**1. Что такое транзакция?**

Транза́кция (англ. transaction) — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными.

**2. В чем её отличие JPA от Hibernate?**

Hibernate одна из самых популярных открытых реализаций последней версии спецификации (JPA 2.1). Даже скорее самая популярная, почти стандарт де-факто. То есть JPA только описывает правила и API, а Hibernate реализует эти описания, впрочем у Hibernate (как и у многих других реализаций JPA) есть дополнительные возможности, не описанные в JPA (и не переносимые на другие реализации JPA).

**3. Что такое Entity?**

Сущность (Entity) – это легковесный объект домена с сохраняемостью. Обычно сущность представляет таблицу в реляционной базе данных, и каждый экземпляр сущности соответствует строке в этой таблице. Основным артефактом программирования сущности является класс сущности, хотя сущности могут использовать вспомогательные классы.

Постоянное состояние сущности представлено либо постоянными полями, либо постоянными свойствами. Эти поля или свойства используют аннотации объектного / реляционного сопоставления для сопоставления сущностей и связей сущностей с реляционными данными в базовом хранилище данных.

**4. Конфигурационный файл Hibernate?**

Hibernate требует заранее знать – где найти информацию о сопоставлении, которая определяет, как ваши классы Java связаны с таблицами базы данных. Hibernate также требует набора параметров конфигурации, связанных с базой данных и других связанных параметров. Вся такая информация обычно предоставляется в виде стандартного файла свойств Java с именем hibernate.properties или в виде файла XML с именем hibernate.cfg.xml/

**5. Аннотация @Autowired?**

Аннотация @Autowired отмечает конструктор, поле или метод как требующий автозаполнения инъекцией зависимости Spring. Используя эту аннотацию, не нужно заботиться о том, как лучше всего передать классу или bean'у экземпляр другого bean'a. Фреймворк Spring сам найдет нужный bean и подставит его значение в свойство, которое отмечено аннотацией @Autowired.

Аннотация @Autowire может быть использована в методе с любым именем и с любым количеством принимаемых параметров. В этом случае Spring попытается присвоить каждому аргументу значение соответствующих bean'а. Метод не обязан быть public.

**6. Что такое JPQL (Java Persistence query language)?**

JPQL (Java Persistence query language) – это язык запросов, практически такой же как SQL, однако вместо имен и колонок таблиц базы данных, он использует имена классов Entity и их атрибуты. В качестве параметров запросов так же используются типы данных атрибутов Entity, а не полей баз данных. В отличии от SQL в JPQL есть автоматический полиморфизм. Также в JPQL используется функции которых нет в SQL: такие как KEY (ключ Map'ы), VALUE (значение Map'ы), TREAT (для приведение суперкласса к его объекту-наследнику), ENTRY и т.п.

**7. Интерфейс JpaRepository?**

**JpaRepository –** это интерфейс фреймворка Spring Data предоставляющий набор стандартных методов JPA для работы с БД. На основе этого интерфейса Spring Data предоставит реализации с методами, которые моггут быть использованы.

**8. Аннотация @Id, @GeneratedValue?**

Аннотации @Id и @GeneratedValue предоставляют возожность не выставлять никакие параметры генерации и значения по умолчанию отработают нормально и в поле будет присваиваться уникальное значение каждый раз при сохранении.

Аннотация @Id наследуется от javax.persistence.Id, указывая, что поле члена ниже является первичным ключом текущего объекта. Следовательно, ваш Hibernate и spring рамки, а также вы можете сделать некоторые reflect работы, основанные на этой аннотации.

Аннотация @GeneratedValue заключается в настройке способа увеличения указанного столбца (поля). Например, при использовании Mysql вы можете указать auto\_increment в определении таблицы, чтобы сделать его самоинкрементным, а затем использовать в коде Java, чтобы обозначить, что вы также признали, что используете эту стратегию на стороне сервера базы данных. Кроме того, вы можете изменить значение в этой аннотации, чтобы соответствовать различным требованиям.

