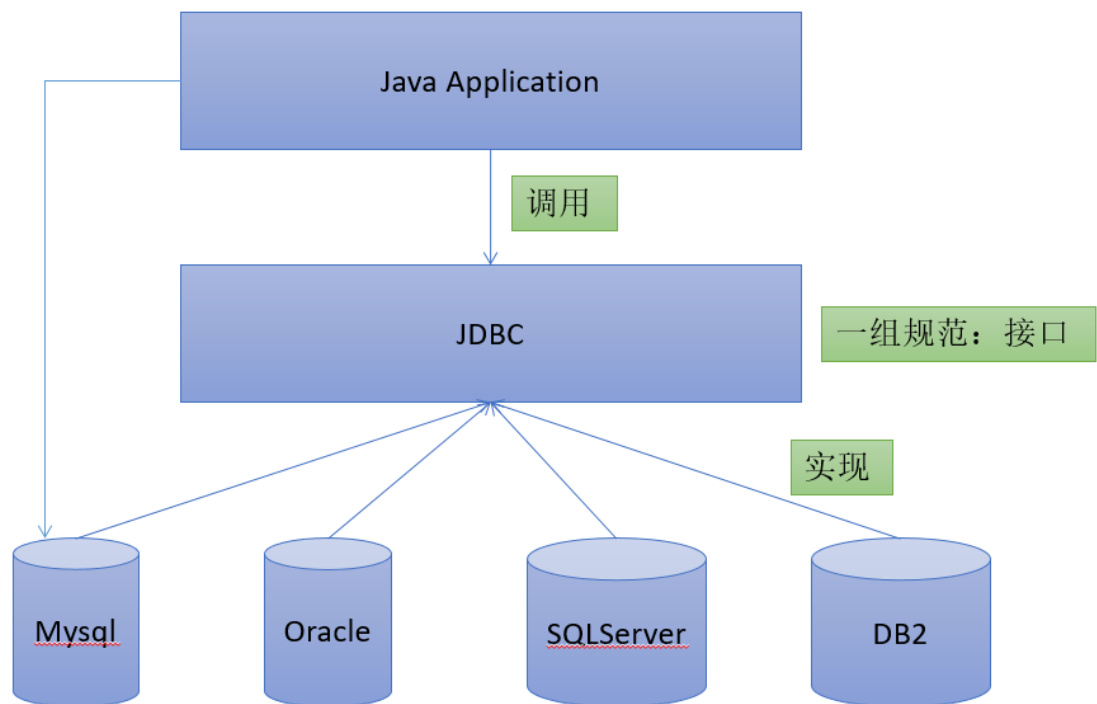


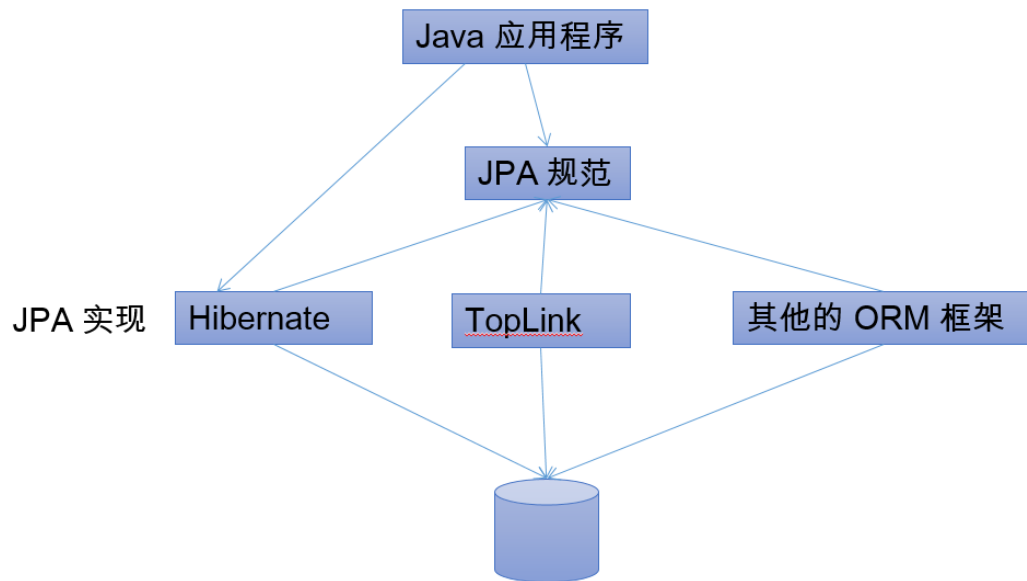
# JPA+SpringData

## 1. JPA 介绍

JPA 是 Java Persistence API 的简称(用于对象持久化的 API)。中文名 Java 持久层 API，是 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。<sup>[1]</sup>

Sun 引入新的 JPA ORM 规范出于两个原因：其一，简化现有 Java EE 和 Java SE 应用开发工作；其二，Sun 希望整合 ORM 技术，实现天下归一。





