**Контрольные вопросы**

**1. Что такое транзакция?**

***Транзакция*** – это набор действий, выполняемых как единое атомарное действие. Либо все действия выполнены, либо ни одно из них не выполнено. Это необходимо для обеспечения того, чтобы целостность состояния приложения никогда не подвергалась риску.

Кроме того, эти транзакции могут включать один или несколько ресурсов, таких как база данных, очередь сообщений, что приводит к различным способам выполнения действий в рамках транзакции. К ним относятся выполнение локальных транзакций ресурсов с отдельными ресурсами. Кроме того, в глобальной транзакции могут участвовать несколько ресурсов.

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

***JPA*** – это спецификация, описывающая объектно-реляционное отображение простых Java объектов и предоставляющая API для сохранения, получения и управления такими объектами.

То есть, как мы помним, реляционные базы данных (БД) представлены в виде множества связанных между собой таблиц. И JPA – общепринятый стандарт, который описывает, как объекты могут взаимодействовать с реляционными базами данных.

В то же время ***Hibernate*** – это конкретная библиотека, реализующая парадигмы JPA. То, есть с помощью этой библиотеки вы можете работать с реляционной базой данных через объекты, которые представляют данные с БД (Entity).

**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.

Требования к классам сущностей:

1. Класс должен быть снабжен аннотацией с javax.persistence.Entity аннотацией.
2. Класс должен иметь общедоступный или защищенный конструктор без аргументов. Класс может иметь другие конструкторы.
3. Класс не должен быть объявлен окончательным. Никакие методы или постоянные переменные экземпляра не должны быть объявлены окончательными.
4. Если экземпляр сущности передается по значению как отдельный объект, например, через удаленный бизнес-интерфейс сеансового компонента, класс должен реализовать Serializable интерфейс.
5. Сущности могут расширять как классы сущностей, так и классы, не являющиеся сущностями, а классы, не являющиеся сущностями, могут расширять классы сущностей.
6. Постоянные переменные экземпляра должны быть объявлены закрытыми, защищенными или закрытыми для пакетов и могут быть доступны напрямую только с помощью методов класса сущности. Клиенты должны получить доступ к состоянию сущности с помощью методов доступа или бизнес-методов.

**4. Конфигурационный файл Hibernate?**

Конфигурационный XML-файл hibernate.cfg.xml определяет параметры СУБД, учетную запись (login, password) подключения к серверу БД, а также дополнительные параметры, которые используются при работе с сервером.

**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. Простыми словами, это интерфейс, от которого можем наследоваться из коробки, получать базовые вопросы из своей базы данных для таблицы, получать удобный синтаксис.

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

Каждый компонент объекта будет иметь первичный ключ, который вы аннотируете в классе с помощью аннотации @Id. Первичный ключ может представлять собой одно поле или комбинацию нескольких полей в зависимости от структуры таблицы.

По умолчанию аннотация @Id автоматически определяет наиболее подходящую стратегию генерации первичного ключа для использования, но можно переопределить это, применив аннотацию @GeneratedValue , которая принимает два параметра strategy и generator.

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

Вначале создаем пустой проект maven. Затем необходимо настроить 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>*

Подписи к зависимостям говорят сами за себя. Первая зависимость Вначале 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>*

* Дескриптор tx:annotation-driven – предназначен для поддержки объявления требований к установлению границ транзакций с использованием аннотаций.
* context:component-scan – сканирование компонентов в указанном пакете и ниже
* context:annotation-config – включает поддержку таких аннотаций как @Require, @Autowired
* бин transactionManager – диспетчер транзакций для фабрики диспетчера сущностей
* EntityManagerFactory – самый главный бин:) В целом с ним и будем работать для доступа к данным

**10. Методы JpaRepository.**