**9. Как проекте Maven подключить JPA?**

Cоздаем пустой проект maven и настраиваем pom.xml:

*<project xmlns="http://maven.apache.org/POM/4.0.0"*

*xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">*

*<modelVersion>4.0.0</modelVersion>*

*<groupId>ru.javastudy</groupId>*

*<artifactId>jpa\_quickStart</artifactId>*

*<version>1.0-SNAPSHOT</version>*

*<properties>*

*<hibernate-version>5.0.1.Final</hibernate-version>*

*<spring-framework-version>4.1.5.RELEASE</spring-framework-version>*

*</properties>*

*<dependencyManagement>*

*<dependencies>*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-framework-bom</artifactId>*

*<version>${spring-framework-version}</version>*

*<scope>import</scope>*

*<type>pom</type>*

*</dependency>*

*</dependencies>*

*</dependencyManagement>*

*<dependencies>*

*<!--driver for connection to MYSql database -->*

*<dependency>*

*<groupId>mysql</groupId>*

*<artifactId>mysql-connector-java</artifactId>*

*<version>5.1.34</version>*

*</dependency>*

*<!-- Hibernate -->*

*<!-- for JPA, use hibernate-entitymanager instead of hibernate-core. Support @PersistenceContext -->*

*<dependency>*

*<groupId>org.hibernate</groupId>*

*<artifactId>hibernate-entitymanager</artifactId>*

*<version>${hibernate-version}</version>*

*</dependency>*

*<!-- Spring core and orm -->*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-core</artifactId>*

*</dependency>*

*<!--support orm classes-->*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-orm</artifactId>*

*</dependency>*

*<!--support @Service, @Repository -->*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-context</artifactId>*

*</dependency>*

*</dependencies>*

*</project>*

Сначала идет общая ссылка на весь проект spring, а далее отдельно зависимости для драйвера подключения к БД MySQL, зависимость Hibernate, и три минимально необходимых библиотеки Spring.

Далее нажимаем на название проекта правой кнопкой и выбираем Add Framework support. Отмечаем галочкой Spring, JavaEE Persistence и Hibernate. Если зависимости подгрузились, то будет предложено выбрать библиотеку maven. После этого в проект добавятся необходимые xml – META-INF\persistence.xml и sping-config.xml.

Далее настраивем spring-config.xml

<?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:tx="http://www.springframework.org/schema/tx"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd" >

<!--@Transaction annotation support -->

<tx:annotation-driven transaction-manager="transactionManager"/>

<!--Scanning components in base-package (look for annotations) -->

<context:component-scan base-package="ru.javastudy"/>

<!--Activates various annotations to be detected in bean classes: Spring's @Required and @Autowired and so on-->

<context:annotation-config/>

<!--Обеспечивает работу с транзакциями в Spring -->

<bean id="transactionManager"

class="org.springframework.orm.jpa.JpaTransactionManager">

<property name="entityManagerFactory" ref="emf"/>

</bean>

<!-- Datasource. Источник данных - база MySQL -->

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver" />

<property name="url" value="jdbc:mysql://localhost:3306/javastudy" />

<property name="username" value="root" />

<property name="password" value="admin" />

</bean>

<!-- EntityManagerFactory -->

<bean id="emf" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">

<property name="dataSource" ref="dataSource" />

<!--Поставщик данных - hibernate-->

<property name="jpaVendorAdapter">

<bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter" />

</property>

<!--поиск сущностей в этом пакете-->

<property name="packagesToScan"

value="ru.javastudy"/>

<!--детали конфигурации поставщика постоянства (hibernate) -->

<property name="jpaProperties">

<props>

<prop key="hibernate.dialect">

org.hibernate.dialect.H2Dialect

</prop>

<prop key="hibernate.max\_fetch\_depth">3</prop>

<prop key="hibernate.jdbc.fetch\_size">50</prop>

<prop key="hibernate.jdbc.batch\_size">10</prop>

<prop key="hibernate.show\_sql">true</prop>

</props>

</property>

</bean>

</beans>

**10. Методы JpaRepository**

• <S extends T> S save(S entity); – сохраняет переданную сущность.

• Optional<T> findById(ID primaryKey); – возвращает сущность, идентифицированную данным ID.

• Iterable<T> findAll(); – возвращает все сущности.

• long count(); – возвращает количество сущностей.

• void delete(T entity); – удаляет переданную сущность.

• boolean existsById(ID primaryKey); – указывает, существует ли сущность с данным ID.

• flush() – производит запись сохраненных данных непосредственно в БД.