



Spring Boot. Spring Data. ORM

Филиппов Евгений Евгеньевич
Java-developer

04.04.2017

О чем пойдет речь

- SQL, РСУБД
- JDBC
- ORM, JPA
- Spring Data JPA



РУСБД



Реляционная СУБД (или РСУБД) - система управления реляционными БД. В реляционных базах данные хранятся в виде таблиц, состоящих из строк и столбцов

attributes

column

SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

tuple

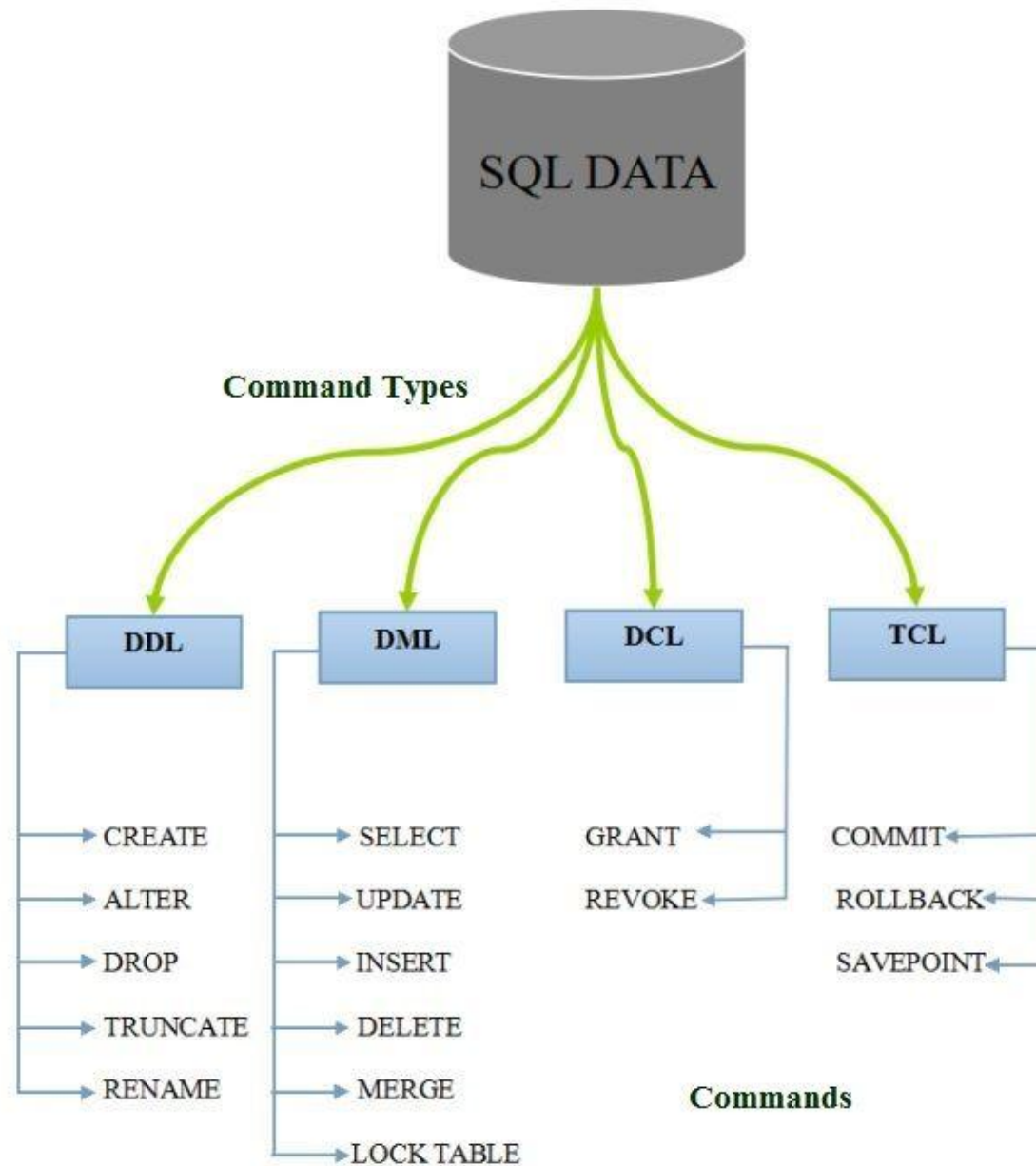
table (relation)

SQL



SQL (*structured query language*) — формальный не процедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных

