1、Spring Data JPA 环境搭建

源码：

https://github.com/zxjin001/Spring-Data-Jpa-Builder

简介：

官网：https://spring.io/projects/spring-data
开源中国：https://www.oschina.net/p/spring-data

Spring Data 项目的目的是为了简化构建基于 Spring 框架应用的数据访问计数，包括非关系数据库、Map-Reduce 框架、云数据服务等等；另外也包含对关系数据库的访问支持

Spring Data 是 Spring 框架的一个子项目，用于简化基于 Spring 框架的应用的数据访问操作

Spring Data JPA 属于 Spring Data 的子项目，用于简化 JPA 的持久层开发

Hibernate、JPA 和 Spring Data JPA：

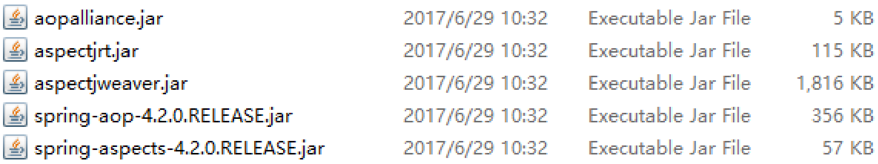
Hibernate：底层封装的是 JDBC 代码，提供一个 ORM 思想框架
JPA：一个 ORM 框架的规范，是接口，底层可以调用 Hibernate 框架
Spring Data JPA：底层调用 JPA 框架，为了简化 JPA 的开发

Spring Data JPA：

持久层开发，无需任何实现类，只需要接口即可

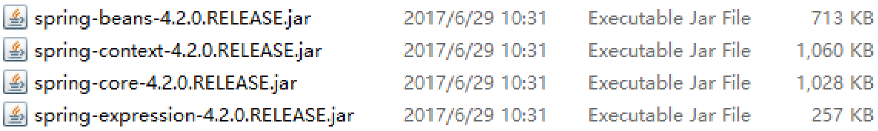
Spring 整合 Hibernate：
导入 jar 包：

spring-aop



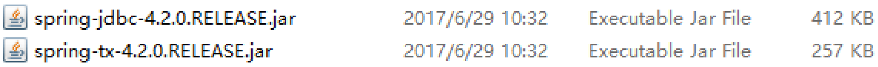
aopalliance.jar	2017/6/29 10:32	Executable Jar File	5 KB
aspectjrt.jar	2017/6/29 10:32	Executable Jar File	115 KB
aspectjweaver.jar	2017/6/29 10:32	Executable Jar File	1,816 KB
spring-aop-4.2.0.RELEASE.jar	2017/6/29 10:32	Executable Jar File	356 KB
spring-aspects-4.2.0.RELEASE.jar	2017/6/29 10:32	Executable Jar File	57 KB

spring-ioc



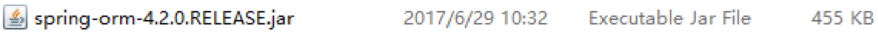
spring-beans-4.2.0.RELEASE.jar	2017/6/29 10:31	Executable Jar File	713 KB
spring-context-4.2.0.RELEASE.jar	2017/6/29 10:31	Executable Jar File	1,060 KB
spring-core-4.2.0.RELEASE.jar	2017/6/29 10:31	Executable Jar File	1,028 KB
spring-expression-4.2.0.RELEASE.jar	2017/6/29 10:31	Executable Jar File	257 KB

spring-jdbc



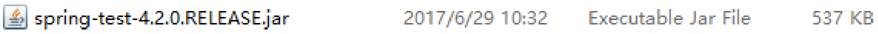
spring-jdbc-4.2.0.RELEASE.jar	2017/6/29 10:32	Executable Jar File	412 KB
spring-tx-4.2.0.RELEASE.jar	2017/6/29 10:32	Executable Jar File	257 KB

spring-orm



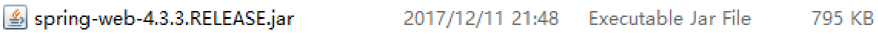
spring-orm-4.2.0.RELEASE.jar	2017/6/29 10:32	Executable Jar File	455 KB

spring-test



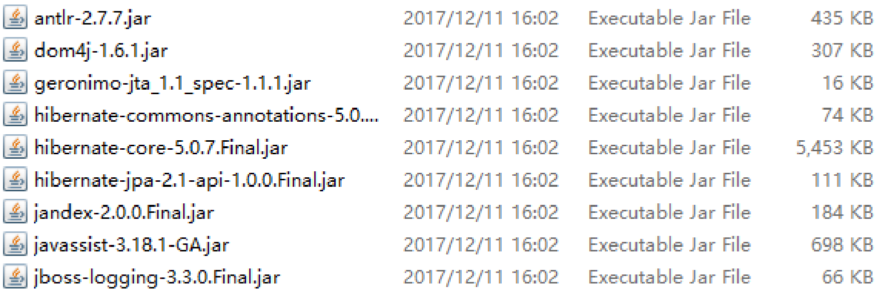
spring-test-4.2.0.RELEASE.jar	2017/6/29 10:32	Executable Jar File	537 KB

spring-web



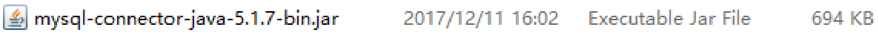
spring-web-4.3.3.RELEASE.jar	2017/12/11 21:48	Executable Jar File	795 KB

hibernate-required



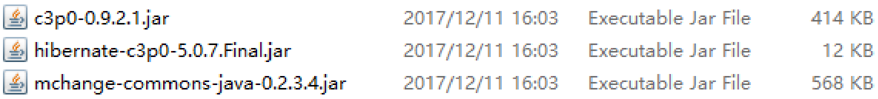
antlr-2.7.7.jar	2017/12/11 16:02	Executable Jar File	435 KB
dom4j-1.6.1.jar	2017/12/11 16:02	Executable Jar File	307 KB
geronimo-jta_1.1_spec-1.1.1.jar	2017/12/11 16:02	Executable Jar File	16 KB
hibernate-commons-annotations-5.0....	2017/12/11 16:02	Executable Jar File	74 KB
hibernate-core-5.0.7.Final.jar	2017/12/11 16:02	Executable Jar File	5,453 KB
hibernate-jpa-2.1-api-1.0.0.Final.jar	2017/12/11 16:02	Executable Jar File	111 KB
jandex-2.0.0.Final.jar	2017/12/11 16:02	Executable Jar File	184 KB
javassist-3.18.1-GA.jar	2017/12/11 16:02	Executable Jar File	698 KB
jboss-logging-3.3.0.Final.jar	2017/12/11 16:02	Executable Jar File	66 KB

mysql-driver



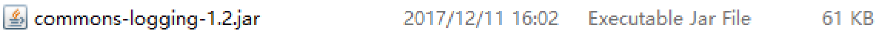
mysql-connector-java-5.1.7-bin.jar	2017/12/11 16:02	Executable Jar File	694 KB

c3p0



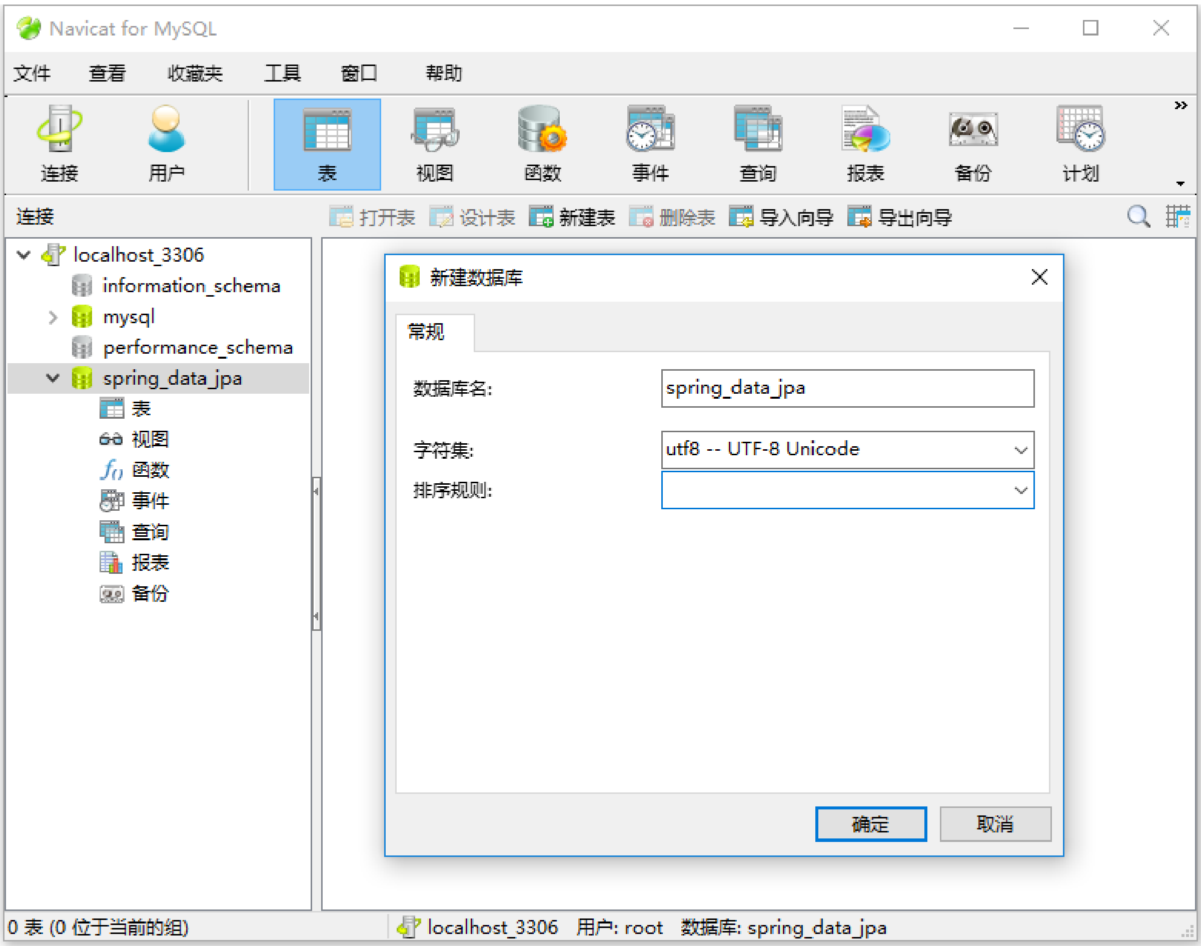
c3p0-0.9.2.1.jar	2017/12/11 16:03	Executable Jar File	414 KB
hibernate-c3p0-5.0.7.Final.jar	2017/12/11 16:03	Executable Jar File	12 KB
mchange-commons-java-0.2.3.4.jar	2017/12/11 16:03	Executable Jar File	568 KB

