

# JISEL

(Java Interface Segregation Library)

## Integration with Spring Data JPA

# Large Interface Definition

*The business requirements for a reporting solution against a company employees database have been modeled as follow:*

ReportsRepository
<b>List&lt;Employee&gt; findByFirstNameContainingOrLastNameContaining(String, String)</b> <b>List&lt;Employee&gt; findByEmailAddressContaining(String)</b> <b>List&lt;Object[]&gt; countEmployeesSharingSameHomeAddress()</b> <b>List&lt;Object[]&gt; totalAccruedVacationsPerDepartment()</b> <b>List&lt;Employee&gt; employeesWithFirstNameAndLastNameListedInMoreThan1Department()</b> <b>List&lt;Employee&gt; employeesWithDifferentIDsUsingSameEmailAddresses()</b> <b>List&lt;Object[]&gt; employeesWithMoreThanXYearsAndSalaryLessThanY(int, long)</b>

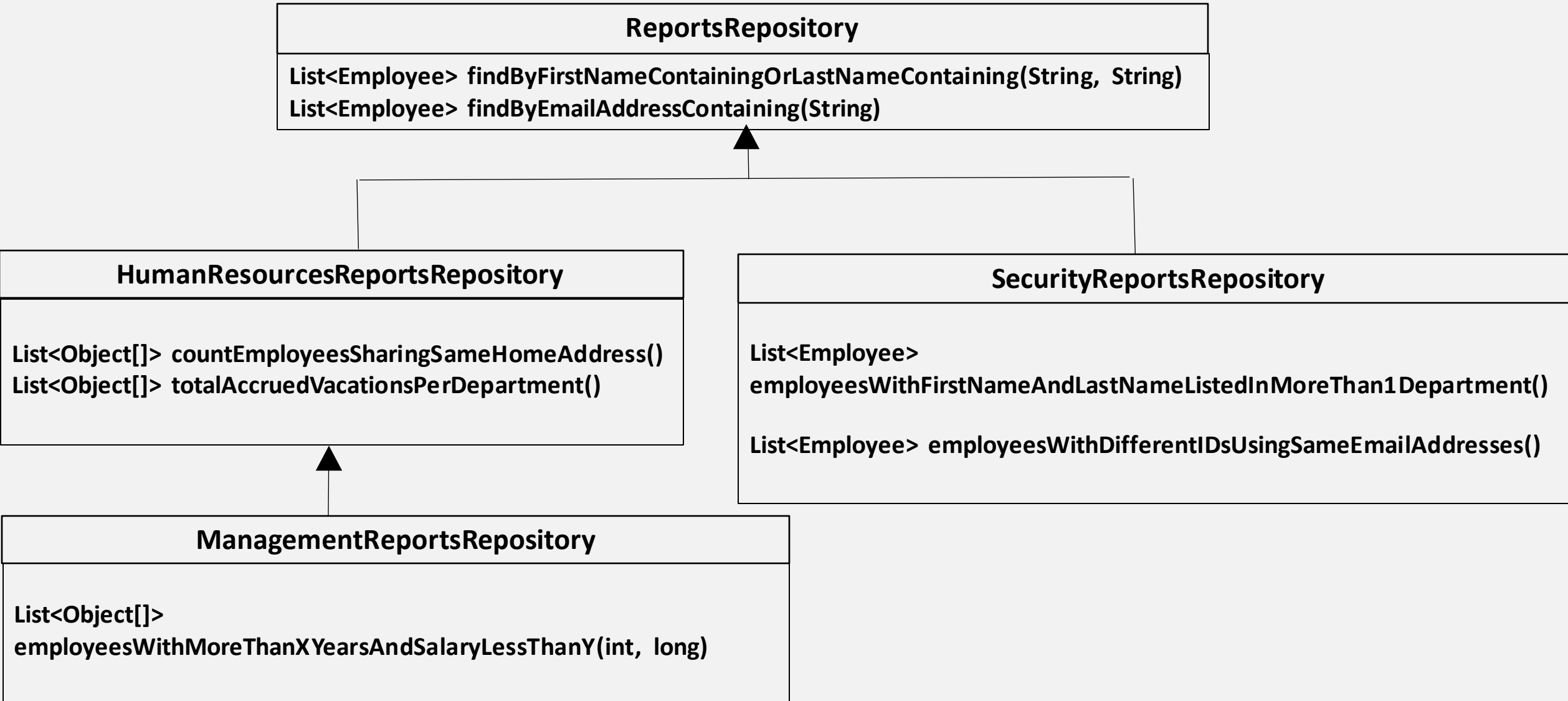
Main goal is to segregate the large interface defined above, so that we create 1 interface for each one of these 3 teams:

Human Resources Team,

Security Team,

Management Team

# Expected Outcome



# Maven Project Setup

## Dependencies:

- Spring Boot (starter-data-jpa & starter-web)
- H2
- JISEL

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.jisel</groupId>
    <artifactId>jisel</artifactId>
    <version>1.2</version>
  </dependency>
</dependencies>
```

## Annotation Processor:

- JISEL

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>${maven-compiler-plugin.version}</version>
      <configuration>
        <release>17</release>
        <compilerArgs>-Xlint:unchecked</compilerArgs>
        <annotationProcessorPaths>
          <path>
            <groupId>org.jisel</groupId>
            <artifactId>jisel</artifactId>
            <version>1.2</version>
          </path>
        </annotationProcessorPaths>
      </configuration>
    </plugin>
  </plugins>
</build>
```

# Entities and Data Setup

## Employee.java

```
@Entity
public class Employee {

    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    private String firstName;

    2 usages
    private String lastName;

    2 usages
    private int age;

    2 usages
    private String homeAddress;

    2 usages
    private String emailAddress;

    2 usages
    private long phoneNumber;

    2 usages
    private String position;

    2 usages
    private String department;

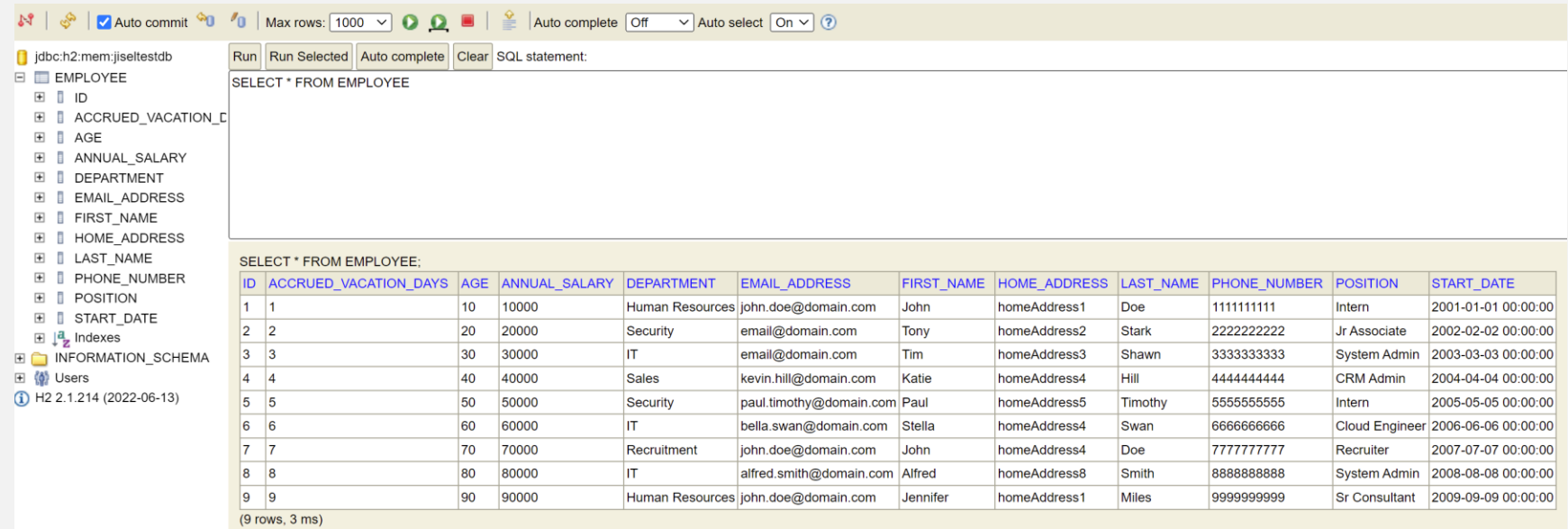
    2 usages
    private Long annualSalary;

    2 usages
    private Date startDate;

    2 usages
    private int accruedVacationDays;

    // getters/setters...
```

