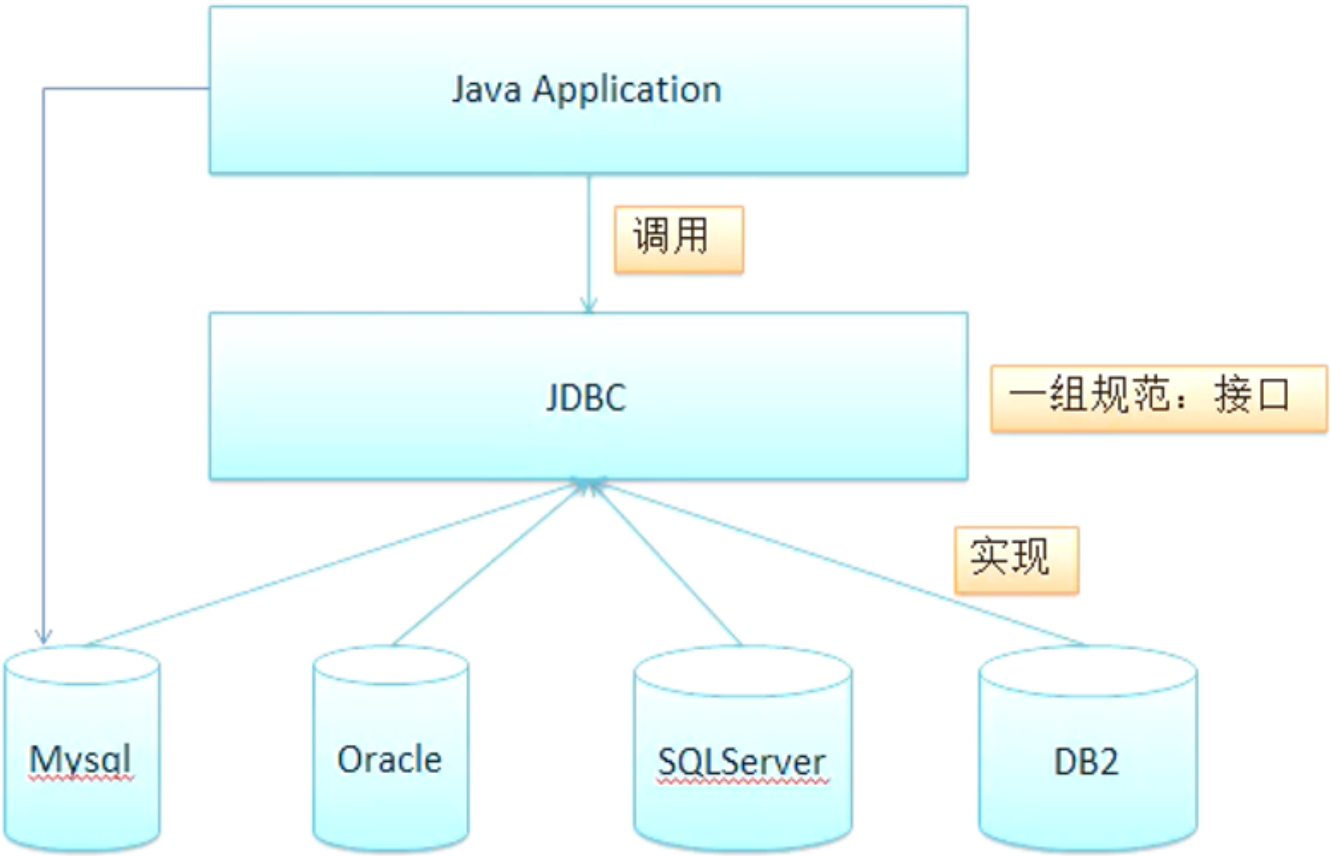
00-JPA 简介

1、JDBC

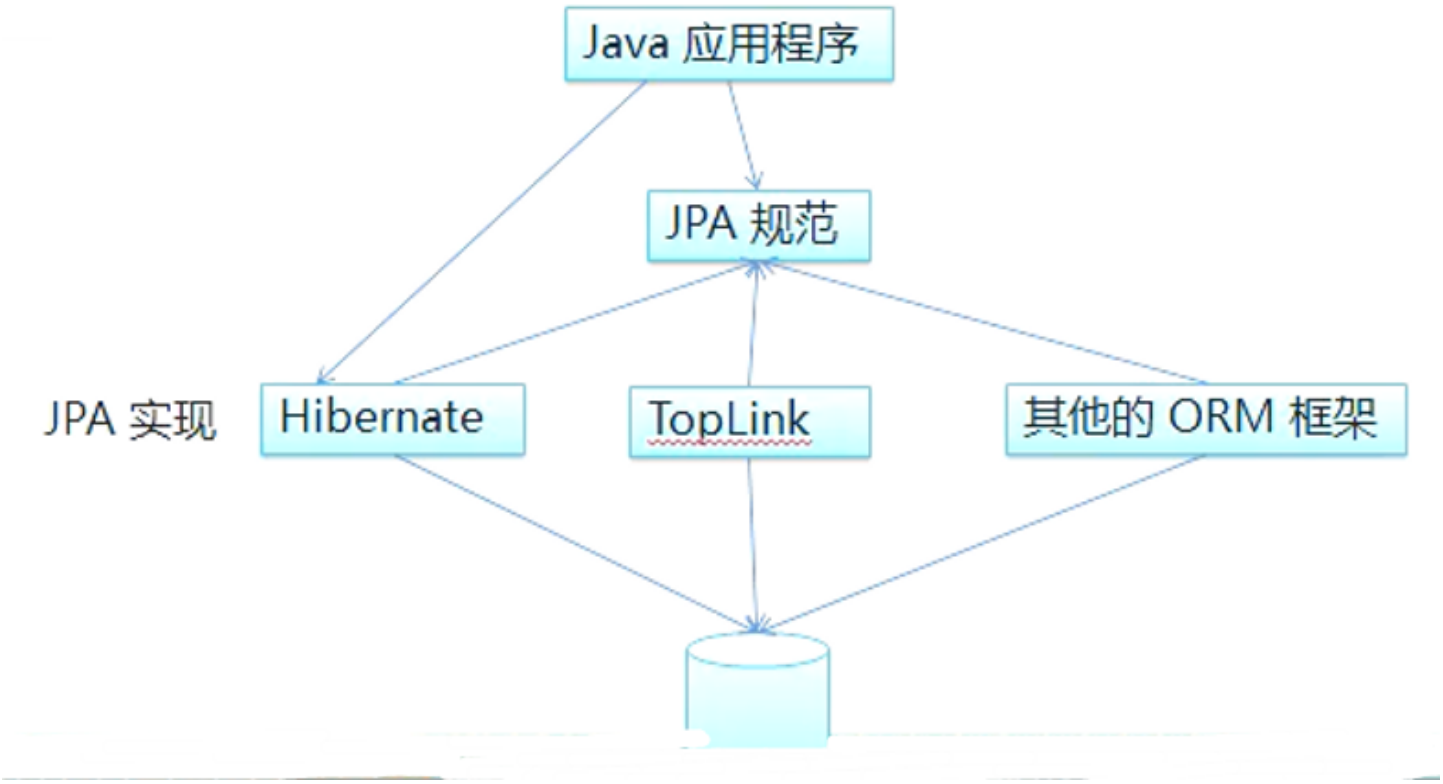
`JDBC`（Java DataBase Connectivity，Java 语言连接数据库），提供了一组规范即接口，由各个数据库厂商提供这些接口的对应实现，而 Java 应用程序只需要通过 `JDBC` 就可以完成对不同数据库（如 MySQL、Oracle、SQLServer 和 DB2 等）的统一调用



2、JPA

`JPA`（Java Persistence API，Java 持久化 API），用于对象持久化的 API

JavaEE5.0 平台标准的 ORM（Object Relational Mapping，对象关系映射）规范，使得应用程序以统一的方式访问持久层



3、JPA 和 Hibernate 的关系

`JPA` 是 `Hibernate` 的一个抽象（就像 JDBC 和 JDBC 驱动的关系）

- JPA 是规范：`JPA` 本质上就是一种 ORM 规范，不是 ORM 框架
  - 因为 `JPA` 并未提供 ORM 实现，它只是制订了一些规范，提供了一些编程的 API 接口，但具体实现则由 ORM 厂商提供实现
- Hibernate 是实现：`Hibernate` 除了作为 ORM 框架之外，它也是一种 `JPA` 实现
- 从功能上来说，JPA 是 Hibernate 功能的一个子集

4、JPA 的供应商

JPA 的目标之一是制定一个可以由很多供应商实现的 API，目前 `Hibernate 3.2+`、`TopLink 10.1+` 以及 `OpenJPA` 都提供了 JPA 的实现

