链滴

springDataJpa 学习笔记

作者：tianyunperfect
原文链接：https://ld246.com/article/1545811569102
来源网站：链滴

前言

ORM框架，会大大减少重复性代码，开发速度快；

底层使用了hibernate；

springData

准备

引用

```
<!--spring相关的依赖-->
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>5.0.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-tx</artifactId>
   <version>5.0.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.aspectj</groupId>
   <artifactId>aspectjweaver</artifactId>
   <version>1.8.7</version>
</dependency>
<dependency>
```

```
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.2.RELEASE</version>
</dependency>

<!--原生Jpa所需依赖-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.6</version>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.12</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-c3p0</artifactId>
    <version>5.0.7.Final</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>5.0.7.Final</version>
</dependency>
<!--spring data jpa 所需依赖-->
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-jpa</artifactId>
    <version>2.0.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>5.0.2.RELEASE</version>
</dependency>
```

xml配置初始化JPA

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

```
amework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema
aop/spring-aop.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/sch
ma/context/spring-context.xsd
        http://www.springframework.org/schema/jdbc http://www.springframework.org/schema
jdbc/spring-jdbc.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/sch
ma/data/jpa/spring-jpa.xsd">

    <!-- 配置要扫描的包 -->
    <context:component-scan base-package="com.alvin"></context:component-scan>

    <!-- 1.dataSource -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="com.mysql.jdbc.Driver" />
        <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/test" />
        <property name="user" value="root" />
        <property name="password" value="admin" />
    </bean>

    <!-- 2.EntityManagerFactory -->
    <bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntit
ManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan" value="com.alvin.domain" />
        <property name="persistenceProvider">
            <bean class="org.hibernate.jpa.HibernatePersistenceProvider" />
        </property><!--必须写-->
        <!--JPA供应商适配器-->
        <property name="jpaVendorAdapter">
            <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
                <!--是否自动生成DDL语句,如果没有表，value=false会报错-->
                <property name="generateDdl" value="false" />
                <property name="database" value="MYSQL" />
                <property name="databasePlatform" value="org.hibernate.dialect.MySQLDialect" /

                <property name="showSql" value="true" />
            </bean>
        </property>
        <!--JPA方言-->
        <property name="jpaDialect">
            <bean class="org.springframework.orm.jpa.vendor.HibernateJpaDialect" />
        </property>
    </bean>


    <!-- 3.事务管理器-->
    <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionMana
er">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
    </bean>
```

```
    <!--整合jpa-->
    <jpa:repositories base-package="com.alvin.dao "
                      transaction-manager-ref="transactionManager"
                      entity-manager-factory-ref="entityManagerFactory">

    </jpa:repositories>

    <!-- 4.txAdvice-->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="get*" read-only="true" propagation="SUPPORTS"/>
            <tx:method name="find*" read-only="true" propagation="SUPPORTS"/>
            <tx:method name="*" propagation="REQUIRED" read-only="false"/>
        </tx:attributes>
    </tx:advice>

    <!-- 5.aop-->
    <aop:config>
        <aop:pointcut id="pointcut" expression="execution(* com.alvin.service.*.*(..))" />
        <aop:advisor advice-ref="txAdvice" pointcut-ref="pointcut" />
    </aop:config>
</beans>
```

pojo

和原生一样

dao层接口

```
public interface ICustomerDao extends JpaRepository<Customer,Integer>,JpaSpecificationExecutor<Customer>{
}
```

使用

新增、修改：save

```
Customer customer = new Customer();
customer.setCustName("hello Spring data JPA");
customerService.save(customer);
```

删除

```
customerService.delete(1);
```

查询所有

```
List<Customer> list = customerService.findAll();
```

根据ID查询findById

```
Customer customer = customerService.findById(1);
System.out.println(customer);
```

命名规则查询（条件查询）

```
/**
 * 根据方法的命名规则查询
 *   拓展：
 *      约定大于配置（很重要的思想）
 * @param custName
 * @param custAddress
 * @return
 */
List<Customer> findByCustNameAndCustAddressLike(StringcustName, String custAddress);
```

自定义JPQL查询（条件查询）

```
/**
 * 查询集合（带条件--JPQL）
 * @param custName
 * @param custAddress
 * @return
 */
@Query("from Customer where custName = ?1 and custAddress like ?2")
List<Customer> findAll(String custName, String custAddress);
```

自定义sql查询（条件查询）

```
/**
 * Sql
 * @param custName
 * @param custAddress
 * @return
 */
@Query(value = "select * from cst_customer where cust_name = ?1 and cust_address like ?2",
ativeQuery=true)
List<Customer> findAllByNative(String custName, StringcustAddress);
```

自定义update、delete

```
/**
 * 自定更新
 * @param custId
 * @param custName
 * @param custAddress
 */
@Query("update Customer set custName = ?2,custAddress = ?3 where custId = ?1")
@Modifying
void update(Integer custId, String custName, StringcustAddress);
```

Specification对象查询和分页

方便拼接查询条件

查询

```
//创建Specification对象 select * from cst_customer where
Specification<Customer> specification = new Specification<Customer>() {
    @Override
    public Predicate toPredicate(Root<Customer> root, CriteriaQuery<?> criteriaQuery, Criteri
Builder criteriaBuilder) {
        // 条件
        Predicate p1 = criteriaBuilder.equal(root.get("custName"), "客户2");
        Predicate p2 = criteriaBuilder.like(root.get("custAddress"), "%北%");
        return criteriaBuilder.and(p1,p2);
    }
};

List<Customer> all = service.findAll(specification);
```

分页

传入参数第几页，多少条！

```
//创建Specification对象 select * from cst_customer
Specification<Customer> specification = new Specification() {
    @Nullable
    @Override
    // 此内部方法只是负责拼接条件及参数
    public Predicate toPredicate(Root root, CriteriaQuery cq, CriteriaBuilder cb) {
        Predicate predicate = cb.and(cb.equal(root.get("custName"),"京西集团"),
                cb.like(root.get("custAddress"),"%山%"));
        return null;
    }
};

Pageable pageable = PageRequest.of(1,2);
Page<Customer> page = customerService.findAllByPage(specification, pageable);
List<Customer> list = page.getContent();
```

一对多和多对多查询

待补充