## data.sql



The screenshot shows a database IDE interface. On the left, a tree view displays the database structure: jdbc:h2:mem:jiseltestdb, EMPLOYEE table, and INFORMATION\_SCHEMA. The EMPLOYEE table has columns: ID, ACCRUED\_VACATION\_DAYS, AGE, ANNUAL\_SALARY, DEPARTMENT, EMAIL\_ADDRESS, FIRST\_NAME, HOME\_ADDRESS, LAST\_NAME, PHONE\_NUMBER, POSITION, and START\_DATE. The main area shows a SQL statement: SELECT \* FROM EMPLOYEE. Below the statement, the query result is displayed as a table with 9 rows and 12 columns. The columns are: ID, ACCRUED\_VACATION\_DAYS, AGE, ANNUAL\_SALARY, DEPARTMENT, EMAIL\_ADDRESS, FIRST\_NAME, HOME\_ADDRESS, LAST\_NAME, PHONE\_NUMBER, POSITION, and START\_DATE. The rows contain data for 9 employees.

ID	ACCRUED_VACATION_DAYS	AGE	ANNUAL_SALARY	DEPARTMENT	EMAIL_ADDRESS	FIRST_NAME	HOME_ADDRESS	LAST_NAME	PHONE_NUMBER	POSITION	START_DATE
1	1	10	10000	Human Resources	john.doe@domain.com	John	homeAddress1	Doe	1111111111	Intern	2001-01-01 00:00:00
2	2	20	20000	Security	email@domain.com	Tony	homeAddress2	Stark	2222222222	Jr Associate	2002-02-02 00:00:00
3	3	30	30000	IT	email@domain.com	Tim	homeAddress3	Shawn	3333333333	System Admin	2003-03-03 00:00:00
4	4	40	40000	Sales	kevin.hill@domain.com	Katie	homeAddress4	Hill	4444444444	CRM Admin	2004-04-04 00:00:00
5	5	50	50000	Security	paul.timothy@domain.com	Paul	homeAddress5	Timothy	5555555555	Intern	2005-05-05 00:00:00
6	6	60	60000	IT	bella.swan@domain.com	Stella	homeAddress4	Swan	6666666666	Cloud Engineer	2006-06-06 00:00:00
7	7	70	70000	Recruitment	john.doe@domain.com	John	homeAddress4	Doe	7777777777	Recruiter	2007-07-07 00:00:00
8	8	80	80000	IT	alfred.smith@domain.com	Alfred	homeAddress8	Smith	8888888888	System Admin	2008-08-08 00:00:00
9	9	90	90000	Human Resources	john.doe@domain.com	Jennifer	homeAddress1	Miles	9999999999	Sr Consultant	2009-09-09 00:00:00

(9 rows, 3 ms)

# Define Finders and Native Queries

## ReportsRepository.java

```
List<Employee> findByFirstNameContainingOrLastNameContaining(String firstNameSearchCriteria, String lastNameSearchCriteria);
```

⚡ Mohamed Ashraf Bayor

```
List<Employee> findByEmailAddressContaining(String emailAddressSearchCriteria);
```

⚡ Mohamed Ashraf Bayor

```
@Query(
    nativeQuery = true,
    value = """
        SELECT home_address, COUNT(id) total
        FROM employee
        GROUP BY home_address
        HAVING total > 1
        ORDER BY total DESC
        """
)
List<Object[]> countEmployeesSharingSameHomeAddress();
```

⚡ Mohamed Ashraf Bayor

```
@Query(
    nativeQuery = true,
    value = """
        SELECT department, SUM(accrued_vacation_days) total
        FROM employee
        GROUP BY department
        ORDER BY total desc
        """
)
List<Object[]> totalAccruedVacationsPerDepartment();
```

⚡ Mohamed Ashraf Bayor

```
@Query(
    nativeQuery = true,
    value = """
        SELECT e1.*
        FROM employee e1
        INNER JOIN employee e2
        ON (e1.department <> e2.department
        AND e1.first_name = e2.first_name
        AND e1.last_name = e2.last_name)
        ORDER BY e1.first_name, e1.last_name
        """
)
List<Employee> employeesWithFirstNameAndLastNameListedInMoreThan1Department();
```

```
@Query(
    nativeQuery = true,
    value = """
        SELECT distinct e1.*
        FROM employee e1
        INNER JOIN employee e2
        ON (e1.email_address = e2.email_address
        AND e1.ID <> e2.ID)
        ORDER BY email_address
        """
)
List<Employee> employeesWithDifferentIDsUsingSameEmailAddresses();
```

