**ServiceProvider与ServiceScopeFactory**

ServiceProvider可以注册三种类型服务实例

Singleton：单例，整个应用中只保存一个实例

Scoped：域，每次站点请求，只生成一个实例

Transient：对于每个依赖，都生成一个实例

Singleton和Transient都好理解，但ServiceProvider是如何实现Scoped的呢？

这就需要ServiceScopeFactory

internal class ServiceScopeFactory : IServiceScopeFactory

{

    private readonly ServiceProvider \_provider;

    public ServiceScopeFactory(ServiceProvider provider)

    {

        \_provider = provider;

    }

    public IServiceScope CreateScope()

    {

        return new ServiceScope(new ServiceProvider(\_provider));

    }

}

ServiceScopeFactory被注测到ServiceProvider中，所以我们从ServiceProvider中可以获取ServiceScopeFactory

**ServiceScopeFactory与ServiceScope**

ServiceScope代表一个服务域（我们每次请求都会新建一个服务域）

每个ServiceScope都有一个ServiceProvider

ServiceScope从ServiceScopeFactory创建

public interface IServiceScopeFactory

{

    IServiceScope CreateScope();

}

public interface IServiceScope : IDisposable

{

    IServiceProvider ServiceProvider { get; }

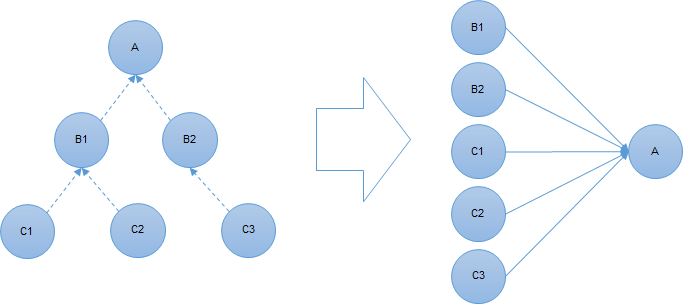
}

ServiceScopeFactory有包含ServiceProvider

ServiceScope也有包含ServiceProvider

这两个ServiceProvider不是同一个对象，而是具有父子关系的ServiceProvider

所有子ServiceProvider或孙ServiceProvider都指向根ServiceProvider，而不是他们的父ServiceProvider



基于右边的图，B1,B2,C1,C2,C3均代表一个域

我们重新定义ServiceProvider三种生命周期

1. Singleton：ServiceProvider创建的服务实例保存在作为根节点的ServiceProvider上
2. Scoped：ServiceProvider创建的服务实例由自己保存
3. Transient：针对每一次服务提供请求，ServiceProvider总是创建一个新的服务实例。

**通过如下示例，理解三种类型的服务实例保存在哪**

class Program

{

    static void Main(string[] args)

    {

        IServiceProvider root = new ServiceCollection()

            .AddTransient<IFoo, Foo>()

            .AddScoped<IBar, Bar>()

            .AddSingleton<IBaz, Baz>()

            .BuildServiceProvider();

        IServiceProvider child1 = root.GetService<IServiceScopeFactory>().CreateScope().ServiceProvider;

        IServiceProvider child2 = root.GetService<IServiceScopeFactory>().CreateScope().ServiceProvider;

        // IFoo为Transient，所以为false

        ReferenceEquals(root.GetService<IFoo>(), root.GetService<IFoo>());

        // IBar为Scoped，所以为true

        ReferenceEquals(child1.GetService<IBar>(), child1.GetService<IBar>());

        // child1，child2是两个不同的域，所以为false

        ReferenceEquals(child1.GetService<IBar>(), child2.GetService<IBar>());

        // IBaz是Singleton，所以为true

        ReferenceEquals(child1.GetService<IBaz>(), child2.GetService<IBaz>());

    }

}

**ServiceProvider是如何回收实例的服务实例？**

当域ServiceProvider被销毁时，由该ServiceProvider所创建的服务实例都会被销毁，即销毁该ServiceProvider的Scoped和Transient类型的服务实例

当根ServiceProvider被销毁时，Singleton类型的服务实例被销毁

如何销毁ServiceProvider

当没有任何对象引用ServiceProvider时，ServiceProvider会被GC销毁，即我们只需要调用IServiceScope的Dispose方法（会解除对ServiceProvider的引用）便可销毁ServiceProvider

于是安全的销毁ServiceProvider代码如下：

public void DoWork(IServiceProvider serviceProvider)

{

    using (IServiceScope serviceScope = serviceProvider.GetService<IServiceScopeFactory>().CreateScope())

    {

        IFoobar foobar = serviceScope.ServiceProvider.GetService<IFoobar>();

        ...

    }

}