appache-logging



commons-logging-1.2.jar	2017/12/11 16:02	Executable Jar File	61 KB

建库：



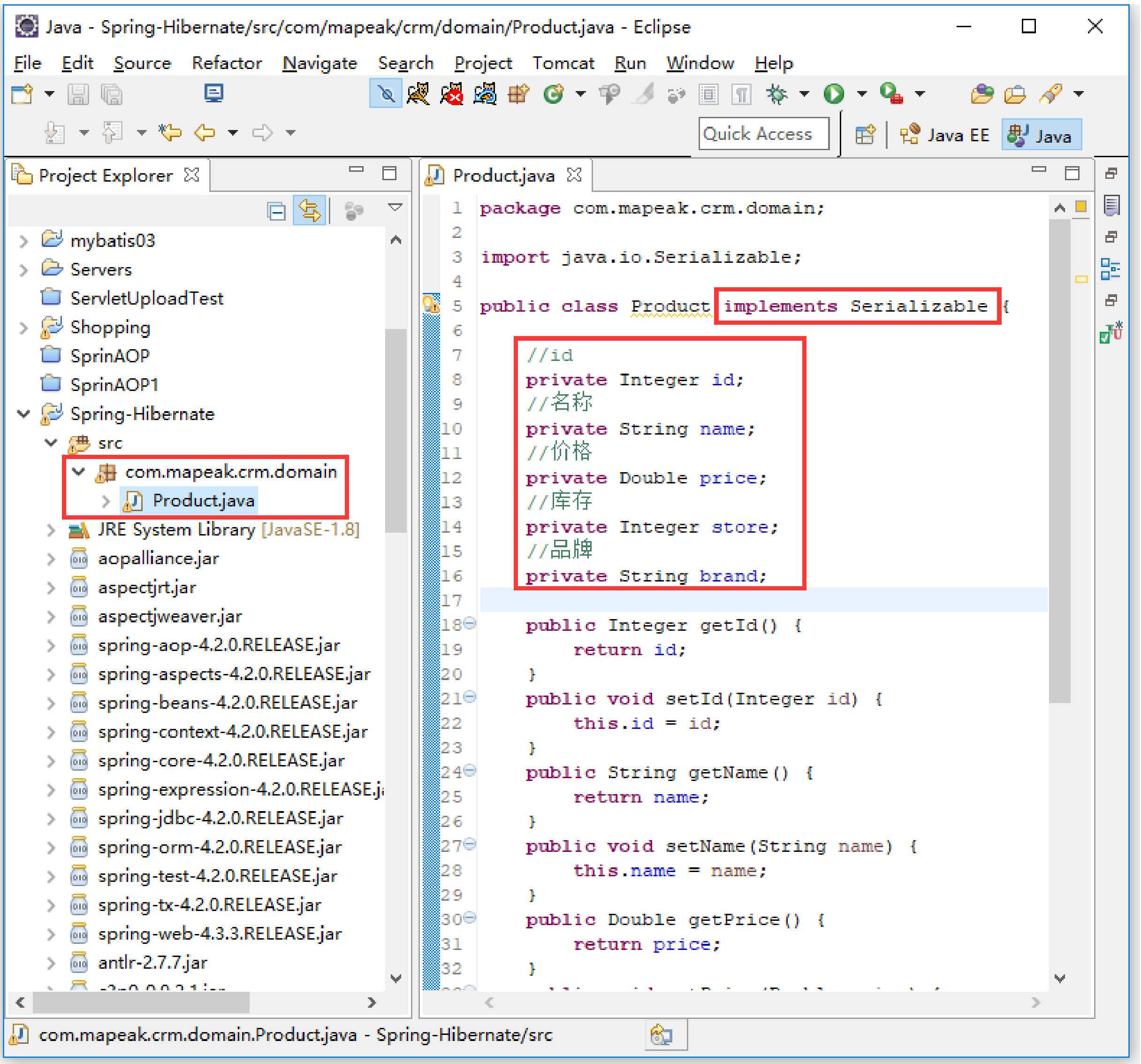
数据库名: spring_data_jpa

字符集: utf8 -- UTF-8 Unicode

排序规则:

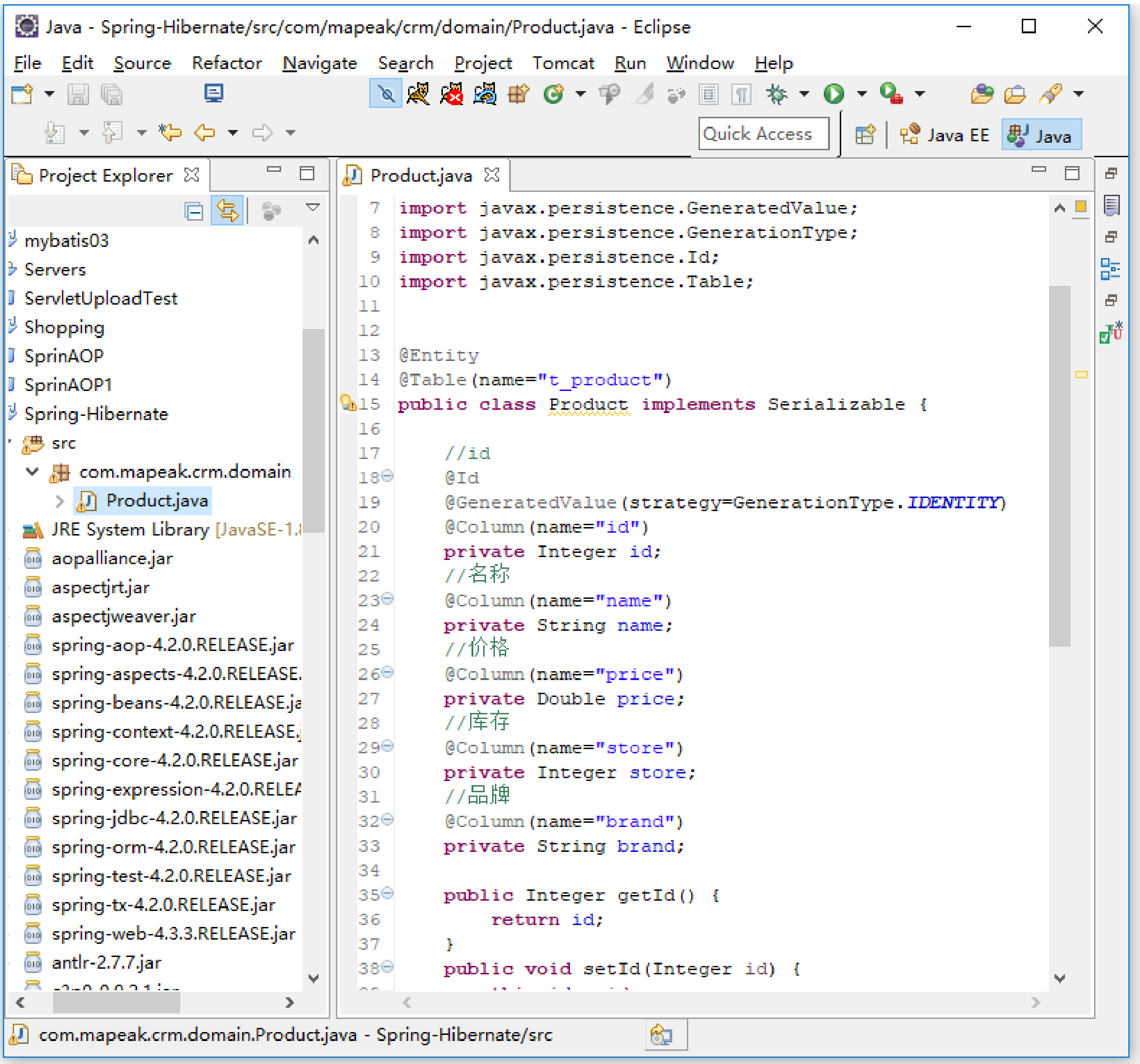
0 表 (0 位于当前的组) localhost_3306 用户: root 数据库: spring_data_jpa

创建实体:


Java - Spring-Hibernate/src/com/mapeak/crm/domain/Product.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Project Explorer
mybatis03
Servers
ServletUploadTest
Shopping
SprinAOP
SprinAOP1
Spring-Hibernate
src
com.mapeak.crm.domain
Product.java
JRE System Library [JavaSE-1.8]
aopalliance.jar
aspectjrt.jar
aspectjweaver.jar
spring-aop-4.2.0.RELEASE.jar
spring-aspects-4.2.0.RELEASE.jar
spring-beans-4.2.0.RELEASE.jar
spring-context-4.2.0.RELEASE.jar
spring-core-4.2.0.RELEASE.jar
spring-expression-4.2.0.RELEASE.j
spring-jdbc-4.2.0.RELEASE.jar
spring-orm-4.2.0.RELEASE.jar
spring-test-4.2.0.RELEASE.jar
spring-tx-4.2.0.RELEASE.jar
spring-web-4.3.3.RELEASE.jar
antlr-2.7.7.jar
Product.java
com.mapeak.crm.domain.Product.java - Spring-Hibernate/src

```java
package com.mapeak.crm.domain;

import java.io.Serializable;

public class Product implements Serializable {

    //id
    private Integer id;
    //名称
    private String name;
    //价格
    private Double price;
    //库存
    private Integer store;
    //品牌
    private String brand;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Double getPrice() {
        return price;
    }
```