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

5、JPA 的优势

- 标准化：提供相同的 API，这保证了基于 JPA 开发的企业应用能够经过少量的修改就能够在不同的 JPA 框架下运行
- 简单易用，集成方便：JPA 的主要目标之一就是提供更加简单的编程模型，在 JPA 框架下创建实体和创建 Java 类一样简单，只需要使用 `javax.persistence.Entity` 进行注释；JPA 的框架和接口也都非常简单
- 可媲美 JDBC 的查询能力：JPA 的查询语言是面向对象的，JPA 定义了独特的 `JPQL`，而且能够支持批量更新和修改、JOIN、GROUP BY、HAVING 等通常只有 SQL 才能够提供的高级查询特性，甚至还能够支持子查询
- 支持面向对象的高级特性：JPA 中能够支持面向对象的高级特性，如类之间的继承、多态和类之间的复杂关系，最大限度的使用面向对象的模型

6、JPA 技术

- ORM 映射元数据：JPA 支持 *XML* 和 *JDK5.0 注解* 两种元数据的形式，元数据描述对象和表之间的映射关系，框架据此将实体对象持久化到数据库表中
- JPA 的 API：用来操作实体对象，执行 CRUD 操作，框架在后台完成所有的事情，开发者从繁琐的 JDBC 和 SQL 代码中解脱出来
- 查询语言（JPQL）：这是持久化操作中很重要的一个方面，通过面向对象而非面向数据库的查询语言查询数据，避免程序和具体的 SQL 紧密耦合

