# 1. Step to query data using JdbcTemplate?

1. JdbcTemplate jdbcTemplate = new JdbcTemplate(datasource)

2. ObjectDto dto = jdbcTemplate.queryForObject(sql, args, dtoMapper)

# 2. What is Result set?

This is cursor that pointing to the current record data.

Result set giống như 1 con trỏ đang trỏ đến 1 record trả về trong data.

# 3. How to create mapper?

1. Create a class implements RowMapper<Dto>

2. Override method mapRow

Object o = new Object();

o.setID(resultSet.getID);

…

# 4. Difference NamedParameterJdbcTemplate and JdbcTemplate?

In JdbcTemplate, we use ? placeholder for the value, like this:

Object[] args = new Object[] {"x", "y"};

String sql = "select \* from foo where a = ? and b = ?";

jdbcTemplate.query(sql, args, resultSetExtractor);

But in NamedParameterJdbcTemplate, we assign names for parameters, like this:

String sql = "select \* from foo where a = :mya and b = :myb";

Map<String, Object> argMap = new HashMap<String, Object>();

argMap.put("mya", "x");

argMap.put("myb", "y");

namedParameterJdbcTemplate.query(sql, argMap, resultSetExtractor);

generating the same SQL as the first example (and running the query is the time-intensive part, the performance of the argument insertion is a non-issue.).

The idea is that matching the arguments by name is less error-prone than having to specify them in a particular order. In a real-life application typically SQL is stored in a separate file from the DAO code, and it may be easy to accidentally get the parameters in the wrong order.

<https://stackoverflow.com/questions/16359316/namedparameterjdbctemplate-vs-jdbctemplate>