## 1.1. 为什么要学习 JPA

## 1.2. 常见的 ORM 框架

### Hibernate

JPA 的始作俑者就是 Hibernate 的作者

Hibernate 从 3.2 开始兼容 JPA

### OpenJPA

OpenJPA 是 Apache 组织提供的开源项目

### TopLink

TopLink 以前需要收费，如今开源了

## 1.3. JPA 功能特点

JPA 包括三个方面技术：

ORM 映射元数据，JPA 支持 XML 与 JDK5.0 注解，元数据描述对象与表之间的映射关系，框架可以将实体对象持久化到数据库当中。

JPA 持久化 API，用来操作实体对象，执行 curd 操作，框架在后台替我

们完成了所有的事情，开发者可以从 JDBC 和 SQL 代码中解脱出来。

查询语言，这是持久化操作很重要的一个概念。通过面向对象而非面向数据库的查询数据，避免程序与 SQL 的紧密耦合。

## 2. 创建 JPA 项目

### 2.1. JPA 完成持久化开发流程

创建 `persistence.xml`，在这个文件中配置持久化单元

需要指定跟哪个数据库进行交互；

需要指定 JPA 使用哪个持久化的框架以及配置该框架的基本属性

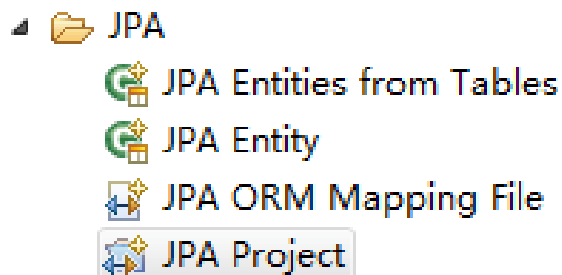
创建实体类，使用 annotation 来描述实体类跟数据库表之间的映射关系。

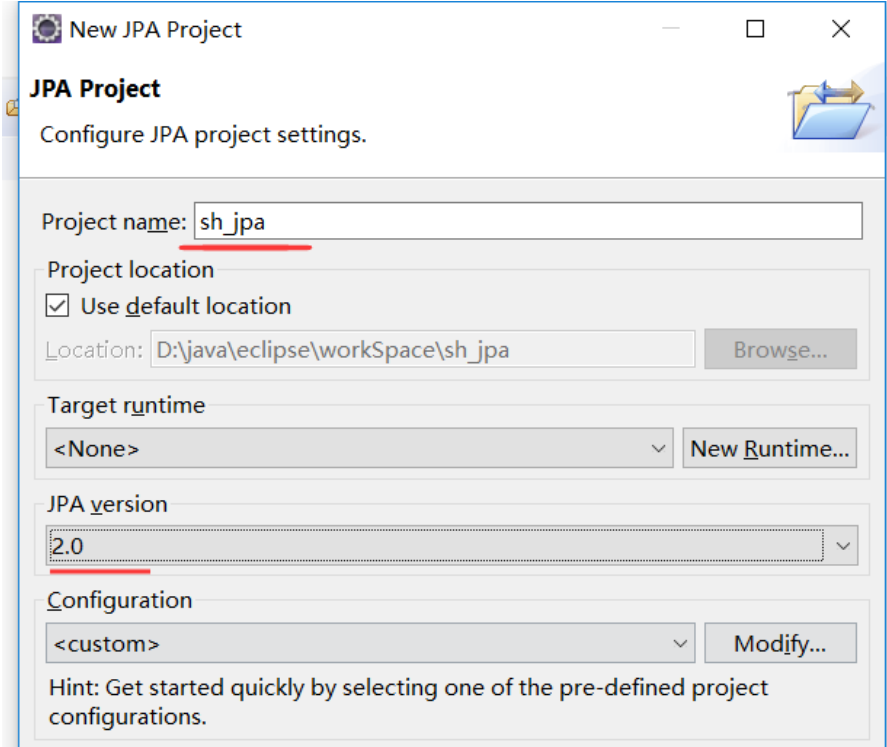
使用 JPA API 完成数据增加、删除、修改和查询操作

创建 EntityManagerFactory (对应 Hibernate 中的 SessionFactory)；

创建 EntityManager (对应 Hibernate 中的 Session)；

### 2.2. 在 Eclipse 下创建 JPA 工程





The image shows the 'New JPA Project' dialog box in Eclipse. The title bar says 'New JPA Project'. Below the title bar, there's a section 'JPA Project' with the subtitle 'Configure JPA project settings.' and a folder icon. The dialog is divided into several sections: 'Project name' with a text field containing 'sh\_jpa'; 'Project location' with a checked 'Use default location' checkbox and a text field showing 'D:\java\eclipse\workspace\sh\_jpa'; 'Target runtime' with a dropdown menu set to '<None>' and a 'New Runtime...' button; 'JPA version' with a dropdown menu set to '2.0'; and 'Configuration' with a dropdown menu set to '<custom>' and a 'Modify...' button. At the bottom, there's a hint: 'Hint: Get started quickly by selecting one of the pre-defined project configurations.'

New JPA Project

JPA Project

Configure JPA project settings.

Project name: sh\_jpa

Project location

☒ Use default location

Location: D:\java\eclipse\workspace\sh\_jpa

Target runtime

<None>

New Runtime...

