# ЛАБОРАТОРНАЯ РАБОТА №7 Применение технологии ORM на базе Hibernate

## 1. Что такое транзакция?

Транзакция ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) transaction) ‒ группа последовательных операций с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), которая представляет собой логическую единицу работы с данными. Транзакции выполняются по принципу «всё или ничего», т. е.транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще, и тогда она не должна произвести никакого эффекта.

## 2. В чем отличие JPA от Hibernate?

JPA (Java Persistence API) это спецификация Java EE и Java SE, описывающая систему управления сохранением java объектов в таблицы реляционных баз данных в удобном виде.

Hibernate – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения.

Hibernate ‒ одна из самых популярных открытых реализаций последней версии спецификации JPA. 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?

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

Entity (Сущность) ‒ [POJO](https://ru.wikipedia.org/wiki/POJO)-класс, связанный с БД с помощью аннотации (@Entity) или через [XML](https://ru.wikipedia.org/wiki/XML). К такому классу предъявляются следующие требования:

* Должен иметь пустой конструктор (public или protected)
* Не может быть вложенным, интерфейсом или enum
* Не может быть final и не может содержать final-полей/свойств
* Должен содержать хотя бы одно @Id-поле

При этом entity может:

* Содержать непустые конструкторы
* Наследоваться и быть наследованным
* Содержать другие методы и реализовывать интерфейсы

Entities могут быть связаны друг с другом (один-к-одному, один-ко-многим, многие-к-одному и многие-ко-многим).

## 4. Конфигурационный файл Hibernate?

Для корректной работы необходимо передать Hibernate подробную информацию, которая связывает Java-классы c таблицами в БД. Нужно также указать значения определённых свойств Hibernate.

Вся эта информация обычно помещается в отдельный XML-файл – hibernate.cfg.xml (либо же это может быть файл hibernate.properties).

Ключевые свойства, которые должны быть настроены в типичном приложении:

* hibernate.dialect

Указывает Hibernate диалект БД. Hibernate же генерирует необходимые SQL-запросы (например, org.hibernate.dialect.MySQLDialect для MySQL).

* hibernate.connection-driver\_class

Указывает класс JDBC драйвера.

* hibernate.connection.url

Указывает URL-ссылку необходимой нам БД.

* hibernate.connection.username

Указывает имя пользователя БД.

* hibernate.connection.password

Указывает пароль к БД.

* hibernate.connection.pool\_size

Ограничивает количество соединений, которые находятся в пуле соединений Hibernate.

* hibernate.connection.autocommit

Указывает режим autocommit для JDBC-соединения.

## 5. Аннотация @Autowired?

Аннотация @Autowired отмечает конструктор, поле или метод как требующий автозаполнения инъекцией зависимости Spring (впервые в Spring 2.5). (Чтобы аннотация @Autowire присвоила переменной значения соответствующего bean-а, необходимо чтобы этот bean либо был объявлен в xml конфигурации приложения, либо существовал класс с соответствующей инъекцией 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.

Свойства класса с аннотацией @Autowired заполняются соответствующими значениями сразу после создания bean'а и перед тем, как любой из методов класса будет вызван.

«Традиционное» использование аннотации ‒ с сеттером.

Только один конструктор может выполнять эту аннотацию. Этот конструктор может быть любого типа (private, protected), а не только public.

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 также не обязан быть public.

При указании в качестве параметра require=false (@Autowire(require=false)) сообщает фреймворку о том, что наличие соответствующего bean-а не является обязательным при компиляции программы. В случае, если аннотация не была объявлена с required=false и bean отсутствует в приложении, пользователь получит исключение.

С помощью аннотации @Qualifier можно отметить конкретного «кандидата» для автозаполнения, если их несколько (например, есть два bean-а).

## 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 также используются типы данных атрибутов Entity, а не полей баз данных.

В отличии от SQL в JPQL есть автоматический полиморфизм, то есть каждый запрос к Entity возвращает не только объекты этого Entity, но также объекты всех его классов-потомков, независимо от стратегии наследования (чтобы исключить такое поведение, используется функция TYPE в where условии).

Также в JPQL используется функции которых нет в SQL (KEY, VALUE, TREAT, ENTRY и т.п.).

## 7. Интерфейс JpaRepository?

**JpaRepository –** это интерфейс фреймворка Spring Data, предоставляющий набор стандартных методов JPA для работы с БД. Интерфейс **JpaRepository наследует интерфейсы** PagingAndSortingRepository и QueryByExampleExecutor.