SQL

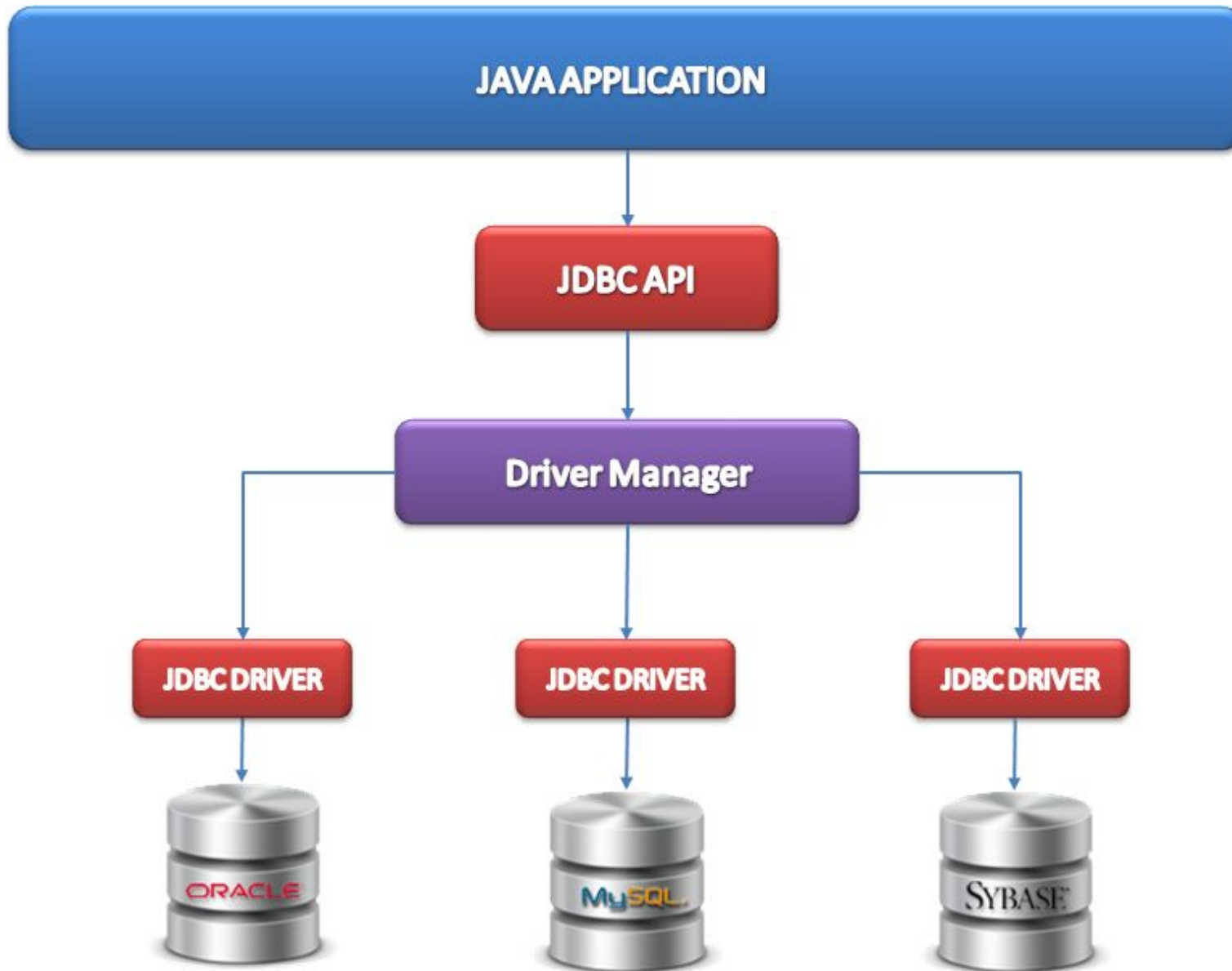


JDBC

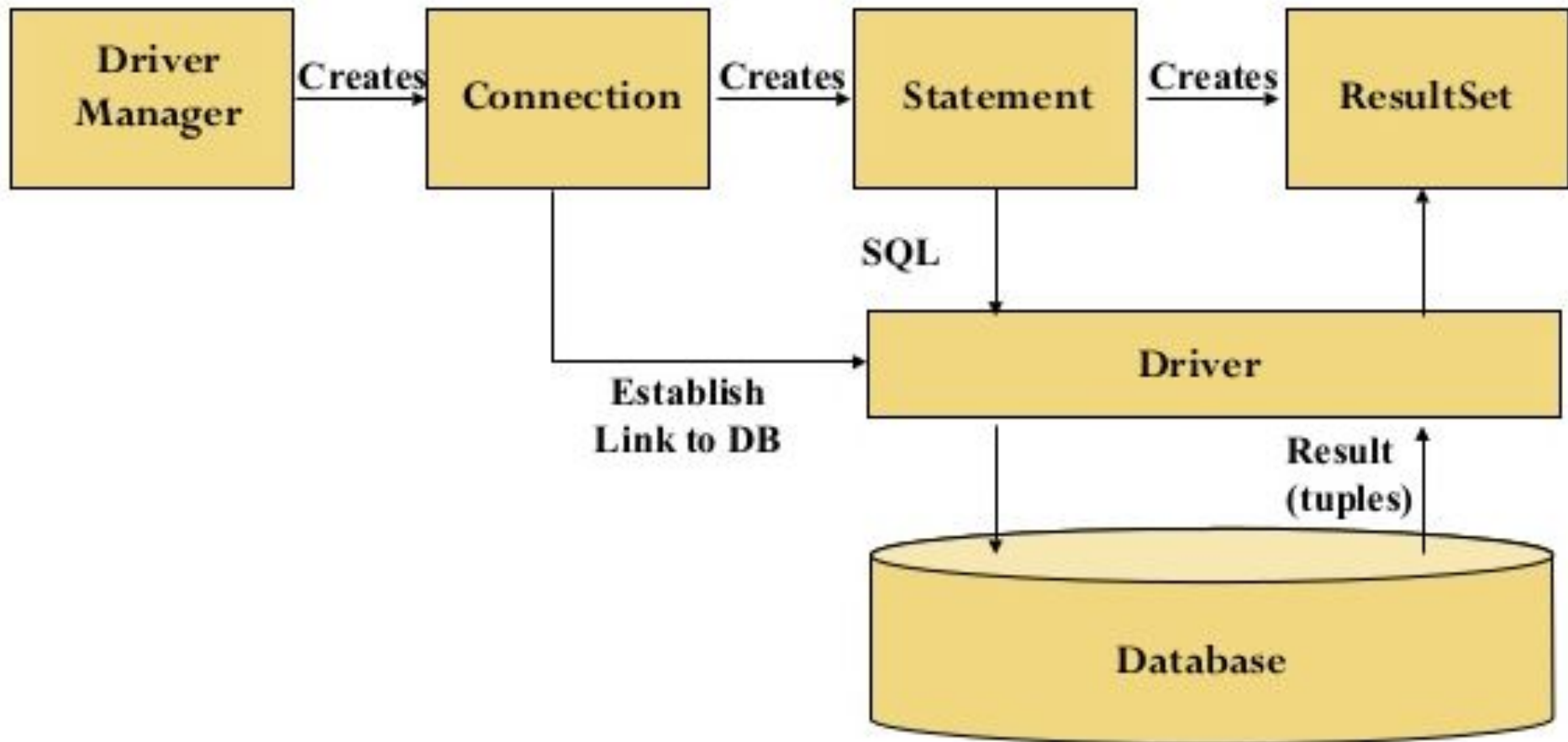


JDBC(Java DataBase Connectivity) - платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета `java.sql`, входящего в состав Java SE

Зачем нужно



JDBC Flow



Example



```
1 package com.simbirsoft.model.domain;
2
3 public class Student {
4     private Long id;
5     private String firstName;
6     private String lastName;
7     private Integer age;
8
9     @Override
10    public String toString() {
11        final StringBuilder sb = new StringBuilder("Student{");
12        sb.append("id=").append(id);
13        sb.append(", firstName=").append(firstName).append('\n');
14        sb.append(", lastName=").append(lastName).append('\n');
15        sb.append(", age=").append(age);
16        sb.append('}');
17        return sb.toString();
18    }
19
20    // ... Геттеры и сеттеры опущены
```

SELECT * FROM STUDENT ;

ID	FIRST_NAME	LAST_NAME	AGE
1	Вася	Пупкин	17
2	Петя	Дудкин	18
3	Вася	Васильев	18

Example

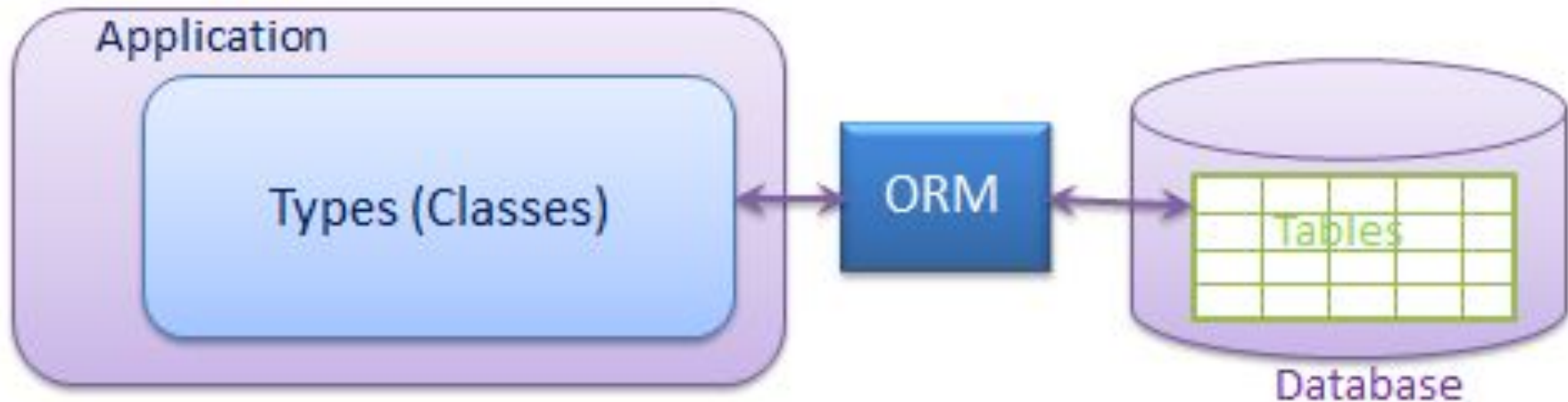


```
1 package com.simbirsoft;
2
3 import ...
4
12
13 public class JdbcApp {
14     private static final String URL =
15         "jdbc:h2:mem:test;INIT=RUNSCRIPT FROM 'classpath:init.sql'\\\\";
16
17     public static void main(String[] args) throws ClassNotFoundException {
18         Class.forName("org.h2.Driver");
19
20         try (Connection connection = DriverManager.getConnection(URL, "sa", "")) {
21             PreparedStatement preparedStatement =
22                 connection.prepareStatement("SELECT * FROM STUDENT WHERE FIRST_NAME=?");
23             preparedStatement.setString(1, "Вася");
24
25             ResultSet resultSet = preparedStatement.executeQuery();
26
27             List<Student> students = new ArrayList<>();
28             while (resultSet.next()) {
29                 Student student = new Student();
30                 student.setId(resultSet.getLong("ID"));
31                 student.setFirstName(resultSet.getString("FIRST_NAME"));
32                 student.setLastName(resultSet.getString("LAST_NAME"));
33                 student.setAge(resultSet.getInt("AGE"));
34                 students.add(student);
35             }
36
37             System.out.println(students);
38         } catch (SQLException e) {
39             throw new RuntimeException(e);
40         }
41     }
42 }
```