JPA version

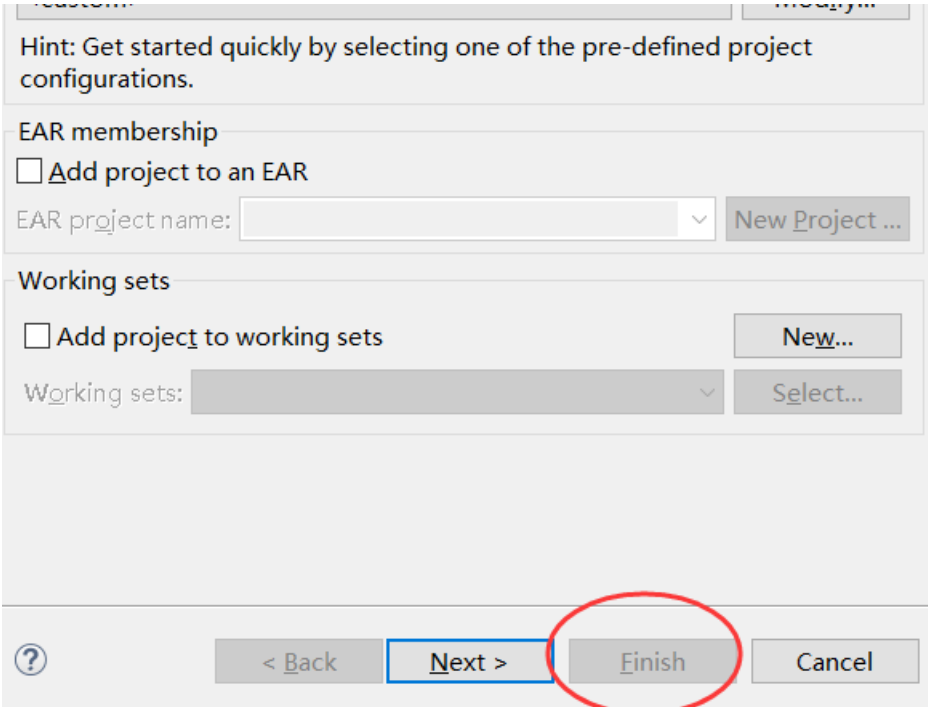
2.0

Configuration

<custom>

Modify...

Hint: Get started quickly by selecting one of the pre-defined project configurations.



This image shows the bottom portion of the 'New JPA Project' dialog box. It includes the same hint as the previous image. Below the hint are two sections: 'EAR membership' with an unchecked 'Add project to an EAR' checkbox and an 'EAR project name' field; and 'Working sets' with an unchecked 'Add project to working sets' checkbox and a 'Working sets' field. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish'. The 'Finish' button is circled in red, and the 'Next >' button is highlighted with a blue border.

Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR membership

☐ Add project to an EAR

EAR project name:

Working sets

☐ Add project to working sets

Working sets:

< Back Next > Finish Cancel

通过观察，如果 Finish 不能成功需要导入需要支持的

### 2.2.1. 安装 EclipseLink JAR 文件

#### 1、安装 EclipseLink JAR 文件

从 <https://www.eclipse.org/eclipselink/downloads/> 处下载 EclipseLink 2.6.4 Installer Zip (38 MB) 资源（也可以选择其它版本）。EclipseLink 中提供了所有持久化服务的实现。

持久化服务将会用到下面三个 jar 包：

- ①、eclipselink.jar
- ②、javax.persistence.source\_2.1.0.v201304241213.jar
- ③、javax.persistence\_2.1.0.v201304241213.jar

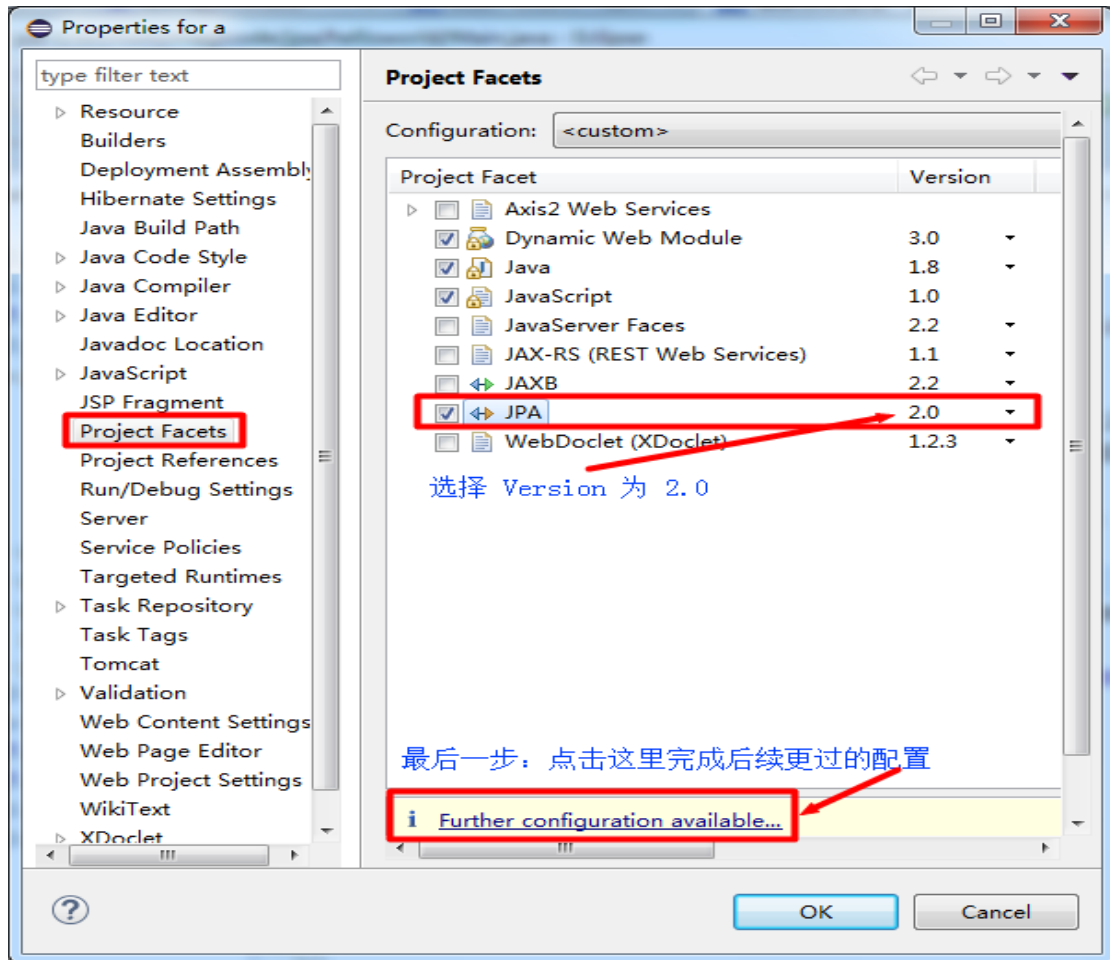
将下载好的 EclipseLink 压缩包解压到任何你想用于完成安装的文件夹中。我将文件解压到 “. \workspace\libraries\EclipseLink” 文件夹下

