(https://www.c-sharpcorner.com/UploadFile/dhananjaycoder/unity-framework/)

UNITY Framework

This article will explain how to use UNITY Framewrok to achieve DI and IOC design pattern.

-
- Dhananjay Kumar
- May 04 2009

UnityTesting.zip

This tutorial will explain step by step explanation of how to use UNITY Framework in code. UNITY Framework is a framework of Microsoft to achieve Inversion of Control Design Pattern.

To know more about, what is Inversion of Control and Dependency Pattern design pattern read my previous article here.

Two very good articles on same topic by Mr ShivPrasad Koirala could be read at following links

Design pattern Inversion of Control and DI and IOC

How to use UNITY Container

Step 1

Download UNITY application block from here

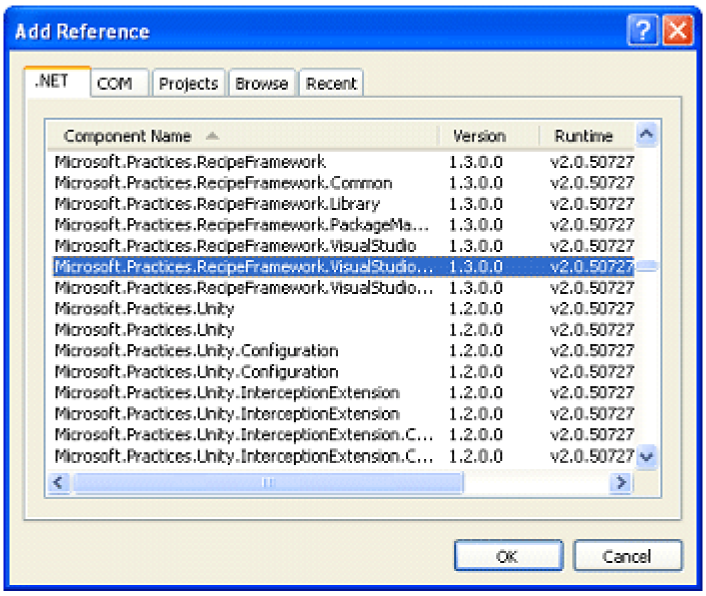
Step 2

Create New Project and Console application.

Step 3

Add reference to

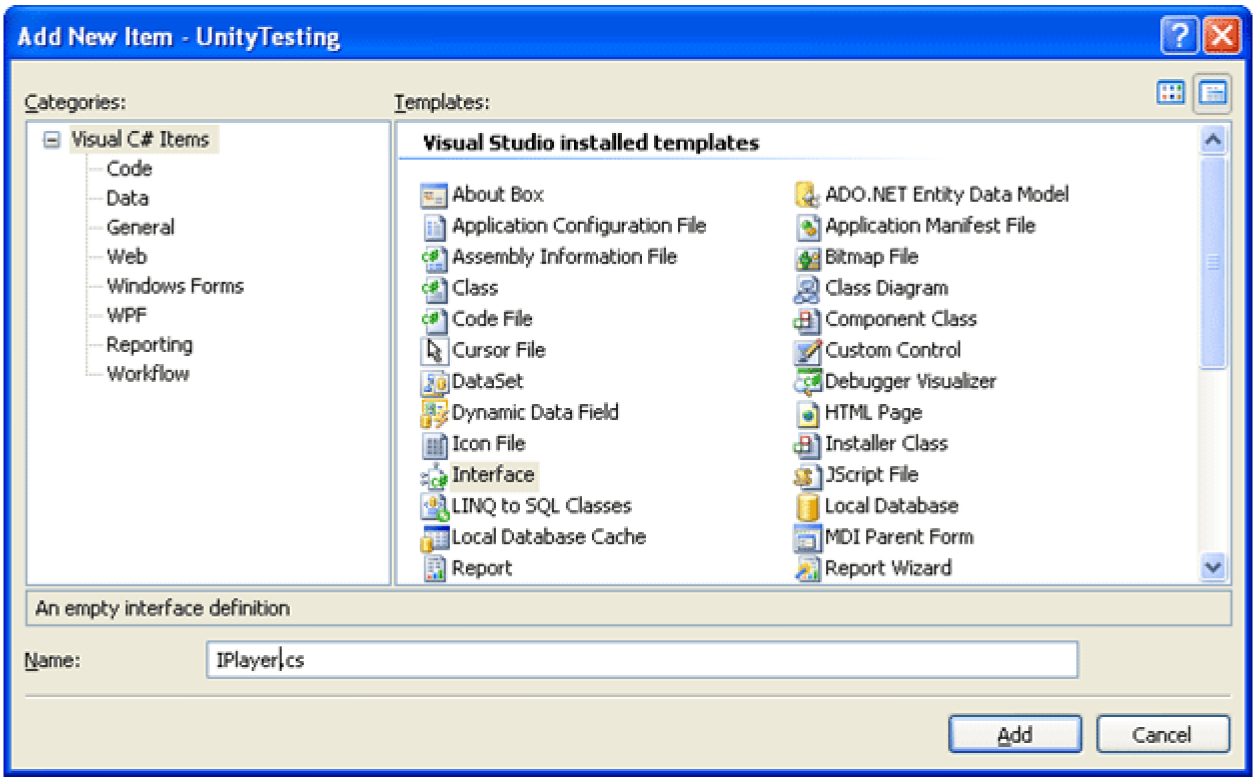
Microsoft.Practices.Unity, Microsoft.Practices.Unity.Configuration;



