参考：<http://lib.csdn.net/article/java/64306?knId=224>

什么是JavaBean、bean?

什么是POJO、PO、DTO、VO、BO ?

什么是EJB、EntityBean？

前言：

在Java开发中经常遇到这些概念问题，有的可能理解混淆，有的可能理解不到位，特此花了很多时间理顺了这些概念。不过有些概念实际开发中并没有使用到，可能理解还不够准确，只能靠后续不断纠正了。

◆◆◆什么是POJO ？

      POJO（Plain Old Java Object）这种叫法是Martin Fowler、Rebecca Parsons和Josh MacKenzie在2000年的一次演讲的时候提出来的。

       按照Martin Fowler的解释是“Plain Old Java Object”，从字面上翻译为“纯洁老式的Java对象”，但大家都使用“简单java对象”来称呼它。POJO的内在含义是指：那些没有继承任何类、也没有实现任何接口，更没有被其它框架侵入的java对象。

◆先给一个定义吧:

**POJO是一个简单的、普通Java对象，它包含业务逻辑处理或持久化逻辑等，但不是JavaBean、EntityBean等，不具有任何特殊角色，不继承或不实现任何其它Java框架的类或接口。**可以包含类似于JavaBean属性和对属性访问的setter和getter方法的。

◆POJO(Plain Old Java Object)这个名字用来强调它是一个普通java对象，而不是一个特殊的对象。

◇2005年11月时，“POJO”主要用来指代那些没有遵从特定的Java对象模型、约定或框架如EJB的Java对象。

理想地讲，一个POJO是一个不受任何限制的Java对象（除了Java语言规范）。例如一个POJO不应该是

   1. 扩展预定的类，如       public class Foo extends javax.servlet.http.HttpServlet { ...

   2. 实现预定的接口，如   public class Bar implements javax.ejb.EntityBean { ...

   3. 包含预定的标注，如   @javax.ejb.Entity public class Baz{ ...

然后，因为技术上的困难及其他原因，许多兼容POJO风格的软件产品或框架事实上仍然要求使用预定的标注，譬如用于更方便的持久化。

**◇◆ 一般在web应用程序中建立一个数据库的映射对象时，我们只能称它为POJO。**

------------------------------------

◆正确官方理解思路：

        我在做Java EE培训中，发现我的很多学生问我什么是POJO，后来我在写书的时候发现POJO这个概念无法回避。现在网上对于POJO的解释很多，但是很多都是有错误的或者不够准确。

      对此我一开始也是存在误区的，我原来是这样理解的： POJO是这样的一种“纯粹的”JavaBean，在它里面除了JavaBean规范的方法和属性没有别的东西，即private属性以及对这个属性方法的public的get和set方法。我们会发现这样的JavaBean很“单纯”，它只能装载数据，作为数据存储的载体，而不具有业务逻辑处理的能力。所以下面的代码被认为是POJO了。

package com.demo.spring;

public class DbHello implements Hello { //实现了接口，就不能称之为POJO，这已经不是简单的Java类了

   　　private DictionaryDAOdao;

   　　public void setDao(DictionaryDAOdao) {

          　　this.dao = dao;

   　　}

}

     其实，这样的认为是错误的，我仔细阅读了《POJOs in Action》这本书的有关部分和POJO的最原始的出处http://martinfowler.com/bliki/POJO.html，

 　 The term was coined while Rebecca Parsons, Josh MacKenzie and I were preparing for a talk at a conference in September 2000.

     In the talk we were pointing out the many benefits of encoding business logic into regular java objects rather than using Entity Beans.We wondered why people were so against using regular objects in their systems and concluded that it was because simple objects lacked a fancy name. So we gave them one, and it's caught on very nicely.

**基本的意思是，将业务逻辑写进规则的Java对象**（regular java objects）中，比使用Entity Beans更有好处。另外，我们要给具有业务逻辑处理的规则的Java对象（regular java objects）起了一个名字——POJO，这些Java对象不是EntityBeans（EJB规范中专门用于封装数据库访问的一种组件，以面向对象的方式进行数据库的访问）。

 　　我又在http://www.webopedia.com/TERM/P/POJO.htm查到解释如下：

 　　POJO, or Plain Old Java Object, is a normal Java object class (that is, not a JavaBean, EntityBean etc.)  and does not serve any other special role nor does it implement any special interfaces of any of the Java frameworks. This term was coined by Martin Fowler, Rebbecca Parsons and Josh MacKenzie who believed that by creating the acronym POJO, such objects would have a "fancy name", thereby convincing people that they were worthy of use.