⚡ Mohamed Ashraf Bayor

```
@Query(
    nativeQuery = true,
    value = """
        SELECT first_name, last_name, FORMATDATETIME(start_date, 'yyyy-MM-dd'), annual_salary, DATEDIFF(year, start_date, NOW()) totalYears
        FROM employee
        WHERE DATEDIFF(year, start_date, NOW()) >= :totalYears
        AND annual_salary < :annualSalary
        ORDER BY totalYears DESC
        """
)
List<Object[]> employeesWithMoreThanXYearsAndSalaryLessThanY(@Param("totalYears") int totalYearsCriteria, @Param("annualSalary") long annualSalaryCriteria);
```

# Apply JISEL Annotations

## ReportsRepository.java

```
@Detach(profile = TOPLEVEL_KW,
    rename = "CommonReportsRepository",
    applyAnnotations = "@org.springframework.stereotype.Repository",
    superInterfaces = Repository.class,
    firstSuperInterfaceGenerics = {Employee.class, Long.class}
)
@Detach(profile = HUMAN_RESOURCES,
    rename = HUMAN_RESOURCES + "ReportsRepository",
    applyAnnotations = "@org.springframework.stereotype.Repository",
    superInterfaces = PagingAndSortingRepository.class,
    firstSuperInterfaceGenerics = {Employee.class, Long.class}
)
@Detach(profile = SECURITY,
    rename = SECURITY + "ReportsRepository",
    applyAnnotations = "@org.springframework.stereotype.Repository",
    superInterfaces = CrudRepository.class,
    firstSuperInterfaceGenerics = {Employee.class, Long.class}
)
@Detach(profile = MANAGEMENT,
    rename = MANAGEMENT + "ReportsRepository",
    applyAnnotations = "@org.springframework.stereotype.Repository",
    superInterfaces = JpaRepository.class,
    firstSuperInterfaceGenerics = {Employee.class, Long.class}
)
public interface ReportsRepository {

    2 usages
    String TOPLEVEL_KW = "(topLevel)"; // keyword used to tell jisel to detach the

    5 usages
    String HUMAN_RESOURCES = "HumanResources";
    5 usages
    String SECURITY = "Security";
    6 usages
    String MANAGEMENT = "Management";

}
```

```
@TopLevel
List<Employee> findByFirstNameContainingOrLastNameContaining(String firstNameSearchCriteria, String lastNameSearchCriteria);

 Mohamed Ashraf Bayor
@TopLevel