#### 2、在 Eclipse IDE 中设置 JPA Facet

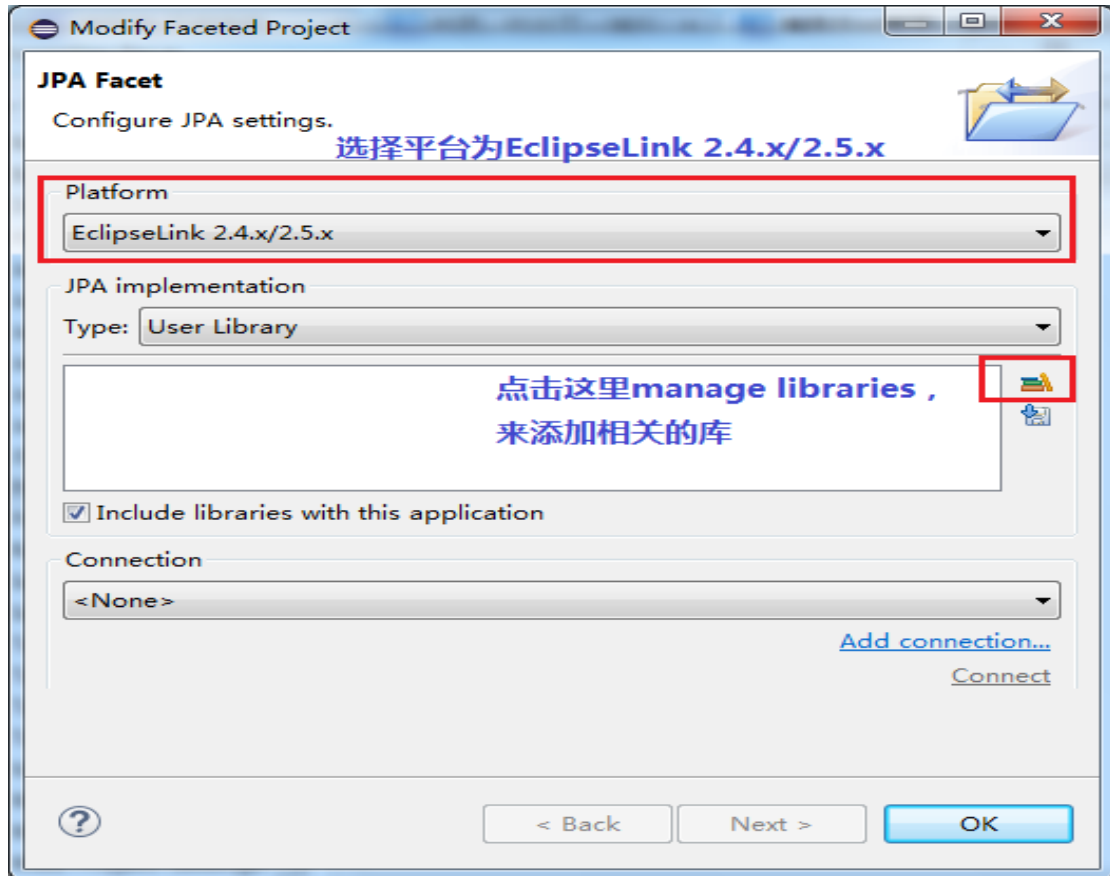
首先在 Eclipse 中创建一个动态 web 工程，为我们后续添加 JPA facet 作准备。步骤如下：

- ①、打开 Eclipse IDE
- ②、创建一个动态的 Web 工程
- ③、在刚刚创建好的工程上点击右键，选择 properties
- ④、在左侧那一列中点击 “Project Facets”
- ⑤、然后在右侧选中 JPA 选项