ORM



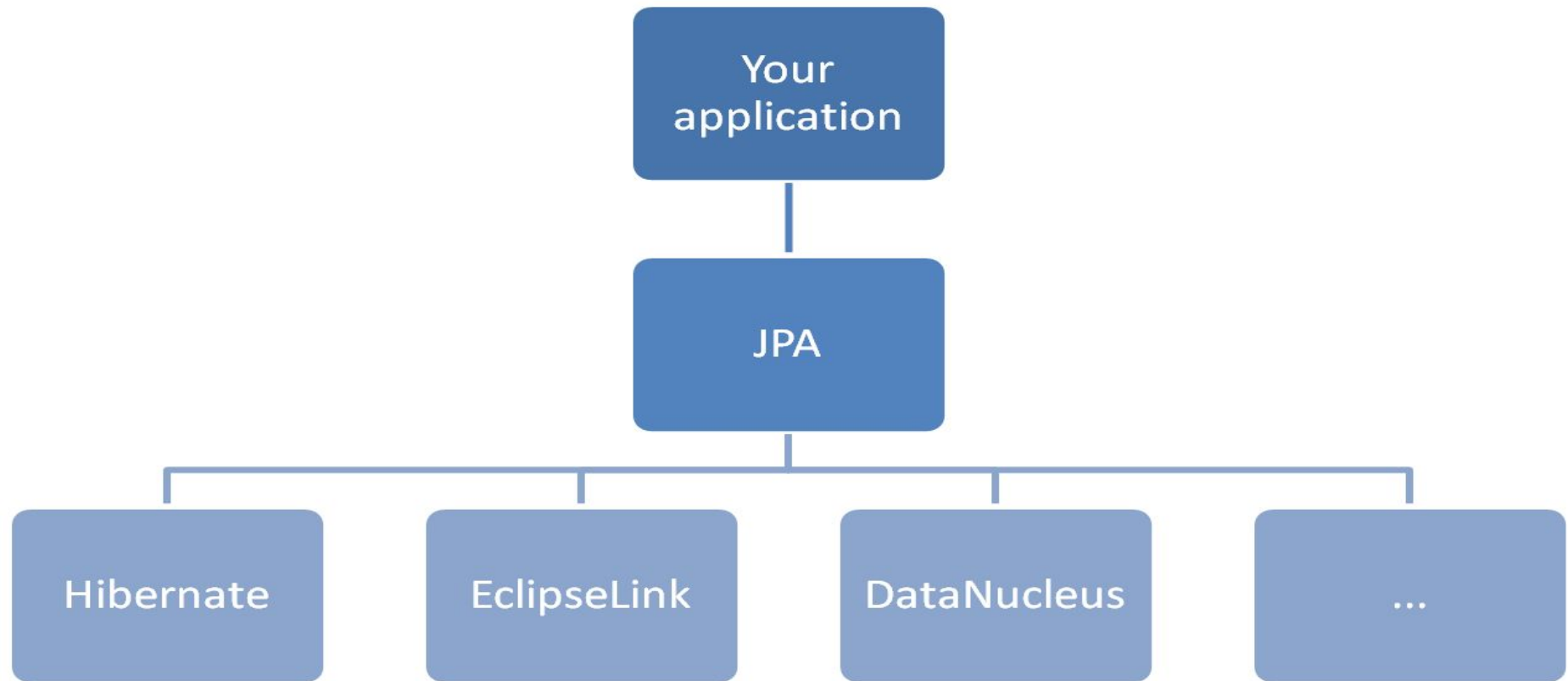
ORM(Object-Relational Mapping, рус. объектно-реляционное отображение) - технология программирования, которая позволяет обеспечить работу с данными в терминах классов, а не таблиц данных и напротив, преобразовать термины и данные классов в данные, пригодные для хранения в СУБД



JPA



Java Persistence API (JPA) — спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных. Существует несколько реализаций этого интерфейса, одна из самых популярных использует для этого Hibernate. JPA реализует концепцию ORM.