Component Name	Version	Runtime
Microsoft.Practices.RecipeFramework	1.3.0.0	v2.0.50727
Microsoft.Practices.RecipeFramework.Common	1.3.0.0	v2.0.50727
Microsoft.Practices.RecipeFramework.Library	1.3.0.0	v2.0.50727
Microsoft.Practices.RecipeFramework.PackageMa...	1.3.0.0	v2.0.50727
Microsoft.Practices.RecipeFramework.VisualStudio	1.3.0.0	v2.0.50727
Microsoft.Practices.RecipeFramework.VisualStudio...	1.3.0.0	v2.0.50727
Microsoft.Practices.RecipeFramework.VisualStudio...	1.3.0.0	v2.0.50727
Microsoft.Practices.Unity	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity.Configuration	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity.Configuration	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity.InterceptionExtension	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity.InterceptionExtension	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity.InterceptionExtension.C...	1.2.0.0	v2.0.50727
Microsoft.Practices.Unity.InterceptionExtension.C...	1.2.0.0	v2.0.50727

Step 4

Right click on project and add new item and then select interface. Give name of the interface as IPlayer.



An empty interface definition

Name: IPlayer.cs

Type below code in IPlayer.cs . This interface will registered as type to Unity Container.

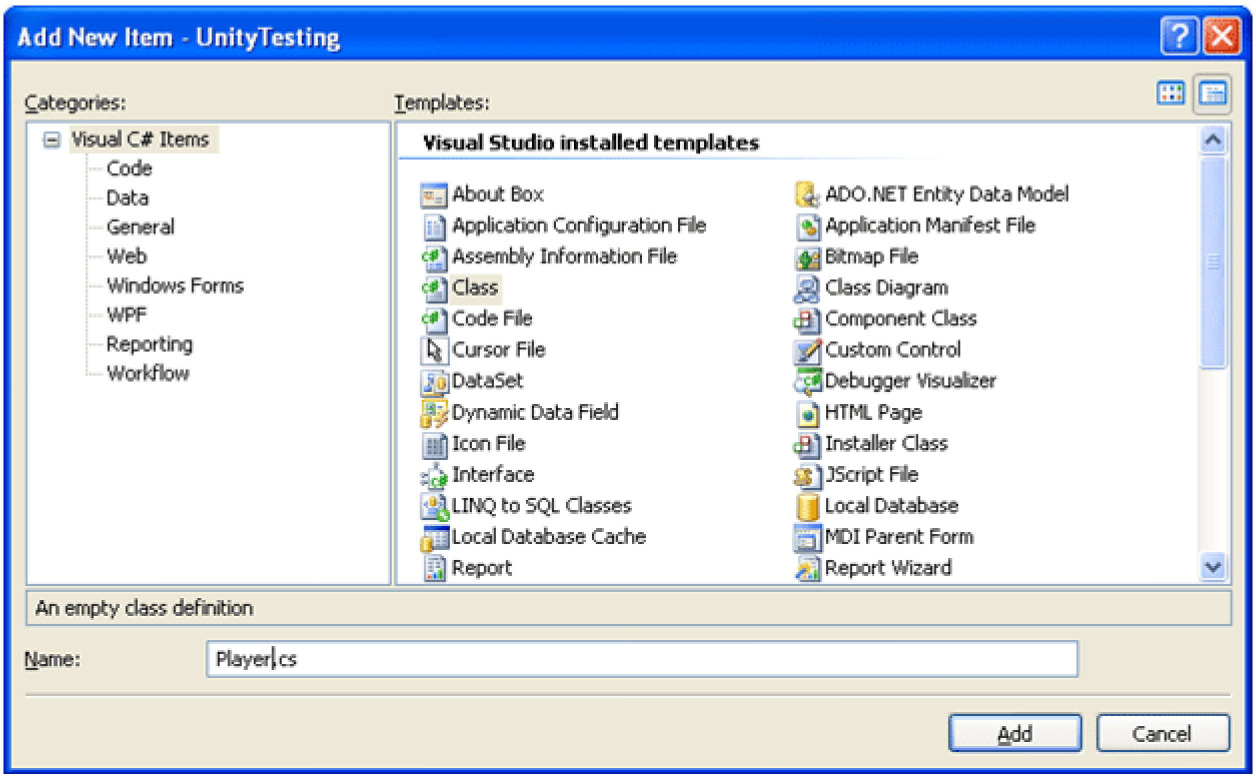
IPlayer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace UnityTesting
{
  public  interface IPlayer
   {
      string PlayerName
      {
         get;
         set;
      }
      string TeamName
      {
         get;
         set;
      }

        void DisplayDetails();
   }}
```

Step 5

Right click on project and add new class. Give name of the class as Player.cs

__Note__: Player class is implementing IPlayer interface.



Name: Player.cs

Type below code in Player.cs.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace UnityTesting
{
    public class Player:IPlayer
   {
        public Player()
        {
        }

        string name;
        string teamName;

        public string PlayerName
        {
            get
            {
```