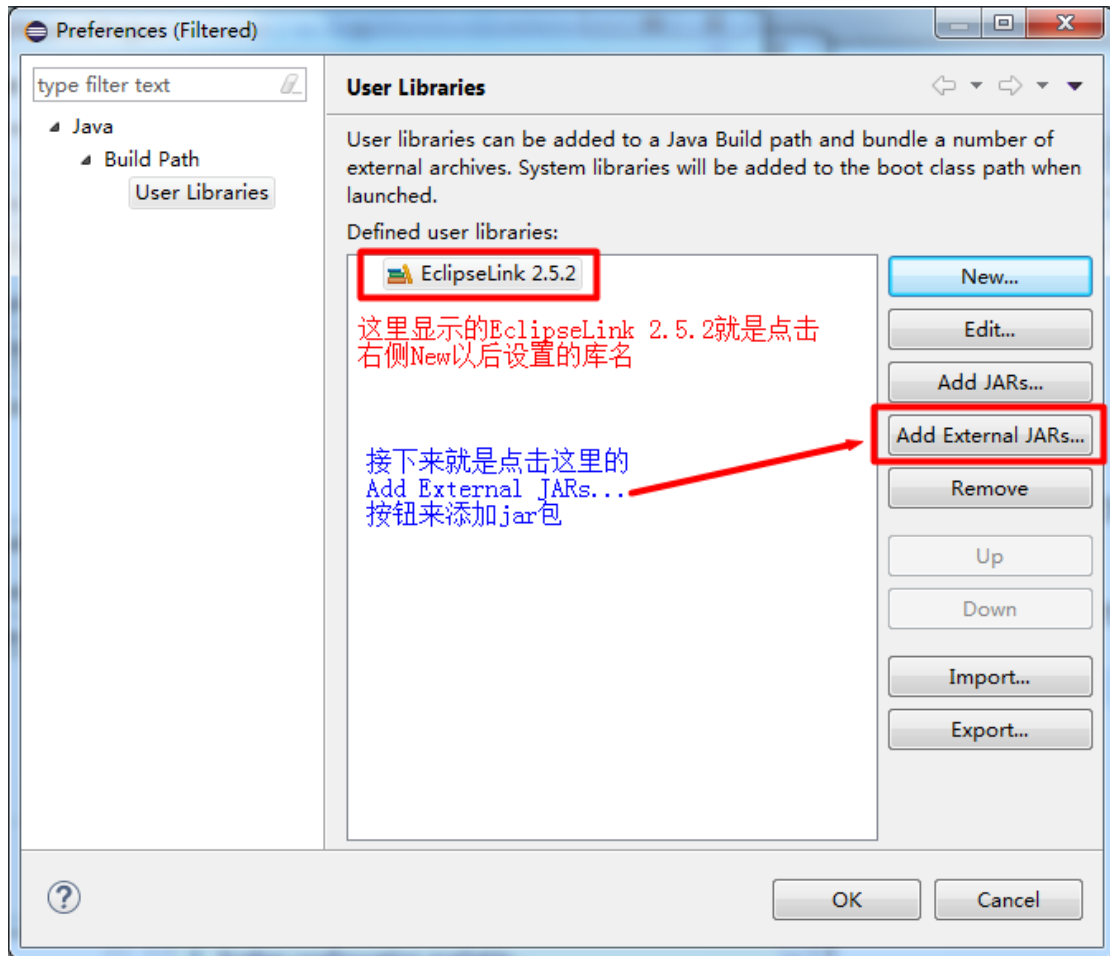
如下图：



新弹出的对话框如下图 Figure 2.3 所示。用户可以在这个对话框中添加必须的库来实现 JPA。



“New” 以后又会弹出一个对话框，要我们填写库的名字，我填写为“EclipseLink 2.5.2”。点击 OK



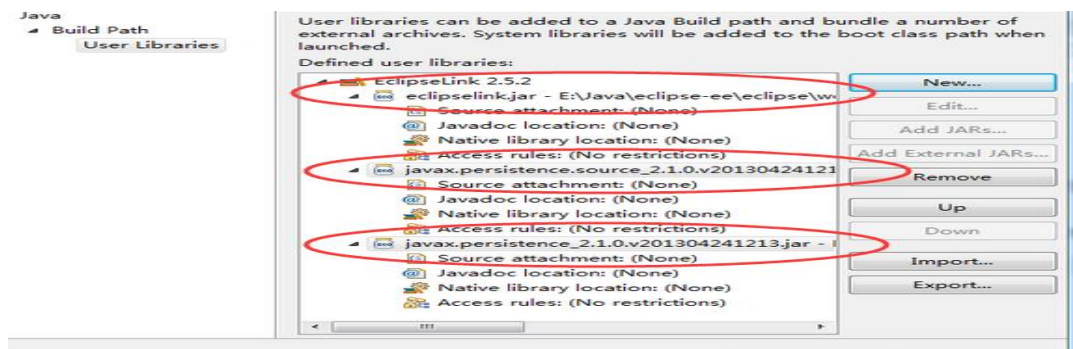
到目前为止，我们已经做好了添加 jar 包的前期准备。接下来就是添加上面“1、安装 EclipseLink JAR 文件”这一步中提到的那三个 jar 包了

配置 Library 需要我们添加 JPA 实现所必须的 jar 包。按照下面的步骤来完成配置：

①、点击“Add External JARs...”（也就是 Figure 2.4 图中红色框中的按钮）

②、添加 eclipselink.jar，它位于“.\workspace\libraries\EclipseLink 2.5.2\jlib”文件夹下面

③、添加“.\workspace\libraries\EclipseLink 2.5.2\jlib\jpa”文件夹下 javax.persistence.\* 开头的 jar 包（也就是前面说的后两个 jar 包）