Entity



```
1 package com.simbirsoft.model.domain;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.Id;
6
7 @Entity
8 public class Student {
9     @Id
10    @GeneratedValue
11    private Long id;
12    private String firstName;
13    private String lastName;
14    private Integer age;
15
16    public Long getId() {
17        return id;
18    }
19
20    public void setId(Long id) {
21        this.id = id;
22    }
23
24    //Остальные сеттеры и геттеры опущены для наглядности
25 }
```

SELECT * FROM STUDENT;

ID	AGE	FIRST_NAME	LAST_NAME
1	17	Вася	Пупкин
2	18	Петя	Дудкин

Основные аннотации



- @Entity(name)
- @Table(name, schema, uniqueConstraints, indexes, catalog)
- @Column(columnDefinition, insertable, length, name, nullable, precision, scale, table, unique, updatable)
- @Id
- @GeneratedValue(generator, strategy)
- @Transient
- @Temporal(TemporalType)
- @Enumerated(EnumType)

Основные аннотации



```
@Entity
@Table(name = "STUDENTS", uniqueConstraints = {@UniqueConstraint(columnNames = {"firstName"})})
public class Student {
    @Id
    @GeneratedValue
    private Long id;

    @Column(insertable = false, columnDefinition = "VARCHAR(100)")
    private String firstName;

    @Column(name = "STUDENT_LAST_NAME", length = 100, unique = true, nullable = false)
    private String lastName;

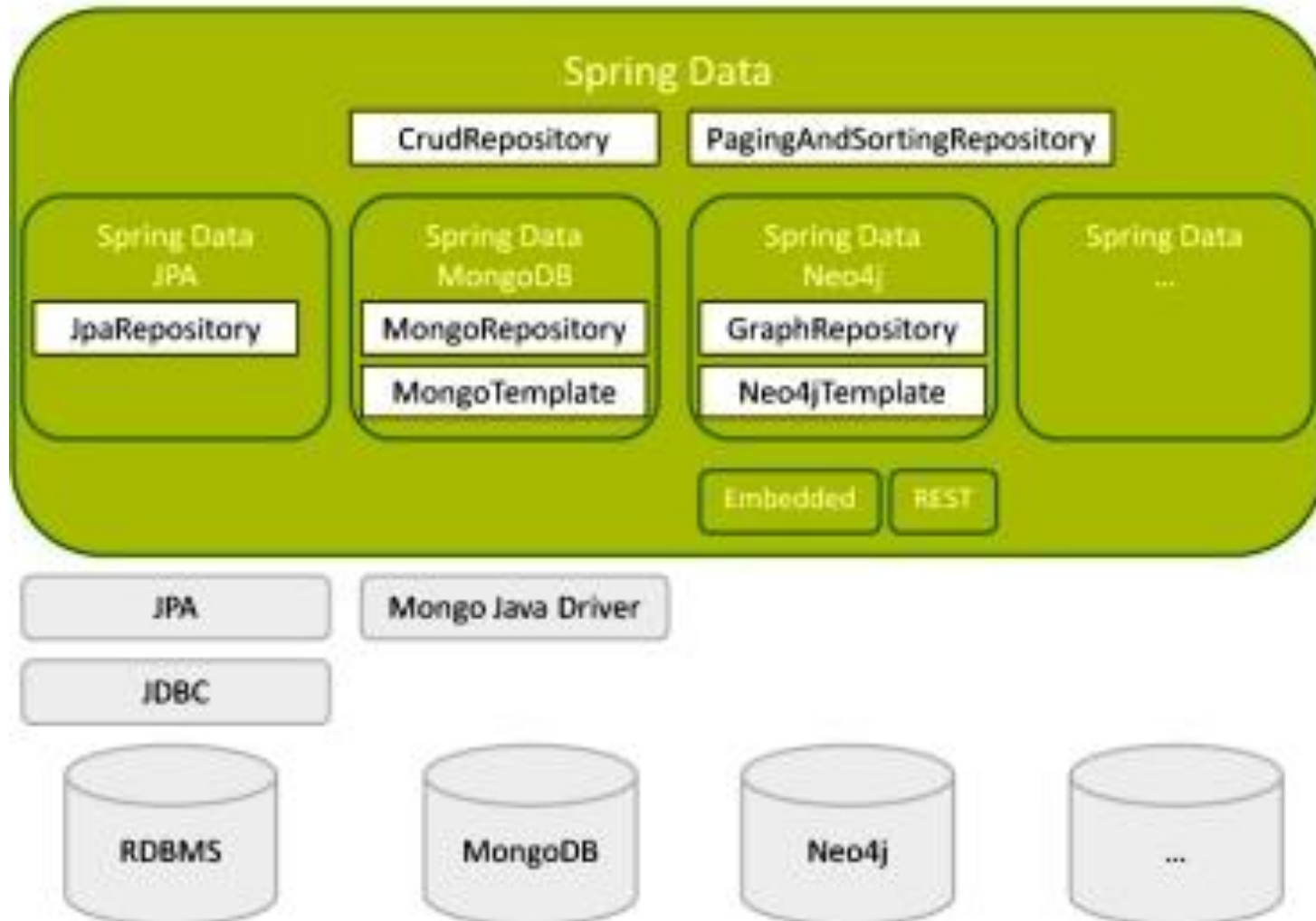
    @Transient
    private Integer age;

    @Temporal(TemporalType.DATE)
    private Date birthday;

    @Enumerated(EnumType.STRING)
    private Gender gender;
}

public Student() {
    this.age = DateUtil.calculateAge(this.birthday);
}
```