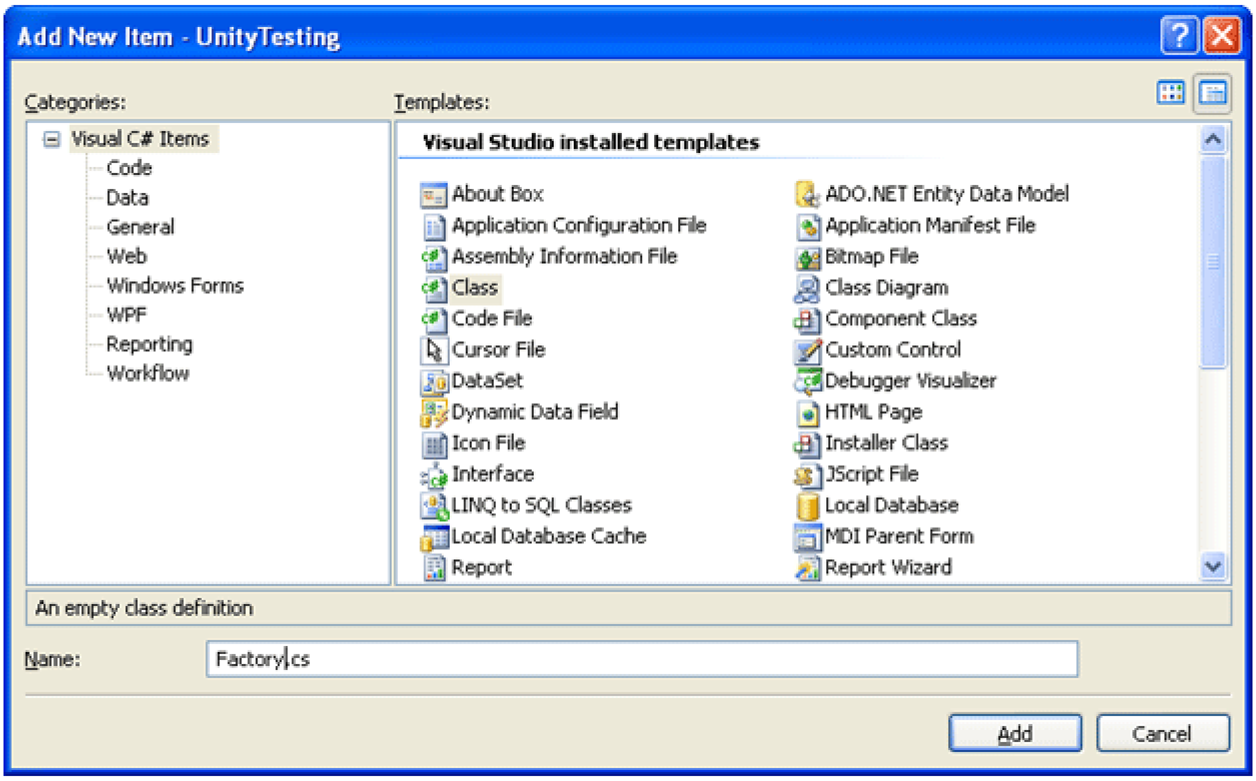
```
            return name;
        }
        set
        {
            name = value;
        }
    }
     public string TeamName
     {
        get
        {
            return teamName;
        }
        set
        {
            teamName = value;
        }
     }
    public void DisplayDetails()
     {
        Console.WriteLine("Player Name = \t" + name + " \tPlayer Team Name = \t" + teamName);
        //Console.WriteLine(p.PlayerName);
     }
  }
}
```

Step 6

Factory Class: This is a class, which is responsible for

1. Setting up the Unity Container. (Task A)
2. Registering interface as a type to the container. (Task B)
3. Returning instance of the registered type or Resolve an object by the type. (Task C)

Add a new class in project by right clicking and give name of the class as Factory. Factory is name here; give any name as per desire.



An empty class definition

Name: Factory.cs

In Factory.cs add namespaces

```
using Microsoft.Practices.Unity;
using Microsoft.Practices.Unity.Configuration;
```

Task A: Set Up Container

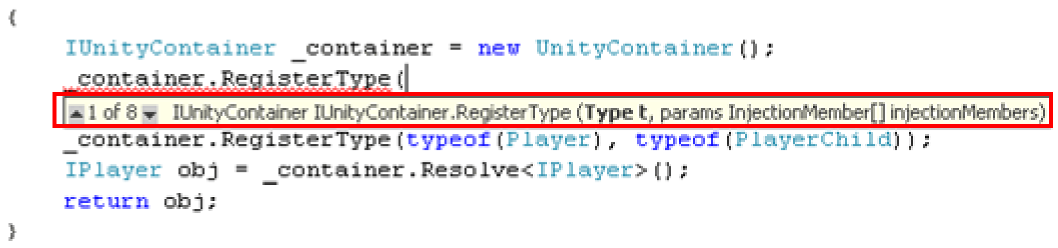
To set up container, create instance of IUnityContainer class.

```
IUnityContainer _container = new UnityContainer();
```

The above code will create an instance of container class.