## 2.3. 创建 JPA Project

新建 JPA Project

导入开发依赖 JAR

## 2.4. persistence.xml

name 属性用于定义持久化单元的名字, 必选

transaction-type : 指定 JPA 的事务处理策略。  
RESOURCE\_LOCAL : 默认值, 数据库级别的事务, 只能针对一种数据库, 不支持分布式事务。  
如果需要使用 JTA : transaction-type="JTA"

```
<persistence-unit name="jpa" transaction-type="RESOURCE_LOCAL">
  <!-- 配置的JPA ORM产品 -->
  <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
  <!-- 添加对应持久化类 -->
  <class>jpa.qfjy.bean.Users</class>
  <properties>
    <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
    <property name="javax.persistence.jdbc.user" value="root"/>
    <property name="javax.persistence.jdbc.password" value="root"/>
    <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/jpa?useUnicode=true&characterEncoding=utf8"/>
    <!-- 配置JPA基本属性、Hibernate基本属性 -->
    <property name="hibernate.show_sql" value="true"/>
    <property name="hibernate.format_sql" value="true"/>
  </properties>
```

显式列出实体类

连接数据库的基本信息

ORM 框架的基本信息

指定ORM框架的 `javax.persistence.spi.PersistenceProvider` 接口的实现类。若项目中只有一个实现可省略

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="5-jpa" transaction-type="RESOURCE_LOCAL">
    <!-- 哪个实现的ORM产品 Hibernate -->
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <!-- 连接数据库的相关配置信息 -->
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.url"
```

```

value="jdbc:mysql://localhost:3306/jpa"/>
    <property name="javax.persistence.jdbc.user" value="root"/>
    <property name="javax.persistence.jdbc.password" value="root"/>

    <!-- 显示其打印的SQL语句 -->
    <property name="hibernate.show_sql" value="true"/>
    <property name="hibernate.format_sql" value="true"/>
</properties>
</persistence-unit>
</persistence>

```

## 2.5. 创建实体 Bean

```

@Entity //作用于：类上方 指出该Java 类为实体类(用于和数据库表做映射的)，
@Table //用于：类上方 指定表名和实体类对映射
@Data
public class Users {
    /**
     * 标注用于声明一个实体类的属性映射为数据库的主键列
     * 作用于：属性上方 or getXX方法前
     * 编码规范：将所有的注解 声明在属性上方（不建议放在getXX）
     */
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;
    private String uname;
    private String upass;
    private Integer age;
    private String remark;
}

```

## 2.6. 流程实现

```

@Test
    public void add(){
        //1得到工厂对象1 JPA EntityManagerFactory
        String persistenceUnitName="jpa";
        EntityManagerFactory
entityManagerFactory=Persistence.createEntityManagerFactory(persistence
UnitName);
        //2 得到会话信息      Hibernate:Session Mybatis:SqlSession
        EntityManager em=entityManagerFactory.createEntityManager();
    }

```

```
//3事务管理
EntityTransaction tran= em.getTransaction();
tran.begin();
//4业务逻辑处理
Users u=new Users();
u.setUname("jpa-用户名2");
u.setUpass("123456");
u.setCreateDate(new Date());

em.persist(u);

//5事务提交
tran.commit();
//6资源关闭（会话）
em.close();
}
```

## 3. JPA 常用注解

### 3.1. @Entity

`@Entity` 标注用于实体类声明语句之前，指出该 Java 类为实体类，将映射到指定的数据库表。如声明一个实体类 `Users`，它将映射到数据库中的 `users` 表上。

### 3.2. @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` 标注的常用选项是 `name`，用于指明数据库的表名

@Table 标注还有一个两个选项 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。uniqueConstraints 选项用于设置约束条件，通常不须设置

### 3.3. @Id

@Id 标注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。

### 3.4. @GeneratedValue

@GeneratedValue 用于标注主键的生成策略，通过 strategy 属性指定。默认情况下，JPA 自动选择一个最适合底层数据库的主键生成策略

在 javax.persistence.GenerationType 中定义了以下几种可供选择的策略：

- **IDENTITY**: 采用数据库 ID 自增长的方式来自增主键字段 (MySQL)，Oracle 不支持这种方式；
- **AUTO**: JPA 自动选择合适的策略，是默认选项，由程序控制；
- **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，框架借由表模拟序列产生主键，使用该策略可以使应用更易于数据库移植。

### 3.5. @Basic

@Basic 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: 表示该属性的读取策略，有 EAGER 和 LAZY 两种，分别表示主支抓取和延迟加载，默认为 EAGER.

optional: 表示该属性是否允许为 null，默认为 true

## 3.6. @Column

当实体的属性与其映射的数据库表的列不同名时需要使用@Column 标注说明, 该属性通常置于实体的属性声明语句之前, 还可与 @Id 标注一起使用。

@Column 标注的常用属性是 name, 用于设置映射数据库表的列名。此外, 该标注还包含其它多个属性, 如: unique、nullable、length 等。

@Column 标注的 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。

## 3.7. @Transient

表示该属性并非一个到数据库表的字段的映射, ORM 框架将忽略该属性。

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

标注置于属性的 getter 方法之前

# 4. JPA API

## 4.1. Persistence

Persistence 类是用于获取 EntityManagerFactory 实例。该类包含一个名为 createEntityManagerFactory 的静态方法。

createEntityManagerFactory 方法有如下两个重载版本。

带有一个参数的方法以 JPA 配置文件 persistence.xml 中的持久化单元名为参数

带有两个参数的方法: 前一个参数含义相同, 后一个参数 Map 类型, 用于设置 JPA 的相关属性, 这时将忽略其它地方设置的属性。Map 对象的属性名必须是 JPA 实现库提供商的名字空间约定的属性名。