7、JPA 持久化对象步骤

1、创建 persistence.xml，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

2、创建实体类，使用 annotation 描述实体类与数据库表间映射关系

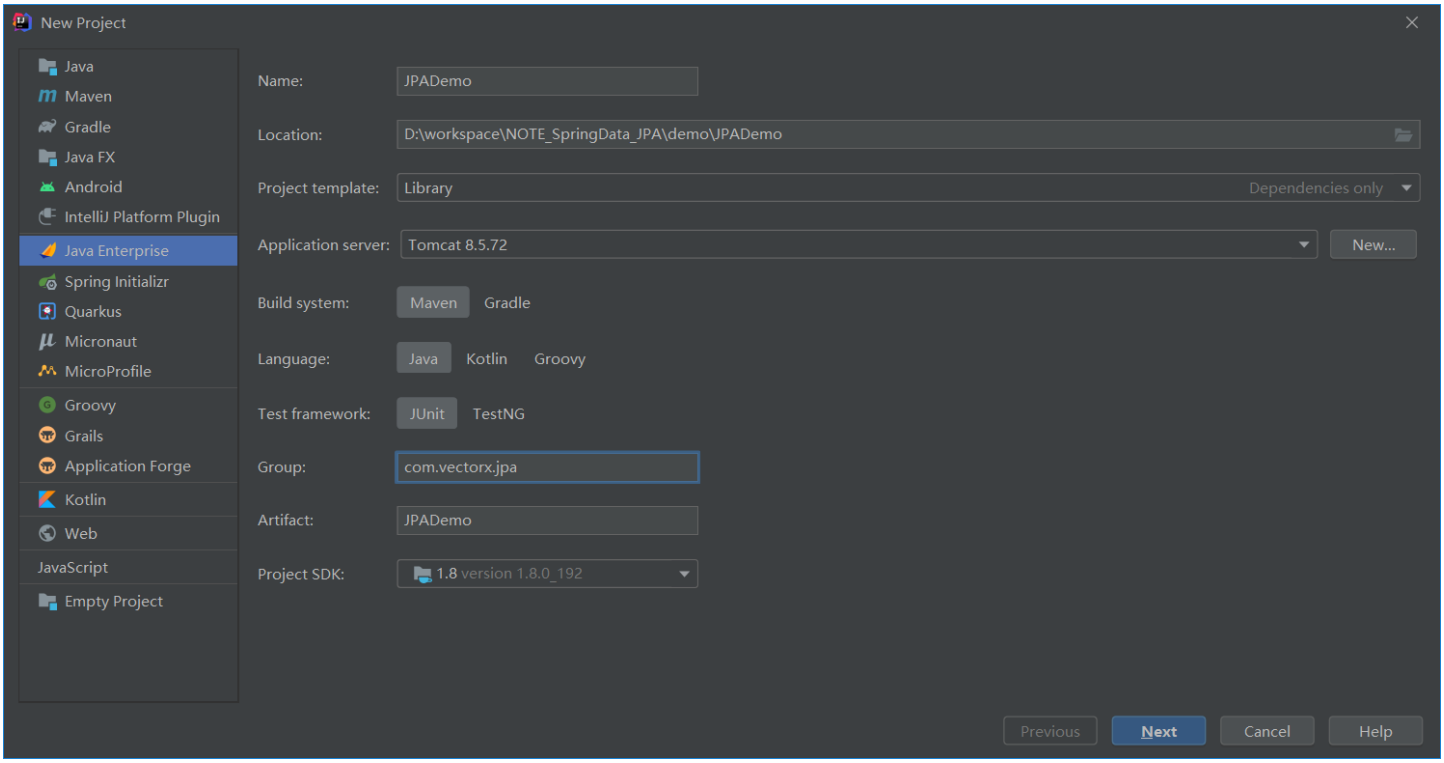
3、使用 JPA API 完成数据增删改查操作

- 创建 `EntityManagerFactory`（对应 Hibernate 中的 SessionFactory）
- 创建 `EntityManager`（对应 Hibernate 中的 Session）