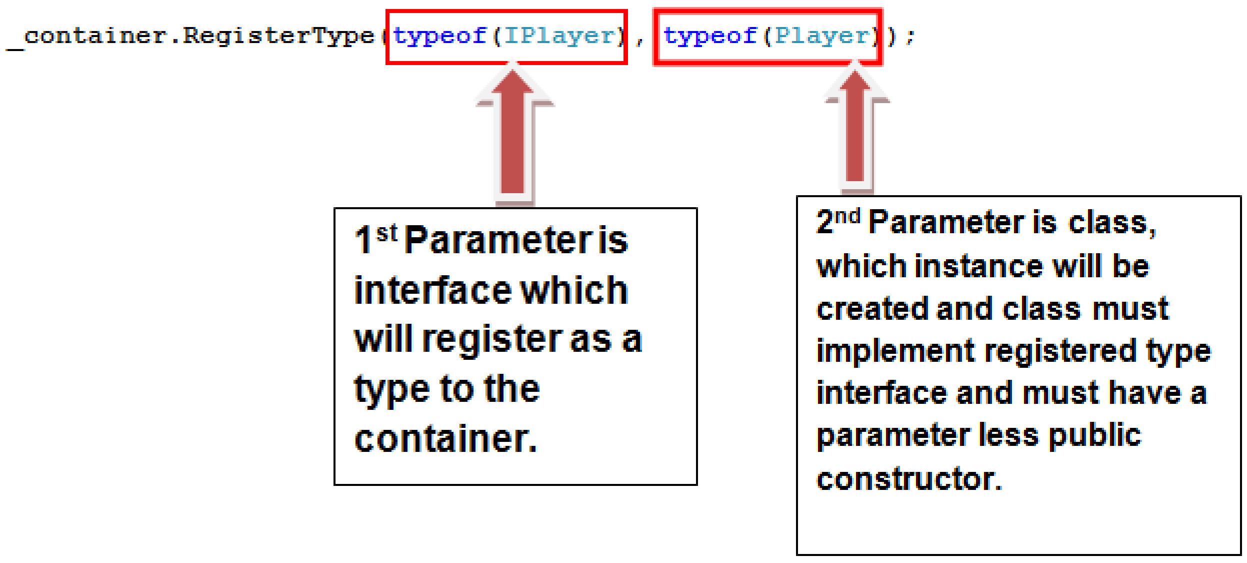
Task B: Register interface as a type

1. RegisterType method of container class is used to register a type into the container.
2. At the appropriate time, the container will build an instance of the type which is specified.
3. The lifetime of the object it builds will correspond to the life time specified in the parameters of the method.
4. If lifetime is not specified then the type will registered for a transient lifetime, which means that a new instance will be created on each call to Resolve. (It is method , which will be discussed later)
5. This method is overloaded with 8 different parameters.



```
{
    IUnityContainer _container = new UnityContainer();
    _container.RegisterType(
    ▲1 of 8▼  IUnityContainer IUnityContainer.RegisterType(Type t, params InjectionMember[] injectionMembers)
    _container.RegisterType(typeof(Player), typeof(PlayerChild));
    IPlayer obj = _container.Resolve<IPlayer>();
    return obj;
}
```

How to register a type?



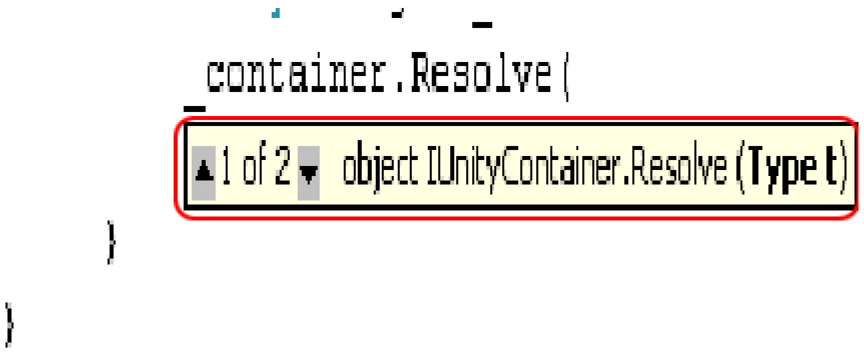
```
_container.RegisterType(typeof(IPlayer), typeof(Player));
```

1st Parameter is interface which will register as a type to the container.

2nd Parameter is class, which instance will be created and class must implement registered type interface and must have a parameter less public constructor.

