# Research for Final Thesis: Comparison of Spring JDBC, Hibernate and MyBatis

This survey is part of a final thesis project comparing three popular Java-based approaches to database interaction: Spring JDBC, Hibernate, and MyBatis. The goal is to gather developers' opinions to better understand strengths and weaknesses of each approach.

* Indicates required question

1. How experienced are you in programming? *(Optional)*

   *Mark only one oval.*

   - ◯ Student
   - ◯ Junior
   - ◯ Mid
   - ◯ Senior
   - ◯ Other: _____

2. How much did you enjoy using the following technologies? *

   *Mark only one oval per row.*

| | Never Used | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|---|
| **Spring JDBC** | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Hibernate** | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| **MyBatis** | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |

3. If you haven't used any of the technologies listed above, what technology do you use or have used instead?*(Optional)*

   _____

Simple query examples

Please provide your opinion based on your experience and provided examples.

There is no need to examine it in high detail.

Response example:

4. Technology 1 necessary code: *

```
@Repository
@AllArgsConstructor
public class DrzavaRepositoryR {

    private final JdbcTemplate jdbcTemplate;

    public List<Drzava> getAllDrzava() {
        var sql =
                """
                SELECT
                    id_drzava,
                    naziv,
                    oznaka
                FROM drzava
                """;

        return jdbcTemplate.query(sql, new BeanPropertyRowMapper<>(Drzava.class));
    }
}
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Learning Curve** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Robustness** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Performance (Expected)** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Customizability** | ◯ | ◯ | ◯ | ◯ | ◯ |

5. Technology 2 necessary code: *

```
public interface DrzavaRepositoryH extends JpaRepository<Drzava, Integer>{}
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Learning Curve** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Robustness** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Performance (Expected)** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Customizability** | ◯ | ◯ | ◯ | ◯ | ◯ |

6. Technology 3 necessary code: *

```java
@Mapper
public interface DrzavaMapperB {
    public List<Drzava> getAllDrzava();
}
```

```xml
<mapper namespace="hr.tvz.hrsustav.mapper.DrzavaMapperB">

    <select id="getAllDrzava" resultType="hr.tvz.hrsustav.entity.Drzava">
        SELECT
            id_drzava,
            naziv,
            oznaka
        FROM drzava
    </select>

</mapper>
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Learning Curve** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Robustness** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Performance (Expected)** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Customizability** | ◯ | ◯ | ◯ | ◯ | ◯ |

## Complex query filter examples

Please provide your opinion based on your experience and provided examples. There is no need to examine it in high detail.

Response example:


```
[
  {
    "idRadnik": 1,
    "radnikNadredFk": null,
    "ime": "Pero",
    "prezime": "Perić",
    "oib": "01234567890",
    "spol": "M",
    "datumRod": "2001-01-01",
    "radnikVrstaNaziv": "Vlastiti",
    "odjelFk": 1,
    "idUgovorRad": 19,
    "datumOdRadnikOdjel": "2025-05-17",
    "datumOdUgovorRad": "2025-07-02"
  }
]
```

7. Technology 1 necessary code: *

```java
@Mapper
public interface RadnikMapperB {
    public List<RadnikDto> getAllRadnik(RadnikGetRequest request);
}
```