 　　基本意思是说POJO一个普通的Java对象（不是JavaBean，EntityBean等），也不担当任何的特殊的角色，也不实现任何Java框架指定的接口。

 　　我觉得上面的解释很准确，POJO应该不是我们开始认为的JavaBean，当然更不是EJB，它不应该依赖于框架（即继承或实现某些框架类或接口）。总之，POJO就是一个很简单的Java类，没那么多复杂的东西。例如：Struts1中的Action和ActionForm当然不属于POJO了，而在Struts2中的Action由于可以不继承任何的接口，所以在这种情况下Action是POJO，但是Struts2中的Action也可以继承ActionSupport类就不再属于POJO了。POJO里面是可以包含业务逻辑处理和持久化逻辑，也可以包含类似与JavaBean属性和对属性访问的set和get方法的。

 　　最后，我们总结一下给一个定义把，POJO是一个简单的、普通Java对象，它包含业务逻辑处理或持久化逻辑等，但不是JavaBean、EntityBean等，不具有任何特殊角色，不继承或不实现任何其它Java框架的类或接口。

◇辅助理解：**POJO可以认为是一个中间对象：**

一个中间对象，可以转化为PO、DTO、VO。

1 ．POJO持久化之后==〉PO

（在运行期，由Hibernate中的cglib动态把POJO转换为PO，PO相对于POJO会增加一些用来管理数据库entity状态的属性和方法。PO对于programmer来说完全透明，由于是运行期生成PO，所以可以支持增量编译，增量调试。）

2 ．POJO传输过程中==〉DTO

3 ．POJO用作表示层==〉VO

◆◆◆什么是JavaBean？

        JavaBean实际上是指一种特殊的Java类，它通常用来实现一些比较常用的简单功能，并可以很容易的被重用或者是插入到其他应用程序中去。所有遵循“一定编程原则”的Java类都可以被称作JavaBean。

◆JavaBean是一个遵循特定写法的Java类，是一种Java语言编写的可重用组件，它的方法命名，构造及行为必须符合特定的约定：

1、这个类必须具有一个公共的(public)无参构造函数；  
2、所有属性私有化（private）；  
3、私有化的属性必须通过public类型的方法（getter和setter）暴露给其他程序，并且方法的命名也必须遵循一定的命名规范。   
4、这个类应是可序列化的。（比如可以实现Serializable 接口，用于实现bean的持久性）

◇◆JavaBean在Java EE开发中，通常用于封装数据，对于遵循以上写法的JavaBean组件，其它程序可以通过反射技术实例化JavaBean对象（内省机制），并且通过反射那些遵循命名规范的方法，从而获知JavaBean的属性，进而调用其属性保存数据。

因为这些要求主要是靠约定而不是靠实现接口，所以许多开发者把JavaBean看作遵从特定命名约定的POJO。（可以这么理解，POJO按JavaBean的规则来，就可以变成JavaBean）。

◆简而言之，当一个POJO可序列化，有一个无参的构造函数，使用getter和setter方法来访问属性时，他就是一个JavaBean。（没毛病！）

------------------------------------

JavaBean是一种组件技术，就好像你做了一个扳手，而这个扳手会在很多地方被拿去用，这个扳子也提供多种功能(你可以拿这个扳手扳、锤、撬等等)，而这个扳手就是一个组件。

◇对于JavaBean，就是一个Java模型组件，他为使用Java类提供了一种标准的格式，在用户程序和可视化管理工具中可以自动获得这种具有标准格式的类的信息，并能够创建和管理这些类。   
◇JavaBean可以使应用程序更加面向对象，可以把数据封装起来，把应用的业务逻辑和显示逻辑分离开，降低了开发的复杂程度和维护成本！  
◇JavaBean 是一种JAVA语言写成的可重用组件。为写成JavaBean，类必须是具体的和公共的，并且具有无参数的构造器。JavaBeans 通过提供符合一致性设计模式的公共方法将内部域暴露称为属性。众所周知，属性名称符合这种模式，其他Java 类可以通过内省机制发现和操作这些JavaBean 属性。

◇通常情况下，由于 Java Bean 是被容器所创建（如 Tomcat) 的，所以Java Bean 应具有一个无参的构造器，另外，通常 Java Bean 还要实现 Serializable 接口用于实现 Bean 的持久性。 Java Bean 是不能被跨进程访问的。