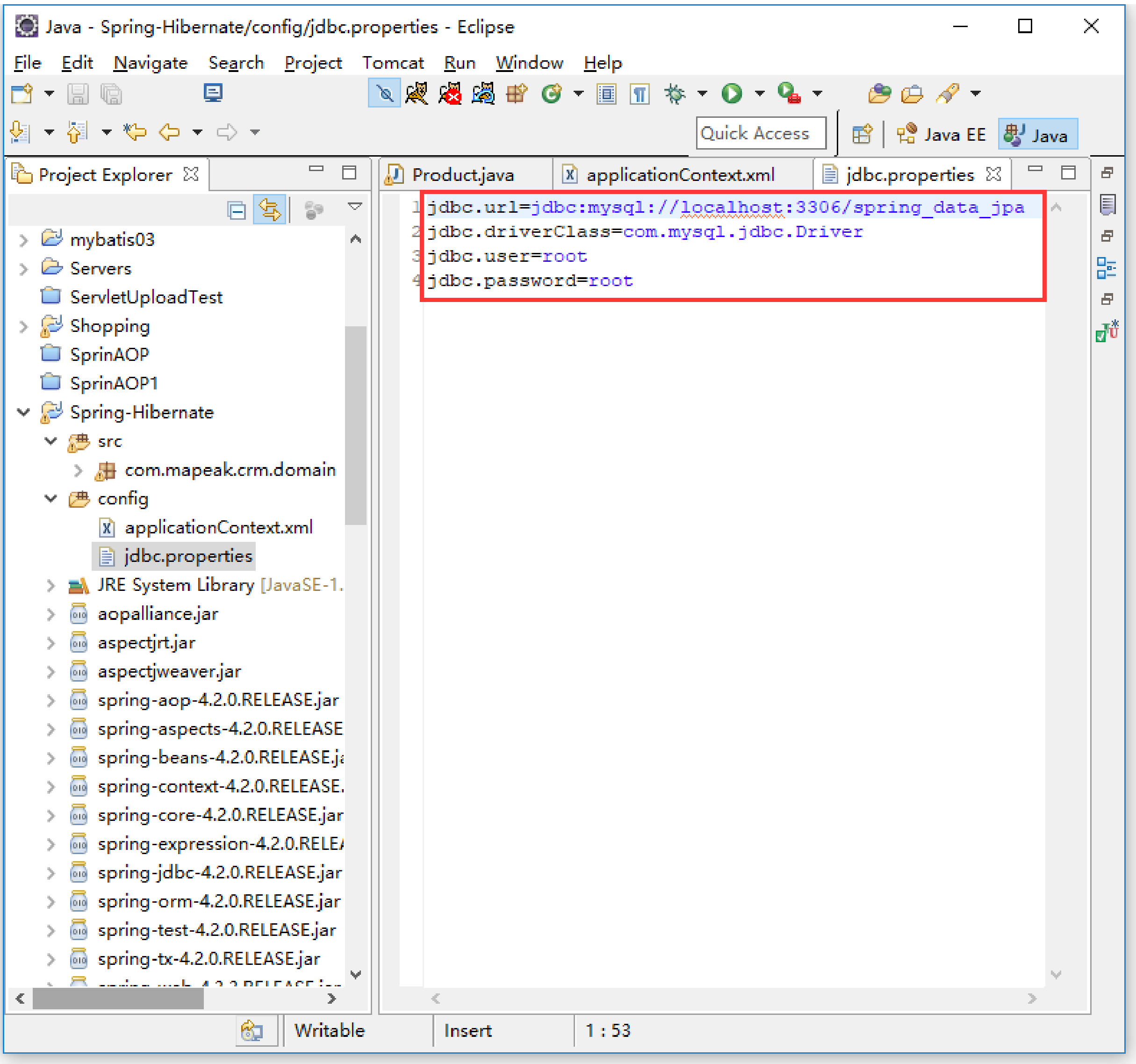
对象映射；


Java - Spring-Hibernate/src/com/mapeak/crm/domain/Product.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Project Explorer
Product.java
mybatis03
Servers
ServletUploadTest
Shopping
SprinAOP
SprinAOP1
Spring-Hibernate
src
com.mapeak.crm.domain
Product.java
JRE System Library [JavaSE-1.
aopalliance.jar
aspectjrt.jar
aspectjweaver.jar
spring-aop-4.2.0.RELEASE.jar
spring-aspects-4.2.0.RELEASE.
spring-beans-4.2.0.RELEASE.ja
spring-context-4.2.0.RELEASE.j
spring-core-4.2.0.RELEASE.jar
spring-expression-4.2.0.RELEA
spring-jdbc-4.2.0.RELEASE.jar
spring-orm-4.2.0.RELEASE.jar
spring-test-4.2.0.RELEASE.jar
spring-tx-4.2.0.RELEASE.jar
spring-web-4.3.3.RELEASE.jar
antlr-2.7.7.jar
com.mapeak.crm.domain.Product.java - Spring-Hibernate/src

```java
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;


@Entity
@Table(name="t_product")
public class Product implements Serializable {

    //id
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private Integer id;
    //名称
    @Column(name="name")
    private String name;
    //价格
    @Column(name="price")
    private Double price;
    //库存
    @Column(name="store")
    private Integer store;
    //品牌
    @Column(name="brand")
    private String brand;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
```

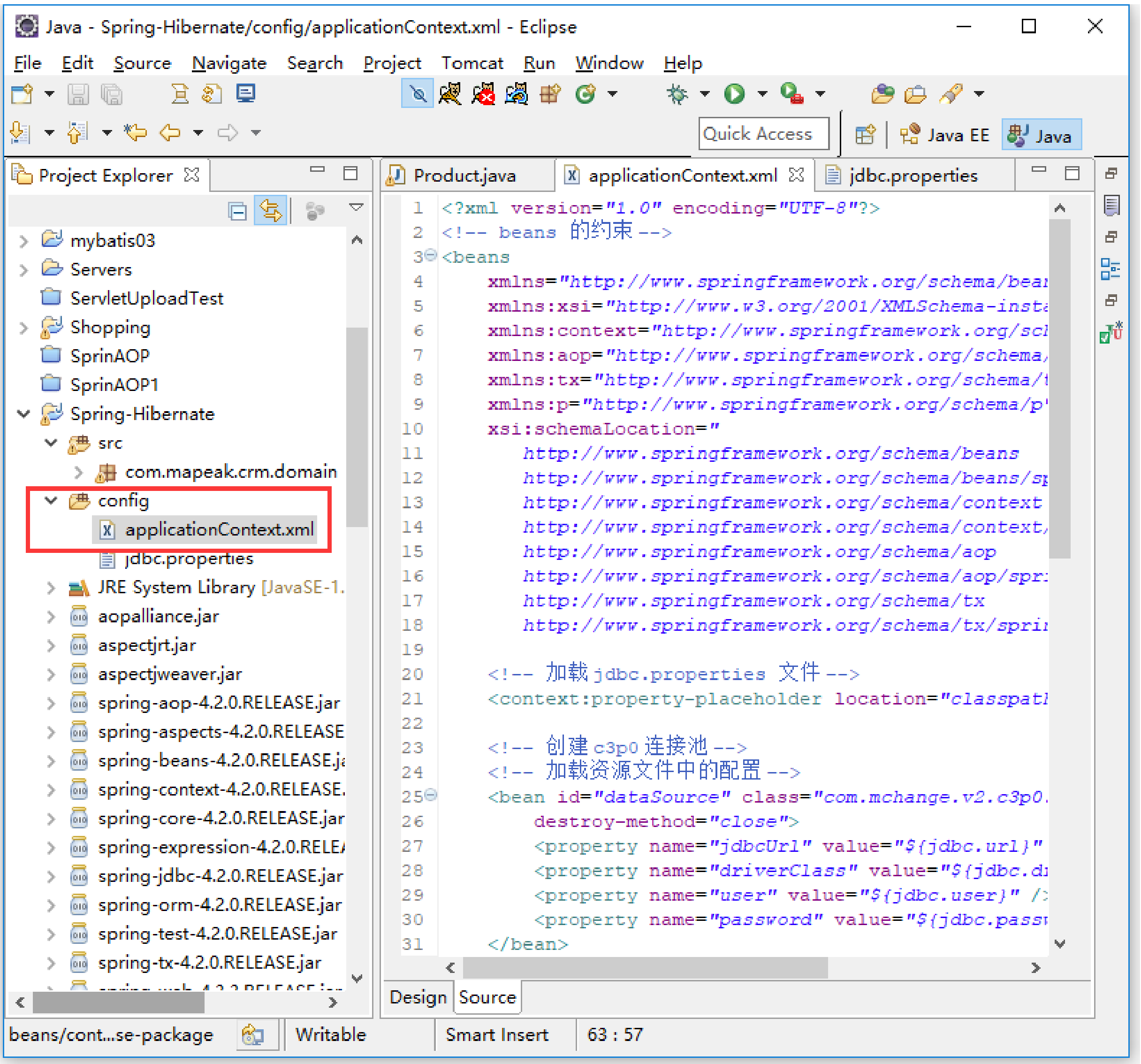
spring 配置文件：

applicationContext.xml



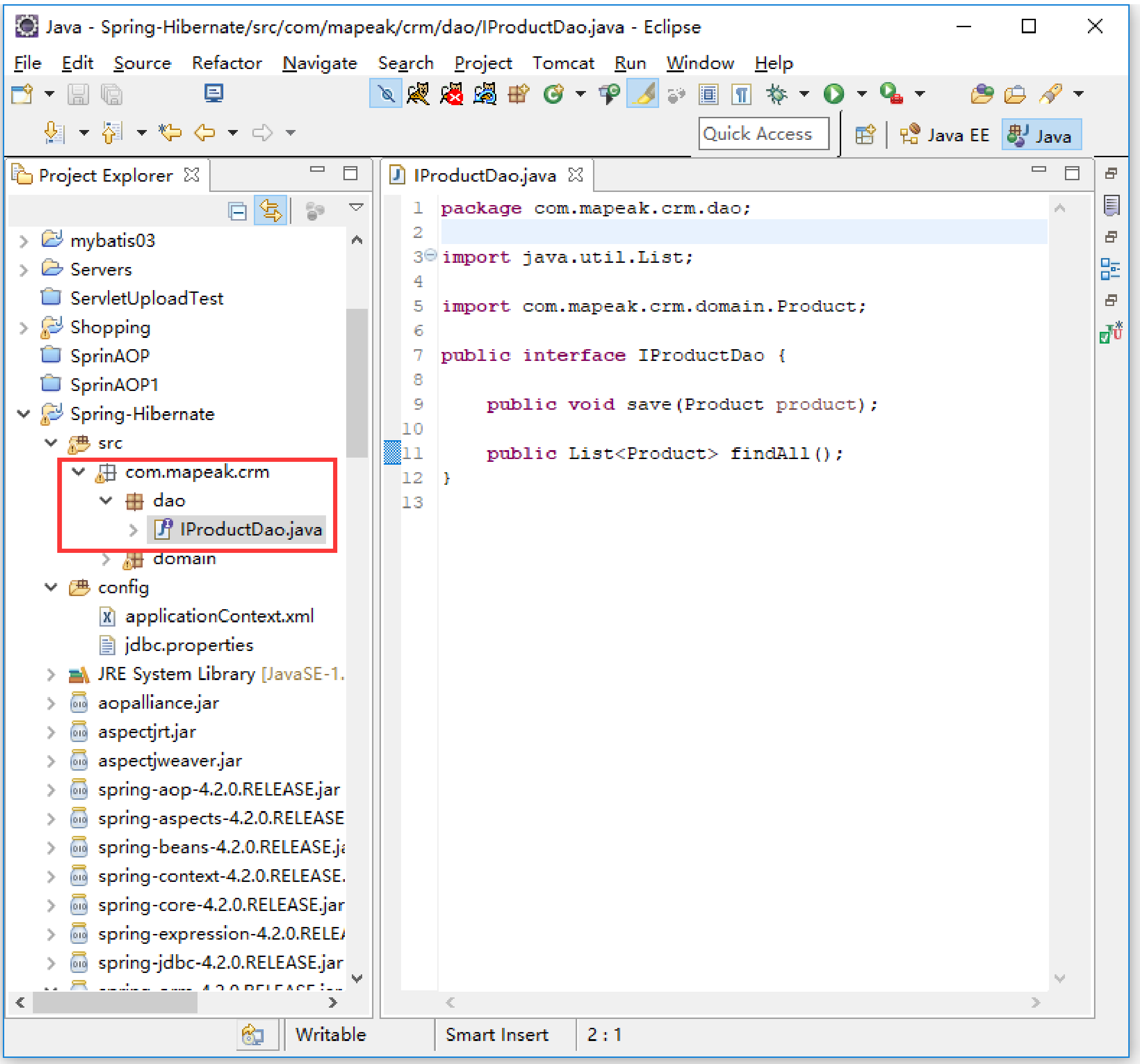
Java - Spring-Hibernate/config/jdbc.properties - Eclipse
File Edit Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Project Explorer
mybatis03
Servers
ServletUploadTest
Shopping
SprinAOP
SprinAOP1
Spring-Hibernate
src
com.mapeak.crm.domain
config
applicationContext.xml
jdbc.properties
JRE System Library [JavaSE-1.
aopalliance.jar
aspectjrt.jar
aspectjweaver.jar
spring-aop-4.2.0.RELEASE.jar
spring-aspects-4.2.0.RELEASE
spring-beans-4.2.0.RELEASE.ja
spring-context-4.2.0.RELEASE.
spring-core-4.2.0.RELEASE.jar
spring-expression-4.2.0.RELEA
spring-jdbc-4.2.0.RELEASE.jar
spring-orm-4.2.0.RELEASE.jar
spring-test-4.2.0.RELEASE.jar
spring-tx-4.2.0.RELEASE.jar
Product.java
applicationContext.xml
jdbc.properties
Writable
Insert
1 : 53

```
jdbc.url=jdbc:mysql://localhost:3306/spring_data_jpa
jdbc.driverClass=com.mysql.jdbc.Driver
jdbc.user=root
jdbc.password=root
```