```xml
<mapper namespace="hr.tvz.hrsustav.mapper.RadnikMapperB">
    <select id="getAllRadnik" resultType="hr.tvz.hrsustav.dto.RadnikDto" parameterType="hr.tvz.hrsustav.request.RadnikGetRequest">
        SELECT DISTINCT ON (r.id_radnik)
            r.id_radnik,
            r.radnik_nadred_fk,
            r.ime,
            r.prezime,
            r.oib,
            s.oznaka AS spol,
            r.datum_rod,
            rv.naziv AS radnikVrstaNaziv,
            ro.odjel_fk,
            ur.id_ugovor_rad,
            ur.datum_od AS datumOdUgovorRad,
            ro.datum_od AS datumOdRadnikOdjel
        FROM radnik r
            INNER JOIN spol s ON s.id_spol = r.spol_fk
            INNER JOIN radnik_vrsta rv ON rv.id_radnik_vrsta = r.radnik_vrsta_fk
            INNER JOIN radnik_odjel ro ON ro.radnik_fk = r.id_radnik
            INNER JOIN ugovor_rad ur ON ur.radnik_fk = r.id_radnik
        <where>
            <if test="idRadnik != null">
                AND r.id_radnik = #{idRadnik}
            </if>
            <if test="radnikNadredFk != null">
                AND r.radnik_nadred_fk = #{radnikNadredFk}
            </if>
            <if test="ime != null">
                AND r.ime = #{ime}
            </if>
            <if test="prezime != null">
                AND r.prezime = #{prezime}
            </if>
            <if test="oib != null">
                AND r.oib = #{oib}
            </if>
            <if test="spolFk != null">
                AND r.spol_fk = #{spolFk}
            </if>
            <if test="datumRod != null">
                AND r.datum_rod = #{datumRod}
            </if>
            <if test="radnikVrstaFk != null">
                AND r.radnik_vrsta_fk = #{radnikVrstaFk}
            </if>
            <if test="inOdjel == true">
                AND ro.datum_od <![CDATA[<=]]> CURRENT_DATE
                AND (datum_do <![CDATA[>=]]> CURRENT_DATE OR datum_do IS NULL)
            </if>
            <if test="isZaposlen == true">
                AND ur.datum_od <![CDATA[<=]]> CURRENT_DATE
                AND (ur.datum_do IS NULL OR ur.datum_do <![CDATA[>=]]> CURRENT_DATE)
            </if>
        </where>
        ORDER BY
            r.id_radnik,
            ro.datum_od DESC,
            ur.datum_od DESC;
    </select>
</mapper>
```

*Mark only one oval per row.*

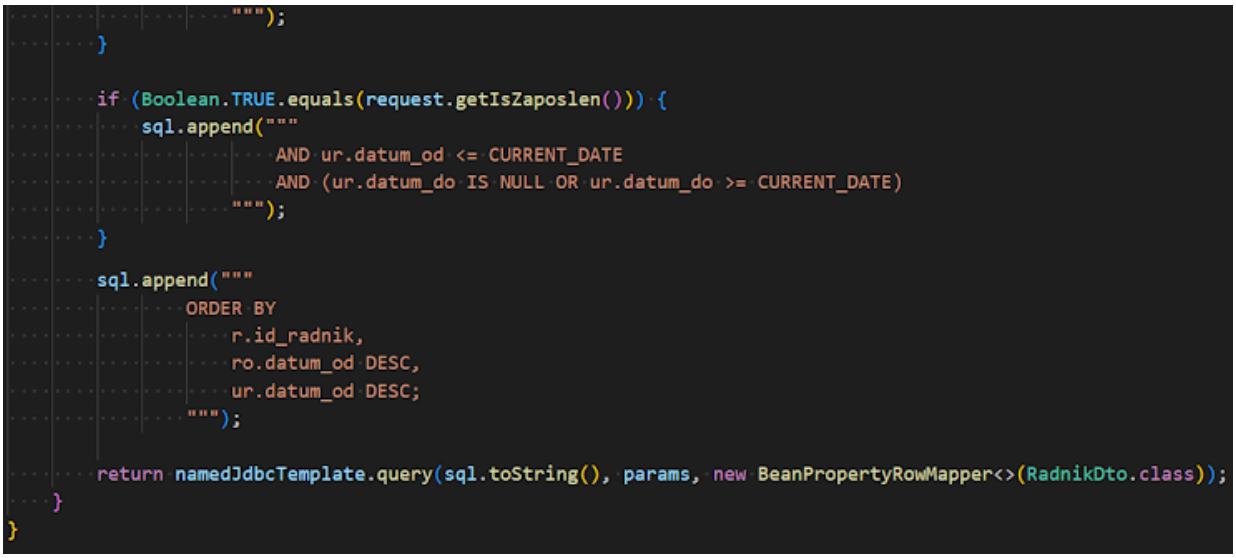
| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |

Learning Curve

| | | | | | |
|---|---|---|---|---|---|
| Learning Curve | | | | | |
| Robustness | | | | | |
| Performance (Expected) | | | | | |
| Customizability | | | | | |

8. Technology 2 necessary code: *