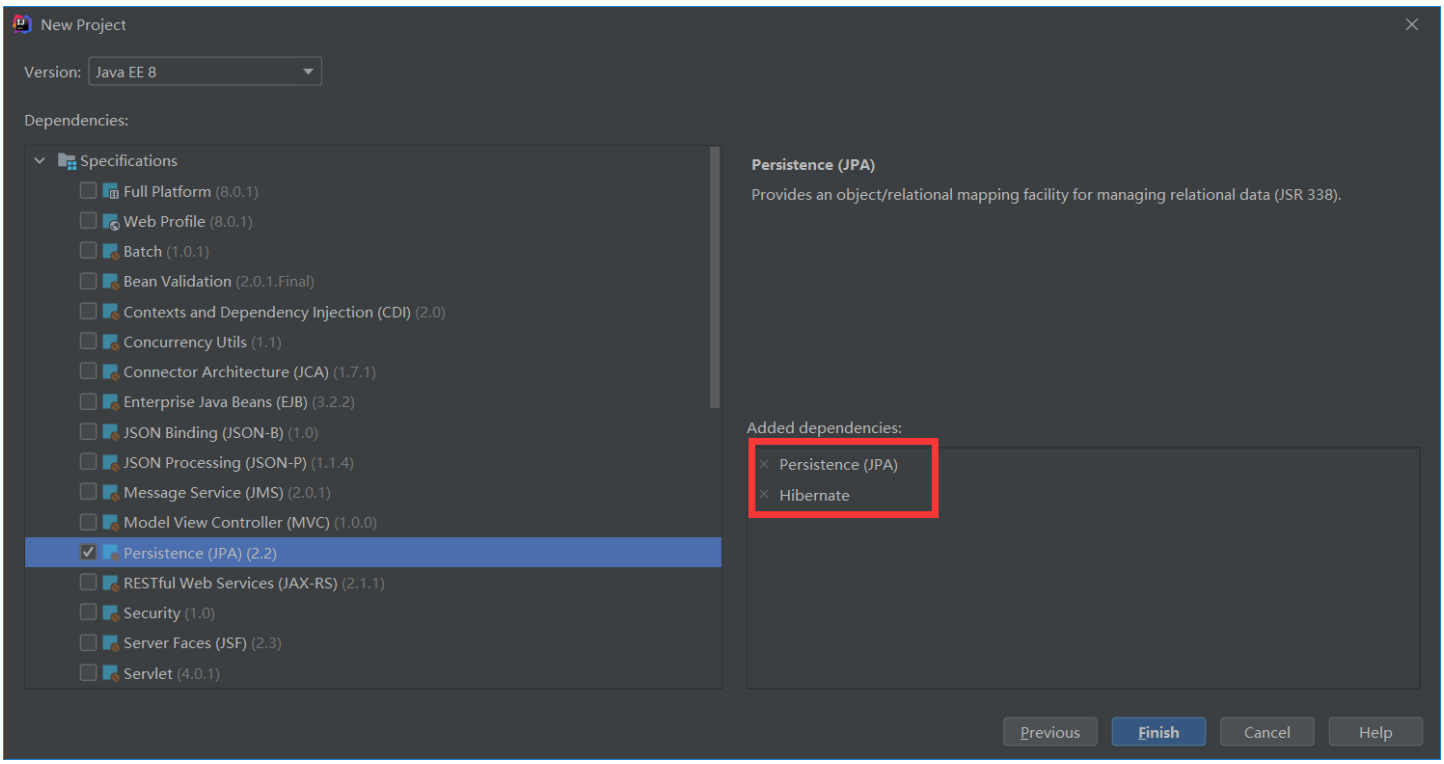
HelloWorld

1）创建 JPA 工程

1、填写工程相关信息（PS：IDEA 貌似不能像 Eclipse 直接创建 JPA 工程）



2、勾选 `Persistence（JPA）` 和 `Hibernate` （需要注意的是这里 JPA 版本默认是 2.2，后面需要修改配置文件），点击 `FINISH` 创建完成



2）配置 POM

相关依赖，清单如下

- `antlr`

- `dom4j`
- `hibernate-commons-annotations`
- `hibernate-core`
- `hibernate-entitymanager`
- `hibernate-jpa-2.0-api`
- `javassist`
- `jboss-logging`
- `jboss-transaction-api_1.1_spec`
- `mysql-connector-java`

`pom.xml` 中添加依赖

```
<dependency>
    <groupId>antlr</groupId>
    <artifactId>antlr</artifactId>
    <version>2.7.7</version>
</dependency>
<dependency>
    <groupId>dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>1.6.1</version>
</dependency>
<dependency>
    <groupId>org.hibernate.common</groupId>
    <artifactId>hibernate-commons-annotations</artifactId>
