Spring Boot. Spring Data. ORM

Филиппов Евгений Евгеньевич
Java-developer

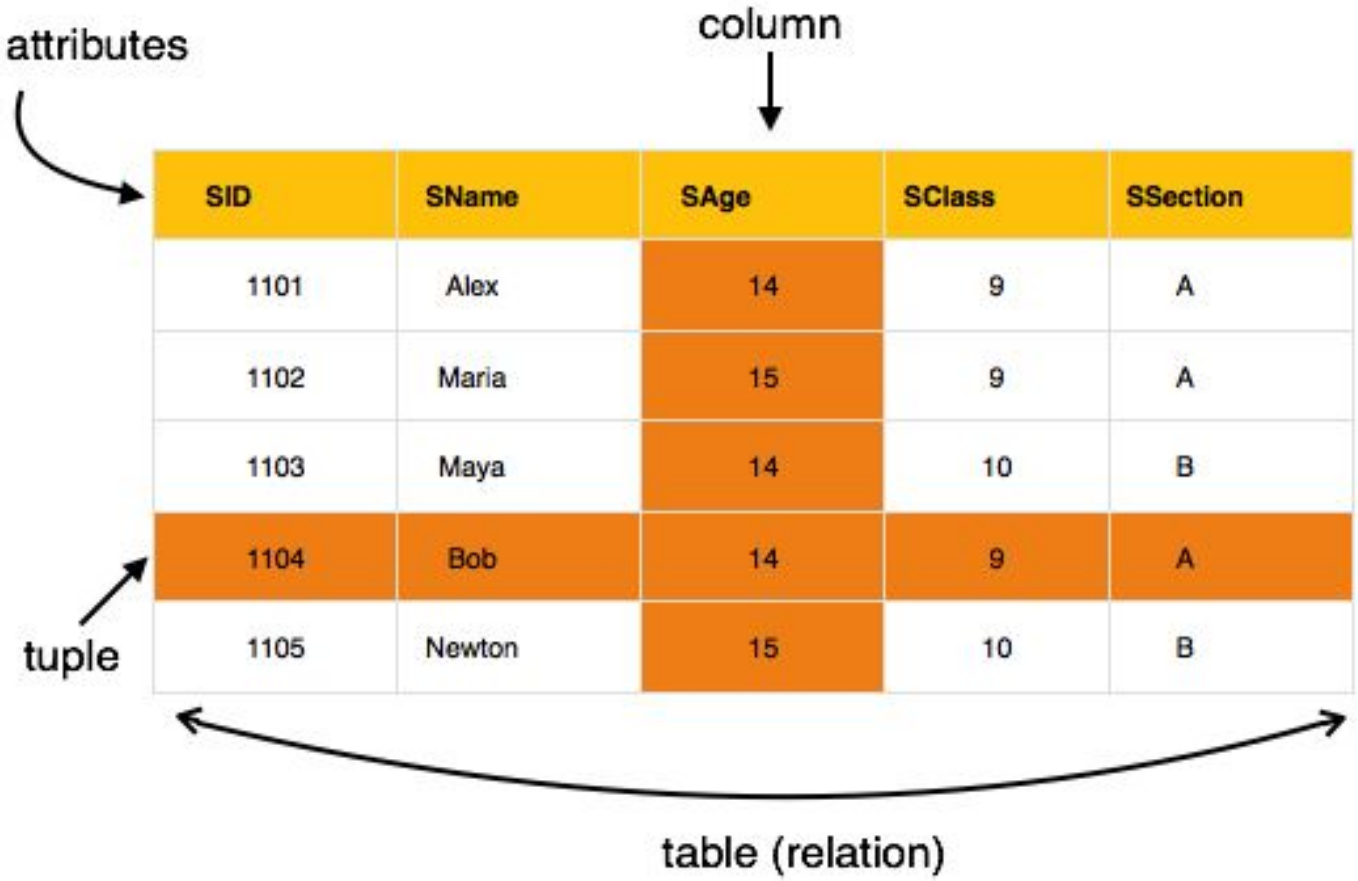
04.04.2017

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

