**什么是MVC，对MVC的理解？**

MVC是一种设计模式，它强制的将应用程序的输入、处理、和输出分开。

使用了MVC的应用程序被分为3个核心部件:试图(View)、模型(Model)、控制器(Controller)。它们各司其职，既分工明确又相互合作。

首先视图接受用户输入请求，然后将请求传递给控制器，控制器再调用某个模型来处理用户的请求，在控制器的控制下，再将处理后的结果交给某个视图进行格式化输出给用户.这是经典的MVC设计执行的基本流程。MVC不仅实现了功能模块和显示模块的分离，同时它还提高了系统的可维护性、可扩展性和组件的可复用性，是一个优秀的创建软件的途径。

J2EE把MVC引入了web领域，并在此基础上架构出了一套称为Model2的体系。Model2体系及jsp作为视图显示输出和接受用户输入数据，Servlet作为控制器处理用户请求并根据用户请求选择相应的jsp返回给用户，javaBean作为模型组件用作业务逻辑处理、管理数据的存储和一致性。

**MVC模式的缺点：**增加了程序源码的复杂性。

**MVC分层的优点**

开发人员可以只关注整个结构中的其中某一层；

可以很容易的用新的实现来替换原有层次的实现；

可以降低层与层之间的依赖；

有利于标准化；

利于各层逻辑的复用。

结构更加的明确

在后期维护的时候，极大地降低了维护成本和维护时间

**MVC分层的缺点**

降低了系统的性能。这是不言而喻的。如果不采用分层式结构，很多业务可以直接造访数据库，以此获取相应的数据，如今却必须通过中间层来完成。

有时会导致级联的修改。这种修改尤其体现在自上而下的方向。如果在表示层中需要增加一个功能，为保证其设计符合分层式结构，可能需要在相应的业务逻辑层和数据访问层中都增加相应的代码。

增加了开发成本。

**MVC分层**

视图层view：用于展示数据，与用户进行交互。

控制层controller：用于分发控制到来的请求,并调用模型层与数据库进行交互,以及将数据返回给视图层展示。

模型层model：数据模型，它与数据库进行交互，进行CURD操作。

当请求来了，controller就会将相应的请求分发到相应的service层，在service层中再调用dao层进行数据库交互。这里的dao层其实就是之前的model层，封装了对数据库的操作。这样一来，就把业务处理逻辑从controller中分离出来，从而实现了解耦。

**三层架构**

三层架构(3-tier application) 通常意义上的三层架构就是将整个业务应用划分为：表现层（UI）、业务逻辑层（BLL）、数据访问层（DAL）。区分层次的目的即为了“高内聚，低耦合”的思想。

1、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。

2、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。

3、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。

**三层架构的实现：**

表现层：UI层只能作为一个外壳，不能包含任何的逻辑处理过程。

业务逻辑层：是系统架构中体现核心价值的部分，主要负责对数据层的操作。也就是说把一些数据层的操作进行组合。

数据访问层：该层所做事务直接操作数据库，针对数据的增添、删除、修改、查找等具体为业务逻辑层或表示层提供数据服务。

**三层与MVC的区别：**

MVC是 Model-View-Controller，严格说这三个加起来以后才是三层架构中的UI层，也就是说，MVC把三层架构中的UI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。

mvc可以是三层中的一个表现层框架，属于表现层。三层和mvc可以共存。

三层是基于业务逻辑来分的，而mvc是基于页面来分的。

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是表现模式（Presentation Pattern）

三层架构是典型的架构模式（Architecture Pattern）

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。