**三层架构**

**三层架构的详解：**

--> 标准三层：

1. UI层：UI层负责提供给用户所使用的UI界面，它调用BLL层并从BLL层中获取返回的结果
2. 业务逻辑层（BLL）：BLL层负责业务上的逻辑处理，它调用DAL层并从DAL层中获取返回的结果
3. 数据访问层（DAL）：DAL层负责与数据库进行交互，它访问数据库并从数据库中获取返回的结果

--> 其他层：

1. 数据模型层（Model）：Model负责提供一些数据上的载体，该层能够为三层之间的调用与返回提供数据上的载体
2. 通用层（Common）：Common负责提供一些工具类，三层中的任何一层都能够去调用它，没有绝对的依赖关

**标准三层架构所存在的问题：**

使用标准三层架构的时候存在一个非常严重的问题，虽然我们是通过三层架构来把数据访问层、业务逻辑层和UI展现层进行了解耦分离，但是实际上业务层和数据访问层之间还是存在着强耦合性，举个例子：业务逻辑层是需要直接访问数据访问层获取数据库中的数据的，所以在业务逻辑层中就需要直接性的创建一个与之所对应业务的数据访问层的实例，而项目的初期使用的是SqlServer的数据库，当到了后期的时候项目需要换成MySql数据库，因为我们不能够删除原有用于SqlServer数据库的数据访问层的解决方案，所以关于MySql数据库的数据访问层我们也要随之建立一个所对应的数据访问层的解决方案，这时候就出现问题了，如果我们要更换MySql数据库的话，那与之所对应的业务逻辑层中所创建的关于SqlServer数据库的数据访问层实例都需要改成MySql数据库的数据库访问层实例，强耦合性就这样体现出来了

**三层图**