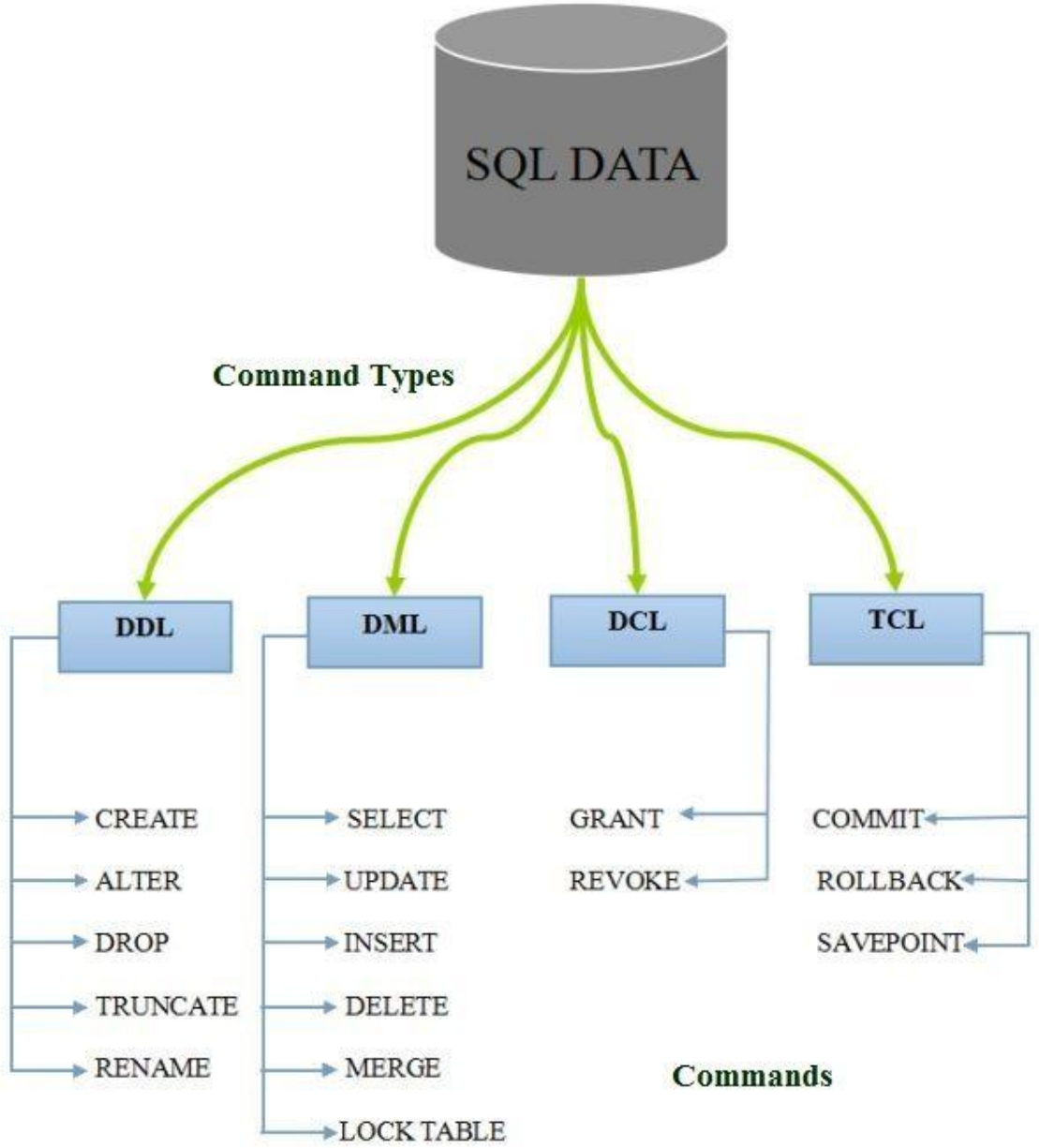


SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

SQL

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

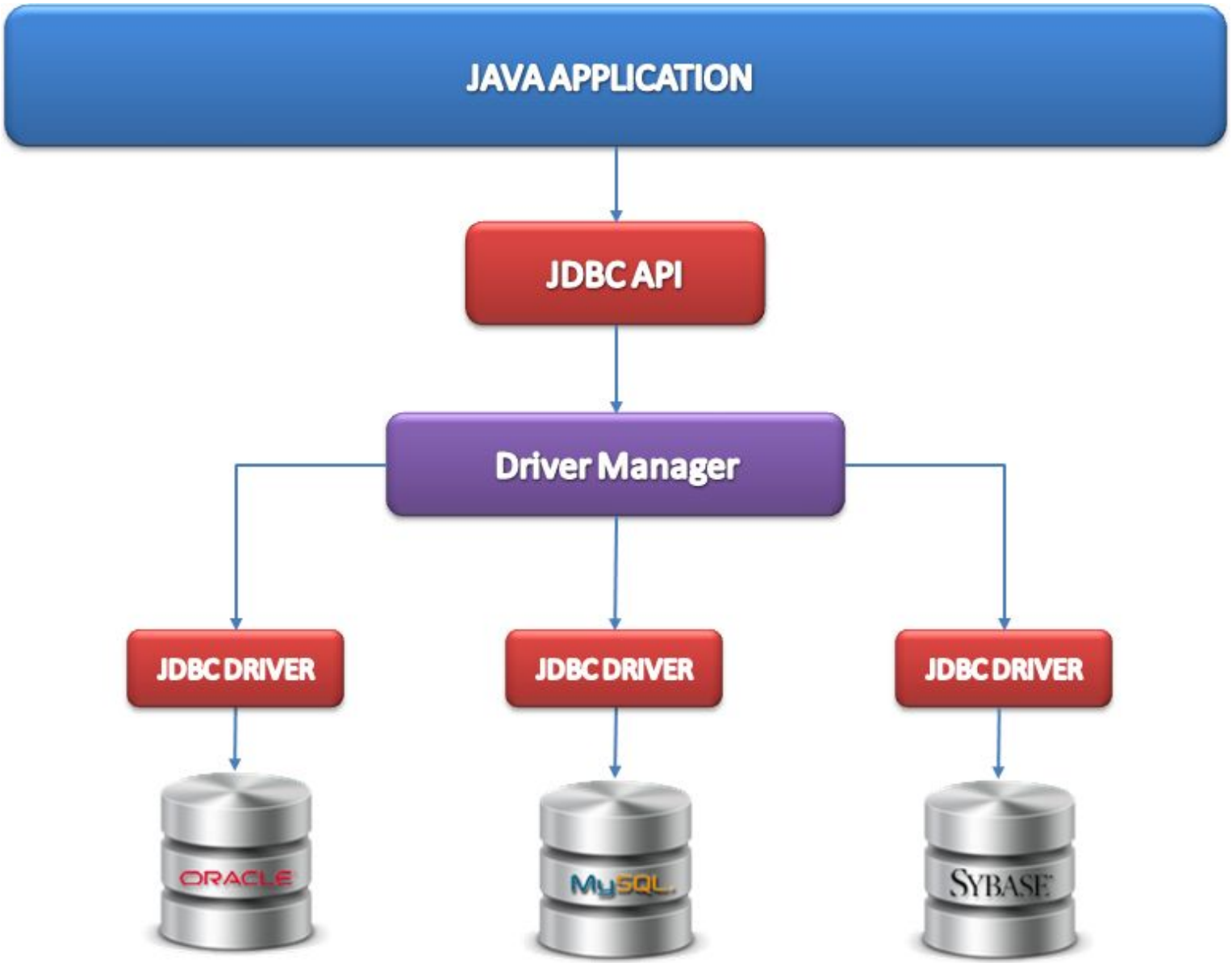
SQL


SQL DATA
Command Types
DDL
DML
DCL
TCL
CREATE
ALTER
DROP
TRUNCATE
RENAME
SELECT
UPDATE
INSERT
DELETE
MERGE
LOCK TABLE
GRANT
REVOKE
COMMIT
ROLLBACK
SAVEPOINT
Commands

