**<http://blog.csdn.net/mikemiller2/article/details/52251887>**

**C# 对轻量级(IoC Container)依赖注入Unity的使用**

转载 2016年08月19日 16:08:49

**概述**

Unity是一个轻量级的可扩展的依赖注入容器，支持构造函数，属性和方法调用注入。Unity可以处理那些从事基于组件的软件工程的开发人员所面对的问题。构建一个成功应用程序的关键是实现非常松散的耦合设计。松散耦合的应用程序更灵活，更易于维护。这样的程序也更容易在开发期间进行测试。你可以模拟对象，具有较强的具体依赖关系的垫片（轻量级模拟实现），如数据库连接，网络连接，ERP连接，和丰富的用户界面组件。例如，处理客户信息的对象可能依赖于其他对象访问的数据存储，验证信息，并检查该用户是否被授权执行更新。依赖注入技术，可确保客户类正确实例化和填充所有这些对象，尤其是在依赖可能是抽象的 。

**Unity 配置文件**复制代码

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

<section name="unity" type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection,Microsoft.Practices.Unity.Configuration"/>

</configSections>

<unity xmlns="http://schemas.microsoft.com/practices/2010/unity">

<container>

<!--register type="full class name,namespace"-->

<register type="UnityTest.ISqlHelper,UnityTest" mapTo="UnityTest.MysqlHelper,UnityTest">

<lifetime type="singleton"/>

</register>

</container>

</unity>

</configuration>

复制代码

需要注意的是type和mapTo的值，用逗号隔开两部分，一是类的全部，包括命名空间，二是命名空间。

那么，也有其他的方法，先设置好命名空间，那就直接写类名即可，这个就不说了。

这里是简单的配置，详细的的配置自行搜索。

**下载与引用**

到官方下载：[http://unity.codeplex.com/](http://unity.codeplex.com/" \t "_blank)

项目里引用dll

Microsoft.Practices.Unity.dll

Microsoft.Practices.Unity.Configuration.dll

**程序**

假设对数据库操作类进行更换，那先建立一个操作类的接口，具体实现留着派生的类。

操作类接口

复制代码

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace UnityTest

{

public interface ISqlHelper

{

string SqlConnection();

}

public interface IOtherHelper

{

string GetSqlConnection();

}

}

派生类一：Ms SQL Server

复制代码

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace UnityTest

{

public class MssqlHelper : ISqlHelper

{

public string SqlConnection()

{

return "this mssql.";

}

}

}

复制代码

派生类二：MySQL

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace UnityTest

{

public class MysqlHelper : ISqlHelper

{

public string SqlConnection()

{

return "this mysql.";

}

}

}

复制代码

其他类

复制代码

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace UnityTest

{

public class MyOtherHelper : IOtherHelper

{

ISqlHelper sql;

public MyOtherHelper(ISqlHelper sql)

{

this.sql = sql;

}

public string GetSqlConnection()

{

return this.sql.SqlConnection();

}

}

}

复制代码

主程序调用

复制代码

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Configuration;

using Microsoft.Practices.Unity;

using Microsoft.Practices.Unity.Configuration;

namespace UnityTest

{

class Program

{

static void Main(string[] args)

{

IUnityContainer mycontainer = new UnityContainer();

//已有对象实例的配置容器注册

// MysqlHelper d = new MysqlHelper();

//mycontainer.RegisterInstance<ISqlHelper>(d);

//类型的配置容器注册

//mycontainer.RegisterType<ISqlHelper, MysqlHelper>();

//配置文件注册

UnityConfigurationSection section = (UnityConfigurationSection)ConfigurationManager.GetSection("unity");

section.Configure(mycontainer);

//mycontainer.LoadConfiguration();

//调用依赖

ISqlHelper mysql = mycontainer.Resolve<ISqlHelper>();

Console.WriteLine(mysql.SqlConnection());

//构造函数注入

mycontainer.RegisterType<IOtherHelper, MyOtherHelper>();

IOtherHelper other = mycontainer.Resolve<IOtherHelper>();

Console.WriteLine(other.GetSqlConnection());

Console.ReadKey();

}

}

}