Java - Spring-Hibernate/config/applicationContext.xml - Eclipse
File Edit Source Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Project Explorer
mybatis03
Servers
ServletUploadTest
Shopping
SprinAOP
SprinAOP1
Spring-Hibernate
src
com.mapeak.crm.domain
config
applicationContext.xml
jdbc.properties
JRE System Library [JavaSE-1.
aopalliance.jar
aspectjrt.jar
aspectjweaver.jar
spring-aop-4.2.0.RELEASE.jar
spring-aspects-4.2.0.RELEASE
spring-beans-4.2.0.RELEASE.ja
spring-context-4.2.0.RELEASE.
spring-core-4.2.0.RELEASE.jar
spring-expression-4.2.0.RELEA
spring-jdbc-4.2.0.RELEASE.jar
spring-orm-4.2.0.RELEASE.jar
spring-test-4.2.0.RELEASE.jar
spring-tx-4.2.0.RELEASE.jar
Product.java
applicationContext.xml
jdbc.properties
Design
Source
beans/cont...se-package
Writable
Smart Insert
63 : 57

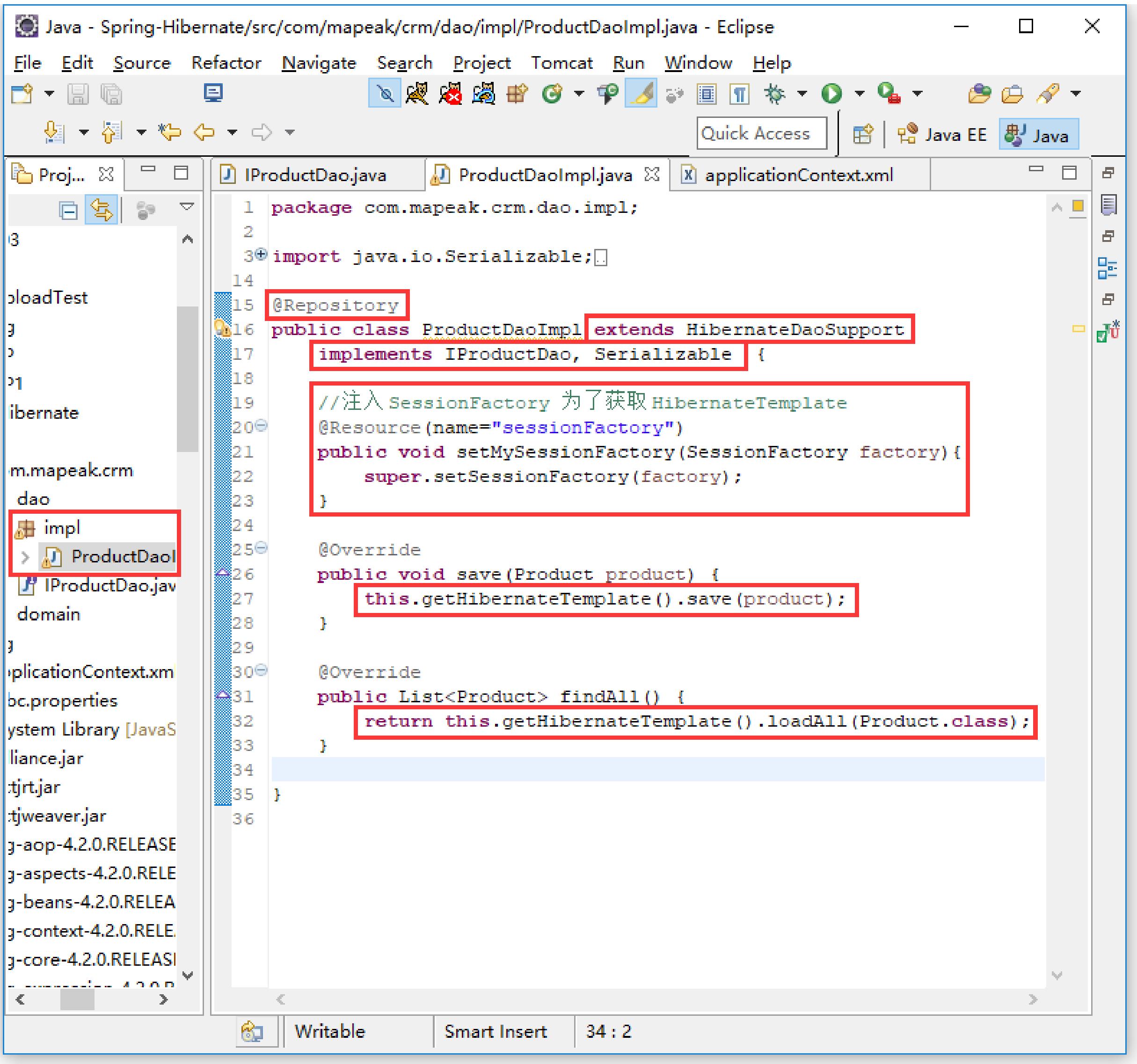
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- beans 的约束 -->
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="
       http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd
        http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd">

    <!-- 加载 jdbc.properties 文件 -->
    <context:property-placeholder location="classpath:jdbc.properties" />

    <!-- 创建 c3p0 连接池 -->
    <!-- 加载资源文件中的配置 -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
        destroy-method="close">
        <property name="jdbcUrl" value="${jdbc.url}" />
        <property name="driverClass" value="${jdbc.driverClass}" />
        <property name="user" value="${jdbc.user}" />
        <property name="password" value="${jdbc.password}" />
    </bean>
    <!-- Spring 整合 Hibernate -->
    <bean id="sessionFactory"
        class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="hibernateProperties">
            <props>
                <!-- 是否输出 SQL 语句 -->
                <prop key="hibernate.show_sql">true</prop>
                <!-- 自动维护表 -->
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
        <property name="packagesToScan">
            <!-- 扫描实体所在的包 -->
            <list>
                <value>com.mapeak.crm.domain</value>
            </list>
        </property>
    </bean>

    <!-- 开启 Spring 的事务管理 -->
    <!-- 使用注解的方式使用事务 -->
    <tx:annotation-driven transaction-manager="transactionManager" />
    <bean id="transactionManager"
        class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>

    <!-- 开启 Spring 注解功能 -->
    <!-- base-package：IOC 注解的类所在包，即包的范围，Spring 也会扫描该包的子包 -->
    <context:component-scan base-package="com.mapeak.crm" />
</beans>
```

DAO:


Java - Spring-Hibernate/src/com/mapeak/crm/dao/IProductDao.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Project Explorer
mybatis03
Servers
ServletUploadTest
Shopping
SprinAOP
SprinAOP1
Spring-Hibernate
src
com.mapeak.crm
dao
IProductDao.java
domain
config
applicationContext.xml
jdbc.properties
JRE System Library [JavaSE-1.
aopalliance.jar
aspectjrt.jar
aspectjweaver.jar
spring-aop-4.2.0.RELEASE.jar
spring-aspects-4.2.0.RELEASE
spring-beans-4.2.0.RELEASE.j
spring-context-4.2.0.RELEASE.
spring-core-4.2.0.RELEASE.jar
spring-expression-4.2.0.RELE
spring-jdbc-4.2.0.RELEASE.jar
IProductDao.java
Writable
Smart Insert
2 : 1

```java
package com.mapeak.crm.dao;

import java.util.List;

import com.mapeak.crm.domain.Product;

public interface IProductDao {

    public void save(Product product);

    public List<Product> findAll();
}
```

DAO-Impl：

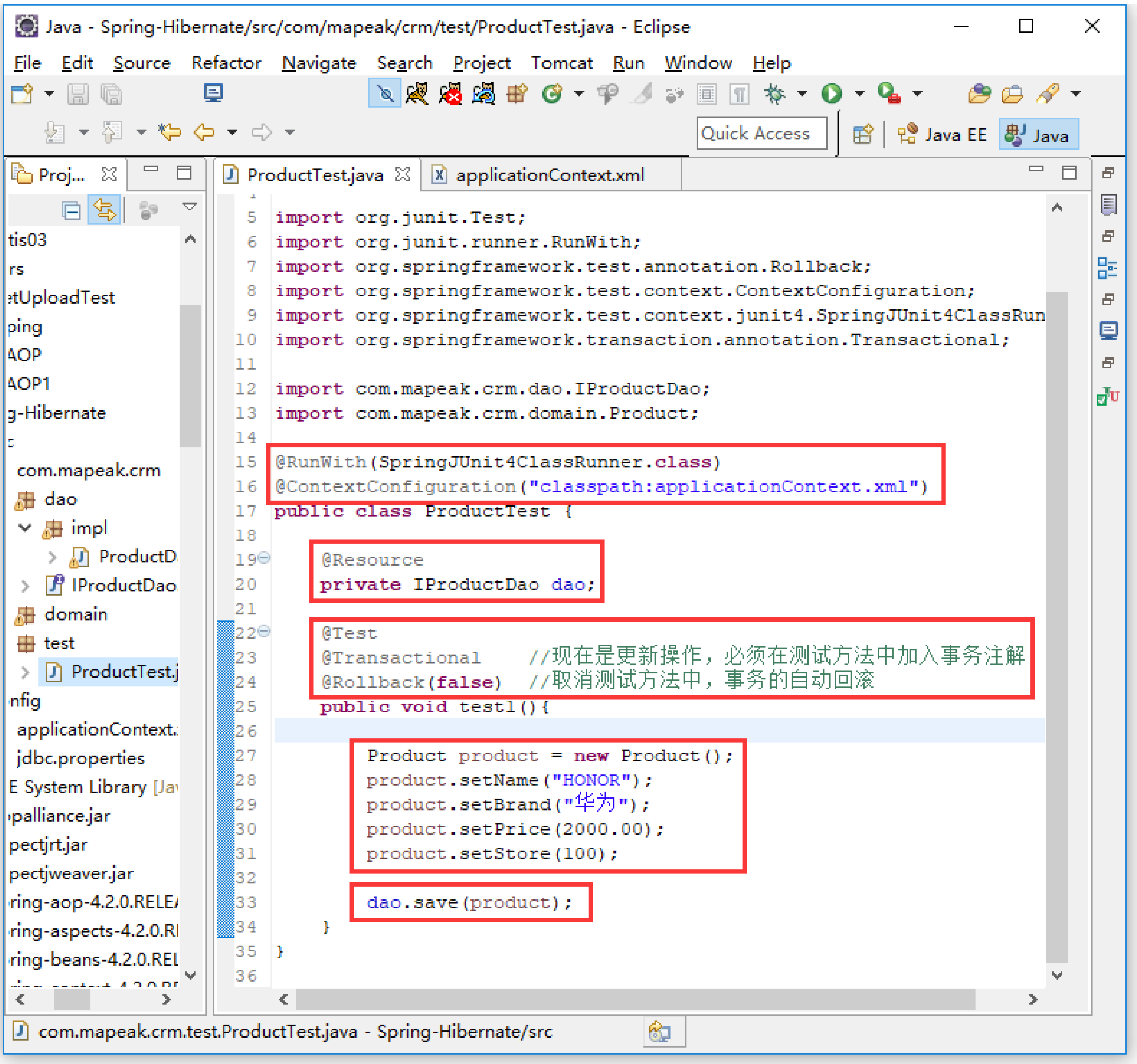


```java
package com.mapeak.crm.dao.impl;

import java.io.Serializable;

@Repository
public class ProductDaoImpl extends HibernateDaoSupport
    implements IProductDao, Serializable {

    //注入 SessionFactory 为了获取 HibernateTemplate
    @Resource(name="sessionFactory")
    public void setMySessionFactory(SessionFactory factory){
        super.setSessionFactory(factory);
    }

    @Override
    public void save(Product product) {
        this.getHibernateTemplate().save(product);
    }

    @Override
    public List<Product> findAll() {
        return this.getHibernateTemplate().loadAll(Product.class);
    }

