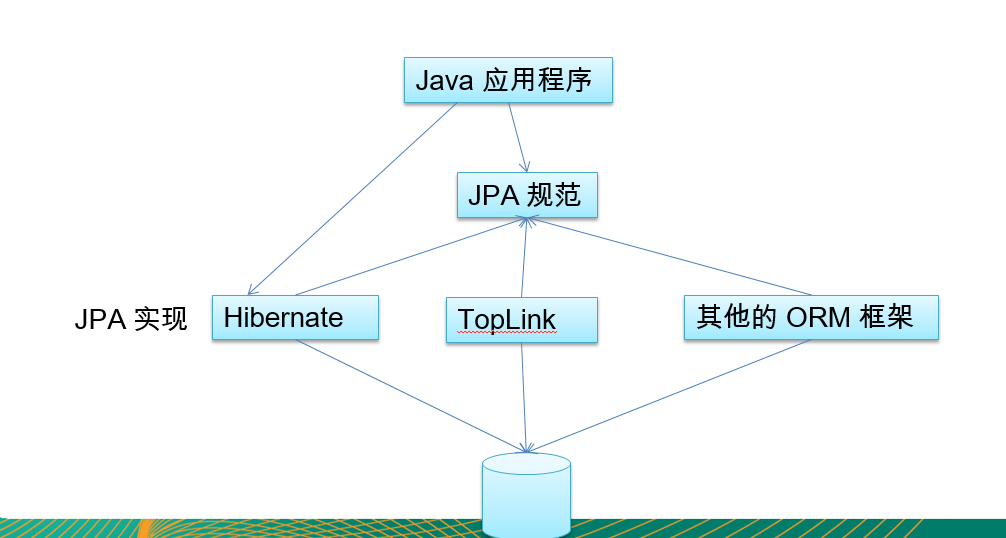
**JPA是什么**

JPA全称Java Persistence API，是一组用于将数据存入数据库的类和方法的集合。JPA通过JDK 5.0注解或XML描述对象－关系表的映射关系，并将运行期的实体对象持久化到数据库中。



**JPA统一了java应用程序访问ORM框架的规范。**

**以前我们的应用程序直接使用ORM框架，如hibernate，mybatis。但是不同的框架使用方法不一样，而JPA让我们以同样的方式访问不同的ORM框架。**常用的框架是hibernate。

## JPA Providers

JPA是开源API，各企业经营商Oracle, Redhat, Eclipse等，提供各种有特色的JPA产品，其中包括: Hiberate, Eclipselink, Toplink, Spring Data JPA等等

## JPA架构

JPA展示如何将Plain Oriented Java Object(POJO)定义为entity，以及如何管理entity之间的关系。

### 类级架构

JPA的类级架构包含几个核心组件

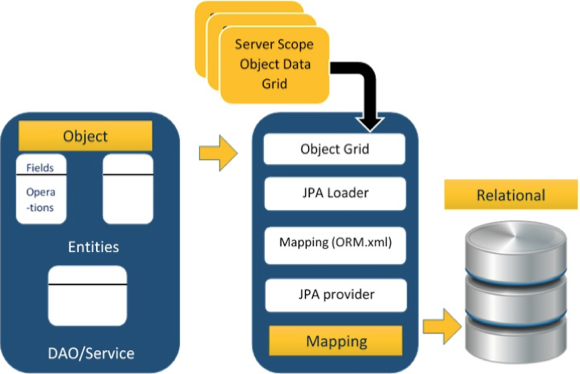
> EntityManagerFactory: 创建和管理多个EntityManager实例  
> EntityManager: 接口，管理对象的操作(create, update, delete, Query)  
> Entity: 持久化对象，在数据库中以record存储  
> EntityTransaction: 与EntityManager一对一  
> Persistence: 包含获取EntityManagerFactory实例的静态方法  
> Query: 运营商必须实现的接口，获取满足creteria的关系对象（relational object）