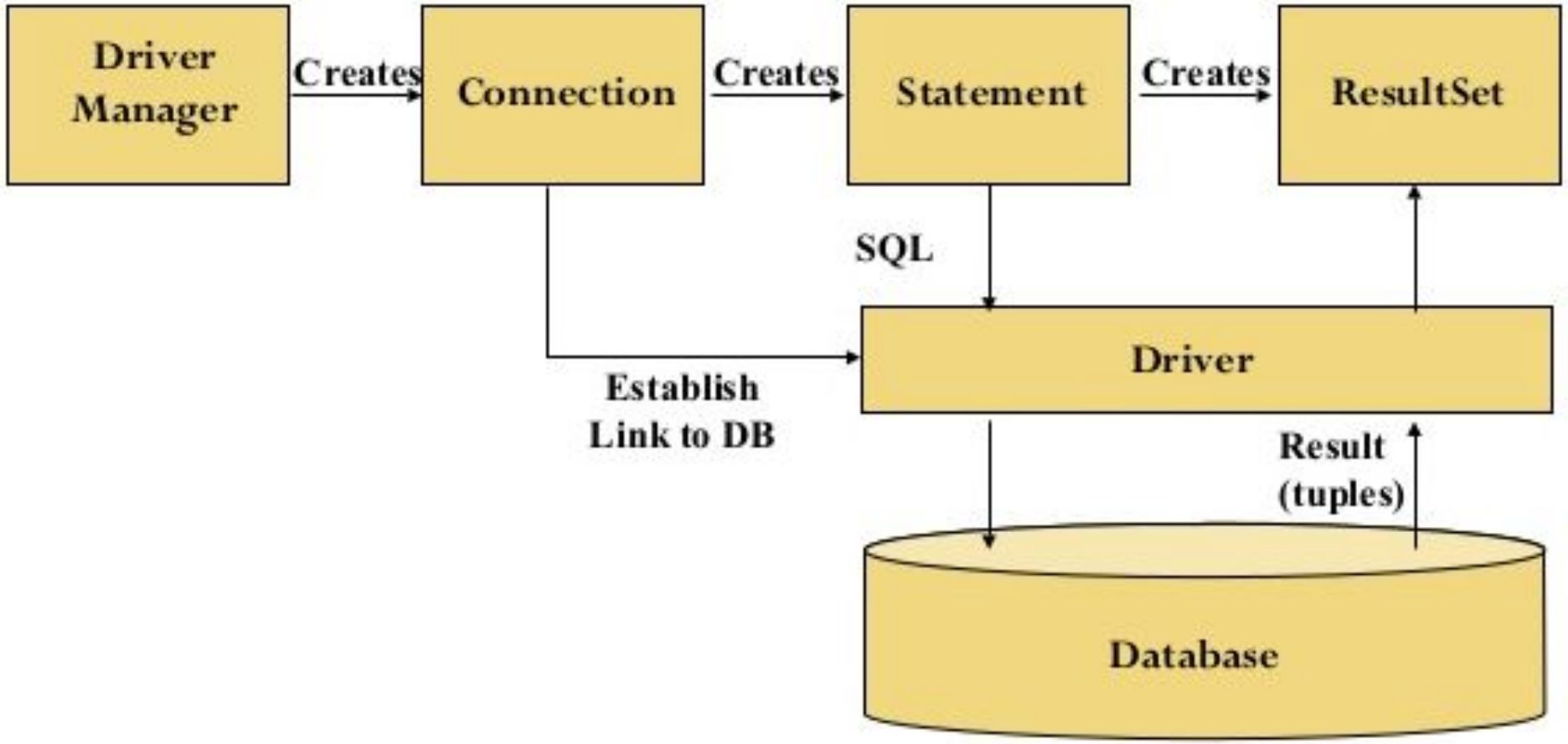
JDBC

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


JAVA APPLICATION
JDBC API
Driver Manager
JDBC DRIVER
JDBC DRIVER
JDBC DRIVER
ORACLE
MySQL
SYBASE

JDBC Flow


Driver Manager
Creates
Connection
Creates
Statement
Creates
ResultSet
SQL
Establish Link to DB
Driver
Result (tuples)
Database

Example

```java
package com.simbirsoft.model.domain;

public class Student {
    private Long id;
    private String firstName;
    private String lastName;
    private Integer age;

    @Override
    public String toString() {
        final StringBuilder sb = new StringBuilder("Student{");
        sb.append("id=").append(id);
        sb.append(", firstName='").append(firstName).append('\'');
        sb.append(", lastName='").append(lastName).append('\'');
        sb.append(", age=").append(age);
        sb.append('}');
        return sb.toString();
    }

//    Геттеры и сеттеры опущены
```

SELECT * FROM STUDENT ;