◇JavaBean 是使用java.beans包开发的，它是 Java 2 标准版的一部分。JavaBean 是一台机器上同一个地址空间中运行的组件。JavaBean 是进程内组件。

◆◆◆什么是Bean？

◇Bean的中文含义是“豆子”，Bean的含义是可重复使用的Java组件。所谓组件就是一个由可以自行进行内部管理的一个或几个类所组成、外界不了解其内部信息和运行方式的群体。使用它的对象只能通过接口来操作。  
◇Bean并不需要继承特别的基类(BaseClass)或实现特定的接口(Interface)。Bean的编写规范使Bean的容器(Container)能够分析一个Java类文件，并将其方法(Methods)翻译成属性(Properties)，即把Java类作为一个Bean类使用。Bean的编写规范包括Bean类的构造方法、定义属性和访问方法编写规则。  
◇Java Bean是基于Java的组件模型，由属性、方法和事件3部分组成。在该模型中，JavaBean可以被修改或与其他组件结合以生成新组件或完整的程序。它是一种Java类，通过封装成为具有某种功能或者处理某个业务的对象。因此，也可以通过嵌在JSP页面内的Java代码访问Bean及其属性。

◆◆◆什么是EJB 、Entity Bean？

◇Enterprise Bean，也就是Enterprise JavaBean（EJB），是J2EE的一部分，定义了一个用于开发基于组件的企业多重应用程序的标准。其特点包括网络服务支持和核心开发工具(SDK)。

◇在 J2EE里，Enterprise Java Beans(EJB)称为Java 企业Bean，是Java的核心代码，分别是会话 Bean（Session Bean），实体Bean（Entity Bean）和消息驱动Bean（MessageDriven Bean）。

◇1.Session Bean用于实现业务逻辑，它可以是有状态的，也可以是无状态的。每当客户端请求时，容器就会选择一个Session Bean来为客户端服务。Session Bean可以直接访问数据库，但更多时候，它会通过Entity Bean实现数据访问。这个类一般用单例模式来实现，因为每次连接都需要用到它。

◇2.Entity Bean是域模型对象，用于实现O/R映射，负责将数据库中的表记录映射为内存中的Entity对象，事实上，创建一个Entity Bean对象相当于新建一条记录，删除一个 Entity Bean会同时从数据库中删除对应记录，修改一个Entity Bean时，容器会自动将Entity Bean的状态和数据库同步。

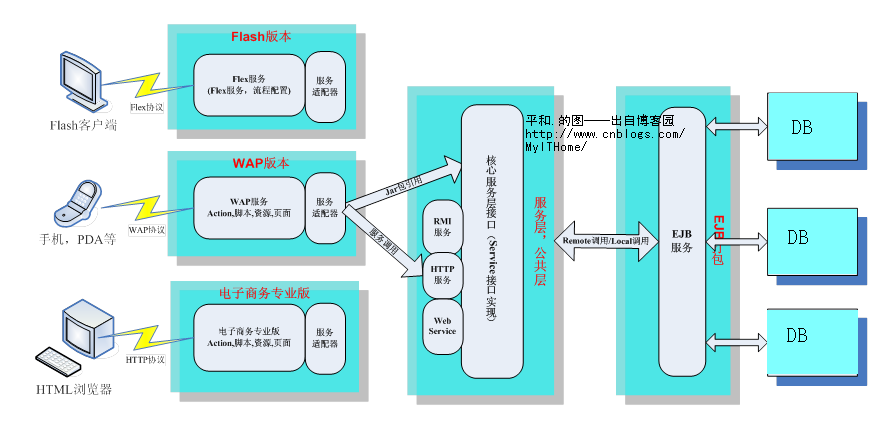
◆Enterprise Bean 是使用javax.ejb包开发的，它是标准 JDK 的扩展，是 Java 2 Enterprise Edition 的一部分。Enterprise Bean 是在多台机器上跨几个地址空间运行的组件。因此 Enterprise Bean 是进程间组件。

