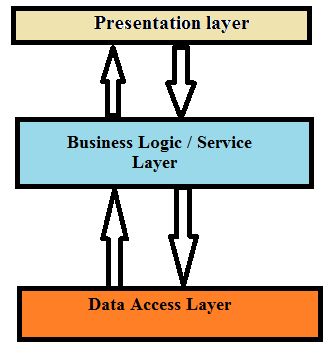
**Layered Architecture**  
  
When the various components in a system are organized systematically we call it a system architecture. The architecture is the enterprise-scale division of a system into layers or tiers, each having responsibility for a major part of the system and with as little direct influence on other layers.  
  
**One, two, three and n-tier applications**  
  
There are plenty of ways for a system to be split into multiple logical tiers.  
  
**One-tier applications**  
Single-tier applications are basically simple standalone programs.  
  
It's not necessary to consider network communication and the risk of network failure  
  
in these cases since they do not require network access.  
  
Since all the data resides within the same application, these programs do not focus on synchronization of data. When separating the tiers physically the application is slower since the communication over the network will result in a loss of performance, therefore one-tier applications certainly have better performance.  
  
**Two-tier applications**  
  
A two-tier application, in comparison to the one-tier application as described, does not combine all functions into a single process but into separate functions. For example a chat application. This kind of application contains two separated tiers, client and a server.  
  
The client has the responsibility of capturing user input and displaying the actual messages.  
  
The server is responsible of the communication between the people that use the chat client.  
  
**Three-tier applications**  
  
A three-tier application adds another tier to the previous mentioned chat application, this could be in the form of a database. One could think of a three-tier application as a dynamic web application, which has a user interface, business logic, services and a database each placed in different tiers, as illustrated in Figure 1. As mentioned in the previous section a two-tier architecture separates the user interface from the business logic, in the same way the three tier architecture separates the database from the business logic.  
  
  
Figure 1 - Three-tier application  
  
**N-tier applications**  
  
A logical n-tier application is an application where all logical parts are separated into discrete classes. In a typical business application, this generally involves a Presentation Layer, Business Logic Layer and a Data Access Layer. This separation makes the application easier to maintain. The advantages of this architecture are that all business rules are centralized that make them easy to create, use and re-use. The data access is also centralized, that has the same advantage as the centralization of the business rules. Centralizing the data access routines are also good when it comes to maintenance since changes must only be implemented in one location. There are really not that many disadvantages of this kind of architecture, however, it takes a bit longer to get up and running since several separate components need to be developed, complicated to understand.