List<Employee> findByEmailAddressContaining(String emailAddressSearchCriteria);

 Mohamed Ashraf Bayor
@SealFor(HUMAN_RESOURCES)
@SealFor(MANAGEMENT)
@Query(
    nativeQuery = true,
    value = """
        SELECT home_address, COUNT(id) total
        FROM employee
        GROUP BY home_address
        HAVING total > 1
        ORDER BY total DESC
        """
)
List<Object[]> countEmployeesSharingSameHomeAddress();

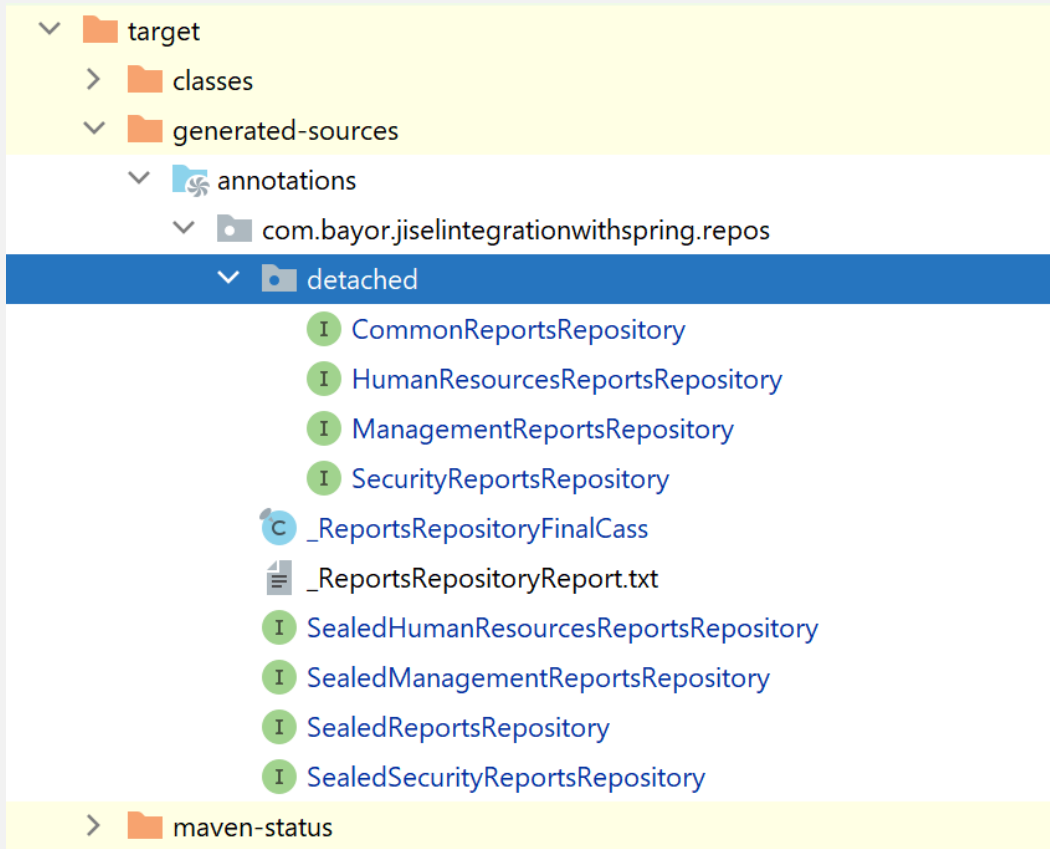
 Mohamed Ashraf Bayor
@SealFor(HUMAN_RESOURCES)
@SealFor(MANAGEMENT)
@Query(
    nativeQuery = true,
    value = """
        SELECT department, SUM(accrued_vacation_days) total
        FROM employee
        GROUP BY department
        ORDER BY total desc
        """
)
List<Object[]> totalAccruedVacationsPerDepartment();
```

```
@SealFor(SECURITY)
@Query(
    nativeQuery = true,
    value = """
        SELECT e1.*
        FROM employee e1
        INNER JOIN employee e2
        ON (e1.department <> e2.department
        AND e1.first_name = e2.first_name
        AND e1.last_name = e2.last_name)
        ORDER BY e1.first_name, e1.last_name
        """
)
List<Employee> employeesWithFirstNameAndLastNameListedInMoreThan1Department();

 Mohamed Ashraf Bayor
@SealFor(SECURITY)
@Query(
    nativeQuery = true,
    value = """
        SELECT distinct e1.*
        FROM employee e1
        INNER JOIN employee e2
        ON (e1.email_address = e2.email_address
        AND e1.ID <> e2.ID)
        ORDER BY email_address
        """
)
List<Employee> employeesWithDifferentIDsUsingSameEmailAddresses();

 Mohamed Ashraf Bayor
@SealFor(MANAGEMENT)
@Query(
    nativeQuery = true,
    value = """
        SELECT first_name, last_name, FORMATDATETIME(start_date, 'yyyy-MM-dd'), annual_salary, DATEDIFF(year, start_date, NOW()) totalYears
        FROM employee
        WHERE DATEDIFF(year, start_date, NOW()) >= :totalYears
        AND annual_salary < :annualSalary
        ORDER BY totalYears DESC
        """
)
List<Object[]> employeesWithMoreThanXYearsAndSalaryLessThanY(@Param("totalYears") int totalYearsCriteria, @Param("annualSalary") long annualSalaryCriteria);
```