}
```

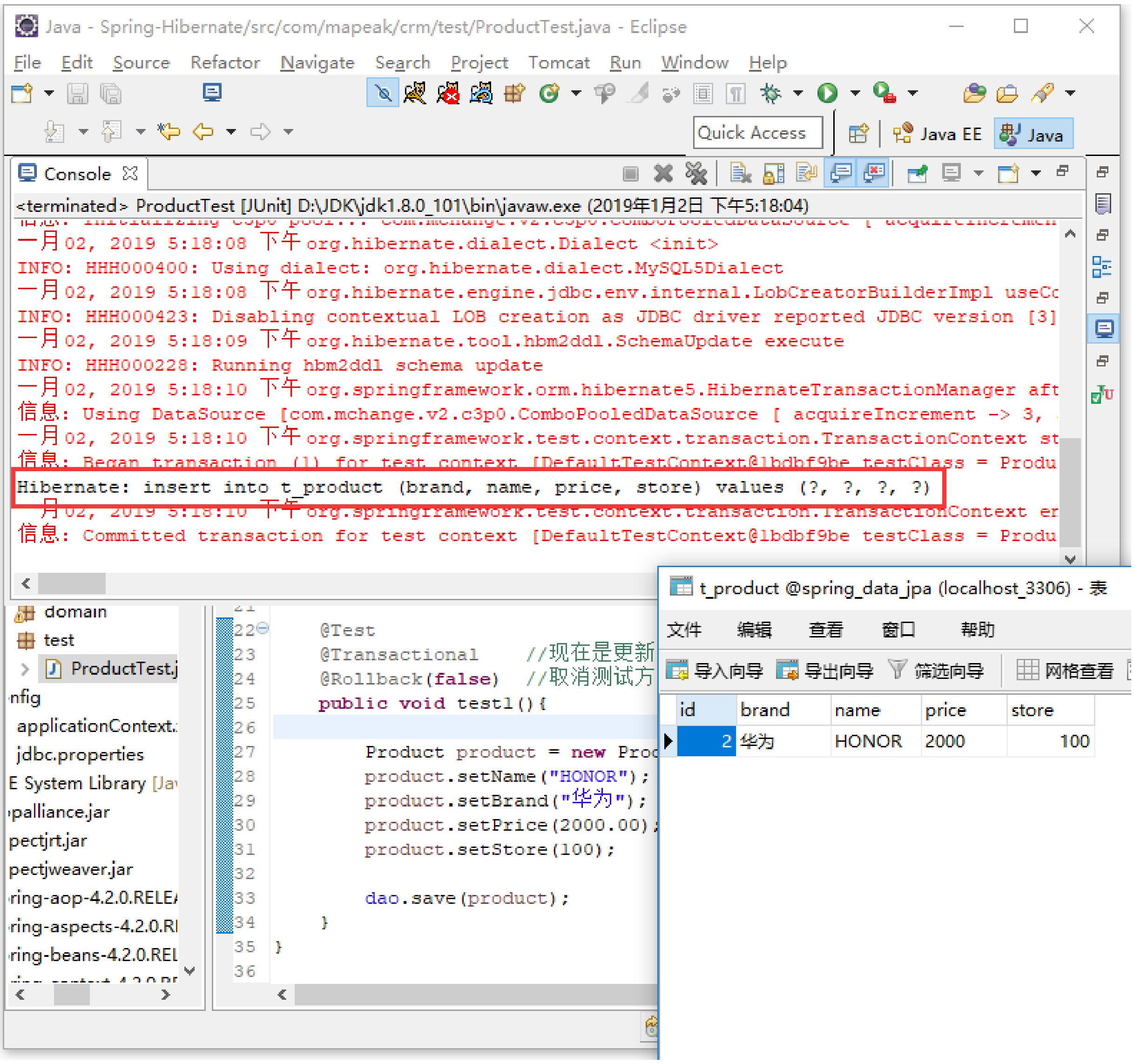
测试：

@Transactional：增删改操作，必须在测试方法中加入事务注解

@Rollback(false)：取消测试方法中，事务的自动回滚



```java
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.annotation.Rollback;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRun
import org.springframework.transaction.annotation.Transactional;

import com.mapeak.crm.dao.IProductDao;
import com.mapeak.crm.domain.Product;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class ProductTest {

    @Resource
    private IProductDao dao;

    @Test
    @Transactional    //现在是更新操作，必须在测试方法中加入事务注解
    @Rollback(false)  //取消测试方法中，事务的自动回滚
    public void test1(){

        Product product = new Product();
        product.setName("HONOR");
        product.setBrand("华为");
        product.setPrice(2000.00);
        product.setStore(100);

        dao.save(product);
    }
}
```



```
<terminated> ProductTest [JUnit] D:\JDK\jdk1.8.0_101\bin\javaw.exe (2019年1月2日 下午5:18:04)
信息: Initializing c3p0 pool... com.mchange.v2.c3p0.ComboPooledDataSource [ acquireIncremen
一月 02, 2019 5:18:08 下午 org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
一月 02, 2019 5:18:08 下午 org.hibernate.engine.jdbc.env.internal.LobCreatorBuilderImpl useCo
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3]
一月 02, 2019 5:18:09 下午 org.hibernate.tool.hbm2ddl.SchemaUpdate execute
INFO: HHH000228: Running hbm2ddl schema update
一月 02, 2019 5:18:10 下午 org.springframework.orm.hibernate5.HibernateTransactionManager aft
信息: Using DataSource [com.mchange.v2.c3p0.ComboPooledDataSource [ acquireIncrement -> 3,
一月 02, 2019 5:18:10 下午 org.springframework.test.context.transaction.TransactionContext st
信息: Began transaction (1) for test context [DefaultTestContext@1bdbf9be testClass = Produ
Hibernate: insert into t_product (brand, name, price, store) values (?, ?, ?, ?)
一月 02, 2019 5:18:10 下午 org.springframework.test.context.transaction.TransactionContext en
信息: Committed transaction for test context [DefaultTestContext@1bdbf9be testClass = Produ
```

```
    @Test
    @Transactional    //现在是更新
    @Rollback(false)  //取消测试方
    public void test1(){

        Product product = new Proc
        product.setName("HONOR");
        product.setBrand("华为");
        product.setPrice(2000.00);
        product.setStore(100);

        dao.save(product);
    }
}
```

id	brand	name	price	store
2	华为	HONOR	2000	100

Spring 整合 JPA：

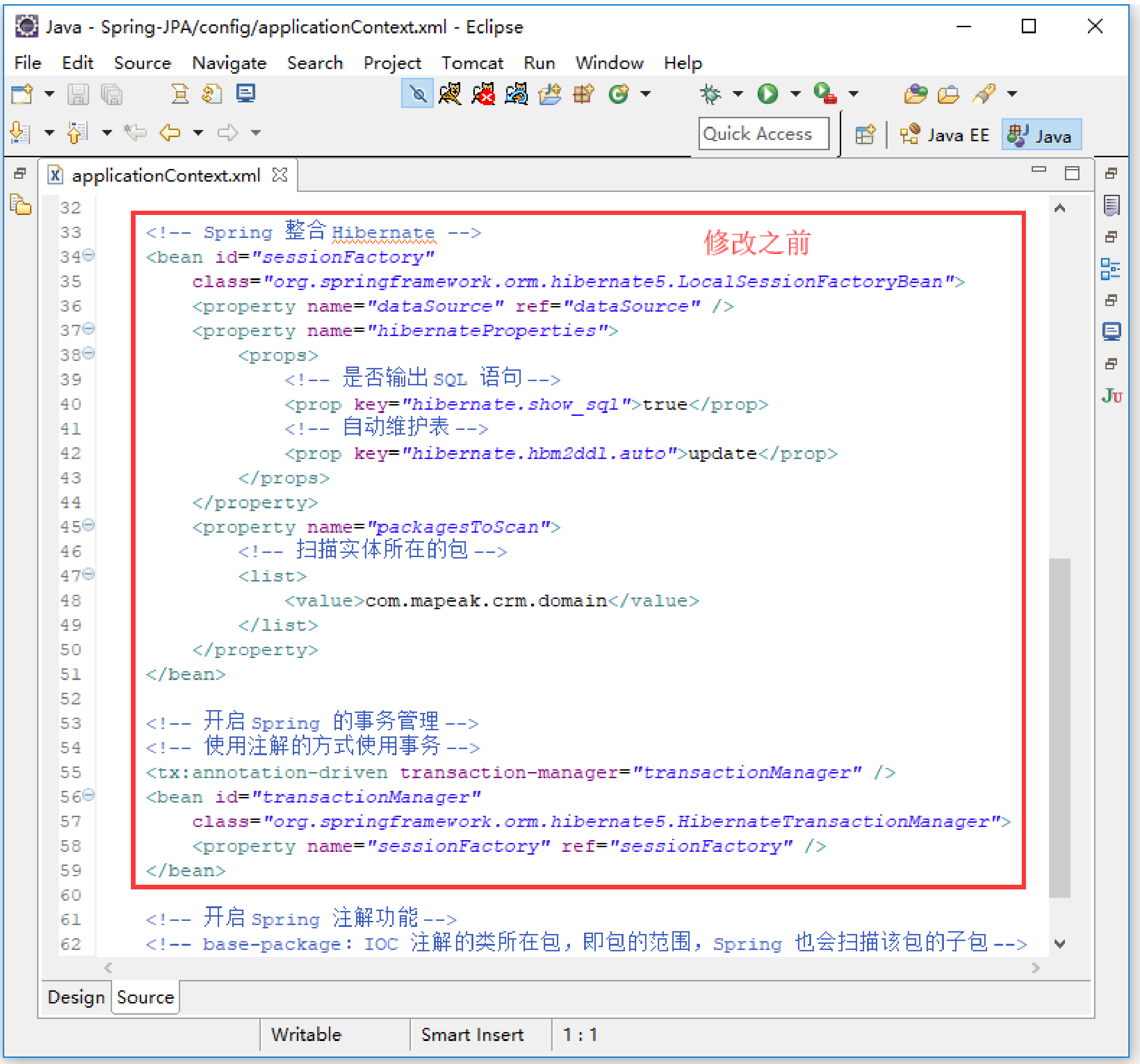
前提：在 Spring 框架已经整合了 Hibernate 框架的基础之上

导入 jar 包：

hibernate-entitymanager-5.0.7.Final.jar   2017/6/29 11:18   Executable Jar File   585 KB

修改 applicationContext.xml 配置：

1、Spring 整合 Hibernate 的配置：

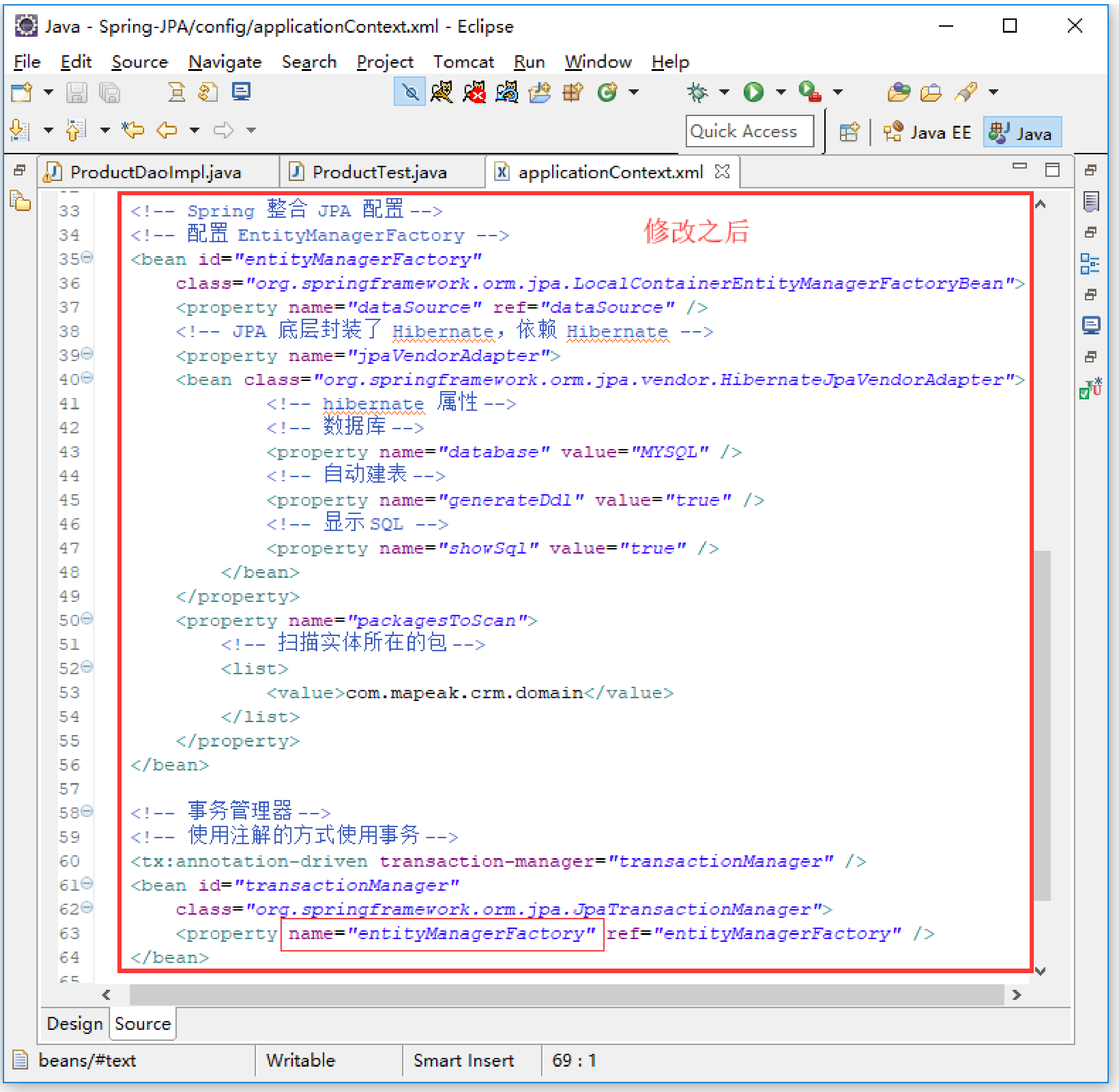


Java - Spring-JPA/config/applicationContext.xml - Eclipse
File Edit Source Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
applicationContext.xml
Design
Source
Writable
Smart Insert
1 : 1

修改之前