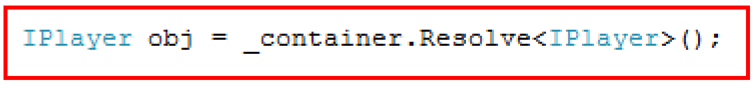
Task C: Resolve an Object by the type

1. To resolve an object by type , Resolve method will be used.
2. To retrieve object instances from the container.
3. When this method is used without specifying a value for the optional name property, the method returns the object registered with the default mapping.
4. This method is overloaded with two types of arguments. One takes only type and other argument takes type as well as string name of the registered type.

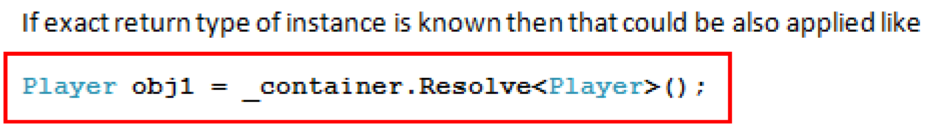


```
_container.Resolve(
    }
}
```

How to resolve instance of registered type?



```
IPlayer obj = _container.Resolve<IPlayer>();
```



If exact return type of instance is known then that could be also applied like

```
Player obj1 = _container.Resolve<Player>();
```

Putting altogether the Factory class.

Factory.cs

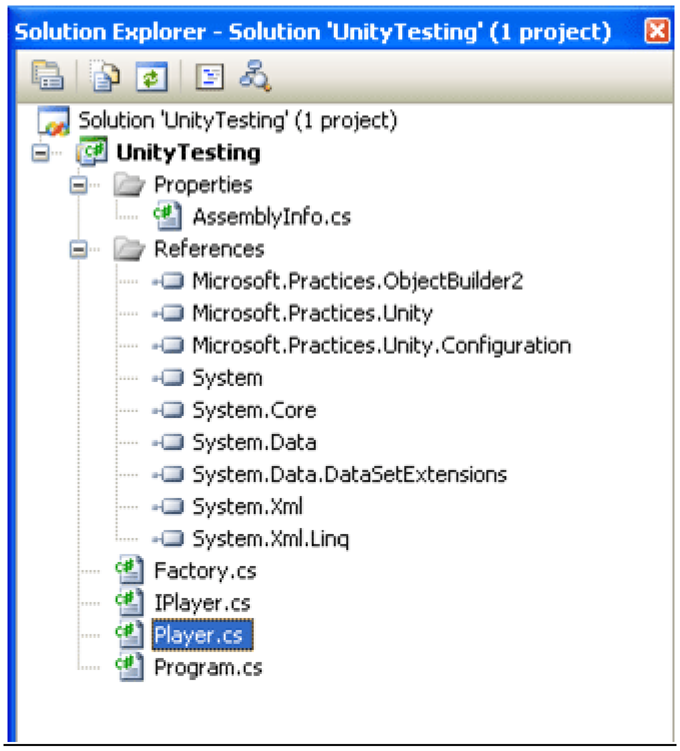
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Practices.Unity;
using Microsoft.Practices.Unity.Configuration;

namespace UnityTesting
{
    public class Factory
    {
        static public IPlayer CreateInstance()
        {
            IUnityContainer _container = new UnityContainer();
            _container.RegisterType(typeof(IPlayer), typeof(Player));
            IPlayer obj = _container.Resolve<IPlayer>();
            return obj;
        }
    }
}
```

Explanation:

CreateInstance() is a static method. This method will return instance of type registered to the container.

Step 7

Now solution explorer should look like as below


Solution Explorer - Solution 'UnityTesting' (1 project)

```
Solution 'UnityTesting' (1 project)
UnityTesting
    Properties
        AssemblyInfo.cs
    References
        Microsoft.Practices.ObjectBuilder2
        Microsoft.Practices.Unity
        Microsoft.Practices.Unity.Configuration
        System
        System.Core
        System.Data
        System.Data.DataSetExtensions
        System.Xml
        System.Xml.Linq
    Factory.cs
    IPlayer.cs
    Player.cs
    Program.cs
```

Step 8

Click on Program.cs and paste below code

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Practices.Unity;
using Microsoft.Practices.Unity.Configuration;
using System.Configuration;

namespace UnityTesting
{
    class Program
    {
        static void Main(string[] args)
        {
            IPlayer obj = Factory.CreateInstance();
            obj.PlayerName = " Sachin Tendulkar ";
            obj.TeamName = " India ";
            obj.DisplayDetails();
            IPlayer obj1 = Factory.CreateInstance();
            obj.PlayerName = " Shane warne";
            obj.TeamName = " Australia ";
            obj.DisplayDetails();
            Console.Read();
        }
    }
}
```

