

Namespace IoCContainer.Binder

Interfaces

[IContext](#)

Represents a context for managing instances and dependencies.

[ICrossContext](#)

[IInstanceProvider](#)

Interface IContext

Namespace: [locContainer.Binder](#)

Assembly: locContainer.dll

Represents a context for managing instances and dependencies.

```
public interface IContext : IDisposable, IInstanceProvider
```

Inherited Members

[IDisposable.Dispose\(\)](#), [IInstanceProvider.GetInstance<T>\(\)](#),
[IInstanceProvider.GetInstance\(Type\)](#)

Methods

AddContext(IContext)

Registers a new context to this one.

```
void AddContext(IContext context)
```

Parameters

context [IContext](#)

The context to add.

BindCrossContextAndSingleton<TKey, TValue>()

Binds a type as both a cross-context and singleton service.

```
void BindCrossContextAndSingleton<TKey, TValue>()
```

Type Parameters

TKey

The type of the key to associate with the service.

TValue

The type of the service instance.

BindCrossContextAndSingleton<TKey, TValue>(object)

Binds a type as both a cross-context and singleton service with a given name.

```
void BindCrossContextAndSingleton<TKey, TValue>(object name)
```

Parameters

name [object](#)

The name to associate with the service.

Type Parameters

TKey

The type of the key to associate with the service.

TValue

The type of the service instance.

BindInstanceAsCrossContext<TKey>(object)

Binds an instance as a cross-context service.

```
void BindInstanceAsCrossContext<TKey>(object value)
```

Parameters

value [object](#)

The instance to bind.

Type Parameters

TKey

The type of the key to associate with the instance.

BindInstanceAsCrossContext<TKey>(object, object)

Binds an instance as a cross-context service with a given name.

```
void BindInstanceAsCrossContext<TKey>(object value, object name)
```

Parameters

value [object](#)[↗]

The instance to bind.

name [object](#)[↗]

The name to associate with the binding.

Type Parameters

TKey

The type of the key to associate with the instance.

BindInstance<T>(object)

Binds an instance to a specific type.

```
void BindInstance<T>(object value)
```

Parameters

value [object](#)[↗]

The instance to bind.

Type Parameters

T

The type to bind the instance to.

BindInstance<T>(object, string)

Binds an instance to a specific type with a given name.

```
void BindInstance<T>(object value, string name)
```

Parameters

value [object](#)

The instance to bind.

name [string](#)

The name to associate with the binding.

Type Parameters

T

The type to bind the instance to.

BindSingleton<TKey, TValue>()

Binds a type as a singleton service.

```
void BindSingleton<TKey, TValue>()
```

Type Parameters

TKey

The type of the key to associate with the singleton.

TValue

The type of the singleton instance.

BindSingleton<TKey, TValue>(object)

Binds a type as a singleton service with a given name.

```
void BindSingleton<TKey, TValue>(object name)
```

Parameters

name [object](#)

The name to associate with the singleton.

Type Parameters

TKey

The type of the key to associate with the singleton.

TValue

The type of the singleton instance.

Deject(object)

Clears the injections from the provided instance. Note that only public fields will be uninjected, not constructor injections.

```
void Deject(object target)
```

Parameters

target [object](#)

The instance to clear injections from.

Inject(object)

Requests that the provided target be injected with dependencies.

```
object Inject(object target)
```

Parameters

target [object](#)

The target to inject dependencies into.

Returns

[object](#)

The target with injected dependencies.

RemoveComponents()

Removes all components from this context.

```
void RemoveComponents()
```

RemoveContext(IContext)

Removes a context from this one.

```
void RemoveContext(IContext context)
```

Parameters

context [IContext](#)

The context to remove.

Unbind<T>(object)

Unbinds a type or named instance from the context.

```
void Unbind<T>(object name = null)
```

Parameters

name [object](#) 

The name associated with the binding to unbind. If `null`, unbinds all instances of the type.

Type Parameters

T

The type to unbind.


Interface ICrossContext

Namespace: [locContainer.Binder](#)

Assembly: locContainer.dll

```
public interface ICrossContext : IContext, IDisposable, IInstanceProvider
```

Inherited Members

[IContext.AddContext\(IContext\)](#) , [IContext.RemoveContext\(IContext\)](#) ,
[IContext.RemoveComponents\(\)](#) , [IContext.BindInstance<T>\(object\)](#) ,
[IContext.BindInstance<T>\(object, string\)](#) , [IContext.BindInstanceAsCrossContext<TKey>\(object\)](#) ,
[IContext.BindInstanceAsCrossContext<TKey>\(object, object\)](#) ,
[IContext.BindSingleton<TKey, TValue>\(\)](#) , [IContext.BindSingleton<TKey, TValue>\(object\)](#) ,
[IContext.BindCrossContextAndSingleton<TKey, TValue>\(\)](#) ,
[IContext.BindCrossContextAndSingleton<TKey, TValue>\(object\)](#) , [IContext.Unbind<T>\(object\)](#) ,
[IContext.Inject\(object\)](#) , [IContext.Deject\(object\)](#) , [IDisposable.Dispose\(\)](#)  ,
[IInstanceProvider.GetInstance<T>\(\)](#) , [IInstanceProvider.GetInstance\(Type\)](#).

Interface IInstanceProvider

Namespace: [locContainer.Binder](#)

Assembly: locContainer.dll

```
public interface IInstanceProvider
```

Methods

GetInstance(Type)

Retrieve an Instance based on the key. ex. `injectionBinder.Get(typeof(ISomeInterface));`

```
object GetInstance(Type key)
```

Parameters

key [Type](#)

Returns

[object](#)

GetInstance<T>()

```
T GetInstance<T>()
```

Returns

T

Type Parameters

T