```xml

<!-- Spring 整合 Hibernate -->
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="hibernateProperties">
        <props>
            <!-- 是否输出 SQL 语句 -->
            <prop key="hibernate.show_sql">true</prop>
            <!-- 自动维护表 -->
            <prop key="hibernate.hbm2ddl.auto">update</prop>
        </props>
    </property>
    <property name="packagesToScan">
        <!-- 扫描实体所在的包 -->
        <list>
            <value>com.mapeak.crm.domain</value>
        </list>
    </property>
</bean>

<!-- 开启 Spring 的事务管理 -->
<!-- 使用注解的方式使用事务 -->
<tx:annotation-driven transaction-manager="transactionManager" />
<bean id="transactionManager"
    class="org.springframework.orm.hibernate5.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>

<!-- 开启 Spring 注解功能 -->
<!-- base-package: IOC 注解的类所在包，即包的范围，Spring 也会扫描该包的子包 -->
```

2、Spring 整合 JPA 的配置:



修改之后

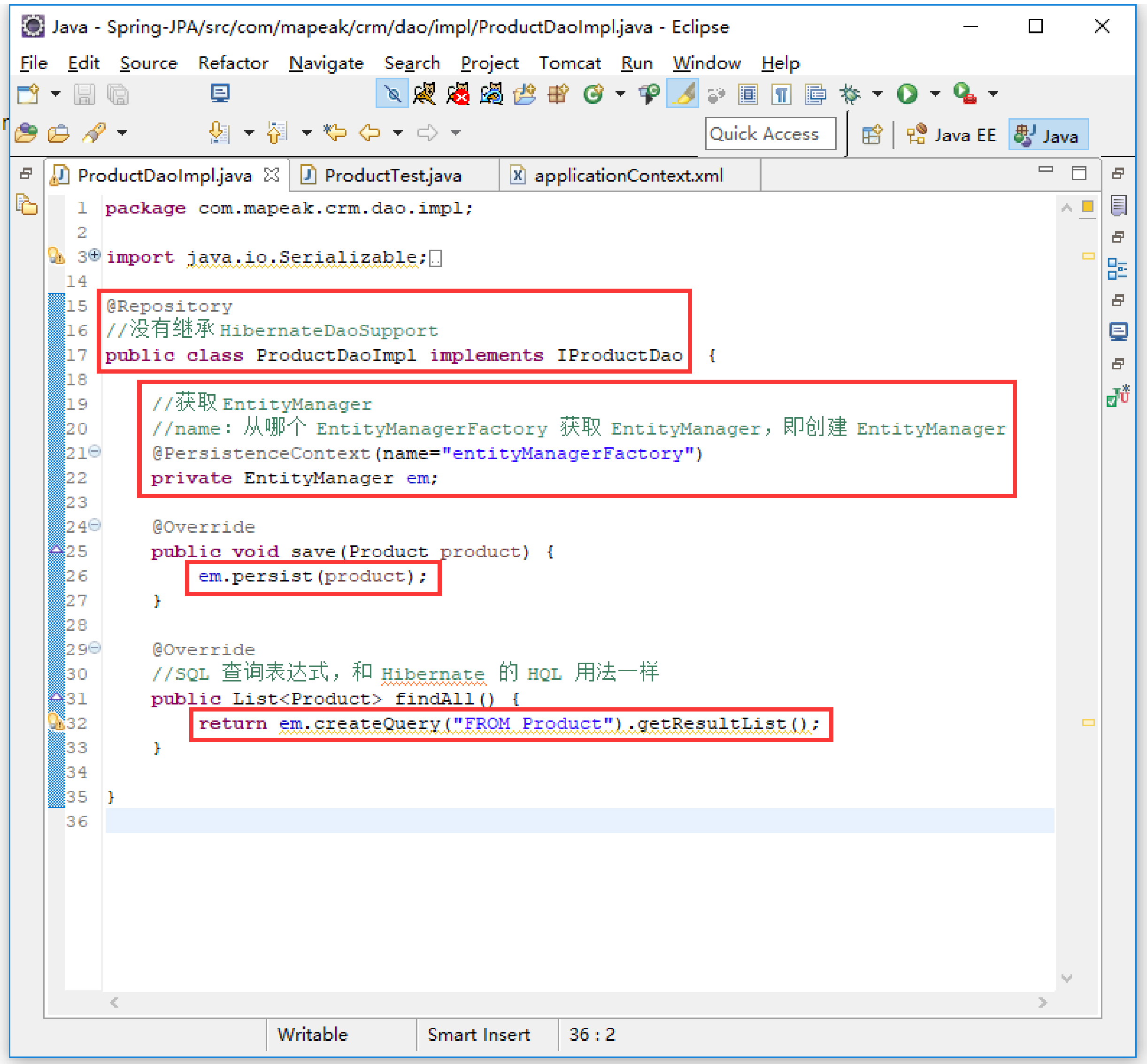
```xml
<!-- Spring 整合 JPA 配置 -->
<!-- 配置 EntityManagerFactory -->
<bean id="entityManagerFactory"
    class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <!-- JPA 底层封装了 Hibernate，依赖 Hibernate -->
    <property name="jpaVendorAdapter">
    <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
            <!-- hibernate 属性 -->
            <!-- 数据库 -->
            <property name="database" value="MYSQL" />
            <!-- 自动建表 -->
            <property name="generateDdl" value="true" />
            <!-- 显示 SQL -->
            <property name="showSql" value="true" />
        </bean>
    </property>
    <property name="packagesToScan">
        <!-- 扫描实体所在的包 -->
        <list>
            <value>com.mapeak.crm.domain</value>
        </list>
    </property>
</bean>

<!-- 事务管理器 -->
<!-- 使用注解的方式使用事务 -->
<tx:annotation-driven transaction-manager="transactionManager" />
<bean id="transactionManager"
    class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
```

修改 DAO 实现：

@PersistenceContext(name="entityManagerFactory")：从哪个 EntityManagerFactory 获取 EntityManager，即创建 EntityManager

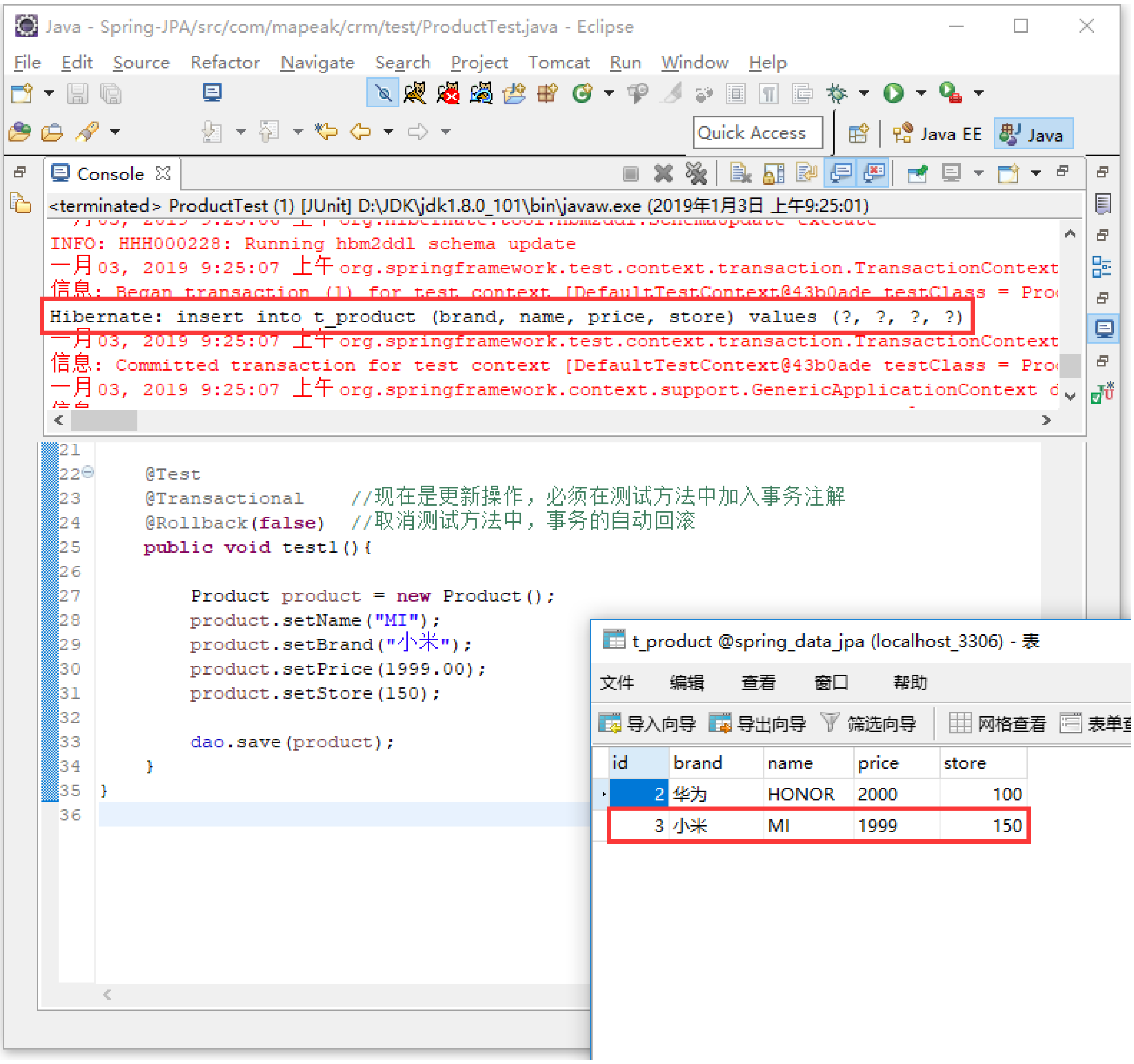
SQL 查询表达式：和 Hibernate 的 HQL 用法一样

PS：没有继承 HibernateDaoSupport


Java - Spring-JPA/src/com/mapeak/crm/dao/impl/ProductDaoImpl.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
ProductDaoImpl.java
ProductTest.java
applicationContext.xml
Writable
Smart Insert
36 : 2