◇我们一般所熟悉的tomcat仅仅只实现了j2ee的一小部分规范，它只是一个serlvet的容器（Web）容器，它不能跑J2EE的程序，EJB说到底也是种规范，它是j2EE下面的一个子分类（核心类），所以j2ee包含EJB，同时我们都可以说JBOSS，Weblogic，WebSphere是J2EE容器，也可以叫EJB容器。因为它们能跑EJB组件。那么什么是EJB组件呢？其实就是java写出来的一段程序被打包成EAR包，这个EAR包放在某个EJB的容器的特定目录下启动就可以跑了。类似于互联网公司经常使用的WAR包（部署在tomcat上）。

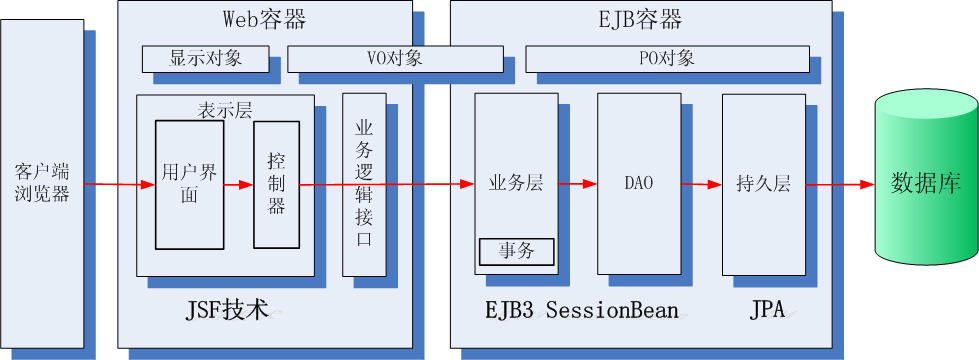
◇然后要说的是EJB是一种是很老、很繁琐的技术标准（规范）了，现如今基本已经被淘汰了。因为EJB的繁琐、难用，spring的出现彻底革了EJB的命，不然怎么说是Java的春天（spring）来了呢。

EJB实现原理： 就是把原来放到客户端实现的代码放到服务器端，并依靠RMI进行通信。

这篇博文对EJB的原理和实质的解释非常清晰： <http://blog.csdn.net/jojo52013145/article/details/5783677>



EJB在架构中所处的位置如上图，EJB在J2EE开发中的详细过程：



◆◆◆什么是PO？

◆ PO ：persistent object持久对象。  
O/R Mapping 是 Object Relational Mapping（对象关系映射）的缩写。通俗点讲，就是将对象与关系数据库绑定，用对象来表示关系数据。

PO是在O/R映射的时候出现的概念，如果没有O/R映射，没有这个概念存在了。常用的O/R映射框架有hibernate等。通常对应数据模型(数据库)，本身还有部分业务逻辑的处理。可以看成是与数据库中的表相映射的java对象。最简单的PO就是对应数据库中某个表中的一条记录，多个记录可以用PO的集合。PO中应该不包含任何对数据库的操作（me:对数据库的操作放在DAO层）。

1、有时也被称为Data对象，对应数据库中的entity，可以简单认为一个PO对应数据库中的一条记录。  
2、在hibernate持久化框架中与insert/delet操作密切相关。  
3、PO中不应该包含任何对数据库的操作。  
4、PO的属性是跟数据库表的字段一一对应的。  
5、PO对象需要实现序列化接口。

就是说在一些Object/Relation Mapping工具中，能够做到维护数据库表记录的persisent object完全是一个符合Java Bean规范的纯Java对象，没有增加别的属性和方法。全都是这样子的：

**public** **class** User {

    private long id;

    private String name;

    public void **setId**(long id) {

**this**.id = id;

    }

    public void **setName**(String name) {

**this**.name = name;

    }

    public long getId() {

**return** id;

    }

    public String getName() {

        return name;

    }

}

◆◆◆什么是DTO？

◆ DTO （TO）：Data Transfer Object 数据传输对象。  
主要用于远程调用等需要大量传输对象的地方。

可以将PO中的部分属性抽取出来，就形成了DTO。

◇比如我们一张表有100个字段，那么对应的PO就有100个属性。  
但是我们界面上只要显示10个字段，客户端用WEB service来获取数据，没有必要把整个PO对象传递到客户端，这时我们就可以用只有这10个属性的DTO来传递结果到客户端，这样也不会暴露服务端表结构.到达客户端以后，如果用这个对象来对应界面显示，那此时它的身份就转为VO（View Object）。  
  