    <version>4.0.2.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.2.4.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>4.2.4.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate.javax.persistence</groupId>
    <artifactId>hibernate-jpa-2.0-api</artifactId>
    <version>1.0.1.Final</version>
</dependency>
```

```
<dependency>
    <groupId>org.javassist</groupId>
    <artifactId>javassist</artifactId>
    <version>3.15.0-GA</version>
</dependency>
<dependency>
    <groupId>org.jboss.logging</groupId>
    <artifactId>jboss-logging</artifactId>
    <version>3.1.0.GA</version>
</dependency>
<dependency>
    <groupId>org.jboss.spec.javax.transaction</groupId>
    <artifactId>jboss-transaction-api_1.1_spec</artifactId>
    <version>1.0.1.Final</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>
```

3）配置 persistence

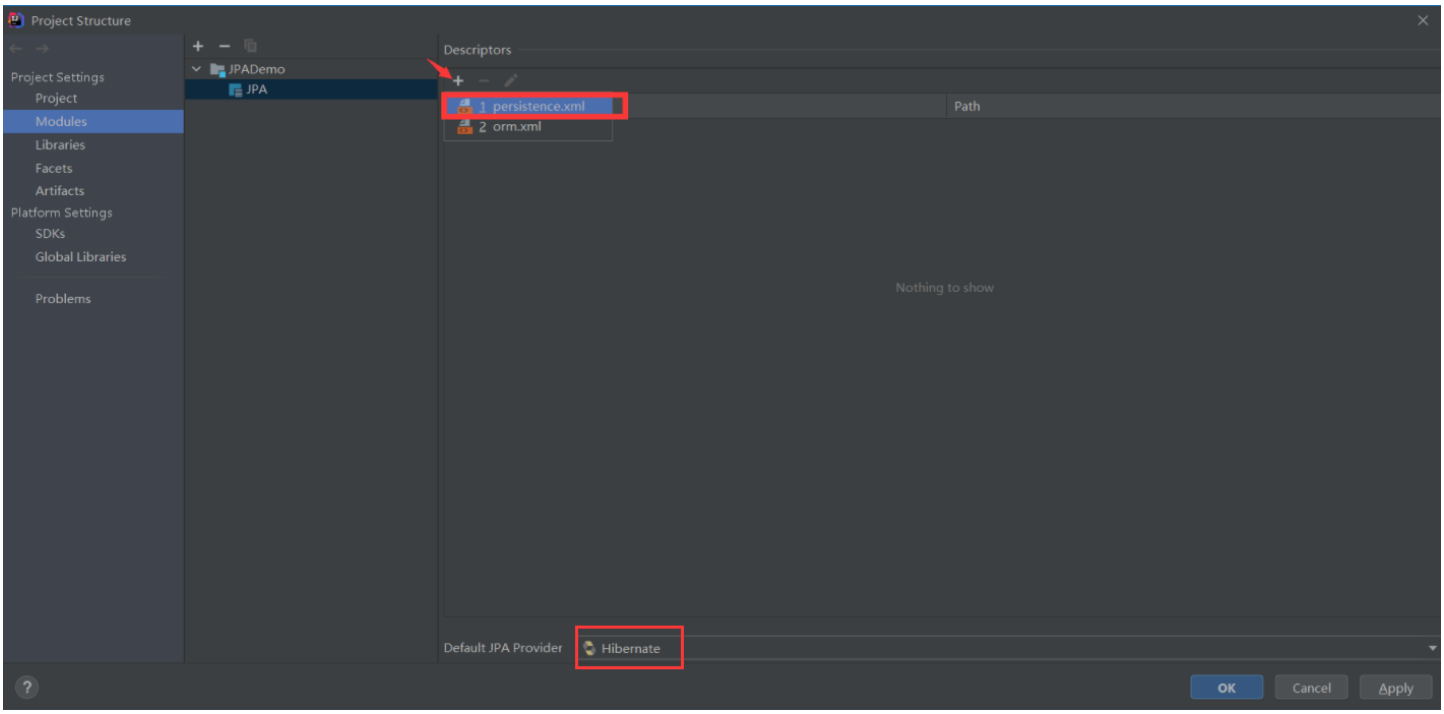
> **注意**：JPA 规范要求在类路径的 META-INF 目录下放置 persistence.xml，文件的名称是固定的。换句话说，
>
> persistence.xml 的名称和位置都是固定的

