Foundation Injector

3/24/2015 | Nicholas Ventimiglia | AvariceOnline.com

The Injector is a static application service for resolving dependencies similar to GameObject.Find. Objects may be added to the injector as "Exports" and then these exports may be "Imported" into consumers worry free. The injector supports collections, abstract types, lazy late loading and the [Import] Annotation.

- Supports inheritance for resolving dependencies using an interface or base class.
- Supports string keys. resolve dependencies by a magic string.
- Supports GetFirst and GetAll. Get all of a exported dependency type.
- Supports [Import] annotation. Using this annotation unlocks the Import(object) helper method. This helper method allows for the importing of all decorated members automagically.
- Consumers may subscribe. Once subscribed imported members will be updated as items are added / removed from the injector. This allows for conducting all setup in Awake worry free.
- Use the [InjectorService] annotation on singleton services or ScriptableObjects and they will be loaded as needed for you.

Platforms

Desktop, Webplayer, Android, iOS, Windows Store

Use

Exporting

Exporting is the process of adding instances to the injector for use elsewhere.

```
// Optional, Add a string Key
[Export("MagicString")]

// This export may also resolved by asking for MyExport, MonoBehaviour or IMyExport
// in the GetAll or GetFirst method.
public class MyExport: MonoBehaviour, IMyExport
{
    void Awake()
    {
        // add this to the injector
```

```
            Injector.AddExport(this);
        }

        // Use this for initialization
        void OnDestroy()
        {
            // Remove this from the injector
            Injector.RemoveExport(this);
        }
}
```

Exporting Scriptable Objects

The InjectorInitialized annotation automates the initialization of application services. You may use it to make sure that services are loaded with the injector. To use simply decorate your services. Both CLR services and Scriptable objects are supported. While optional, it is suggested that you create a static accessor property (Singleton) as demonstrated below.

```
 // Initialized at runtime by the injector
[InjectorInitialized("ResourceName")]
public class MyScriptableObject : ScriptableObject{

        // optionally
        private static MyScriptableObject _instance;
        public static MyScriptableObject Instance
        {
            get { return _instance ?? (_instance = Create()); }
        }

        static MyScriptableObject Create()
        {
            return Resources.Load<MyScriptableObject>("ResourceName");
        }
}
```

Exporting Singltons

```
 // Initialized at runtime by the injector
[InjectorInitialized]
public class MyService {
    public static readonly MyService = new MyService();
}
```

Importing

Importing is the process of consuming an export.

```
    public class MyConsumer: MonoBehaviour
    {
        // Import using the Key
        [Import("MagicString")]
        public IMyExport Import1 { get; set; }
```

```
    // Import many using the Key
    [Import("MagicString")]
    public IEnumerable<IMyExport> Import2 { get; set; }

    // Import first by type
    [Import]
    public IMyExport Import3 { get; set; }

    // Import all by type
    [Import]
    public IEnumerable<IMyExport> Import4 { get; set; }

    void Awake()
    {
        // Asks the Injector to resolve dependencies now
        Injector.Import(this);

        // or

        // Asks the Injector to resolve dependencies now and later
        // Members will be updated if items are added / removed from export
        Injector.Subscribe(this);
    }

    void Start()
    {
        // Get import manually
        var manualImport = Injector.GetFirst<IMyExport>();
        // Get imports manually
        var manualImports = Injector.GetAll<IMyExport>();
    }

    void OnDestroy()
    {
          // Removes subscription
          Injector.Unsubscribe(this);
    }
}
```