|  |
| --- |
| Entity |
| // @Entity  @Data  public class RolesOfUser      implements EntityInterface<RolesOfUser, RolesOfUserKey>  {      private RolesOfUserKey rouKey;      @Override      public void updateNotId(RolesOfUser o2) {          // ko lam gi ca          return;      }      public RolesOfUser(RolesOfUserKey rouKey) {          this.rouKey = rouKey;      }    } |

|  |
| --- |
| EntityKey |
| @Data  public class RolesOfUserKey {      private Long userId;      private Long roleId;        public RolesOfUserKey(Long userId, Long roleId) {          this.userId = userId;          this.roleId = roleId;      }    } |

|  |
| --- |
| RowMapper |
| public class RolesOfUserRowMapper      implements RowMapper<RolesOfUser>  {      @Override      @Nullable      public RolesOfUser mapRow(ResultSet ketQuaTraVe, int sttHang) throws SQLException {          Long Userid = ketQuaTraVe.getLong("Userid");          Long Roleid = ketQuaTraVe.getLong("Roleid");          RolesOfUserKey rouKey1 = new RolesOfUserKey(Userid, Roleid);          RolesOfUser rou1 = new RolesOfUser(rouKey1);          return rou1;      }    } |

|  |
| --- |
| Repo |
| @Repository  public class RolesOfUserRepo      implements RepoInterface<RolesOfUser, RolesOfUserKey>  {      @Autowired      private JdbcTemplate jdbcTemplate;      private RolesOfUserRowMapper rouRowMapper = new RolesOfUserRowMapper();      @Override      public RolesOfUser findById(RolesOfUserKey tKey) {          String sql =              " SELECT "              + " \* "              + " FROM RolesOfUser "              + " WHERE "                  + " Userid= ? "                  + " AND Roleid= ? ";          PreparedStatementCreator psCreator = con->{              PreparedStatement ps = con.prepareStatement(sql);                ps.setLong(1, tKey.getUserId());              ps.setLong(2, tKey.getRoleId());              return ps;          };          return this.jdbcTemplate.query(psCreator, this.rouRowMapper).stream().findFirst().orElse(null);      }      @Override      public List<RolesOfUser> getList(Integer sttPage, Integer sizePage) {          String sql =              "SELECT "              + " DISTINCT "              + " \* "              + " FROM RolesOfUser "              + " WHERE 1=1 "              + " ORDER BY Userid DESC, Roleid DESC "              + " OFFSET ? ROW "              + " FETCH NEXT ? ROW ONLY ";          PreparedStatementCreator psc = con-> {              PreparedStatement ps = con.prepareStatement(sql);              ps.setInt(1, ( (sizePage-1)\*sttPage) );              ps.setInt(2, sizePage);              return ps;          };          return this.jdbcTemplate.query(psc, this.rouRowMapper);      }      @Override      public RolesOfUser create(RolesOfUser t) {          // PK của rou là FK nên không được set null          String sql =              " INSERT INTO RolesOfUser "              + " (Userid, Roleid) "              + " VALUES (?, ?) ";          // PK của rou là FK nên không cần get Key trả về          PreparedStatementCreator psc = con->{              PreparedStatement ps = con.prepareStatement(sql);              ps.setLong(1, t.getRouKey().getUserId());              ps.setLong(2, t.getRouKey().getRoleId());              return ps;          };          jdbcTemplate.update(psc);          return this.findById(t.getRouKey());      }      @Override      public RolesOfUser update(RolesOfUserKey tKey, RolesOfUser t) {          t.setRouKey(tKey);          String sql =              " UPDATE RolesOfUser "                  + " SET "                      + " "                  + " WHERE "                      + " Userid = ? "                      + " AND Roleid = ? ";          PreparedStatementCreator psc = con->{              PreparedStatement ps = con.prepareStatement(sql);                ps.setLong(1, t.getRouKey().getUserId());              ps.setLong(2, t.getRouKey().getRoleId());              return ps;          };          jdbcTemplate.update(psc);          return this.findById(t.getRouKey());      }      @Override      public Boolean delete(RolesOfUserKey tKey) {          String sql=              " DELETE FROM RolesOfUser "              + " WHERE "                  + " Userid = ? "                  + " AND Roleid = ? ";          PreparedStatementCreator psc = con->{              PreparedStatement ps = con.prepareStatement(sql);              ps.setLong(1, tKey.getUserId());              ps.setLong(2, tKey.getRoleId());              return ps;          };          jdbcTemplate.update(psc);          return true;      }      //🟩🟩🟩🟩🟩🟩hết phần implements🟩🟩🟩🟩🟩🟩//      public List<RolesOfUser> findByIdUser(Long idUser){          String sql = "SELECT \* FROM RolesOfUser WHERE Userid = ?";          PreparedStatementCreator psc = con->{              PreparedStatement ps = con.prepareStatement(sql);              ps.setLong(1, idUser);              return ps;          };          return jdbcTemplate.query(psc, this.rouRowMapper);      }  } |

|  |
| --- |
| Service |
| @Service  public class RolesOfUserService      implements ServiceInterface<RolesOfUser, RolesOfUserKey>  {      @Autowired      RolesOfUserRepo rouRepo;      @Override      public RolesOfUser findById(RolesOfUserKey tKey) {          return this.rouRepo.findById(tKey);      }      @Override      public List<RolesOfUser> getList(Integer sttPage, Integer sizePage) {          return this.rouRepo.getList(sttPage, sizePage);      }      @Override      public RolesOfUser create(RolesOfUser t) {          // PK là FK nên không được set PK thành null          RolesOfUser rou1 = this.findById(t.getRouKey());          if(rou1!= null){              return null;          }          return this.rouRepo.create(t);      }      @Override      public RolesOfUser update(RolesOfUserKey tKey, RolesOfUser t) {            RolesOfUser rou1 = this.findById(tKey);          if(rou1==null){              return null;          }          return this.update(tKey, t);      }      @Override      public Boolean delete(RolesOfUserKey tKey) {          return this.rouRepo.delete(tKey);      }        //🟩🟩🟩🟩🟩🟩hết phần implements🟩🟩🟩🟩🟩🟩//      public List<RolesOfUser> findByIdUser(Long idUser){          return this.rouRepo.findByIdUser(idUser);      }  } |