## 8. Аннотация @Id, @GeneratedValue?

Каждый Entity bean будет иметь первичный ключ, который аннотируется в классе с помощью аннотации @Id. Первичный ключ может быть одним полем или комбинацией 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?

Для работы с БД только зависимости spring-webmvc недостатно, необходимо также подключить spring-jdbc для работы с базой данных, spring-tx для поддержки транзакций и spring-orm для работы с Hibernate. Т. е. в pom.xml нужно добавить:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>5.1.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.3.7.Final</version>

</dependency>

Этих двух зависимостей достаточно: javax.persistence-api подключится вместе с hibernate-core, а spring-jdbc и spring-tx ‒ вместе со spring-orm.

## 10. Методы JpaRepository?

void flush()

- Сбрасывает все ожидающие изменения в базе данных.

<S extends [T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> S saveAndFlush(S entity)

- Сохраняет объект и мгновенно удаляет изменения.

<S extends [T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<S> saveAllAndFlush([Iterable](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Iterable.html" \o "class or interface in java.lang)<S> entities)

- Сохраняет все объекты и мгновенно сбрасывает изменения.

default void deleteInBatch([Iterable](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Iterable.html" \o "class or interface in java.lang)<[T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> entities)

- Удаляет указанные объекты в пакете, что означает, что будет создан один запрос. Этот тип операции оставляет кеш первого уровня JPA и базу данных несинхронизированными. (Предлагается сбросить EntityManager перед вызовом этого метода.) Считается устаревшим, предлагается использовать [deleteAllInBatch(Iterable)](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html#deleteAllInBatch(java.lang.Iterable)).

void deleteAllInBatch([Iterable](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Iterable.html" \o "class or interface in java.lang)<[T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> entities)

- Удаляет указанные объекты в пакете, что означает, что будет создан один запрос. Этот тип операции оставляет кеш первого уровня JPA и базу данных несинхронизированными. (Предлагается сбросить EntityManager перед вызовом этого метода.)

void deleteAllByIdInBatch([Iterable](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Iterable.html" \o "class or interface in java.lang)<[ID](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> ids)

- Удаляет объекты, идентифицированные заданными идентификаторами, с помощью одного запроса. Этот тип операции оставляет кеш первого уровня JPA и базу данных несинхронизированными. (Предлагается сбросить EntityManager перед вызовом этого метода.)

void deleteAllInBatch()

- Удаляет все объекты в пакетном вызове.

[T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html) getOne([ID](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html) id)

- Возвращает ссылку на объект с заданным идентификатором. В зависимости от того, как реализован провайдер персистентности JPA, очень вероятно, что он всегда будет возвращать экземпляр и выдавать EntityNotFoundException при первом доступе. Некоторые из них будут немедленно отклонять недействительные идентификаторы. Считается устаревшим, предлагается использовать [getReferenceById(ID)](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html#getReferenceById(ID)).

[T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html) getById([ID](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html) id)

- Возвращает ссылку на объект с заданным идентификатором. В зависимости от того, как реализован провайдер персистентности JPA, очень вероятно, что он всегда будет возвращать экземпляр и выдавать EntityNotFoundException при первом доступе. Некоторые из них будут немедленно отклонять недействительные идентификаторы. Считается устаревшим, предлагается использовать [getReferenceById(ID)](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html#getReferenceById(ID)).

[T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html) getReferenceById([ID](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html) id)

- Возвращает ссылку на объект с заданным идентификатором. В зависимости от того, как реализован провайдер персистентности JPA, очень вероятно, что он всегда будет возвращать экземпляр и выдавать EntityNotFoundException при первом доступе. Некоторые из них будут немедленно отклонять недействительные идентификаторы.

<S extends [T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<S> findAll([Example](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Example.html)<S> example)

- Возвращает все объекты, соответствующие заданному [Example](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Example.html). Если совпадений не найдено, возвращается пустое [Iterable](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Iterable.html) значение.

<S extends [T](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)> [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<S> findAll([Example](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Example.html)<S> example, [Sort](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Sort.html) sort)

- Возвращает все объекты, соответствующие заданному [Example](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Example.html), применяя указанный Sort. Если совпадений не найдено, возвращается пустое [Iterable](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Iterable.html) значение.