**Первое приложение с Hibernate**

Сейчас рассмотрим работу с H**ibernate** без использования **Spring**. С помощью **Hiberanate** пообщаемся с базой данных.

Для того, чтобы создать файл с конфигурацией нам нужно поместить его в папку **resources**. В этой папке лежат статические ресурсы, которые используются в нашем **maven** проекте.

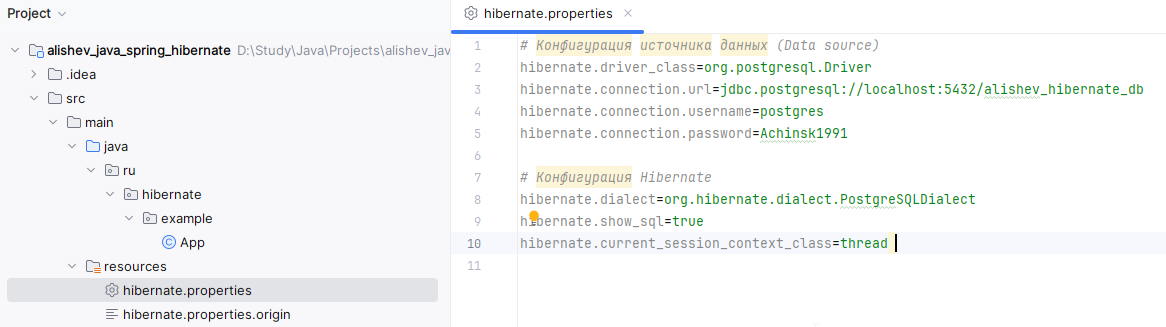
Файл с настройками должен называться именно **hibernate.properties**. В этой конфигурации мы должны описать конфигурацию нашего источника данных, тоесть конфигурацию **jdbc** драйвера и должны описать конфигурацию самого **hiberante**.

**1.** Сначала указываем конфигарацию источника данных. По сути это настройки подключения к БД.

Далее указываем настройки самого **Hibernate:**

**hibernate.dialect** – так как **hibernate** за нас пишет запросы к базе данных мы должны указывать тот диалект базы данной с которой мы будем работать.

**hibernate.show\_sql** – показывает тот sql запрос который hibernate генерирует.

**current\_session\_context.class** —

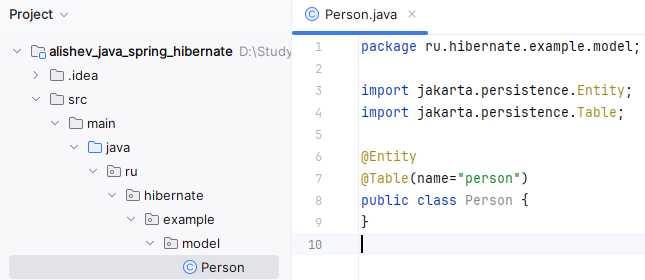
**2.** Создаем таблицу в базе данных, например: **person**



**3.** Так как мы создали таблицу **person** в базе данных, теперь создаем **java** сущность (класс **person**) которая будет соответствовать нашей таблице в БД. Создаем эту сущность в директории **model** так как обычно они всегда хранятся в папке **model**.

Созданный класс person нужно пометить аннотацией **@Entity**. Эта аннотация помечает те классы которые связанны с базой данных. Так как у нас класс **person —** это класс для сущности человек и так как у нас в базе данных есть таблица **person**, то у нас между этим классом и таблицей person в базе данных будет связь. Тоест ьвсе классы которые мы хотим связать с базой данных помечаются аннотацией **@Entity**.

Так же помечаем этот класс аннотацией **@Table**.

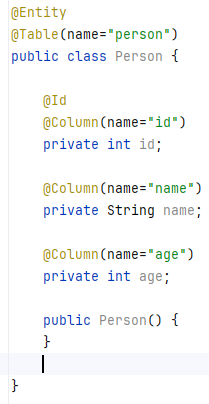


**4.** Теперь в класе **person** опишем все поля которые есть у таблице **person** в базе данных.  
Все классы с аннотацией **@Entity** должны иметь:

**1.** Пустой конструктор

**2.** Хотя бы одно поле с аннотацией **@Id**.

Теперь нужно каждому полю класса person сопоставить название столбца из таблицы **person**. Для этого мы используем аннотацию **@Column.**

Так же нужно создать пустой конструктор, конструктор с полями, геттеры и сеттеры.  


**5.** После того как создали сущность можем начать работать с **hibernate**, для этого:

**Пример получения сущности из базы данных**

1. Получаем конифугурацию.

Этой конфигурации мы должны передать класс который является сущносью(класс который помечен аннотацией **@Entity**). Теперь **hibernate** будет видеть наш класс **Person,** будет понимать, что у нас есть сущность человек и будет понимать, что где то в базе данных должна быть таблица **person** и эта таблица должна соответствовать классу(сущности).