```java
package com.mapeak.crm.dao.impl;

import java.io.Serializable;

@Repository
//没有继承 HibernateDaoSupport
public class ProductDaoImpl implements IProductDao  {

    //获取 EntityManager
    //name: 从哪个 EntityManagerFactory 获取 EntityManager，即创建 EntityManager
    @PersistenceContext(name="entityManagerFactory")
    private EntityManager em;

    @Override
    public void save(Product product) {
        em.persist(product);
    }

    @Override
    //SQL 查询表达式，和 Hibernate 的 HQL 用法一样
    public List<Product> findAll() {
        return em.createQuery("FROM Product").getResultList();
    }

}
```

测试:



Java - Spring-JPA/src/com/mapeak/crm/test/ProductTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Console

```
<terminated> ProductTest (1) [JUnit] D:\JDK\jdk1.8.0_101\bin\javaw.exe (2019年1月3日 上午9:25:01)
INFO: HHH000228: Running hbm2ddl schema update
一月 03, 2019 9:25:07 上午 org.springframework.test.context.transaction.TransactionContext
信息: Began transaction (1) for test context [DefaultTestContext@43b0ade testClass = Pro
Hibernate: insert into t_product (brand, name, price, store) values (?, ?, ?, ?)
一月 03, 2019 9:25:07 上午 org.springframework.test.context.transaction.TransactionContext
信息: Committed transaction for test context [DefaultTestContext@43b0ade testClass = Pro
一月 03, 2019 9:25:07 上午 org.springframework.context.support.GenericApplicationContext c
```

```java

    @Test
    @Transactional    //现在是更新操作，必须在测试方法中加入事务注解
    @Rollback(false)  //取消测试方法中，事务的自动回滚
    public void test1(){

        Product product = new Product();
        product.setName("MI");
        product.setBrand("小米");
        product.setPrice(1999.00);
        product.setStore(150);

        dao.save(product);
    }
}
```

t_product @spring_data_jpa (localhost_3306) - 表
文件 编辑 查看 窗口 帮助
导入向导 导出向导 筛选向导 网格查看 表单查

id	brand	name	price	store
2	华为	HONOR	2000	100
3	小米	MI	1999	150

Spring Data 整合 JPA：

前提：在 Spring 框架已经整合了 JPA 框架的基础之上

导入 spring-Data-jpa jar 包：

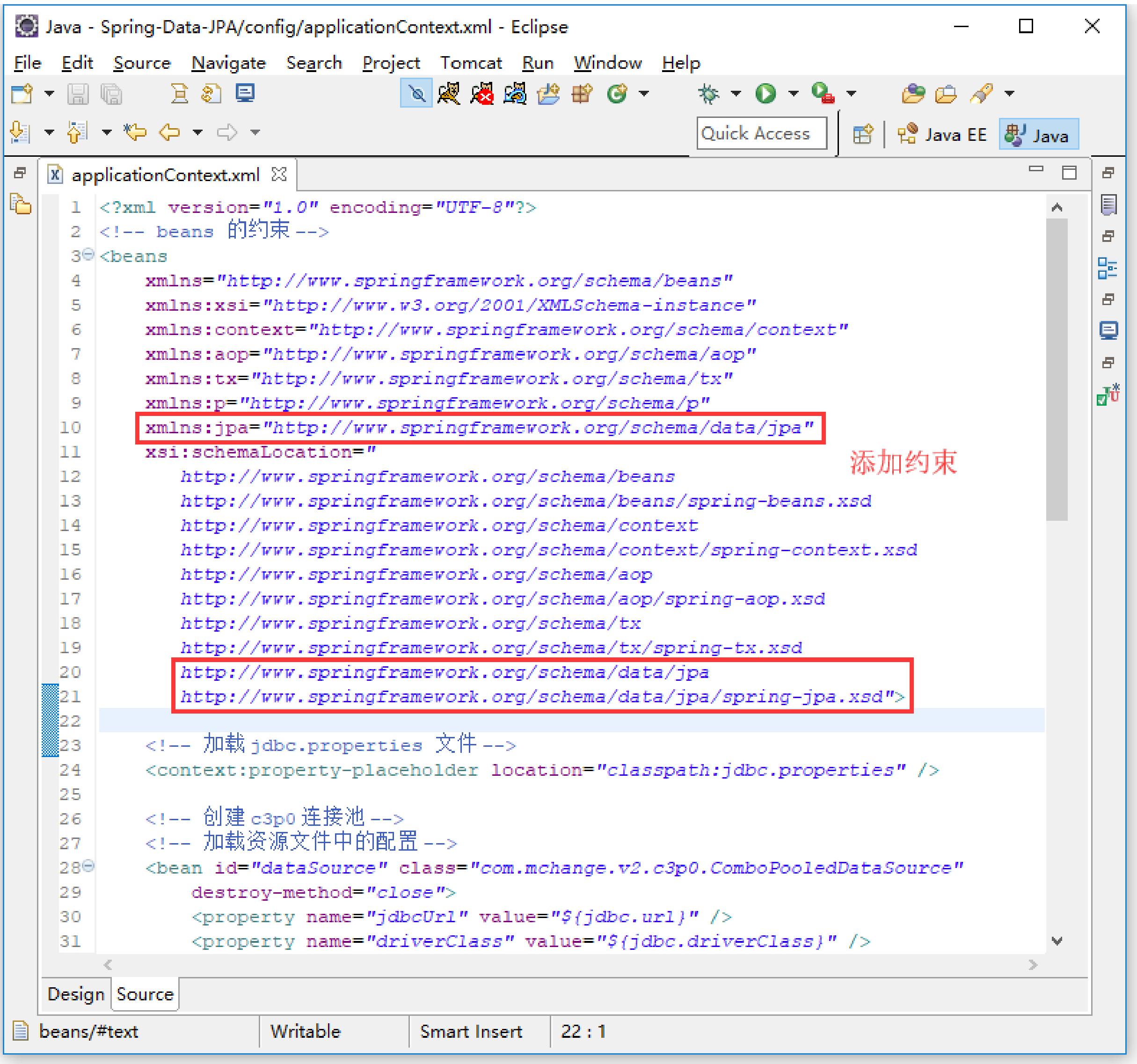
spring-data-commons-1.11.0.RELEAS...	2017/6/29 11:36	Executable Jar File	621 KB
spring-data-jpa-1.9.0.RELEASE.jar	2017/6/29 11:36	Executable Jar File	249 KB

导入日志 jar 包：

log4j-1.2.16.jar	2017/6/29 11:46	Executable Jar File	471 KB
slf4j-api-1.6.1.jar	2017/6/29 11:46	Executable Jar File	25 KB
slf4j-log4j12-1.7.2.jar	2017/6/29 11:46	Executable Jar File	9 KB

修改 applicationContext.xml 配置：

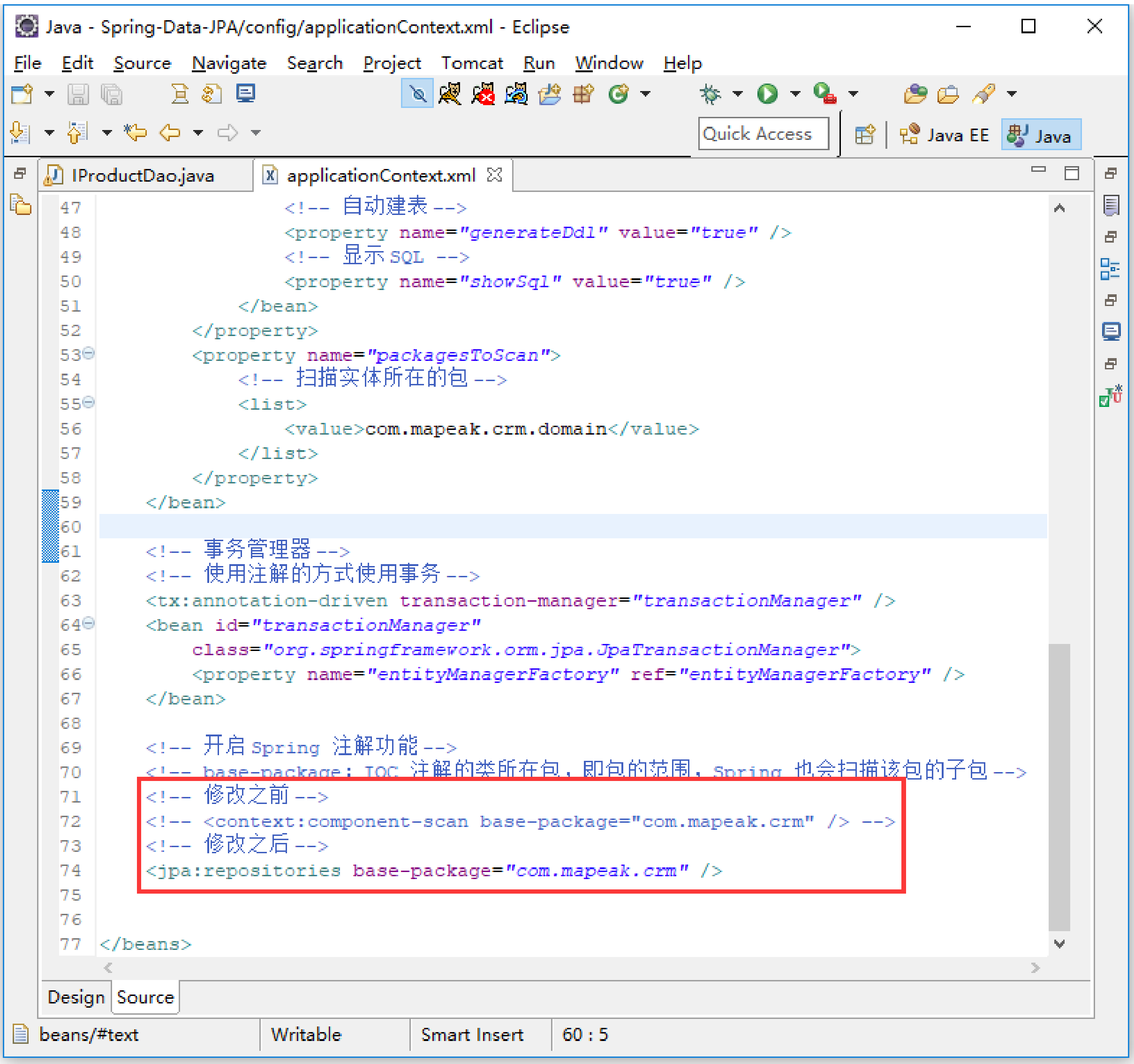
1、添加约束


Java - Spring-Data-JPA/config/applicationContext.xml - Eclipse
File Edit Source Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
applicationContext.xml
Design
Source
beans/#text
Writable
Smart Insert
22 : 1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- beans 的约束 -->
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd
       http://www.springframework.org/schema/data/jpa
       http://www.springframework.org/schema/data/jpa/spring-jpa.xsd">

    <!-- 加载 jdbc.properties 文件 -->
    <context:property-placeholder location="classpath:jdbc.properties" />

    <!-- 创建 c3p0 连接池 -->
    <!-- 加载资源文件中的配置 -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
        destroy-method="close">
        <property name="jdbcUrl" value="${jdbc.url}" />
        <property name="driverClass" value="${jdbc.driverClass}" />
```

