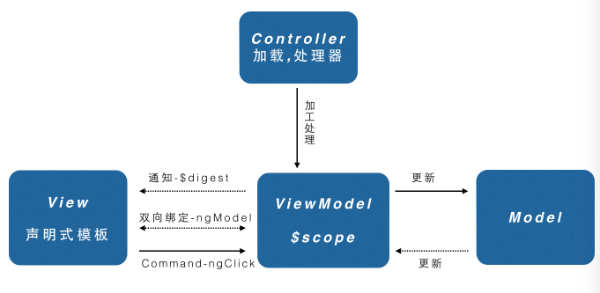
Angular mvvm model-view-viewModel (模型-视图-视图模型)

￥scope可以被认为是ViewModel ，而controller则是装饰、加工这个ViewModel的js函数。

用服务代替指令，让指令尽可能保持简约， 模型驱动

1. 低耦合：View可以独立于Model变化和修改，同一个ViewModel可以被多个View复用；并且可以做到View和Model的变化互不影响；
2. 可重用性：可以把一些视图的逻辑放在ViewModel，让多个View复用；
3. 独立开发：开发人员可以专注与业务逻辑和数据的开发（ViewModel），界面设计人员可以专注于UI(View)的设计；
4. 可测试性：清晰的View分层，使得针对表现层业务逻辑的测试更容易，更简单。



1. View：它专注于界面的显示和渲染，在angular中则是包含一堆声明式Directive的视图模板。
2. ViewModel：它是View和Model的粘合体，负责View和Model的交互和协作，它负责给View提供显示的数据，以及提供了View中Command事件操作Model的途径；在angular中$scope对象充当了这个ViewModel的角色；
3. Model：它是与应用程序的业务逻辑相关的数据的封装载体，它是业务领域的对象，Model并不关心会被如何显示或操作，所以模型也不会包含任何界面显示相关的逻辑。在web页面中，大部分Model都是来自Ajax的服务端返回数据或者是全局的配置对象；而angular中的service则是封装和处理这些与Model相关的业务逻辑的场所，这类的业务服务是可以被多个Controller或者其他service复用的领域服务。
4. Controller：这并不是MVVM模式的核心元素，但它负责ViewModel对象的初始化，它将组合一个或者多个service来获取业务领域Model放在ViewModel对象上，使得应用界面在启动加载的时候达到一种可用的状态。

创建一个scope，找到一个控制然后用控制器对scope进行初始化，最后把scope绑定到视图，把生成的live dom渲染出来。

