Tier

Tier: Think of a tier as a logical separation of components in an application or a service. And when I say separation, I mean physical separation at the component level, not the code level.

What do I mean by components?

- *Database*
- *Backend application server*
- *User interface*
- *Messaging*
- *Caching*

Single Tier:

A single-tier application is an application where the user interface, backend business logic & the database all reside in the same machine.

优点

There are no data requests to the backend server every now and then, which would make the user experience slow. In single-tier apps, the data is easily & quickly available since it is located in the same machine.

缺点

- The business has no control over the software once it's shipped. 无法release new version
- The code can be easily reversed engineered
- Also, the applications' performance & the look and feel can get inconsistent as it largely depends on the configuration of the user's machine.

Two Tier Application:

A Two-tier application involves a client and a server. The client would contain the user interface & the business logic in one machine.

And the backend server would be the database running on a different machine. The database server is hosted by the business & has control over it.

也就是client包括了实际的逻辑和UI，business包含了database

举例子：

1. 元神这一类的游戏。Game files are pretty heavy, they get downloaded on the client just once when the user uses the application for the first time. Moreover, they make the network calls only to keep the game state persistent.

Three Tier Application:

Mostly popular.

So, if we take the example of a simple blog, the user interface would be written using Html, JavaScript, CSS, the backend application logic would run on a server like Apache & the database would be MySQL. A three-tier architecture works best for simple use cases.

N Tier (Distributed Applications):

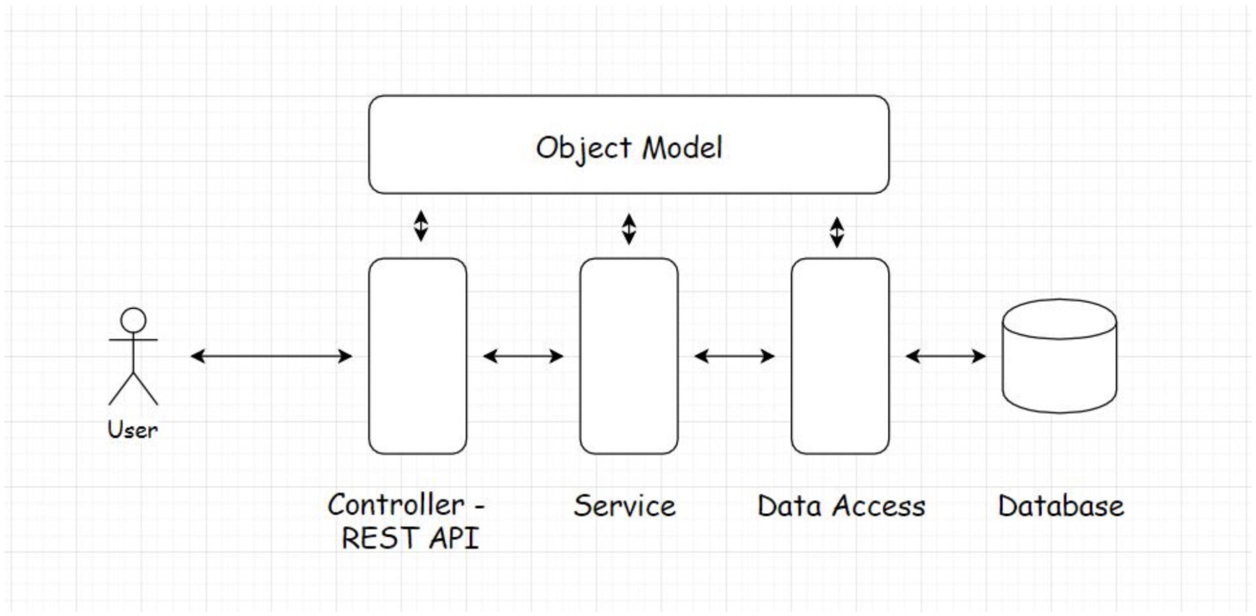
还包括了:

- *Cache*
- *Message queues for asynchronous behaviour*
- *Load balancers*
- *Search servers for searching through massive amounts of data*
- *Components involved in processing massive amounts of data*
- *Components running heterogeneous tech commonly known as web services* etc.

Single Responsibility Principle simply means giving one, just one responsibility to a component & letting it execute it with perfection. Be it saving data, running the application logic or ensuring the delivery of the messages throughout the system.

举例子： For instance, when upgrading a database server. Like when installing a new OS or a patch, it wouldn't impact the other components of the service running & even if something amiss happens during the OS installation process, just the database component would go down. The application as a whole would still be up & would only impact the features requiring the database.

> Note: Don't confuse tiers with the layers of the application. Some prefer to use them interchangeably. But in the industry layers of an application typically means the user interface layer, business layer, service layer, or the data access layer.


Object Model
User
Controller - REST API
Service
Data Access
Database