***1.PO: (persistent object)****, persistent object****2. VO: (value object)****, value object****3. DAO: (Data Access Objects)****, data access object interface****4. BO: (Business Object)****, business object layer  
5.****DTO****Data Transfer Object data transfer object  
6.****POJO****:****(Plain Old Java Objects)****, simple Java objects*

**1.PO: (persistent object), persistent object**

PO can be thought of as a Java object that is linked to a database table. Hibernate is an excellent choice for generating PO.

**2. VO: (value object), value object**

VO, like PO, is typically used for data transfer between business layers and only contains data. However, it should be an abstract business object that can or cannot correspond to the table, depending on the needs of the business.

PO is only used in the data layer, whereas VO is used in the business logic layer and the presentation layer. Each layer operates its own data objects, which can reduce coupling between layers and make future system maintenance and expansion easier.

**3. DAO: (Data Access Objects), data access object interface**

DAO stands for Data Access Organization. Object data access interface, data access: as the name suggests, is used to interact with databases. Between the business logic and the database resources.

To separate low-level data access logic from high-level business logic, J2EE developers employ the Data Access Object (DAO) design pattern. When using the DAO pattern, you can concentrate on writing data access code.

One of the most common J2EE design patterns is the DAO pattern. This pattern is used by developers to separate the underlying data access operations from the higher-level business logic.

***A typical DAO implementation includes the following elements:***

***1.****A DAO factory class;****2.****A DAO interface;****3.****A concrete class that implements the DAO interface;****4.****Data transfer objects (sometimes called value objects).*

The logic for accessing data from a specific data source is contained in concrete DAO classes.

**4. BO: (Business Object), business object layer**

BO represents all “things” entity classes in the application domain. These entity classes reside on the server and rely on service classes to carry out their duties.

**5.DTO Data Transfer Object data transfer object**

DTO is most commonly used in situations where a large number of objects must be transferred, such as remote calls. For example, if a table has 100 fields, the corresponding PO will have 100 attributes.

However, we only need to display 10 fields on the interface, and the client obtains data via a WEB service. It is not necessary to send the entire PO object to the client. At this time, we can transfer the result to the client using DTO with only these ten attributes. This will not reveal the server’s table structure. ***If this object is used to display the corresponding interface after reaching the client, its identity will be changed to VO at that time.***

**6.POJO: (Plain Old Java Objects), simple Java objects**

***POJO is regular JavaBeans.***To avoid confusion with EJB, the POJO name is used, and the abbreviation is more straightforward. Some properties, as well as their getter and setter methods, can be used as value objects or dtos (Data Transform Object).

Of course, if you have a Simple operational attributes are also possible, but business methods, such as connection, are not permitted.