```
Map<String, Object> properties = new HashMap<String, Object>();
properties.put("hibernate.show_sql", false);
EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("jpa", properties);
```

## 4.2. EntityManagerFactory

`EntityManagerFactory` 接口主要用来创建 `EntityManager` 实例。该接口约定了如下 4 个方法：

**`createEntityManager()`**：用于创建实体管理器对象实例。

**`createEntityManager(Map map)`**：用于创建实体管理器对象实例的重载方法，`Map` 参数用于提供 `EntityManager` 的属性。

**`isOpen()`**：检查 `EntityManagerFactory` 是否处于打开状态。实体管理器工厂创建后一直处于打开状态，除非调用 `close()` 方法将其关闭。

**`close()`**：关闭 `EntityManagerFactory`。`EntityManagerFactory` 关闭后将释放所有资源，`isOpen()` 方法测试将返回 `false`，其它方法将不能调用，否则将导致 `IllegalStateException` 异常。

## 4.3. EntityManager

在 JPA 规范中，`EntityManager` 是完成持久化操作的核心对象。实体作为普通 Java 对象，只有在调用 `EntityManager` 将其持久化后才会变成持久化对象。`EntityManager` 对象在一组实体类与底层数据源之间进行 O/R 映射的管理。它可以用来管理和更新 Entity Bean，根据主键查找 Entity Bean，还可以通过 JPQL 语句查询实体。

### 4.3.1. 实体的状态：

**新建状态**：（瞬时状态）新创建的对象，尚未拥有持久性主键。

**持久化状态**：已经拥有持久性主键并和持久化建立了上下文环境

**游离状态**：拥有持久化主键，但是没有与持久化建立上下文环境

**删除状态**：拥有持久化主键，已经和持久化建立上下文环境，但是从数据库中删除。

### 4.3.2. 常用方法 CRUD

**`find(Class<T> entityClass, Object primaryKey)`**：返回指定的 OID 对应的实体类对象，如果这个实体存在于当前的持久化环境，则返回一个被缓存的对象；否则会创建一个新的 Entity，并加载数据库中相关信息；若 OID 不存在于数据库中，则返回一个 `null`。第一个参数为被查询的**实体类类型**，第二个参数为待查找实体的**主键值**。

**`getReference(Class<T> entityClass, Object primaryKey)`**：与 `find()` 方法类

似，不同的是：如果缓存中不存在指定的 Entity，EntityManager 会创建一个 Entity 类的代理，但是不会立即加载数据库中的信息，只有第一次真正使用此 Entity 的属性才加载，所以如果此 OID 在数据库不存在，getReference() 不会返回 null 值，而是抛出 EntityNotFoundException

**persist(Object entity)**：用于将新创建的 Entity 纳入到 EntityManager 的管理。该方法执行后，传入 persist() 方法的 Entity 对象转换成持久化状态。

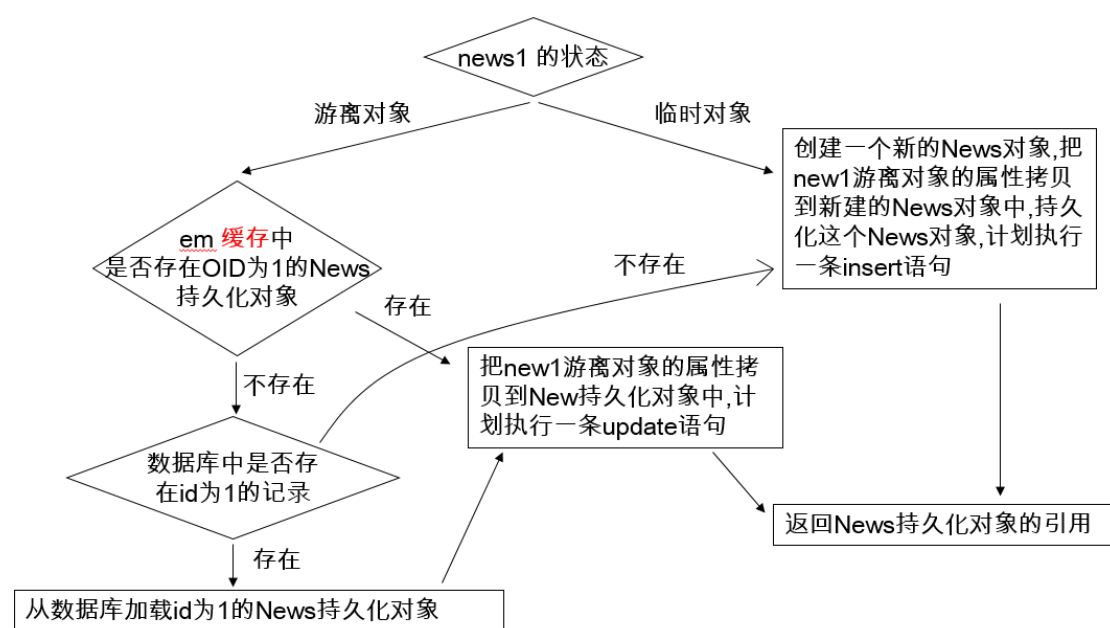
如果传入 persist() 方法的 Entity 对象已经处于持久化状态，则 persist() 方法什么都不做。

如果对删除状态的 Entity 进行 persist() 操作，会转换为持久化状态。

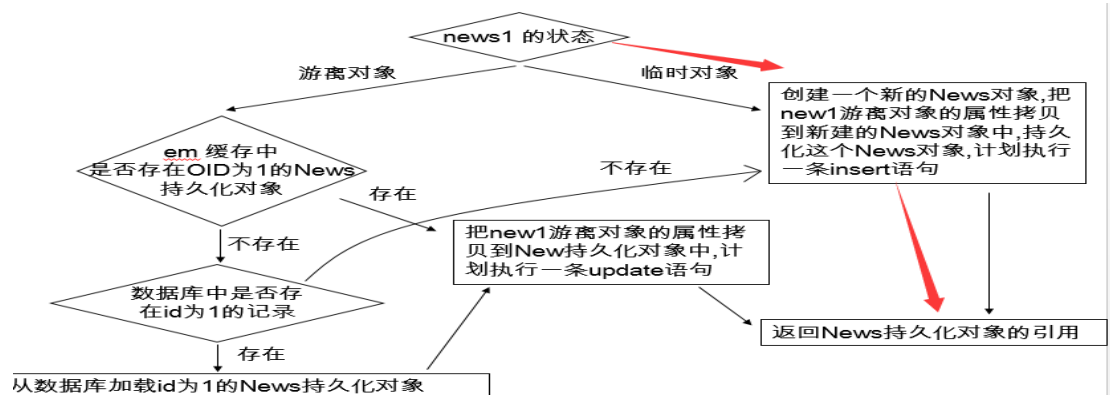
如果对游离状态的实体执行 persist() 操作，可能会在 persist() 方法抛出 EntityExistException(也有可能是在 flush 或事务提交后抛出)。

**remove(Object entity)**：删除实例。如果实例是被管理的，即与数据库实体记录关联，则同时会删除关联的数据库记录。该方法只能移除持久化对象，不能删除游离对象

**merge(T entity)**：merge() 用于处理 Entity 的同步。即数据库的插入和更新操作







