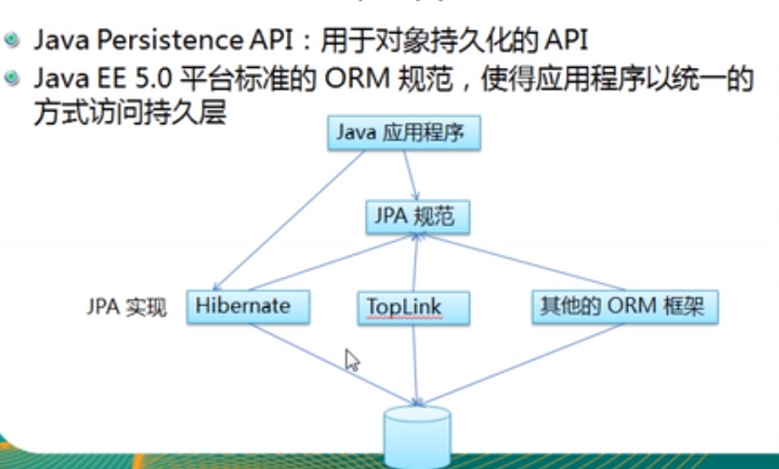
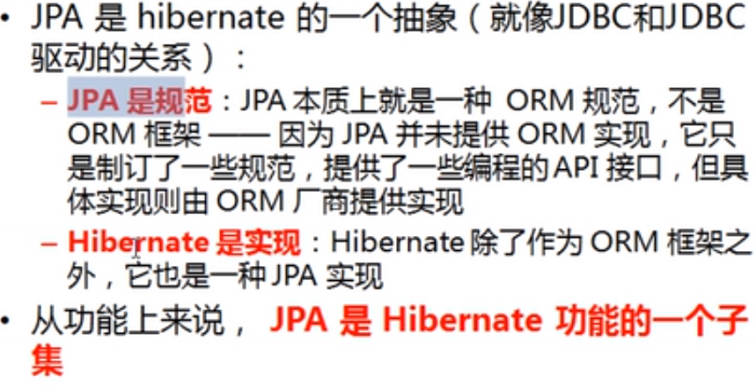
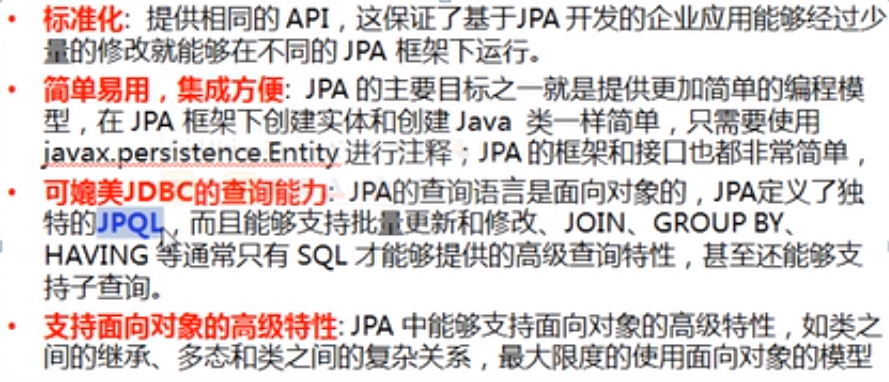
# JPA是什么？



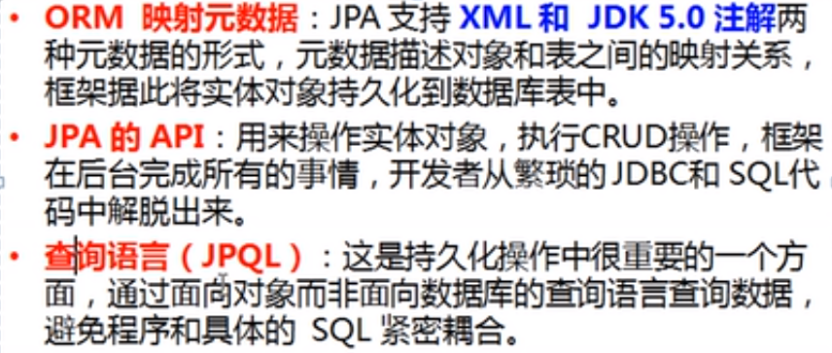
# JPA和Hibernate的关系



# JPA的优势



# JPA的三方面技术



# 一、JPA的基本注解

## 1.1、六个基本注解

@Entity、@Table、@Id、@GeneratedValue、@Column、@Basic

### @Entity

此注解标注用于实体类声明语句前，**指出该Java类为实体类，将映射到指定的数据库表中**。

### @Table

当实体类与其映射的数据库表名不同名时，需要使用@Table标注说明，**该标注与@Entity标注并列使用**，置于实体类声明语句之前，可写于单独语句行，也可与声明语句同行。

