StrangeIoc的理解

1. 什么是StrangeIoc

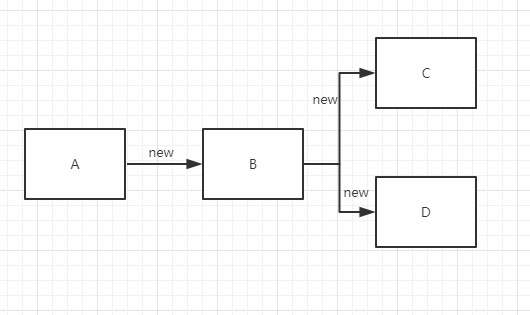
说的比较多的一句话是，一个针对Unity3，超级轻量、高度可扩展的、IOC的框架。

已知公司内部使用StrangeIOC框架的游戏有：腾讯桌球、欢乐麻将、植物大战僵尸Online，数据来自KM。

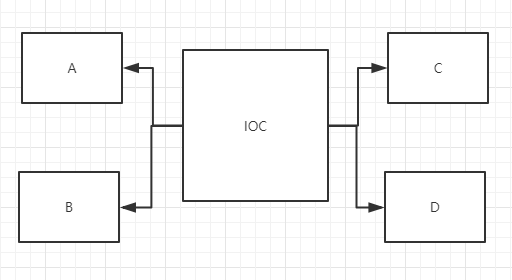
1. 什么是IOC

Ioc：Inversion of Control （控制反转）

为什么有这个?先看下原来的对象管理方式，面向对象开发过程会有很多对象。当A需要B时，会new一个B，当B需要C和D时同样会new出来，这样所有对象都是耦合比较重的。



通过IOC创建的对象



所有实体的管理都在IOC这个容器中完成。那么A是怎么获得B这个对象的呢？

这里有另一个概念DI(Dependency Injection依赖注入)。

通过注入的方法把对象设置进来。

我们看下代码中我们要怎么做：

Class A {

[Inject]

public B b {get;set;}

}

Class B {

[Inject]

public C c {get;set;}

[Inject]

public D d {get;set;}

}

这句话表示我需要一个B的对象，[Inject]标签表示需要通过注入的方式来赋值，实际上在这个脚本创建出A对象的同时，B对象就会在IOC容器中去找是否已经创建成功，如果创建成功就会赋值到变量上，如果没创建就会先创建再赋值。

看下IOC做了什么呢？

实际上是个容器，所有的实例对象都在IOC这个容器中产生。

1. 框架介绍

StrangeIoc采用MVCS结构，Model、View、Controller、Services。

有一个MVCSContext脚本实际对应的就是IOC对象。

View:包含View和Mediator。

View主要处理界面相关逻辑，提供修改界面接口给Mediator。

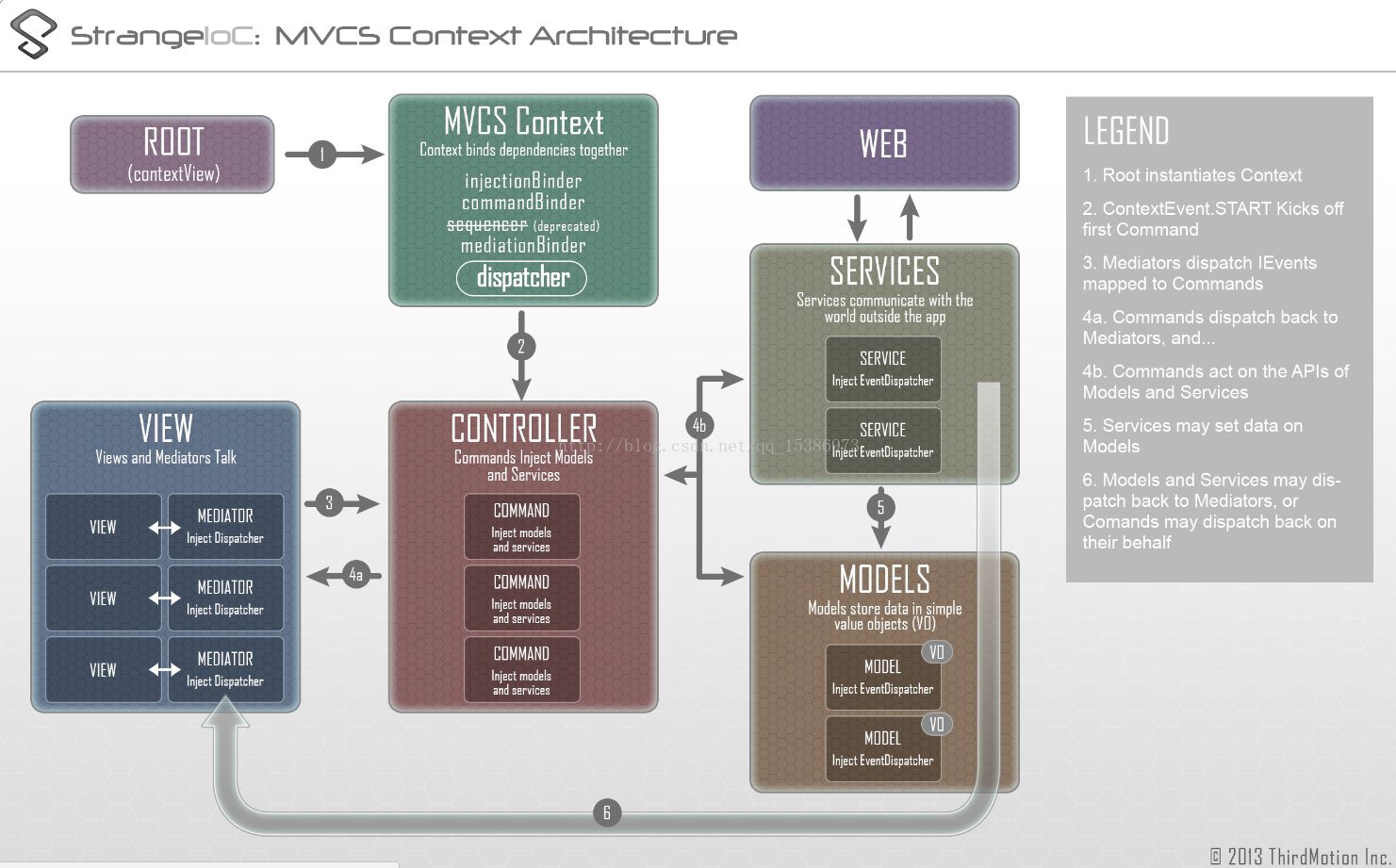
Mediator与Controller之间通过事件交互，并调用View的界面修改接口。