Spring Data

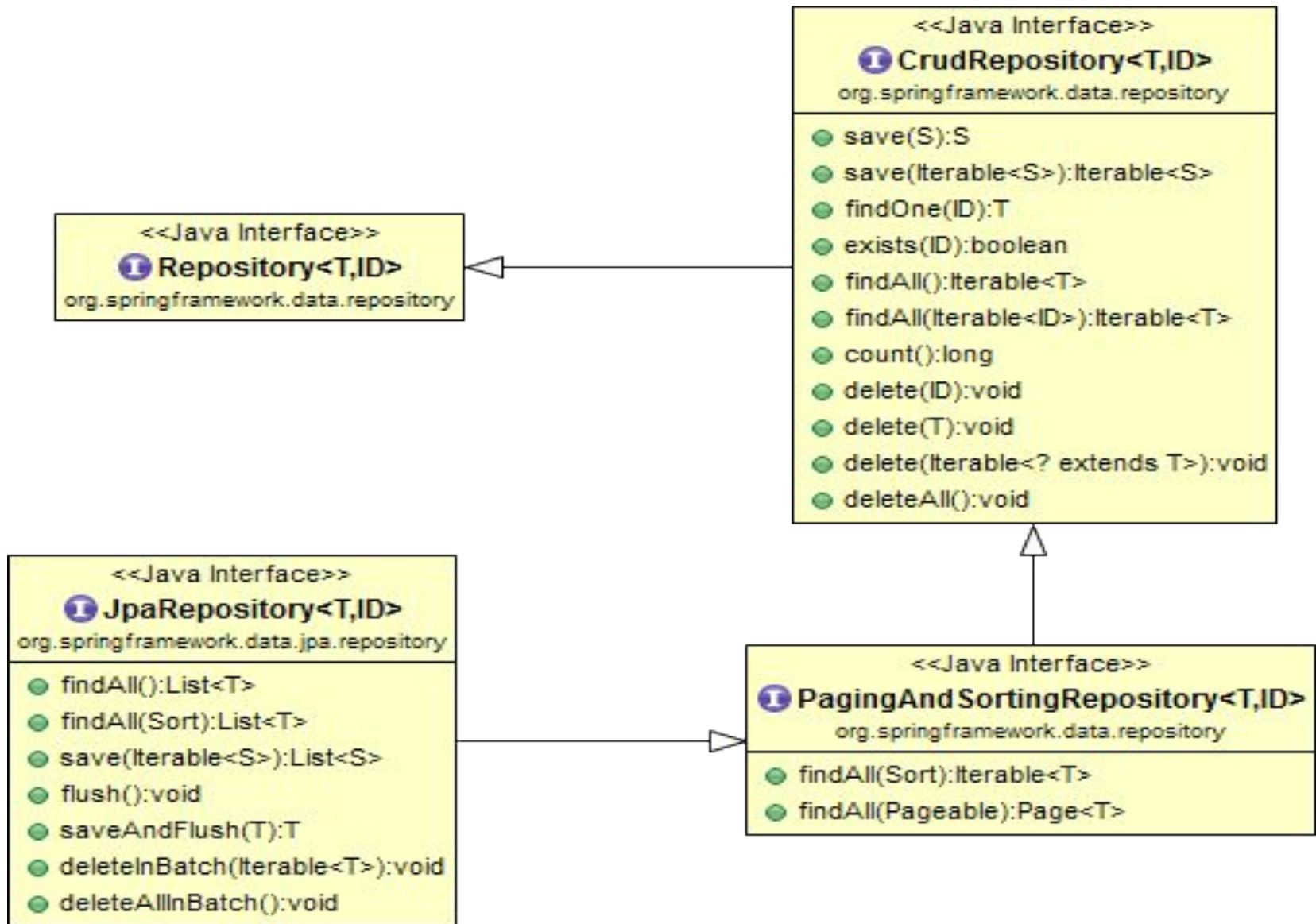


Spring Data JPA



Spring Data JPA - часть проекта Spring Data, которая упрощает реализацию классов доступа данных (Repositories) основанных на технологии JPA