```java
@Repository
@AllArgsConstructor
public class RadnikRepositoryR {

    private NamedParameterJdbcTemplate namedJdbcTemplate;

    public List<RadnikDto> getAllRadnik(RadnikGetRequest request) {
        var sql = new StringBuilder("""
                    SELECT DISTINCT ON (r.id_radnik)
                        r.id_radnik,
                        r.radnik_nadred_fk,
                        r.ime,
                        r.prezime,
                        r.oib,
                        s.oznaka AS spol,
                        r.datum_rod,
                        rv.naziv AS radnikVrstaNaziv,
                        ro.odjel_fk,
                        ur.id_ugovor_rad,
                        ur.datum_od AS datumOdUgovorRad,
                        ro.datum_od AS datumOdRadnikOdjel
                    FROM radnik r
                        INNER JOIN spol s ON s.id_spol = r.spol_fk
                        INNER JOIN radnik_vrsta rv ON rv.id_radnik_vrsta = r.radnik_vrsta_fk
                        INNER JOIN radnik_odjel ro ON ro.radnik_fk = r.id_radnik
                        INNER JOIN ugovor_rad ur ON ur.radnik_fk = r.id_radnik
                    WHERE 1=1
                """);

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

        if (request.getIdRadnik() != null) {
            sql.append(str:" AND r.id_radnik = :idRadnik");
            params.put(key:"idRadnik", request.getIdRadnik());
        }

        if (request.getRadnikNadredFk() != null) {
            sql.append(str:" AND r.radnik_nadred_fk = :radnikNadredFk");
            params.put(key:"radnikNadredFk", request.getRadnikNadredFk());
        }

        if (request.getIme() != null) {
            sql.append(str:" AND r.ime = :ime");
            params.put(key:"ime", request.getIme());
        }

        if (request.getPrezime() != null) {
            sql.append(str:" AND r.prezime = :prezime");
            params.put(key:"prezime", request.getPrezime());
        }

        if (request.getOib() != null) {
            sql.append(str:" AND r.oib = :oib");
            params.put(key:"oib", request.getOib());
        }

        if (request.getSpolFk() != null) {
            sql.append(str:" AND r.spol_fk = :spolFk");
            params.put(key:"spolFk", request.getSpolFk());
        }

        if (request.getDatumRod() != null) {
            sql.append(str:" AND r.datum_rod = :datumRod");
            params.put(key:"datumRod", request.getDatumRod());
        }

        if (request.getRadnikVrstaFk() != null) {
            sql.append(str:" AND r.radnik_vrsta_fk = :radnikVrstaFk");
            params.put(key:"radnikVrstaFk", request.getRadnikVrstaFk());
        }

        if (Boolean.TRUE.equals(request.getInOdjel())) {
            sql.append("""
                        AND ro.datum_od <= CURRENT_DATE
                        AND (ro.datum_do >= CURRENT_DATE OR ro.datum_do IS NULL)
```



```
                """);
        }

        if (Boolean.TRUE.equals(request.getIsZaposlen())) {
            sql.append("""
                    AND ur.datum_od <= CURRENT_DATE
                    AND (ur.datum_do IS NULL OR ur.datum_do >= CURRENT_DATE)
                """);
        }

        sql.append("""
                ORDER BY
                    r.id_radnik,
                    ro.datum_od DESC,
                    ur.datum_od DESC;
                """);

        return namedJdbcTemplate.query(sql.toString(), params, new BeanPropertyRowMapper<>(RadnikDto.class));
    }
}
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Learning Curve** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Robustness** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Performance (Expected)** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Customizability** | ◯ | ◯ | ◯ | ◯ | ◯ |