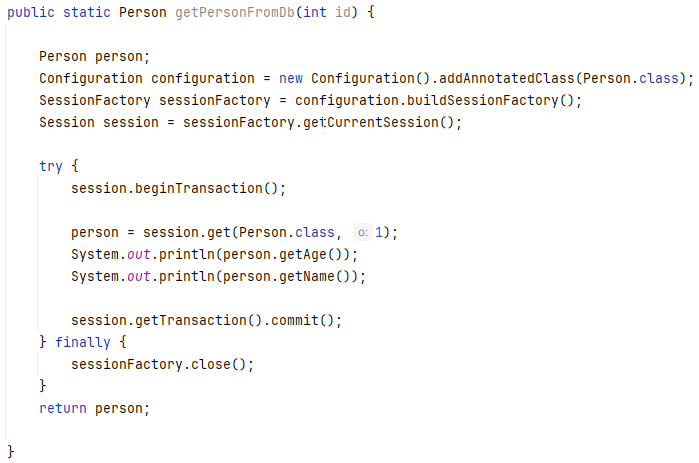
По умолчанию класс **Configuration** читает настройки из файла под названием **hibernate.properties** (файл с настройками подключения).

1. Создаем сессию для работы с **hibernate** с помощью **SessionFactory**.
2. Начинаем транзакцию ***beginTransaction()***
3. Получим человека с **id**=1.

Для этого на этой сессии вызовем метод ***get().*** Этому методу мы должны передать два аргумента: 1. Сущность которую мы хотим получить из БД. 2. Значение id, которое мы хотим найти в базе данных.

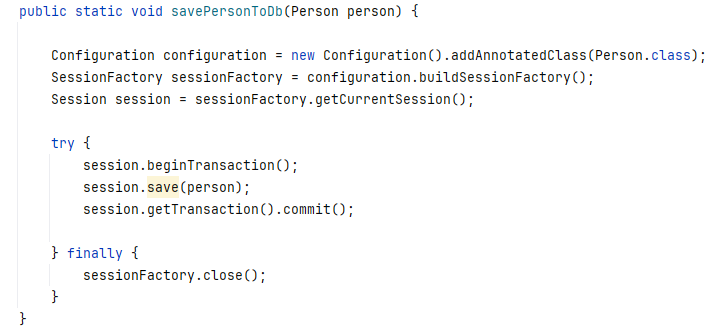
После того как **hibernate** получит нужную строку из таблицы он сам преобразует её в нужный нам объект.

1. Закроем нашу транзацию с помощью метода ***commit()***.
2. Закроем **SessionFactory.**

****

**Пример** **сoхранения сущностей в БД**

У сессии вызывается метод ***save()***.

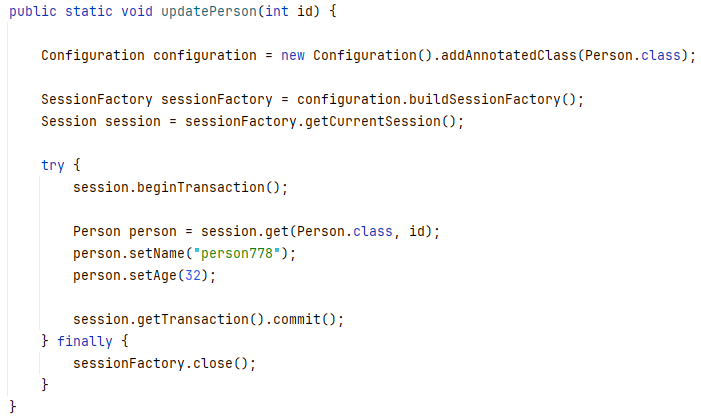
****

**Пример обновления сущностей в БД**

**1.** Сначала нужно получить обновляемую сущность из базы данных с помощью **hibernate**.

**2.** Чтобы обновить данные у полученного объекта в таблице нужно вызвать сеттер и установить новое значение.

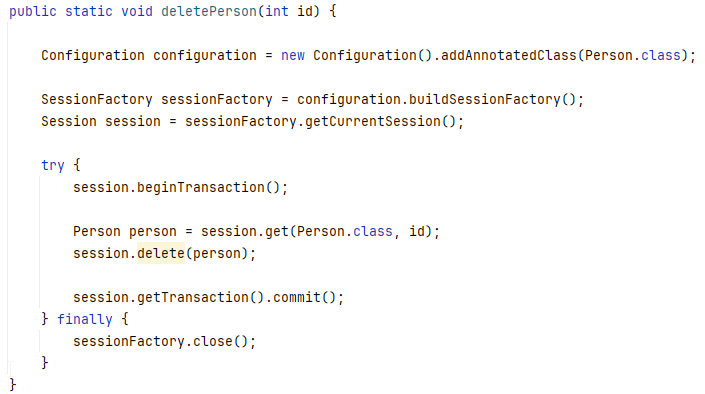
**3.** После того как транзакция будет закоммичена, новое значение сущности появится в таблице базы данных.



**Пример удаления сущностей в БД**

**1.** Сначала нужно получить удаляемую сущность из базы данных с помощью **hibernate**.

**2.** Чтобы удалить сущность из таблицы нужно на сессии вызвать метод **delete()**.



**Дополнительные методы сессии**

**detach()** - отвязать объект от контекста, отсоединяем от hibernate  
**persist()** - Добавить объект к контексту  
**remove()** - удалить объект из таблицы  
**merge()** - вернуть удаленный объект пока транзакция не закомичена