iesi, TestDriven, TestFu, UserAgentStringLibrary, VJSharpCodeProvider, WebActivator, WebDev, WebGrease

### WebAppTypeFinder

Extiende la clase AppDomainTypeFinder y establece la direccion de la ruta de la cual va a hacer la busqueda de los ensamblados. Por otro lado esta clase puede realizar la busqueda de acuerdo a la configuracion del archivo web.config a travez del tag *softConfig*

### IstartupTask

Interfaz que define los metodos de inicializacion de las tareas que deberan ser ejecutadas una vez inicializando el sistema.

* void Execute();
* int Order { get; }

### Singleton

Patron de diseño que permite tener una instancia de un objeto en memoria, se define bajo el tipo:

* public static readonly IDictionary<Type, object> AllSingletons;

### Singleton<T>

Establece una instancia de un tipo T bajo el patron de singleton

* private static T \_instance;

### SingletonList<T>

Es una implementacion del tipo *Singleton<IList<T>>* el cual maneja instancias de tipo *IList<T>*

### SingletonDictionary<TKey, TValue>

Es una implementacion del tipo *Singleton<IDictionary<TKey, TValue>>* el cual maneja instancias de tipo *IDictionary<TKey, TValue>*

### IEngine

Es la interfaz que define los metodos para el manejo de los ensamblados, dependencias y tareas programadas registrandolas debidamente en el containerManager

* ContainerManager ContainerManager { get; }
* void Initialize(SoftConfig config);
* T Resolve<T>() where T : class;
* object Resolve(Type type);
* T[] ResolveAll<T>();

### SoftEngine

Es la implementación de la interfaz IEngine el cual permite el registro de ensamblados en tiempo real. Entre los ensamblados que se cargan tenemos:

* var typeFinder = new WebAppTypeFinder(config);

//dlls propias soft.core, soft.data. soft.service, soft.web

* builder.RegisterInstance(config).As<SoftConfig>().SingleInstance();
* builder.RegisterInstance(this).As<IEngine>().SingleInstance();
* builder.RegisterInstance(typeFinder).As<ITypeFinder>().SingleInstance();
* typeFinder.FindClassesOfType<IDependencyRegistrar>();

Ademas ejecuta las tareas que se implementaron

* foreach (var startUpTask in startUpTasks) startUpTask.Execute();

### ContainerManager

Es la clase encargada de resolver las dependencias asignadas mediante una llave y un *ILifetimeScope*

* public IContainer Container { get; private set; }

### EngineContext

Provee los accesos a la instancia del singleton del tipo SoftEngine

## Especificaciones

|  |  |
| --- | --- |
| SCI1 | Permitir registrar una clase deribada dentro de un builder de un tipo interfaz |
| SCI2 | Permite registrar clases deribadas en diferente builder bajo diferente LifetimeScope |
| SCI3 | Manejar de manera diferentes LifetimeScope los cuales son instancias diferentes de hilos |