9. Technology 3 necessary code: *

```java
@Repository
public class RadnikRepositoryC {

    private final EntityManager entityManager;

    public RadnikRepositoryC(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    public List<RadnikDto> findAllRadnik(RadnikGetRequest request) {

        CriteriaBuilder cb = entityManager.getCriteriaBuilder();
        CriteriaQuery<RadnikDto> query = cb.createQuery(RadnikDto.class);
        Root<Radnik> r = query.from(Radnik.class);

        Join<Radnik, Spol> s = r.join("spolFk", JoinType.INNER);
        Join<Radnik, RadnikVrsta> rv = r.join("radnikVrstaFk", JoinType.INNER);
        Join<Radnik, RadnikOdjel> ro = r.join("radnikOdjel", JoinType.INNER);
        Join<Radnik, UgovorRad> ur = r.join("ugovorRad", JoinType.INNER);

        query.select(cb.construct(
                RadnikDto.class,
                r.get("idRadnik"),
                r.get("radnikNadredFk").get("idRadnik"),
                r.get("ime"),
                r.get("prezime"),
                r.get("oib"),
                s.get("oznaka"),
                r.get("datumRod"),
                rv.get("naziv"),
                ro.get("odjelFk").get("idOdjel"),
                ur.get("idUgovorRad"),
                ro.get("datumOd"),
                ur.get("datumOd")));

        List<Predicate> predicates = new ArrayList<>();

        addPredicateIfNotNull(predicates, cb, r.get("idRadnik"), request.getIdRadnik());
        addPredicateIfNotNull(predicates, cb, r.get("radnikNadredFk"), request.getRadnikNadredFk());
        addPredicateIfNotNull(predicates, cb, r.get("ime"), request.getIme());
        addPredicateIfNotNull(predicates, cb, r.get("prezime"), request.getPrezime());
        addPredicateIfNotNull(predicates, cb, r.get("oib"), request.getOib());
        addPredicateIfNotNull(predicates, cb, r.get("spolFk"), request.getSpolFk());
        addPredicateIfNotNull(predicates, cb, r.get("datumRod"), request.getDatumRod());
        addPredicateIfNotNull(predicates, cb, r.get("radnikVrstaFk"), request.getRadnikVrstaFk());

        if (Boolean.TRUE.equals(request.getInOdjel())) {
            predicates.add(cb.lessThanOrEqualTo(ro.get("datumOd"), cb.currentDate()));
            predicates.add(cb.or(
                    cb.greaterThanOrEqualTo(ro.get("datumDo"), cb.currentDate()),
                    cb.isNull(ro.get("datumDo"))));
        }

        if (Boolean.TRUE.equals(request.getIsZaposlen())) {
            predicates.add(cb.lessThanOrEqualTo(ur.get("datumOd"), cb.currentDate()));
            predicates.add(cb.or(
                    cb.greaterThanOrEqualTo(ur.get("datumDo"), cb.currentDate()),
                    cb.isNull(ur.get("datumDo"))));
        }

        query.where(cb.and(predicates.toArray(new Predicate[0])));
        query.distinct(true);

        query.orderBy(
                cb.asc(r.get("idRadnik")),
                cb.desc(ro.get("datumOd")),
                cb.desc(ur.get("datumOd")));

        return entityManager.createQuery(query).getResultList();
    }

    private <T> void addPredicateIfNotNull(List<Predicate> predicates, CriteriaBuilder cb, Path<T> path, T value) {
        if (value != null) {
            predicates.add(cb.equal(path, value));
        }
    }
}
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | | | | | |
| **Ease of Maintenance** | | | | | |
| **Learning Curve** | | | | | |
| **Robustness** | | | | | |
| **Performance (Expected)** | | | | | |
| **Customizability** | | | | | |

Nested object query

Please provide your opinion based on your experience and provided examples. There is no need to examine it in high detail.

Response example:

```
[
  {
    "idMjesto": 1,
    "naziv": "Split",
    "postBr": "21000",
    "drzavaFk": {
      "idDrzava": 1,
      "naziv": "Republika Hrvatska",
      "oznaka": "RH"
    }
  },
  {
    "idMjesto": 2,
    "naziv": "Zagreb",
    "postBr": "10000",
    "drzavaFk": {
      "idDrzava": 1,
      "naziv": "Republika Hrvatska",
      "oznaka": "RH"
    }
  }
]
```

10. Technology 1 code: *

```
public interface MjestoRepositoryH extends JpaRepository<Mjesto, Integer> {}
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | | | | | |
| **Ease of Maintenance** | | | | | |
| **Learning Curve** | | | | | |
| **Robustness** | | | | | |
| **Performance (Expected)** | | | | | |
| **Customizability** | | | | | |

11. Technology 2 code: *

```xml
<mapper namespace="hr.tvz.hrsustav.mapper.MjestoMapperB">

    <resultMap id="mjestoWithDrzavaResultMap" type="hr.tvz.hrsustav.entity.Mjesto">
        <id property="idMjesto" column="id_mjesto"/>
        <result property="naziv" column="naziv"/>
        <result property="postBr" column="post_br"/>

        <association property="drzavaFk" javaType="hr.tvz.hrsustav.entity.Drzava">
            <id property="idDrzava" column="id_drzava"/>
            <result property="naziv" column="drzava_naziv"/>
            <result property="oznaka" column="drzava_oznaka"/>
        </association>
    </resultMap>

    <select id="getAllMjesto" resultMap="mjestoWithDrzavaResultMap">
        SELECT
            m.id_mjesto,
            m.naziv,
            m.post_br,
            d.id_drzava,
            d.naziv AS drzava_naziv,
            d.oznaka AS drzava_oznaka
        FROM mjesto m
            JOIN drzava d ON m.drzava_fk = d.id_drzava
    </select>

</mapper>
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Learning Curve** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Robustness** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Performance (Expected)** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Customizability** | ◯ | ◯ | ◯ | ◯ | ◯ |

12. Technology 3 necessary code: *

```java
@Repository
@AllArgsConstructor
public class MjestoRepositoryR {

    private final JdbcTemplate jdbcTemplate;

    public List<Mjesto> getAllMjesto() {
        var sql = """
                SELECT
                    m.id_mjesto,
                    m.naziv,
                    m.post_br,
                    d.id_drzava,
                    d.naziv AS drzava_naziv,
                    d.oznaka AS drzava_oznaka
                FROM mjesto m
                JOIN drzava d ON m.drzava_fk = d.id_drzava
                """;

        return jdbcTemplate.query(sql, (rs, rowNum) -> {
            var drzava = new Drzava();
            drzava.setIdDrzava(rs.getInt(columnLabel:"id_drzava"));
            drzava.setNaziv(rs.getString(columnLabel:"drzava_naziv"));
            drzava.setOznaka(rs.getString(columnLabel:"drzava_oznaka"));

            var mjesto = new Mjesto();
            mjesto.setIdMjesto(rs.getInt(columnLabel:"id_mjesto"));
            mjesto.setNaziv(rs.getString(columnLabel:"naziv"));
            mjesto.setPostBr(rs.getString(columnLabel:"post_br"));

            mjesto.setDrzavaFk(drzava);

            return mjesto;
        });
    }
}
```

*Mark only one oval per row.*

| | 1 - Very poor | 2 - Poor | 3 - Average | 4 - Good | 5 - Excellent |
|---|---|---|---|---|---|
| **Readability** | ◯ | ◯ | ◯ | ◯ | ◯ |
| **Ease of Maintenance** | ◯ | ◯ | ◯ | ◯ | ◯ |

Learning Curve

| | | | | | |
|---|---|---|---|---|---|
| Learning Curve Robustness | | | | | |
| Robustness Performance | | | | | |
| (Expected) Performance (Expected) Customizability | | | | | |
| Customizability | | | | | |