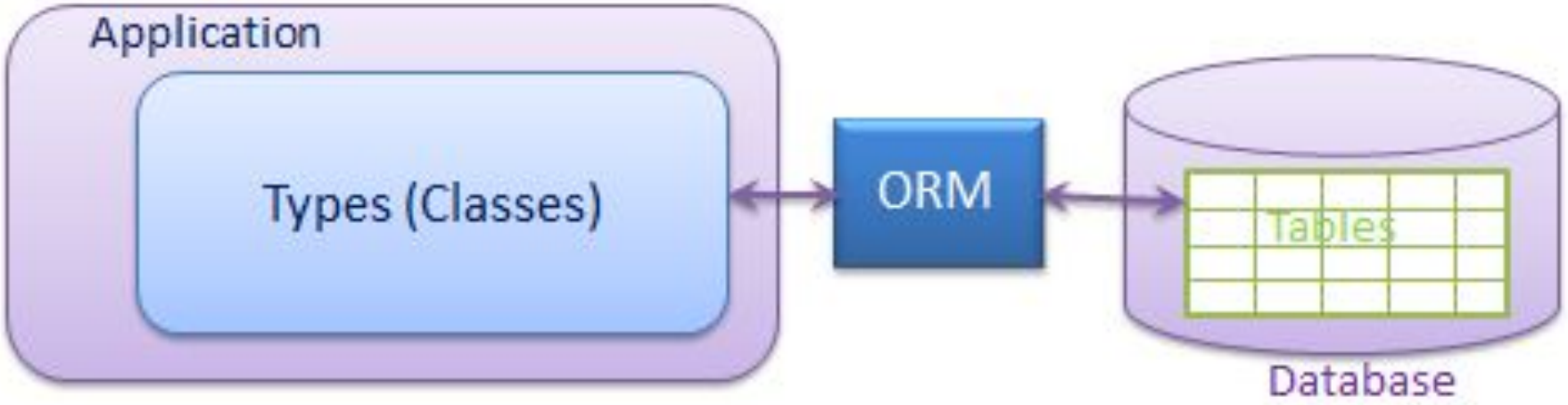
ID	FIRST_NAME	LAST_NAME	AGE
1	Вася	Пупкин	17
2	Петя	Дудкин	18
3	Вася	Васильев	18

Example

```
1  package com.simbirsoft;
2
3  import ...
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 }
```