◇用在需要跨进程或远程传输时，它不应该包含业务逻辑。  
◇比如一张表有100个字段，那么对应的PO就有100个属性（大多数情况下，DTO 内的数据来自多个表）。但view层只需显示10个字段，没有必要把整个PO对象传递到client，这时我们就可以用只有这10个属性的DTO来传输数据到client，这样也不会暴露server端表结构。到达客户端以后，如果用这个对象来对应界面显示，那此时它的身份就转为VO（View Object）。

◆◆◆什么是VO？

◇ VO ：value object 值对象 / view object 表现层（视图）对象。

◇先说 value object 值对象：  
通常用于业务层之间的数据传递，和PO一样也是仅仅包含数据而已。但应是抽象出的业务对象，可以和表对应，也可以不，这根据业务的需要。个人觉得同DTO(数据传输对象)，在web上传递。

◇①VO（value object）是值对象，精确点讲它是业务对象，是存活在业务层的，是业务逻辑使用的，它存活的目的就是为数据提供一个生存的地方。VO的属性是根据当前业务的不同而不同的，也就是说，它的每一个属性都一一对应当前业务逻辑所需要的数据的名称。  
VO是什么？它是值对象，准确地讲，它是业务对象，是生活在业务层的，是业务逻辑需要了解，需要使用的，再简单地讲，它是概念模型转换得到的。

◇首先说PO和VO吧，它们的关系应该是相互独立的，一个VO可以只是PO的部分，也可以是多个PO构成，同样也可以等同于一个PO（当然我是指他们的属性）。正因为这样，PO独立出来，数据持久层也就独立出来了，它不会受到任何业务的干涉。又正因为这样，业务逻辑层也独立开来，它不会受到数据持久层的影响，业务层关心的只是业务逻辑的处理，至于怎么存怎么读交给别人吧！不过，另外一点，如果我们没有使用数据持久层，或者说没有使用hibernate，那么PO和VO也可以是同一个东西，虽然这并不好。

◇② VO（view object）表现层对象，视图对象。  
用一个VO对象对应整个界面的值。

1、主要对应页面显示（web页面/swt、swing界面）的数据对象。  
2、可以和表对应，也可以不，这根据业务的需要。  
注：在struts中，用ActionForm做VO，需要做一个转换，因为PO是面向对象的，而ActionForm是和view对应的，要将几个PO要显示的属性合成一个ActionForm，可以使用BeanUtils的copy方法。

◆◆◆什么是DAO？

◇DAO ：data access object 数据访问对象。  
这个大家最熟悉，和上面几个O区别最大，基本没有互相转化的可能性和必要.  
主要用来封装对数据库的访问。通过它可以把POJO持久化为PO，用PO组装出来VO、DTO。

◇是一个sun的一个标准j2ee设计模式，这个模式中有个接口就是DAO，它负持久层的操作。为业务层提供接口。此对象用于访问数据库。通常和PO结合使用，DAO中包含了各种数据库的操作方法。通过它的方法，结合PO对数据库进行相关的操作。夹在业务逻辑与数据库资源中间。配合VO，提供数据库的CRUD操作...  
1、主要用来封装对DB（数据库）的访问（CRUD操作）。  
2、通过接收业务层的数据，把POJO持久化为PO。

◆◆◆什么是BO？

JavaBean是一种组件技术，就好像你做了一个扳手，而这个扳手会在很多地方被拿去用，这个扳子也提供多种功能(你可以拿这个扳手扳、锤、撬等等)，而这个扳手就是一个组件。

◇BO ：business object 业务对象。  
  
◇主要作用是把业务逻辑封装为一个对象。这个对象可以包括一个或多个其它的对象。  
◇比如一个简历，有教育经历、工作经历、社会关系等等。  
我们可以把教育经历对应一个PO，工作经历对应一个PO，社会关系对应一个PO。  
建立一个对应简历的BO对象处理简历，每个BO包含这些PO。  
这样处理业务逻辑时，我们就可以针对BO去处理。  
  
◇封装业务逻辑的java对象，通过调用DAO方法，结合PO，VO进行业务操作。  
◇封装业务逻辑为一个对象（可以包括多个PO，通常需要将BO转化成PO，才能进行数据的持久化，反之，从数据库中得到的PO，需要转化成BO才能在业务层使用）。  
关于BO主要有三种概念  
1 、只包含业务对象的属性；  
2 、只包含业务方法；  
3 、两者都包含。  
在实际使用中，认为哪一种概念正确并不重要，关键是实际应用中适合自己项目的需要。