## ORM

当前许多应用使用关系数据库存数据。最近，供应商转向对象数据库来减少数据维护的压力。对象关系技术的核心是映射orm.xml文件。  
由于xml不要求兼容，我们可以轻松修改数据源。

### ORM架构

ORM通过编程的方式将对象类型转换成关系类型。主要特点是将object映射成数据库中的数据。在映射的过程中我们必须考虑数据，数据类型和数据之间的关系。



**阶段一**  
**又名Object data阶段**，包括POJO类，服务接口和类。它是主要的业务层，包含业务逻辑操作和属性。比如我们考虑employ数据库的schema  
> Employee POJO类包含属性：ID, name, salary, 和destination，还有这些属性的setter和getter方法  
> Employee DAO/Service类包含创建employee，查找employee，和删除employee等服务方法。

**阶段二**  
**又名Mapping或者persistance阶段**，包括JPA provider, mapping文件(ORM.xml), JPA Loader, 和Object Grid  
> JPA provider: 运营商提供的产品，包括JPA flavor(javax.persistence中)。例如 Eclipselink, Toplink, Hibernate等等。  
> Mapping文件: ORM.xml包括POJO类到关系数据库中数据的映射  
> JPA Loader: JPA loader类似加载关系数据的缓存内存。  
> Oject Grid: 存储关系数据的临时位置。

**阶段三**  
**关系数据阶段，包括和业务逻辑有关的关系数据**。在提交之前，修改的数据导以grid格式存在缓存内存中。

以上展示了ORM的如何在三个阶段中将数据存入数据库中。

### Mapping.xml

mapping.xml指示JPA vendor将entity类映射到数据库表上。

### 注解

一般XML用于配置特定组建或者定义两个不同组建直接直接的关系。在我们的case中，我们在写mapping.xml文件时需要将POJO类的属性和文件中entity tags对应起来。这需一定的维护开销。

另一种方案是: 在类定义中，我们可以用注解完成配置。注解用于class, properties, 和方法前。所有JPA的注解定义在javax.persistence包中。

Entity关系  
Entity可以看作是关系表，因此entity类之间的关系有:  
> @ManyToOne  
> @OneToMany  
> @OneToOne  
> @ManyToMany

@ManyToOne关系  
例子：employee和department的关系是多对一。department的key作为employment的外键。标注在employee内  
生成employee和department两张表，其中employment包含department的key

@OneToMany关系  
TableA与TableB是一对多的关系，那么TableA中的一条记录对映TableB中0或者多条记录。比如department和employee的关系。标注在department上  
department\_employee, department, employee

@OneToOne关系

比如一个employee只属于一个department。标注方式同@ManyToOne，在employee上。  
生成employee和department两张表，其中employment包含department的key

@ManyToMany关系  
比如班级和老师之间的关系。两边都需要标注。  
生成三张表：teacher\_class, teacher, class

## Criteria API

Criteria API用来定义query，是JPQA query的另一种选择。String based JPQL query和JPA criteria based query在性能和效率方面相同。

Criteria Query Structure简单的例子

[复制代码](javascript:void(0);)

EntityManager em = ...;

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Entity class> cq = cb.createQuery(Entity.class);

Root<Entity> from = cq.from(Entity.class);

cq.select(Entity);

TypedQuery<Entity> q = em.createQuery(cq);

List<Entity> allitems = q.getResultList();

[复制代码](javascript:void(0);)

上例展示了创建criteria的基本步骤.

> EntityManager: 创建CriteriaBuilder对象   
> CriteriaQuery: 创建Query对象，属性可以被修改  
> CriteriaQuery.from: 设置query root  
> CriteriaQuery.select: 设置结果list类型  
> TypedQuery<T>: 准备执行的query，并且指明Query result的类型  
> TypedQuery.getResultList: 执行query，返回一组结果，存在list中