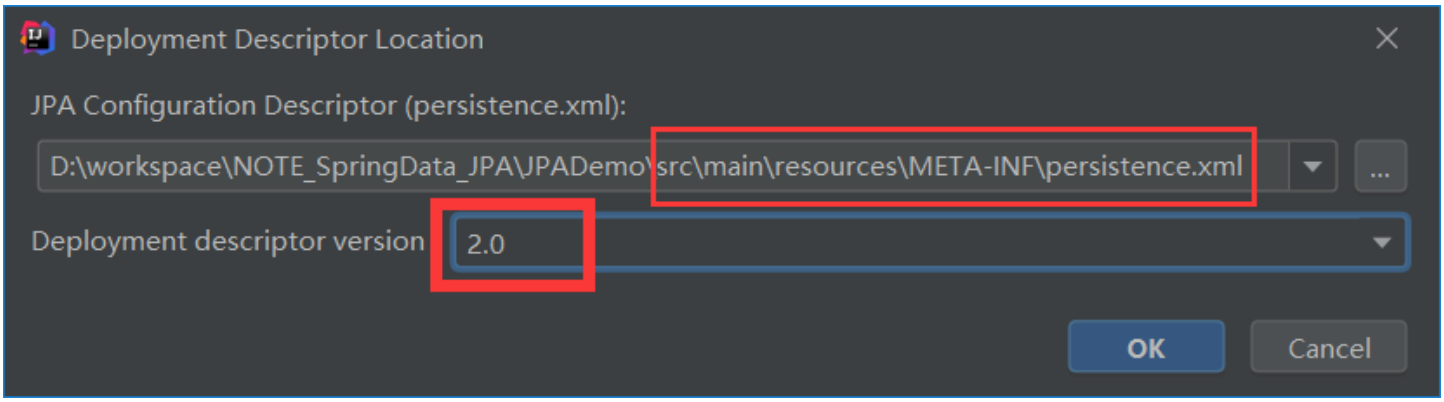
1、进入工程后，查看 `persistence.xml`，其默认内容如下（可以发现，这里版本为 2.2，需要修改）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence 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_2.xsd"
             version="2.2">
  <persistence-unit name="default">

  </persistence-unit>
</persistence>
```

进入工程结构，删除并新建一个 `persistence.xml`（如果想直接替换整个 xml 内容，此步也可跳过）





配置完成后，`persistence.xml`默认内容如下

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">


</persistence>
```

2、修改`persistence.xml`文件，添加如下配置信息

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">
  <persistence-unit name="jpa-1" transaction-type="RESOURCE_LOCAL">
    <!--
    配置使用什么 ORM 产品来作为 JPA 的实现
    1.实际上配置的是 javax.persistence.spi.PersistenceProvider 接口的实现类
    2.若 JPA 项目中只有一个 JPA 的实现产品，则也可以不配置该节点。
    -->
```