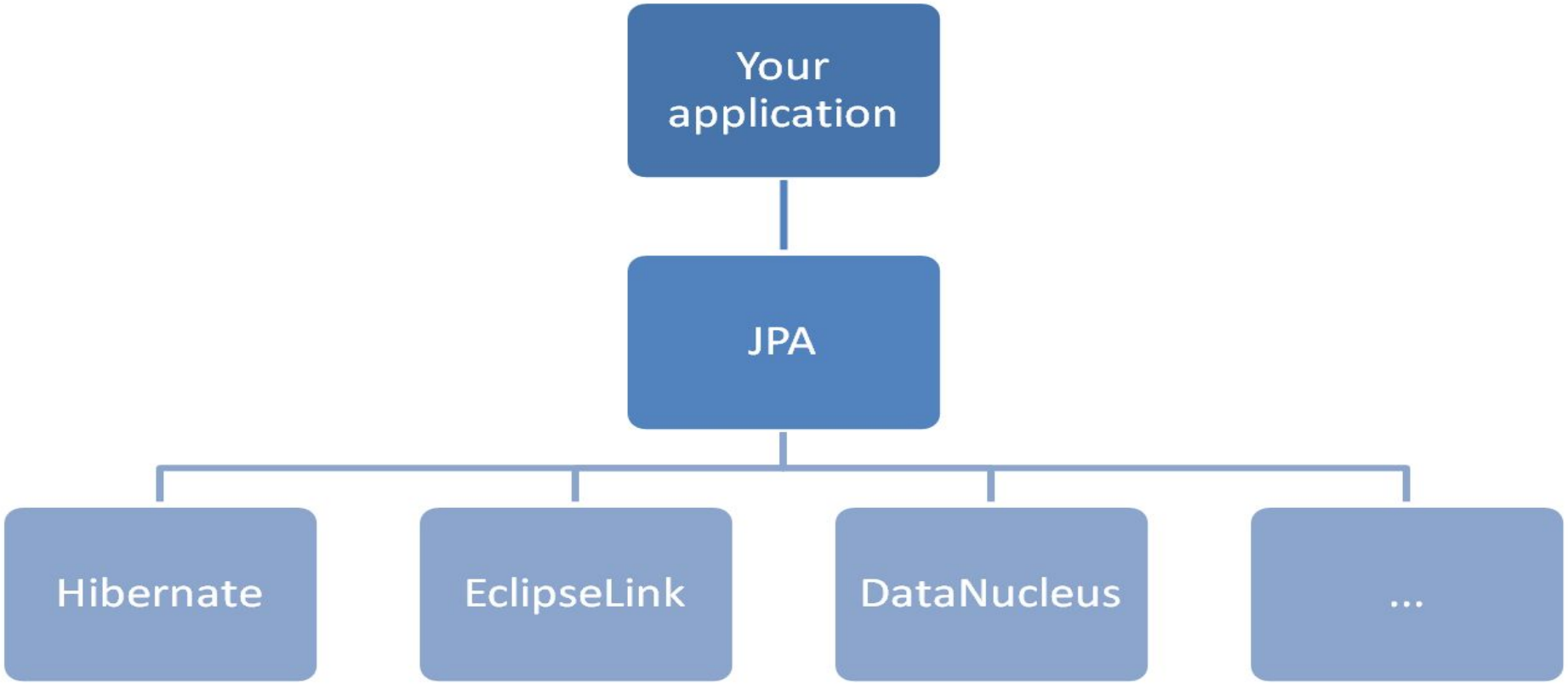
ORM

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

```java
package com.simbirsoft.model.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Student {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    private Integer age;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    //Остальные сеттеры и геттеры опущены для наглядности
}
```

SELECT * FROM STUDENT;