Model:数据层

Services:网络层

Controller:持有Model、Services。与Mediator通过事件交互。

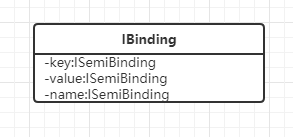
附一张官方介绍图：



1. 核心类介绍

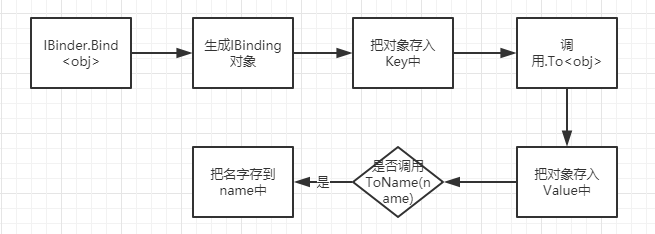
IBinder 对外的接口

IBinding 内部绑定实体类 有3个封装的数组对象组成，分别为Key，Value，Name数组



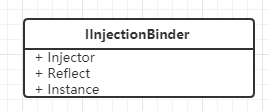
一个完整的绑定流程:

injectionBinder.Bind<IExampleModel>().To<ExampleModel>();



DI相关的接口

IInjectionBinder



IInjector

如果对象没有初始化在这里也进行实例生成

获取Reflect结果复赋值到对应的Inject标签的值上

IReflectionBinder

1.找到[Constructor]标签的构造方法

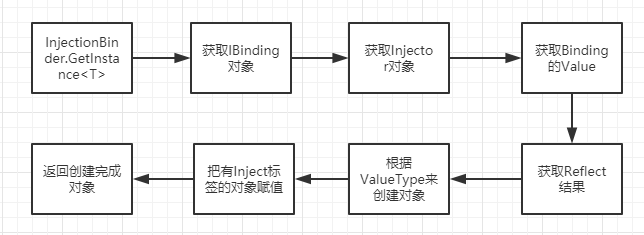
2.找到[Inject]标签的属性

    1.找到私有Inject标签会报错提醒

2.保存对应名字，类型，属性到数组中

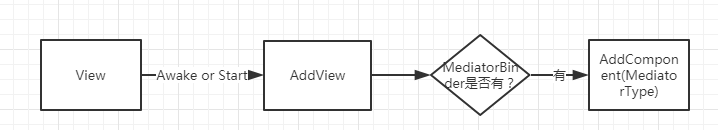
一个完整的初始化流程

injectionBinder.GetInstance<ICommandBinder>();



View与Mediator初始化流程

mediationBinder.Bind<MainView>().To<MainMediator>();



1. 用例

一艘飞船，需要用到武器，这个武器到底是什么武器呢？

class Spaceship : ISpaceship

{

public void input(float angle, float velocity)

{

//do stuff here

}

[Inject] //<----- The magic word!

public IWeapon weapon{get;set;}

}

injectionBinder.Bind<ISpaceship>().To<Spaceship>();

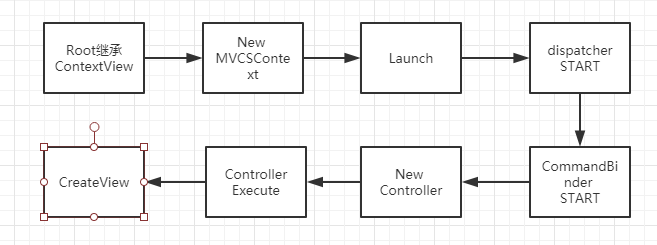
injectionBinder.Bind<IWeapon>().To<PhaserGun>();（激光枪）

injectionBinder.Bind<IWeapon>().To<SquirtCannon>();（散弹炮）

这样就能很方便的只需要Value对应不一样的值就可以很快的初始化不同的武器了。

当然重复绑定是不允许的，先要UnBind，然后重新Bind就可以了。

1. 框架启动



会抛出固定的Start事件。支持事件方式启动框架。

1. 总结

好处：

1. 模块拆分比较清晰
2. 介绍对象之间的耦合，不需要强绑定关系
3. 对后期扩展非常方便、灵活

坏处：

1. 框架理解学习成本
2. 对象创建用到了反射，内部已经做了优化只有第一次时会反射后面缓存了。
3. 框架引入的常规问题，简单问题复杂化，一个很简单界面也需要用到MVCS。

学习框架对于结构、思想有很好的帮助。在开发过程中也能提供很好的帮助。