# Compile and Check Generated Sources



[CommonReportsRepository.java](#)

[HumanResourcesReportsRepository.java](#)

[SecurityReportsRepository.java](#)

[ManagementReportsRepository.java](#)



# Use the Generated Sources in your Code

## JiselIntegrationWithSpringApplication.java

```
@SpringBootApplication
public class JiselIntegrationWithSpringApplication implements CommandLineRunner {

    1 usage
    Logger log = Logger.getLogger(JiselIntegrationWithSpringApplication.class.getName());

    Mohamed Ashraf Bayor
    public static void main(String[] args) {
        SpringApplication.run(JiselIntegrationWithSpringApplication.class, args);
    }

    Mohamed Ashraf Bayor
    @Override
    public void run(String... args) {
        displayReports();
    }

    2 usages
    @Autowired
    CommonReportsRepository commonReportsRepository;

    2 usages
    @Autowired
    HumanResourcesReportsRepository humanResourcesReportsRepository;

    2 usages
    @Autowired
    SecurityReportsRepository securityReportsRepository;

    1 usage
    @Autowired
    ManagementReportsRepository managementReportsRepository;
```

1 usage Mohamed Ashraf Bayor

```
void displayReports() {
    log.info(
```

```
        format(
```

```
            ""
```

```
            ===== COMMON REPORTS =====
```

```
            > Find by FirstName (containing "o") or LastName (containing "il":
            %s
```

```
            > Find by E-mail Address (containing "email"):
            %s
```

```
// ===== COMMON REPORTS =====

commonReportsRepository.findByFirstNameContainingOrLastNameContaining("o", "il").stream() Stream<Employee>
    .map(employee -> format("%s %s", employee.getFirstName(), employee.getLastName())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n")),

commonReportsRepository.findByEmailAddressContaining( emailAddressSearchCriteria: "email").stream() Stream<Employee>
    .map(employee -> format("%s %s - %s", employee.getFirstName(), employee.getLastName(), employee.getEmailAddress())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n")),

// ===== HR REPORTS =====

humanResourcesReportsRepository.countEmployeesSharingSameHomeAddress().stream() Stream<Object[]>
    .map(row -> format("%s %s", row[0].toString(), row[1].toString())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n")),

humanResourcesReportsRepository.totalAccruedVacationsPerDepartment().stream() Stream<Object[]>
    .map(row -> format("%s %s", row[0].toString(), row[1].toString())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n")),

// ===== SECURITY REPORTS =====

securityReportsRepository.employeesWithFirstNameAndLastNameListedInMoreThan10Department().stream() Stream<Employee>
    .map(employee -> format("%s %s (%s), %s - %s, since %s",
        employee.getFirstName(),
        employee.getLastName(),
        employee.getEmailAddress(),
        employee.getPosition(),
        employee.getDepartment(),
        employee.getStartDate())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n")),

securityReportsRepository.employeesWithDifferentIDsUsingSameEmailAddresses().stream() Stream<Employee>
    .map(employee -> format("%s, %s %s (ID#%s) %s - %s",
        employee.getEmailAddress(),
        employee.getFirstName(),
        employee.getLastName(),
        employee.getId(),
        employee.getPosition(),
        employee.getDepartment())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n")),

// ===== MANAGEMENT REPORTS =====

managementReportsRepository.employeesWithMoreThanXYearsAndSalaryLessThanY( totalYearsCriteria: 5, annualSalaryCriteria: 50000).stream() Stream<Object[]>
    .map(row -> format("%s %s, since %s (%s years), salary: %s",
        row[0].toString(), row[1].toString(), row[2].toString(), row[4].toString(), row[3].toString())) Stream<String>
    .collect(Collectors.joining( delimiter: "\n"))
```

# Run and Check Results

## ===== COMMON REPORTS =====

> Find by FirstName (containing "o") or LastName (containing "il"):

John Doe  
Tony Stark  
Katie Hill  
John Doe  
Jennifer Miles

> Find by E-mail Address (containing "email"):

Tony Stark - email@domain.com  
Tim Shawn - email@domain.com

## ===== HR REPORTS =====

> Count Employees sharing Same Home Address:

homeAddress4 3  
homeAddress1 2

> Total Accrued Vacation Days per Department:

IT 17  
Human Resources 10  
Recruitment 7  
Security 7  
Sales 4

## ===== SECURITY REPORTS =====

> Employees with First Name and Last Name listed in More Than 1 Department:

John Doe (john.doe@domain.com), Intern - Human Resources, since 2001-01-01 00:00:00.0  
John Doe (john.doe@domain.com), Recruiter - Recruitment, since 2007-07-07 00:00:00.0

> Employees with Different IDs using Same E-mail Addresses:

email@domain.com, Tony Stark (ID#2) Jr Associate - Security  
email@domain.com, Tim Shawn (ID#3) System Admin - IT  
john.doe@domain.com, John Doe (ID#1) Intern - Human Resources  
john.doe@domain.com, John Doe (ID#7) Recruiter - Recruitment  
john.doe@domain.com, Jennifer Miles (ID#9) Sr Consultant - Human Resources

## ===== MANAGEMENT REPORTS =====

> Employees who Started More than 5 years ago AND with a current Salary Less than 50000:

John Doe, since 2001-01-01 (21 years), salary: 10000  
Tony Stark, since 2002-02-02 (20 years), salary: 20000  
Tim Shawn, since 2003-03-03 (19 years), salary: 30000  
Katie Hill, since 2004-04-04 (18 years), salary: 40000

# Integrating JISEL with Spring Data JPA

---

<https://github.com/mohamed-ashraf-bayor/jisel-integration-with-spring>