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

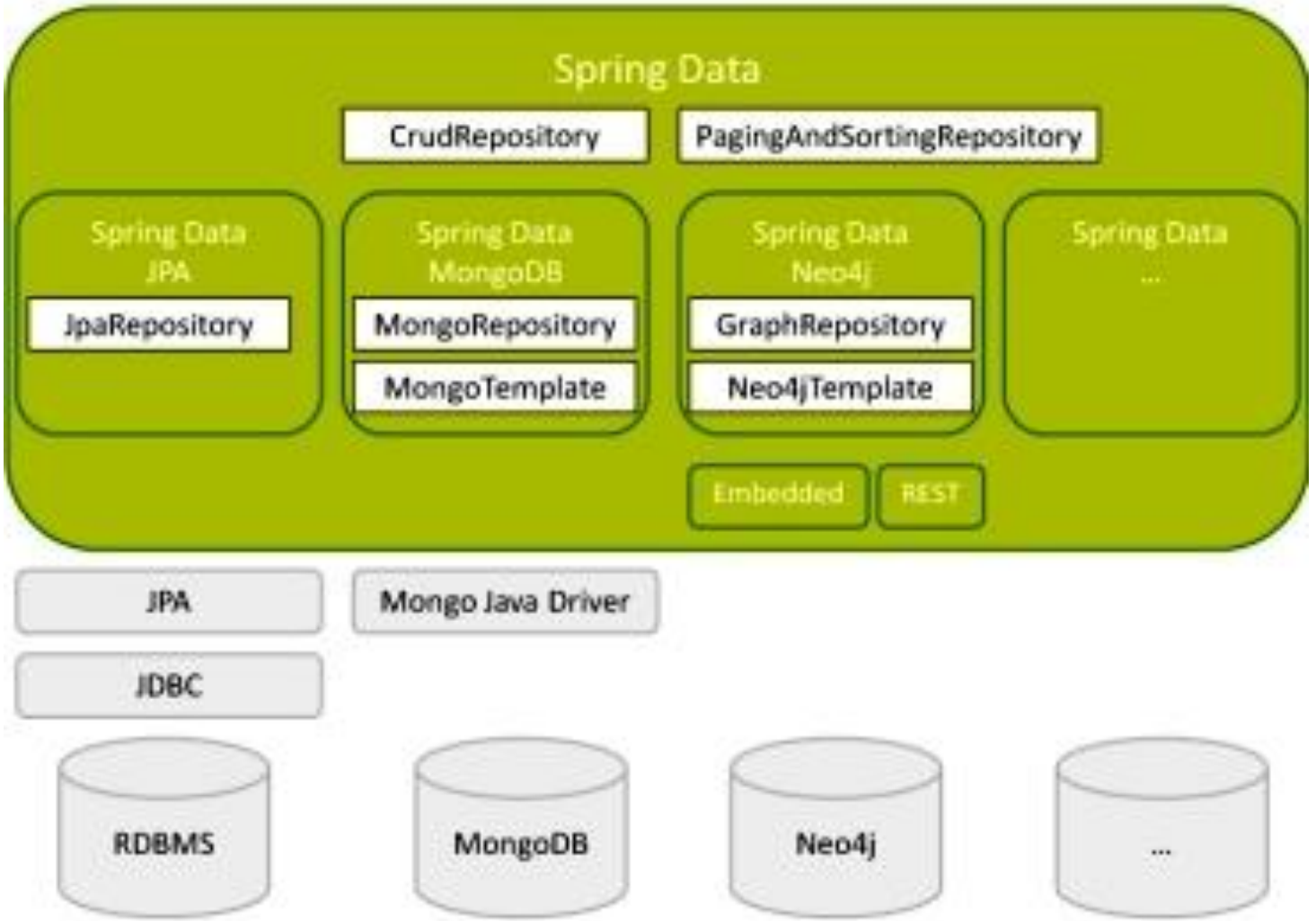
Основные аннотации

- @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

```java
@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);
    }
```