```

EntityManager man=JpaUtils.get();
//操作持久化方法
Users u=new Users();
u.setAge(21);
u.setCurrDate(new Date());
u.setEmail("ccc.@sina.com");
u.setUname("dd");

Users u1=man.merge(u);
System.out.println(u.getId());
System.out.println(u1.getId());
JpaUtils.close();

```

**flush ()**: 同步持久上下文环境，即将持久上下文环境的所有未保存实体的状态信息保存到数据库中。

**setFlushMode** (FlushModeType flushMode): 设置持久上下文环境的 Flush 模式。参数可以取 2 个枚举

- FlushModeType.AUTO 为自动更新数据库实体，
- FlushModeType.COMMIT 为直到提交事务时才更新数据库记录。

**getFlushMode ()**: 获取持久上下文环境的 Flush 模式。返回 FlushModeType 类的枚举值。

**refresh** (Object entity): 用数据库实体记录的值更新实体对象的状态，即更新实例的属性值。

**clear ()**: 清除持久上下文环境，断开所有关联的实体。如果这时还有未提交的更新则会被撤消。

**contains** (Object entity): 判断一个实例是否属于当前持久上下文环境管理的实体。

**isOpen ()**: 判断当前的实体管理器是否是打开状态。

**getTransaction ()**: 返回资源层的事务对象。EntityTransaction 实例可以用于开始和提交多个事务。

**close ()**: 关闭实体管理器。之后若调用实体管理器实例的方法或其派生的查询对象的方法都将抛出 IllegalStateException 异常，除了 getTransaction 和 isOpen 方法(返回 false)。不过，当与实体管理器关联的事务处于活动状态时，调用 close 方法后持久上下文将仍处于被管理状态，直到事务完成。



**createQuery**(String qlString): 创建一个查询对象。

**createNamedQuery**(String name): 根据命名的查询语句块创建查询对象。参数为命名的查询语句。

**createNativeQuery**(String sqlString): 使用标准 SQL 语句创建查询对象。参数为标准 SQL 语句字符串。

**createNativeQuery**(String sqls, String resultSetMapping): 使用标准 SQL 语句创建查询对象，并指定返回结果集 Map 的名称。

## 4.4. EntityManager

EntityManager 接口用来管理资源层实体管理器的事务操作。通过调用实体管理器的 getTransaction 方法获得其实例。

**begin()**

用于启动一个事务，此后的多个数据库操作将作为整体被提交或撤销。若这时事务已启动则会抛出 IllegalStateException 异常。

**commit()**

用于提交当前事务。即将事务启动以后的所有数据库更新操作持久化至数据库中。

**rollback()**

撤销(回滚)当前事务。即撤销事务启动后的所有数据库更新操作，从而不对数据库产生影响。

**setRollbackOnly()**

使当前事务只能被撤销。

**getRollbackOnly()**

- **查看**当前事务是否设置了只能撤销标志。