**JdbcTemplate**

Это технология находится на ступень выше чем обычный **JDBC API**, эта технология является тонкой обёрткой над **JDBC API** и облегчает нам взаимодействие с базой данных.

**Проблемы обычной JDBC API**

**1.** Много лишнего кода (создание connection, statement, работа с ResultSet).

**2.** Дублирование кода. При каждом запросе получаем ResultSet, проходимся по нему в цикле и т. д.

**3.** Неинформативный SQLException, который надо везде обрабатывать (при подключении к бд, не существует таблица, не можем добавить строку в таблицу и т. д. ).

Чтобы использовать **JdbcTemplate** нам нужно создать его объект. Будем создавать этот объект с помощью **Spring,** а затем уже в наших классах с помощью аннотации **@Autowired** подключим **JdbcTemplate** и будем им пользоваться.

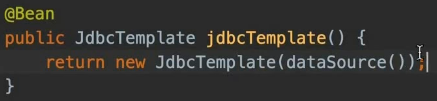
В классе **SpringConfig** можем создать bean **JdbcTemplate** и после этого сможем внедрять его в остальные классы которые работают с базой данных.

Для того, чтобы создать bean **JdbcTemplate** нам нужно в конфигурационном файле **SpringConfig** создать еще один bean, который называется **DaraSource**, он нужен **JdbcTemplate`у,** чтобы указать на нашу базу данных, чтобы он знал в какой базе данных подключаться.

Пример **DataSouce**:



Теперь имея bean **DataSouce** мы можем создать bean **JdbcTemplate:**

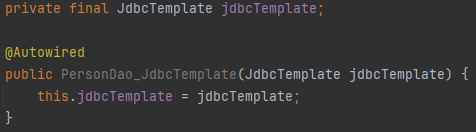
Создаем объект класса **JdbcTemplate** и в конструктор передаем **dataSource**, который создали ранее.

**Важно: таким образом указывать конфигурацию базы данных не правильно, правильно выносить эти данные в отдельный файл,а потом эти данные импортировать. Здесь так сделанно только ради примера.**

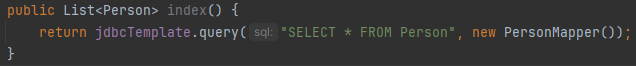
**Работа с JdbcTemplate**

Нужно создать поле которое будет являться нашим **JdbcTemplate.**

Это поле внедрим с помощью аннотации **@Autowired** используя конструктор.



**Чтобы создать запрос нужно использовать метод *query().***

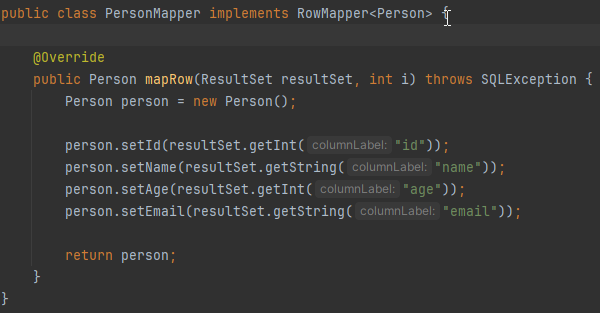


В качестве аругментов нам нужно передать:

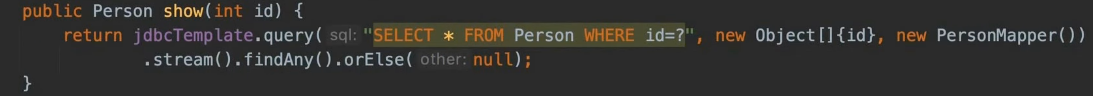
**1.** SQL запрос

**2.** Указать **RowMapper. RowMapper –** это такой объект который отображает строки из таблицы в нашей сущности, тоесть каждую строку полученную в результате этого запроса из нашей таблицы Person? он отобразить в объект класса Person(объект в нашем Java приложении). Этот **RowMapper** мы должны реализовать сами.

Пример реализации **RowMapper`a:**

Этот класс должен реализовывать интерфейс **RowMapper.** В этом интрфейсе есть всего один метод который нужно реализовать который называется ***mapRaw().*** В этом методе мы должны из нашего **ResulSet**`a брать очередую строку и переводить в объект класса **Person.**

**Чтобы создать запрос с параметрами(PreparedStatement) нужно:**

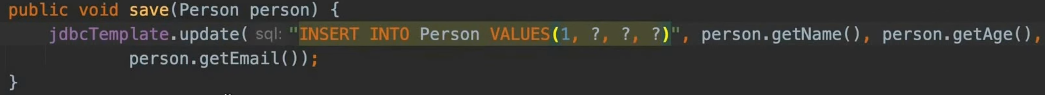
В качестве аругментов нам нужно передать:

**1.** Указать запрос с параметрами.

**2.** У **jdbcTemplate** по умолчанию всегда используется **PreparedStatement** поэтому если мы в запросе пишем вопросительный знак, это значит, что **jdbcTemplate** ждёт, что мы в эти вопросительные подставим какие то значения. Эти значения мы указываем в качестве второго аргумента, для этого мы должны передать массив из значений, которые будут подставлены в наш **PreparedStatement** в качестве вопросительных знаков.

**3.** В качестве третьего аругмента мы используем **RowMapper,** объект который будет переводить строку таблицы в объект **java**.

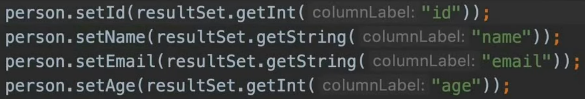
**Чтобы обновить даные в таблице нужно:**

Нужно использовать метод ***update().*** В аругменты этого метода нуэно передать:

1. SQL запрос.

2. Для метода ***update()*** передавать массив значений не нужно как это было с методом ***query().*** Здесь просто поочереди передаём нужные значения. Тоесть сколько у нас вопросительных знаков в SQL запросе столько значений через запятую нужно передать.

**ВАЖНО:** в нашем методе (речь о **RowMapper**) который переводит строку таблицы в объект класса **RowMapper** мы делаем довольно травиальную штуку, мы назначаем поля нашему **Person** колонками у которых такое же название как у наших полей. Тоесть поля в классе **Person** полностью совпадают с полями таблицы **Person.**

Для таких ситуаций нам не обязательно писать свой **RowMapper.** Для этих целей можно использовать готовый **RowMapper,** который есть уже в библиотеке **Spring-jdbc.**

Этот готовый **RowMapper** называется **BeanPropertyRowMapper()<>**, в качестве аругмента

в конструктор передаем тот класс к объектам которого будет производится перевод строк из нашей таблицы. Соответственно это будет работать точно так же как и в случае с аншим собственным **RowMapper`ом** только тперь нам не нужно писать свой собственный класс.

Пример:

****

**Чтобы обновить даные в таблице нужно:**