三大框架和MVC架构的关系

软件的架构是系统的一个草图、阐述了各个组件之间的通讯、划分层次、一旦系统开始详细设计、架构蓝图就很难甚至无法更改、是由软件架构师从无到有设计出来的。三层架构：一种设计软件架构的思想 ，把软件上从逻辑上分为、表示层（UI）业务逻辑层（BLL）数据访问层（DAL），表现层（UI）：通俗讲就是展现给用户的界面，即用户在使用一个系统的时候他的所见所得。业务逻辑层（BLL）：针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。数据访问层（DAL）：该层所做事务直接操作数据库，针对数据的增添、删除、修改、更新、查找等。目的：低耦合、高内聚、各司其职、达到易更换、修改、可以分散部署、编码。软件框架是在一定领域内、别人已经对这个领域制作软件所需的基础架构功能、进行了总结、做出了有代码实体的软件框架结构、如果要制作这一领域的软件、可以在别人写好的框架上、继续设计、编写自己的软件、骨头架上填肉、框架有一定的局限性。MVC（框架），英文  Model View Controller、是针对Web开发、已经写好有代码的框架、分别为M 模型(model)－V视图(view)－C控制器(controller)三部分；目的：模型和视图分离开、使得一个模型可被多个视图使用、简单说就是同样的一个网站、用手机的视图（界面）和电脑的视图、可以共用一个模型。

MVC主要用于表现层，3层主要用于体系架构，3层一般是表现层、中间层、数据层，其中表现层又可以分成M、V、C，(Model View Controller)模型－视图－控制器，MVC模式是GUI界面开发的指导模式，基于表现层分离的思想把程序分为三大部分：Model-View-Controller，呈三角形结构。Model是指数据以及应用程序逻辑，View是指 Model的视图，也就是用户界面。这两者都很好理解，关键点在于Controller的角色以及三者之间的关系。在MVC模式中，Controller和View同属于表现层，通常成对出现。Controller被设计为处理用户交互的逻辑。一个通常的误解是认为Controller负责处理View和Model的交互，而实际上View和Model之间是可以直接通信的。由于用户的交互通常会涉及到Model的改变和View的更新，所以这些可以认为是Controller的副作用。MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开，它们各自处理自己的任务。而在N层架构中，一般还会有一个Model层，用来与数据库的表相对应，也就是所谓O/RM中的O。这个Model可能是POCO，也可能是包含一些验证逻辑的实体类，一般也不包含数据读取。进行数据读取的是数据访问层。而作为UI层的MVC一般不直接操作数据访问层，中间会有一个业务逻辑层封装业务逻辑、调用数据访问层。UI层（Controller）通过业务逻辑层来得到数据（Model），并进行封装（ViewModel），然后选择相应的View。MVC本来是存在于Desktop程序中的，M是指数据模型，V是指用户界面，C则是控制器。使用MVC的目的是将M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据你可以分别用柱状图、饼图来表示。C存在的目的则是确保M和V的同步，一旦M改变，V应该同步更新。MVC如何工作MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。视图V视图是用户看到并与之交互的界面。对老式的Web应用程序来说，视图就是由HTML元素组成的界面，在新式的Web应用程序中，HTML依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括Macromedia Flash和象XHTML，XML/XSL，WML等一些标识语言和Web services.如何处理应用程序的界面变得越来越有挑战性。MVC一个大的好处是它能为你的应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。模型M模型表示企业数据和业务规则。在MVC的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。控制器C控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。模型Model 模型是应用程序的主体部分。模型表示业务数据，或者业务逻辑。 实现具体的业务逻辑、状态管理的功能。视图View 视图是应用程序中用户界面相关的部分，是用户看到并与之交互的界面。 就是与用户实现交互的页面，通常实现数据的输入和输出功能。控制器controller 控制器工作就是根据用户的输入，控制用户界面数据显示和更新model对象状态。起到控制整个业务流程的作用，实现View层跟Model层的协同工作。

三层架构指：表现层（显示层） 业务逻辑层 数据访问层（持久化）如果大家非要“生搬硬套”把它和MVC扯上关系话那我就只能在这里”强扭这个瓜”了即：V 3层架构中”表现层”aspx页面对应MVC中View（继承的类不一样）C 三层架构中”表现层”的aspx.cs页面（类）对应MVC中的Controller，理解这一点并不难，大家想一想我们以前写过的 Redirect，当然它本身就是跳转了一些链接页面，而MVC中的Controller要做的更爽，它控制并显示输出了一个视图。即然所起到的作用都是对业务流程和显示信息的控制，只不过是实现手段不同而已。M 3层架构中业务逻辑层和数据访问层对应MVC中Model（必定View和Controller已找到“婆家”剩下Model只能是业务逻辑层和数据访问层了）。三层架构的分层模式是典型的上下关系，上层依赖于下层。但MVC作为表现模式是不存在上下关系的，而是相互协作关系。即使将MVC当作架构模式，也不是分层模式。MVC和三层架构基本没有可比性，是应用于不同领域的技术。

MVC和三层架构的相同点是分层，解耦。他们相同的设计理念就是：把视图设计与数据持久化进行分离，从而降低耦合性，易于扩展，提高团队开发效率。只所以说MVC和三层毫无关系，是因为它们二者使用范围不同：三层可以应用于任何语言、任何技术的应用程序；而MVC只是为了解决BS应用程序视图层各部分的耦合关系。它们互不冲突，可以同时存在，也可根据情况使用其中一种。三层架构中，DAL、BLL、WEB层各司其职，意在职责分离。MVC是 Model-View-Controller，严格说这三个加起来以后才是三层架构中的WEB层，也就是说，MVC把三层架构中的WEB层再度进行了分化，分成了控制器、视图、实体三个部分，控制器完成页面逻辑，通过实体来与界面层完成通话；而C层直接与三层中的BLL进行对话。