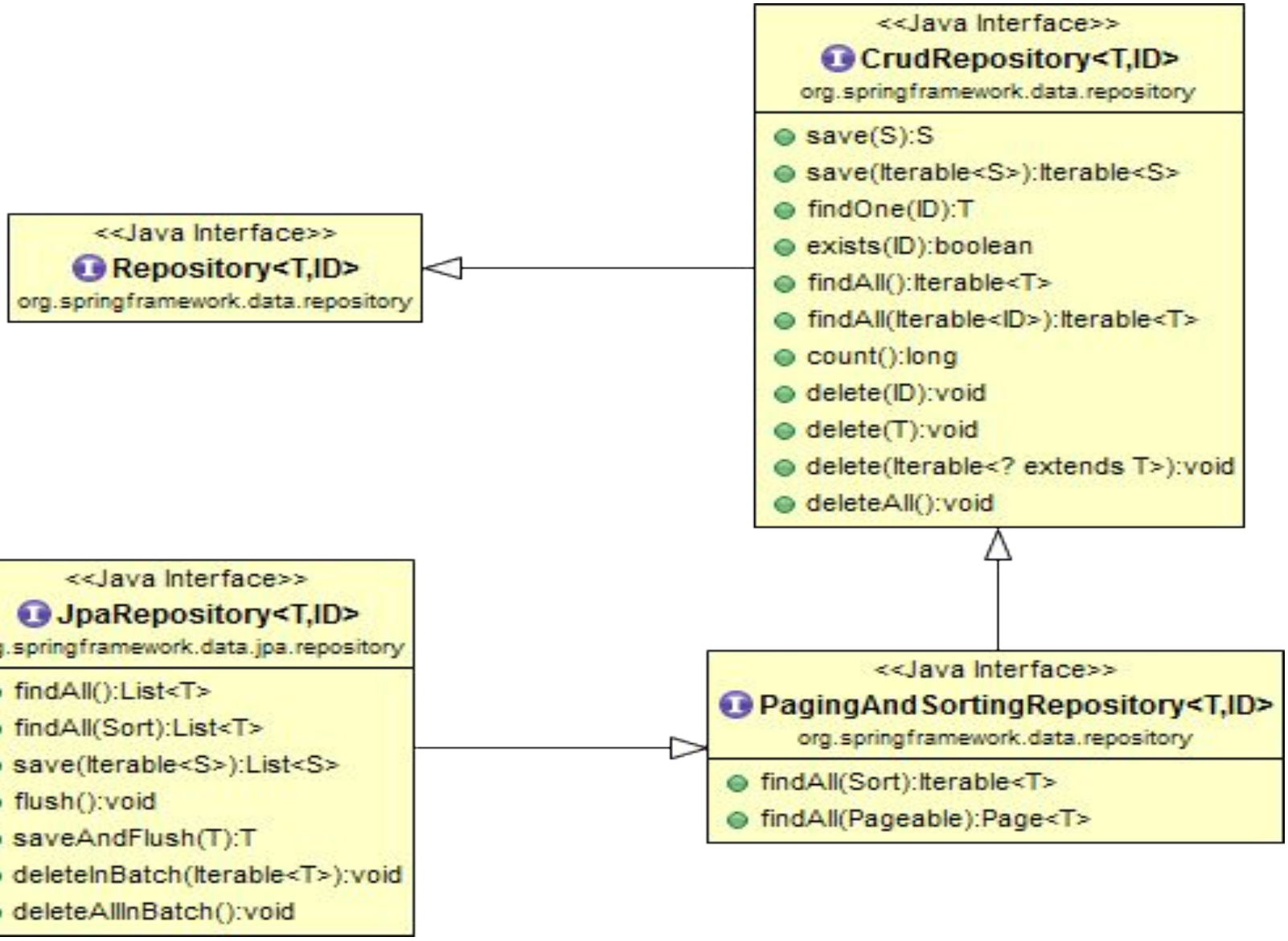
Spring Data


Spring Data
CrudRepository
PagingAndSortingRepository
Spring Data JPA
JpaRepository
Spring Data MongoDB
MongoRepository
MongoTemplate
Spring Data Neo4j
GraphRepository
Neo4jTemplate
Spring Data ...
Embedded
REST
JPA
Mongo Java Driver
JDBC
RDBMS
MongoDB
Neo4j
...

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

```java
package com.simbirsoft.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.simbirsoft.model.domain.Student;

public interface StudentRepository extends JpaRepository<Student, Long> {
}
```

```java
package com.simbirsoft.model.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Student {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    private Integer age;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    //Остальные сеттеры и геттеры опущены для наглядности
}
```

SELECT * FROM STUDENT;

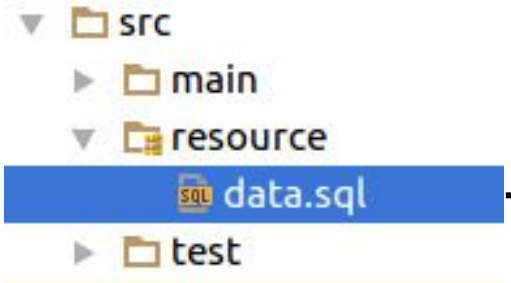
ID	AGE	FIRST_NAME	LAST_NAME
1	17	Вася	Пупкин
2	18	Петя	Дудкин

Example

```java
package com.simbirsoft;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import com.simbirsoft.model.domain.Student;
import com.simbirsoft.repository.StudentRepository;

@Component
public class SomeBean {

    @Autowired
    private StudentRepository repository;

    public void doSmth() {
        Student studentKolya = new Student();
        studentKolya.setFirstName("Коля");
        studentKolya.setLastName("Уткин");
        studentKolya.setAge(20);
        repository.save(studentKolya);
        assert Long.valueOf(3L).equals(studentKolya.getId());

        Student studentVasya = repository.getOne(1L);
        assert "Вася".equals(studentVasya.getFirstName());

        List<Student> students = repository.findAll();
        assert students.size() == 3;

        repository.deleteAll();
        assert repository.findAll().size() == 0;
    }
}
```

Example

```java
package com.simbirsoft;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;


@SpringBootApplication
public class SpringBootApp  {

    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(SpringBootApp.class, args);
        context.getBean(SomeBean.class).doSmth();
    }
}
```




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

Query Creation

```java
package com.simbirsoft.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import com.simbirsoft.model.domain.Student;

public interface StudentRepository extends JpaRepository<Student, Long> {
    List<Student> findByFirstNameIgnoreCase(String firstName);

    List<Student> findByLastNameAndAge(String lastName, Integer age);

    List<Student> findByAgeOrderByLastNameDesc(Integer age);

    List<Student> findByLastNameLike(String firstName);

    List<Student> findByAgeBetween(Integer from, Integer to);
}
```

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

```java
package com.simbirsoft.model.domain;

import ...

@Entity
public class Student {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    private Integer age;

    @ManyToOne
    @JoinColumn(name = "GROUP_ID")
    private Group group;

    //    Геттеры и сеттеры опущены
```

```java
package com.simbirsoft.model.domain;

import ...

@Entity
@Table(name = "GROUPS")
public class Group {
    @Id
    @GeneratedValue
    private Long id;

    private String name;

    @OneToMany(fetch = FetchType.LAZY,
               mappedBy = "group")
    private List<Student> students;

//    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

```java
package com.simbirsoft.repository;

import ...

public interface GroupRepository extends JpaRepository<Group, Long> {
}
```

```java
package com.simbirsoft;

import ...

@Component
public class SomeBean {
    @Autowired
    private GroupRepository groupRepository;

    public void doSmth() {

        List<Group> groups = groupRepository.findAll();

        for (Group group : groups) {
            System.out.println(group.getStudents());
        }
    }
}
```

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