**Лабораторная работа №7**

**КОНТРОЛЬНЫЕ ВОПРОСЫ:**

1. *Что такое транзакция?*

Под транзакцией понимается ряд действий (не обязательно в БД), которые воспринимаются системой, как единый пакет, т.е. или все действия проходят успешно, или все откатываются на исходные позиции.

Есть два типа транзакций – локальные и распределенные. Локальная транзакция работает с одним источником, например одна БД; распределенная использует несколько – например jms source и БД.

1. *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).

1. *Что такое Entity?*

Entity – это легковесный хранимый объект бизнес-логики (persistent domain object). Основная программная сущность – это entity класс, который так же может использовать дополнительные классы, который могут использоваться как вспомогательные классы или для сохранения состояния еntity.

1. *Конфигурационный файл Hibernate?*

Для корректной работы, мы должны передать Hibernate подробную информацию, которая связывает наши Java-классы c таблицами в базе данных (далее – БД). Мы, также, должны укзать значения определённых свойств Hibernate. Обычно, вся эта информация помещена в отдельный файл, либо XML-файл – **hibernate.cfg.xml**, либо – **hibernate.properties**.

Ключевые свойства, которые должны быть настроены в типичном приложении:

|  |  |
| --- | --- |
| **hibernate.dialect** | Указывает HIebrnate диалект БД. Hibernate, в своб очередь, генерирует необходимые SQL-запросы (например, org.hibernate.dialect.MySQLDialect, если мы используем MySQL). |
| **hibernate.connection-driver\_class** | Указывает класс JDBC драйвера. |
| **hibernate.connection.url** | Указывает URL (ссылку) необходимой нам БД (например, jdbc:mysql://localhost:3306/database). |
| **hibernate.connection.username** | Указывает имя пользователя БД (например, root). |
| **hibernate.connection.password** | Указывает пароль к БД (например, password). |
| **hibernate.connection.pool\_size** | Ограничивает количество соединений, которые находятся в пуле соединений Hibernate. |
| **hibernate.connection.autocommit** | Указывает режим autocommit для JDBC-соединения. |

1. *Аннотация @Autowired?*

Аннотация @Autowired отмечает конструктор, поле или метод как требующий автозаполнения инъекцией зависимости Spring.

Чтобы аннотация @Autowired присвоила переменной значения соответствующего bean'а, необходимо чтобы этот bean либо был объявлен в xml конфигурации приложения, либо существовал класс с соответствующей иньекцией управления.

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.

1. *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 (см. следующий вопрос). 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, downcasting), ENTRY и т.п.

1. *Интерфейс JpaRepository?*

JpaRepository – это интерфейс фреймворка Spring Data предоставляющий набор стандартных методов JPA для работы с БД, например, обновление постоянных данных и пакетное удаление.

1. *Аннотация @Id, @GeneratedValue?*

С помощью аннотации @Id мы указываем первичный ключ (Primary Key) данного класса. Аннотация @GeneratedValue используется вместе с аннотацией @Id и определяет такие паметры, как strategy и generator.

В JPA значение примечания @GeneratedValue главным образом заключается в создании уникально идентифицированного первичного ключа для объекта (JPA требует, чтобы каждый объект должен иметь один и только один первичный ключ). @GeneratedValue обеспечивает генерацию первичного ключа. стратегия. Аннотация @GeneratedValue имеет два атрибута, а именно, стратегию и генератор, где значением атрибута генератора является строка со значением по умолчанию "", которое объявляет имя генератора первичного ключа (соответствует генераторам первичного ключа с тем же именем @SequenceGenerator и @TableGenerator).

JPA предоставляет разработчикам четыре стратегии генерации первичных ключей, которые определены в классе перечисления GenerationType, включая GenerationType.TABLE, GenerationType.SEQUENCE, GenerationType.IDENTITY и GenerationType.AUTO.

1. *Как в проекте Maven подключить JPA?*

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.

1. *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.