```
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <!--添加持久化类-->
    <class>com.vectorx.jpa.helloworld.Customer</class>
    <properties>
      <!--连接数据库基本信息-->
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql:///jpa"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.password" value="root"/>
      <!--配置 JPA 实现产品的基本属性。配置 hibernate 的基本属性-->
      <property name="hibernate.format_sql" value="true"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
    </properties>
  </persistence-unit>
</persistence>
```

配置信息详解：

- `<persistence-unit>` 标签中 `name` 属性：用于自定义持久化单元的名称，必选
- `<persistence-unit>` 标签中 `transaction-type` 属性：用于指定 JPA 的事务处理策略
  - `RESOURCE_LOCAL`：默认值，数据库级别事务。只能针对一种数据库，不支持分布式事务
  - 如果想要支持分布式事务，使用 JTA：`transaction-type="JTA"`
- `<provider>` 标签：显式列出实体类
- `<property>` 标签中 `name` 为 `javax.persistence.jdbc.xxx` 系列配置：连接数据库基本信息
- `<property>` 标签中 `name` 为 `hibernate.xxx` 系列配置：ORM 框架的基本信息

4）创建实体类

```
// 标识为持久化类
@Entity
// 设置关联数据表
@Table(name = "JPA_CUSTOMERS")
```

```
public class Customer {
    private Integer id;
    private String lastName;
    private String email;
    private int age;

    // 标识为主键
    @Id
    // 设置生成策略为 AUTO
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    // 设置关联字段名，若同名可省略
    @Column(name = "LAST_NAME")
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

关于注解作用的详细说明：

- `@Entity`：标识当前类为持久化类
- `@Table`：与数据库表建立关联关系，指定 `name` 为对应的表名
- `@Id`：标识当前属性为数据表主键字段
- `@GeneratedValue`：主键生成策略，指定字段的生成方式，指定 `strategy` 为 `TABLE`、`SEQUENCE`、`IDENTITY`、`AUTO` 其中之一，其中
  - `AUTO`：可以根据底层数据库的类型，自动选用主键生成方式
  - `IDENTITY`：显式地指定数据表主键字段自增
- `@Column`：与数据表字段建立关联关系，如果属性名与字段名一致，则该注解可以省略不写

5）测试 JPA API

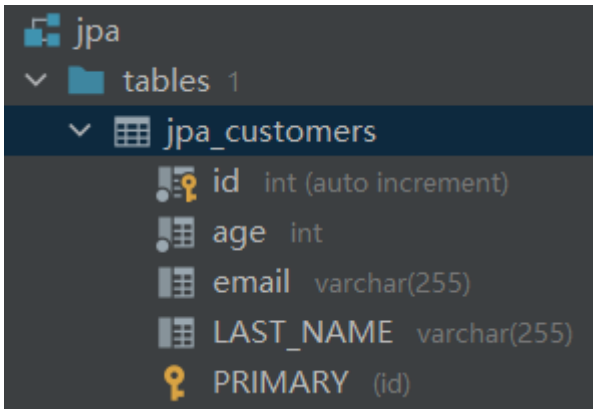
```
//1、创建 EntityManagerFactory
String persistenceUnitName = "jpa-1";
EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory(persistenceUnitName);
//2、创建 EntityManager
EntityManager entityManager = entityManagerFactory.createEntityManager();
//3、开启事务
EntityTransaction transaction = entityManager.getTransaction();
```

```
transaction.begin();
//4、进行持久化操作
Customer customer = new Customer();
customer.setLastName("Vector");
customer.setEmail("vector@qq.com");
customer.setAge(100);
entityManager.persist(customer);
//5、提交事务
transaction.commit();
//6、关闭 EntityManager
entityManager.close();
//6、关闭 EntityManagerFactory
entityManagerFactory.close();
```

后台日志信息，主要内容如下

```
Hibernate:
    insert
    into
        JPA_CUSTOMERS
        (age, email, LAST_NAME)
    values
        (?, ?, ?)
```

查看数据库表结构生成情况



查看数据表数据是否添加成功

	id	age	email	LAST_NAME
1	1	100	vector@qq.com	Vector

可以看到，数据表结构和数据已分别成功生成和添加

总结

本节重点关注：

- 熟悉 JPA 的基本概念与技术定位
- 掌握 JPA 的核心技术点，包括基本注解与其 API
- 熟练 JPA 持久化对象的基本步骤

附上导图，仅供参考