@Table注解常用选项是name，用于指定数据库的表名。另外还有两个选项catalog和schema用于设置表所属的数据库目录或模式，通常为数据库名。uniqueConstrients选项用于设置约束条件，通常不须设置。

### @Id

该注解用于声明一个实体类的属性映射为数据库的主键列。该属性通常置于属性声明语句之前，可与声明语句同行，也可写在单独行上。@Id注解也可以置于属性的getter方法之前。

### @GeneratedValue

该注解用于标注主键的生成策略，通过strategy属性指定。默认情况下，JPA自动选择一个最合适底层数据库的主键生成策略：SqlServer->identity，MySQL->auto increment。

在Javax.persistence.GenerationType中定义了以下几种可选择的策略：

- IDENTITY:采用数据库ID自增长的方式来自增主键字段，Oracle不支持这种方式。

- **AUTO：JPA自动选择合适的策略，就是默认项**。

- SEQUENCE:通过序列产生主键，通过@SequenceGenerator注解指定序列名，MySQL不支持这种方式。

- TABLE:通过表产生主键，框架借由表模拟产生主键，使用这种策略可以使应用更易于数据库移植。

### @Column

当实体的属性与其映射的数据库的列不同名时需要使用@Column注解，该注解通常置于实体的属性声明语句之前，还可与@Id注解一起使用。

@Column主机的常用属性是name，用于设置映射数据库表的列名。此外，其他属性如：unique、nullable、length等。

- columnDefinition属性：表示该字段在数据库中的实际类型。通常ORM框架可以根据属性类型自动判断数据库中字段的类型，但是对于Date类型仍无法确定数据库中字段类型究竟是DATE，TIME还是TIMESTAMP。此外，String的默认映射类型为VARCHAR，如果要将String类型映射到特定数据库的BLOB或TEXT字段类型。

### @Basic

该注解表示一个简单的属性到数据库表的字段的映射，对于没有任何标注的getXxxx()方法，默认即为@Basic。

- fetch属性：表示该属性的读取策略，有EAGER何LAZY两种，分别表示主动抓取和延迟加载，默认为EAGER。

- optional：表示该属性是否允许为null，默认为true。

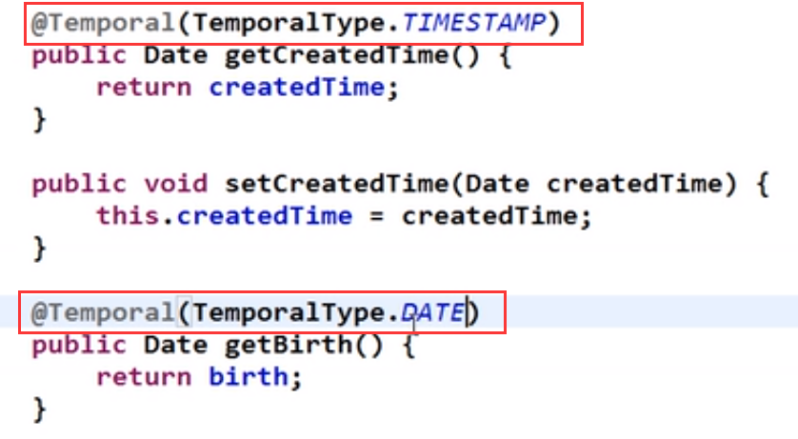
## @Transient

表示该属性并非一个到数据库表的字段的映射，ORM框架将忽略属性。

**如果一个属性并非数据库表的字段映射，就务必将其标注为@Transient，否则，ORM框架默认其注解为@Basic**。

## @Temporal

在核心的JavaAPI中并没有定义Date类型的精度。而在数据库中，表示Date类型的数据有DATE，TIME，和TIMESTAMP三种精度（即单纯的日期，时间，或者两者兼备）在进行属性映射时可以使用@Temporal注解来调整精度。



## 用table来生成主键详解

将当前主键的值单独保存到一个数据表中，主键的值每次都是从指定的表中查询来获得。

这种生成主键的策略可以适用于任何数据库，不必担心数据库兼容问题。

- name属性：表示该主键生成策略的名称，它被引用在@GeneratedValue中设置的generator中。

- table属性：表示主键生成策略所持久化的表名