Spring Data JPA



Example



```
1 package com.simbirsoft.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import com.simbirsoft.model.domain.Student;
5
6 public interface StudentRepository extends JpaRepository<Student, Long> {
7 }
```

```
1 package com.simbirsoft.model.domain;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.Id;
6
7 @Entity
8 public class Student {
9     @Id
10    @GeneratedValue
11    private Long id;
12    private String firstName;
13    private String lastName;
14    private Integer age;
15
16    public Long getId() {
17        return id;
18    }
19
20    public void setId(Long id) {
21        this.id = id;
22    }
23
24    //Остальные сеттеры и геттеры опущены для наглядности
25 }
```

SELECT * FROM STUDENT;

ID	AGE	FIRST_NAME	LAST_NAME
1	17	Вася	Пупкин
2	18	Петя	Дудкин

Example

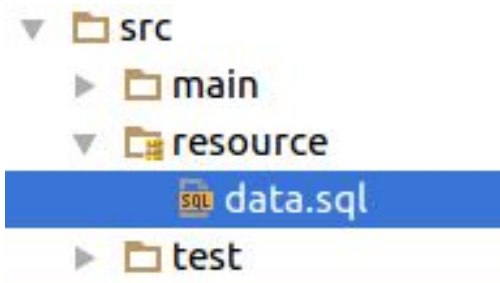


```
1 package com.simbirsoft;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Component;
7
8 import com.simbirsoft.model.domain.Student;
9 import com.simbirsoft.repository.StudentRepository;
10
11 @Component
12 public class SomeBean {
13
14     @Autowired
15     private StudentRepository repository;
16
17     public void doSmt() {
18         Student studentKolya = new Student();
19         studentKolya.setFirstName("Коля");
20         studentKolya.setLastName("Уткин");
21         studentKolya.setAge(20);
22         repository.save(studentKolya);
23         assert Long.valueOf(3L).equals(studentKolya.getId());
24
25         Student studentVasya = repository.getOne(1L);
26         assert "Вася".equals(studentVasya.getFirstName());
27
28         List<Student> students = repository.findAll();
29         assert students.size() == 3;
30
31         repository.deleteAll();
32         assert repository.findAll().size() == 0;
33     }
34 }
```


Example



```
1 package com.simbirsoft;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.ConfigurableApplicationContext;
6
7
8 @SpringBootApplication
9 public class SpringBootTestApp {
10
11     public static void main(String[] args) {
12         ConfigurableApplicationContext context = SpringApplication.run(SpringBootTestApp.class, args);
13         context.getBean(SomeBean.class).doSmt();
14     }
15 }
```



```
1 INSERT INTO STUDENT VALUES
2     (NULL, 17, 'Вася', 'Пупкин'),
3     (NULL, 18, 'Петя', 'Дудкин');
```

Query Creation



```
1  package com.simbirsoft.repository;
2
3  import java.util.List;
4
5  import org.springframework.data.jpa.repository.JpaRepository;
6  import com.simbirsoft.model.domain.Student;
7
8  public interface StudentRepository extends JpaRepository<Student, Long> {
9      List<Student> findByFirstNameIgnoreCase(String firstName);
10
11     List<Student> findByLastNameAndAge(String lastName, Integer age);
12
13     List<Student> findByAgeOrderByLastNameDesc(Integer age);
14
15     List<Student> findByLastNameLike(String firstName);
16
17     List<Student> findByAgeBetween(Integer from, Integer to);
18 }
```

Example



```
List<Student> studentsByAgeBetween = repository.findByAgeBetween(17, 20);  
out.println(studentsByAgeBetween.size()); //1 ?
```

```
List<Student> studentsByLastNameLike = repository.findByLastNameLike("%ин");  
out.println(studentsByLastNameLike.size()); //2 ?
```

```
List<Student> studentsByAgeOrderByLastNameDesc = repository.findByAgeOrderByLastNameDesc(18);  
out.println(studentsByAgeOrderByLastNameDesc.get(0).getFirstName()); //3 ?
```

```
List<Student> studentsByFirstNameIgnoreCase = repository.findByFirstNameIgnoreCase("вася");  
out.println(studentsByFirstNameIgnoreCase.size()); //4 ?
```

```
List<Student> studentsByLastNameAndAge = repository.findByLastNameAndAge("Пупкин", 17);  
out.println(studentsByLastNameAndAge.get(0).getFirstName()); //5 ?
```

SELECT * FROM STUDENT;

ID	AGE	FIRST_NAME	LAST_NAME
1	17	Вася	Пупкин
2	18	Петя	Дудкин
3	21	Коля	Уткин
4	18	Маша	Пирожкова
5	16	вася	Мухин

Example



```
List<Student> studentsByAgeBetween = repository.findByAgeBetween(17, 20);  
out.println(studentsByAgeBetween.size()); //1 - 3
```

```
List<Student> studentsByLastNameLike = repository.findByLastNameLike("%ин");  
out.println(studentsByLastNameLike.size()); //2 - 4
```

```
List<Student> studentsByAgeOrderByLastNameDesc = repository.findByAgeOrderByLastNameDesc(18);  
out.println(studentsByAgeOrderByLastNameDesc.get(0).getFirstName()); //3 - Маша
```

```
List<Student> studentsByFirstNameIgnoreCase = repository.findByFirstNameIgnoreCase("вася");  
out.println(studentsByFirstNameIgnoreCase.size()); //4 - 2
```

```
List<Student> studentsByLastNameAndAge = repository.findByLastNameAndAge("Пупкин", 17);  
out.println(studentsByLastNameAndAge.get(0).getFirstName()); //5 - Вася
```