Explanation:

```
IPlayer obj = Factory.CreateInstance();
```

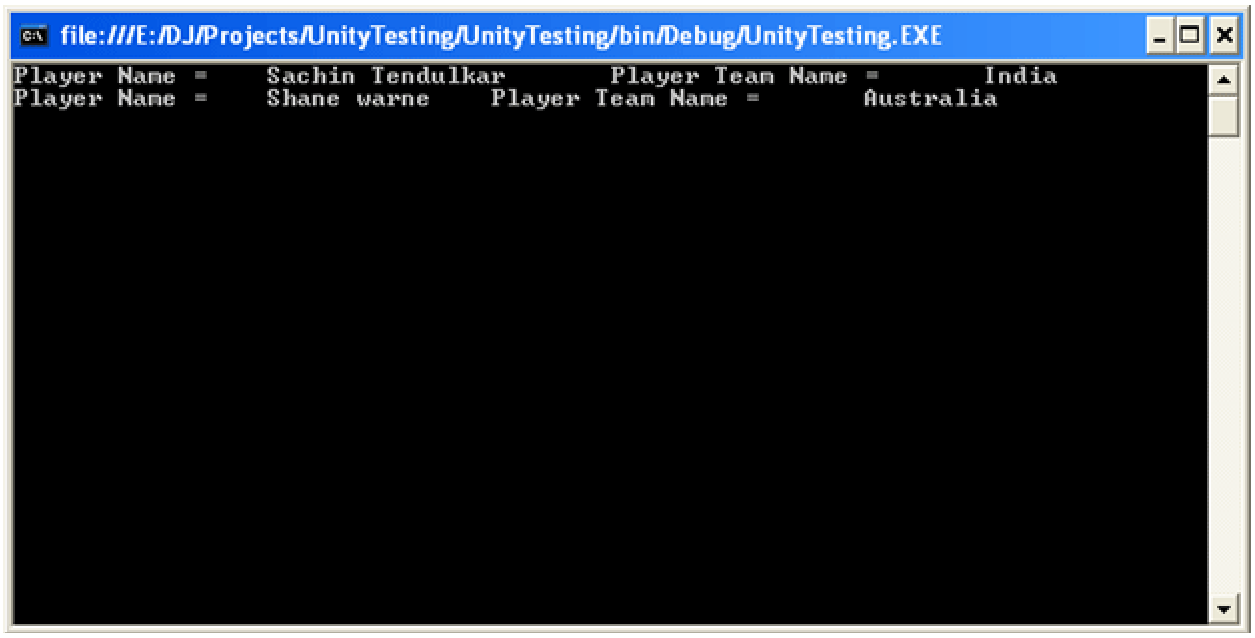
This code is used to create a instance if type Player using Unity Container.

```
obj.PlayerName = " Sachin Tendulkar ";
obj.TeamName = " India ";
obj.DisplayDetails();
```

Now properties and methods of class being called using instance of the class.

Here each time calling Resolve method is returning an instance of registered type, because time line is variant and not specified.

Output


file:///E:/DJ/Projects/UnityTesting/UnityTesting/bin/Debug/UnityTesting.EXE

```
Player Name =    Sachin Tendulkar        Player Team Name =       India
Player Name =    Shane warne     Player Team Name =       Australia
```

Future scope

In the next article, I will explain how

1. To register an existing object instance as an externally controlled instance
2. To register an existing object as a singleton instance

How to use zip file

Just download and use it.

Download UNITY Framework also. Add references as shown in above steps. Then run the code.

Happy Coding.