添加约束

2、包扫描修改


Java - Spring-Data-JPA/config/applicationContext.xml - Eclipse
File Edit Source Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
IProductDao.java
applicationContext.xml
Design
Source
beans/#text
Writable
Smart Insert
60 : 5

```xml
                <!-- 自动建表 -->
                <property name="generateDdl" value="true" />
                <!-- 显示 SQL -->
                <property name="showSql" value="true" />
            </bean>
        </property>
        <property name="packagesToScan">
            <!-- 扫描实体所在的包 -->
            <list>
                <value>com.mapeak.crm.domain</value>
            </list>
        </property>
    </bean>

    <!-- 事务管理器 -->
    <!-- 使用注解的方式使用事务 -->
    <tx:annotation-driven transaction-manager="transactionManager" />
    <bean id="transactionManager"
        class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
    </bean>

    <!-- 开启 Spring 注解功能 -->
    <!-- base-package: IOC 注解的类所在包，即包的范围，Spring 也会扫描该包的子包 -->
    <!-- 修改之前 -->
    <!-- <context:component-scan base-package="com.mapeak.crm" /> -->
    <!-- 修改之后 -->
    <jpa:repositories base-package="com.mapeak.crm" />


</beans>
```

修改 DAO:

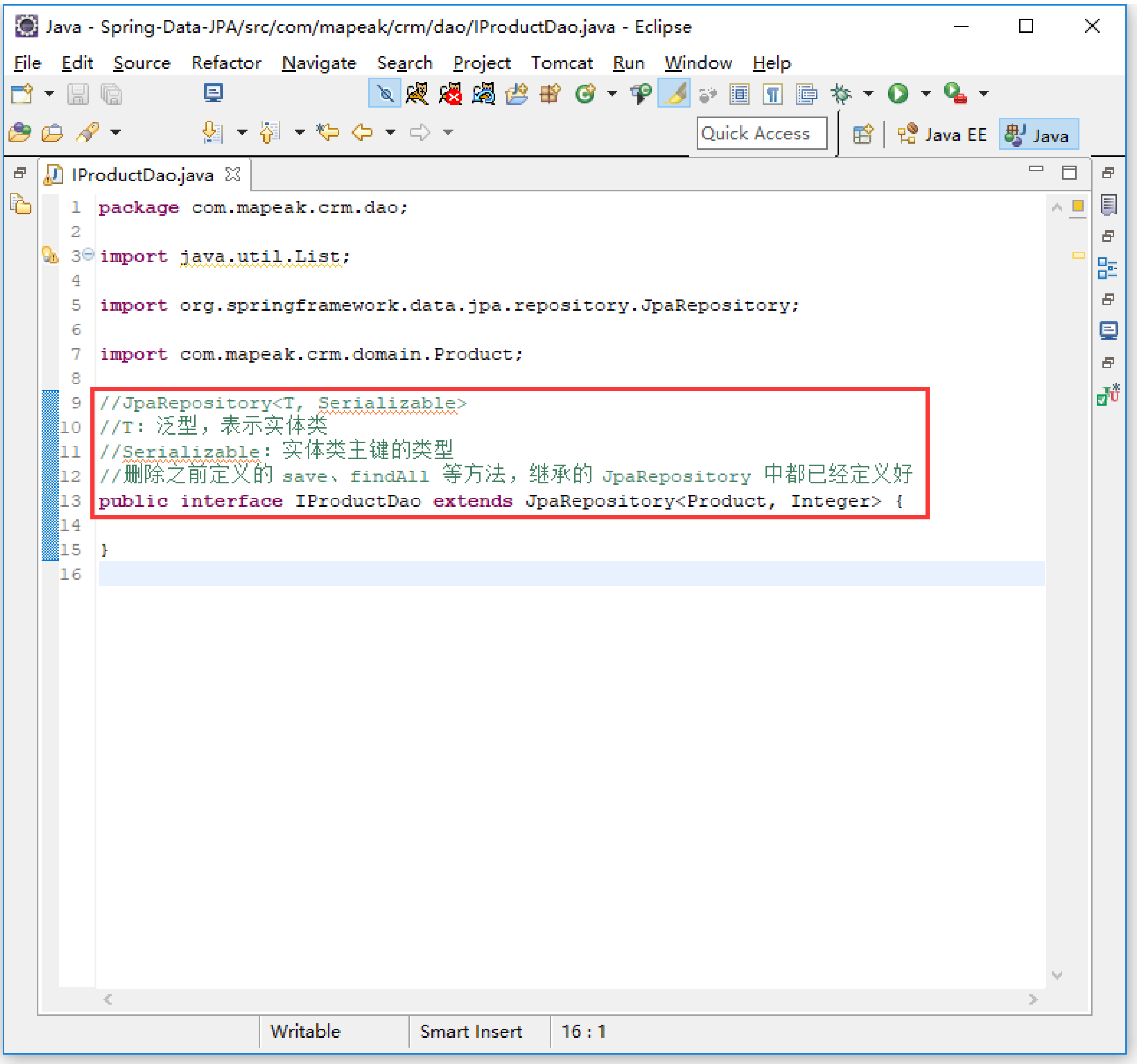
PS: 删除 DAO 实现类

JpaRepository<T, Serializable>

T：泛型，表示实体类

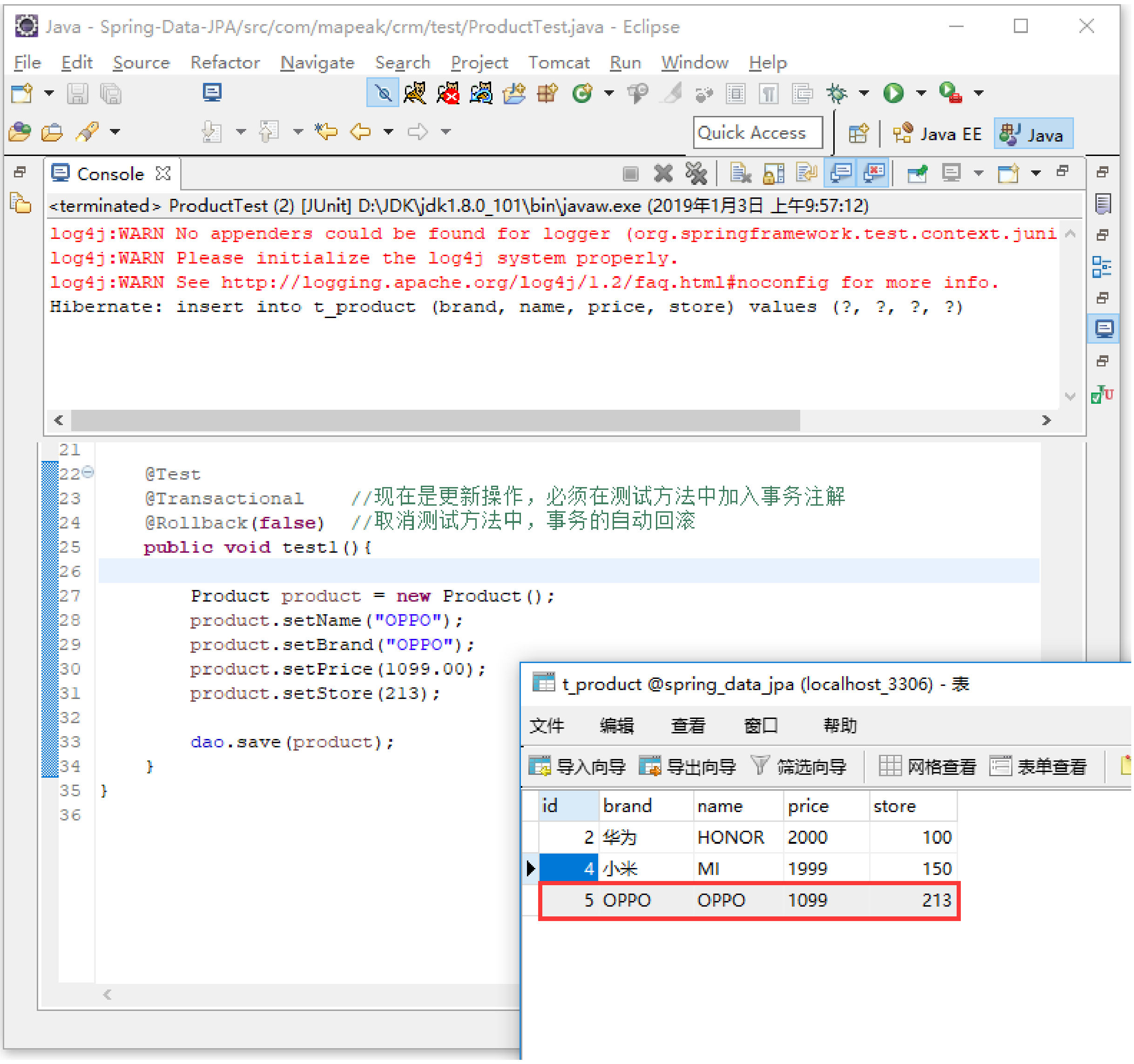
Serializable：实体类主键的类型

删除之前定义的 save、findAll 等方法，继承的 JpaRepository 中都已经定义



```java
package com.mapeak.crm.dao;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.mapeak.crm.domain.Product;

//JpaRepository<T, Serializable>
//T：泛型，表示实体类
//Serializable：实体类主键的类型
//删除之前定义的 save、findAll 等方法，继承的 JpaRepository 中都已经定义好
public interface IProductDao extends JpaRepository<Product, Integer> {

}
```

测试：


Java - Spring-Data-JPA/src/com/mapeak/crm/test/ProductTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access
Java EE
Java
Console

```
<terminated> ProductTest (2) [JUnit] D:\JDK\jdk1.8.0_101\bin\javaw.exe (2019年1月3日 上午9:57:12)
log4j:WARN No appenders could be found for logger (org.springframework.test.context.juni
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Hibernate: insert into t_product (brand, name, price, store) values (?, ?, ?, ?)
```

```java
    @Test
    @Transactional    //现在是更新操作，必须在测试方法中加入事务注解
    @Rollback(false)  //取消测试方法中，事务的自动回滚
    public void test1(){

        Product product = new Product();
        product.setName("OPPO");
        product.setBrand("OPPO");
        product.setPrice(1099.00);
        product.setStore(213);

        dao.save(product);
    }
}
```

t_product @spring_data_jpa (localhost_3306) - 表
文件 编辑 查看 窗口 帮助
导入向导 导出向导 筛选向导 网格查看 表单查看

id	brand	name	price	store
2	华为	HONOR	2000	100
4	小米	MI	1999	150
5	OPPO	OPPO	1099	213