SELECT * FROM STUDENT;

ID	AGE	FIRST_NAME	LAST_NAME
1	17	Вася	Пупкин
2	18	Петя	Дудкин
3	21	Коля	Уткин
4	18	Маша	Пирожкова
5	16	вася	Мухин

Отношения



- @ManyToOne(fetch, cascade, optional, targetEntity, mappedBy)
- @OneToMany(fetch, cascade, targetEntity, orphanRemoval, mappedBy)
- @OneToOne(fetch, cascade, optional, targetEntity, orphanRemoval, mappedBy)
- @ManyToMany(fetch, cascade, targetEntity, mappedBy)
- @JoinColumn(name, foreignKey, referencedColumnName, ..(@Column)
- @JoinTable(name, joinColumns, foreignKey, inverseJoinColumns, inverseForeignKey)

Example



```
1 package com.simbirsoft.model.domain;
2
3 import ...
4
5
6
7
8
9 @Entity
10 public class Student {
11     @Id
12     @GeneratedValue
13     private Long id;
14     private String firstName;
15     private String lastName;
16     private Integer age;
17
18     @ManyToOne
19     @JoinColumn(name = "GROUP_ID")
20     private Group group;
21
22     // ... Геттеры и сеттеры опущены
```

```
1 package com.simbirsoft.model.domain;
2
3 import ...
4
5
6
7
8
9
10
11
12 @Entity
13 @Table(name = "GROUPS")
14 public class Group {
15     @Id
16     @GeneratedValue
17     private Long id;
18
19     private String name;
20
21     @OneToMany(fetch = FetchType.LAZY,
22               mappedBy = "group")
23     private List<Student> students;
24
25     // ... Геттеры и сеттеры опущены
```

SELECT * FROM STUDENT;

ID	AGE	FIRST_NAME	LAST_NAME	GROUP_ID
1	17	Вася	Пупкин	1
2	18	Петя	Дудкин	1
3	21	Коля	Уткин	2
4	18	Маша	Пирожкова	1
5	16	вася	Мухин	2

SELECT * FROM GROUPS;

ID	NAME
1	РТД-1
2	РТД-2

Example



```
1 package com.simbirsoft.repository;
2
3 import ...
4
5
6
7 public interface GroupRepository extends JpaRepository<Group, Long> {
8 }
```

```
1 package com.simbirsoft;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 @Component
19 public class SomeBean {
20     @Autowired
21     private GroupRepository groupRepository;
22
23     public void doSmtth() {
24
25         List<Group> groups = groupRepository.findAll();
26
27         for (Group group : groups) {
28             System.out.println(group.getStudents());
29         }
30     }
31 }
```

Запросы



- `/resource/application.properties - spring.jpa.show-sql=true`
- Hibernate: `select group0_.id as id1_0_, group0_.name as name2_0_ from groups group0_`
- `select students0_.group_id as group_id5_1_0_, students0_.id as id1_1_0_, students0_.id as id1_1_1_, students0_.age as age2_1_1_, students0_.first_name as first_na3_1_1_, students0_.group_id as group_id5_1_1_, students0_.last_name as last_nam4_1_1_ from student students0_ where students0_.group_id=?`

Практическое задание

1. Развернуть Spring Boot проект
2. Создать сущность(Entity) Person с полями name, age
3. Создать PersonRepository (Заимплементить интерфейс спринга)
4. Создать контроллер PersonController (использовать @RestController вместо просто @Controller)
5. По url-у /persons/ и http методу POST (RequestMapping) создать метод save() принимающий name и age в виде параметров (RequestParam)
6. В контроллер заинжектить PersonRepository и в методе контроллера save() создать и сохранить сущность(Person) с помощью репозитория
7. Зайти в <http://localhost:8080/h2-console> и проверить что все сохранилось
8. Заполнить таблицу 3-4 записями с помощью resources/data.sql
9. В репозитории и контроллере создать 2-3 метода поиска по параметрам. Методы должны возвращать список найденных записей (и в контроллере и в репозитории)
10. Как создавать методы поиска можно посмотреть здесь - <http://docs.spring.io/spring-data/jpa/docs/1.7.0.M1/reference/htmlsingle/#jpa.query-methods.query-creation>
11. Для тех кто чувствует себя уверенно - добавить отношение @ManyToOne. Например у каждого Person будет список Phone. Phone так же является Entity и содержит поля id, number. Данные заполнить так же с помощью data.sql
12. Добавить метод в контроллере который будет принимать id Person'а как параметр и выводить список Phone

Спасибо за внимание!

Филиппов Евгений Евгеньевич
Java-developer

+7 (8422) 44-66-91
+7 (495) 133-90-01
www.simbirsoft.com