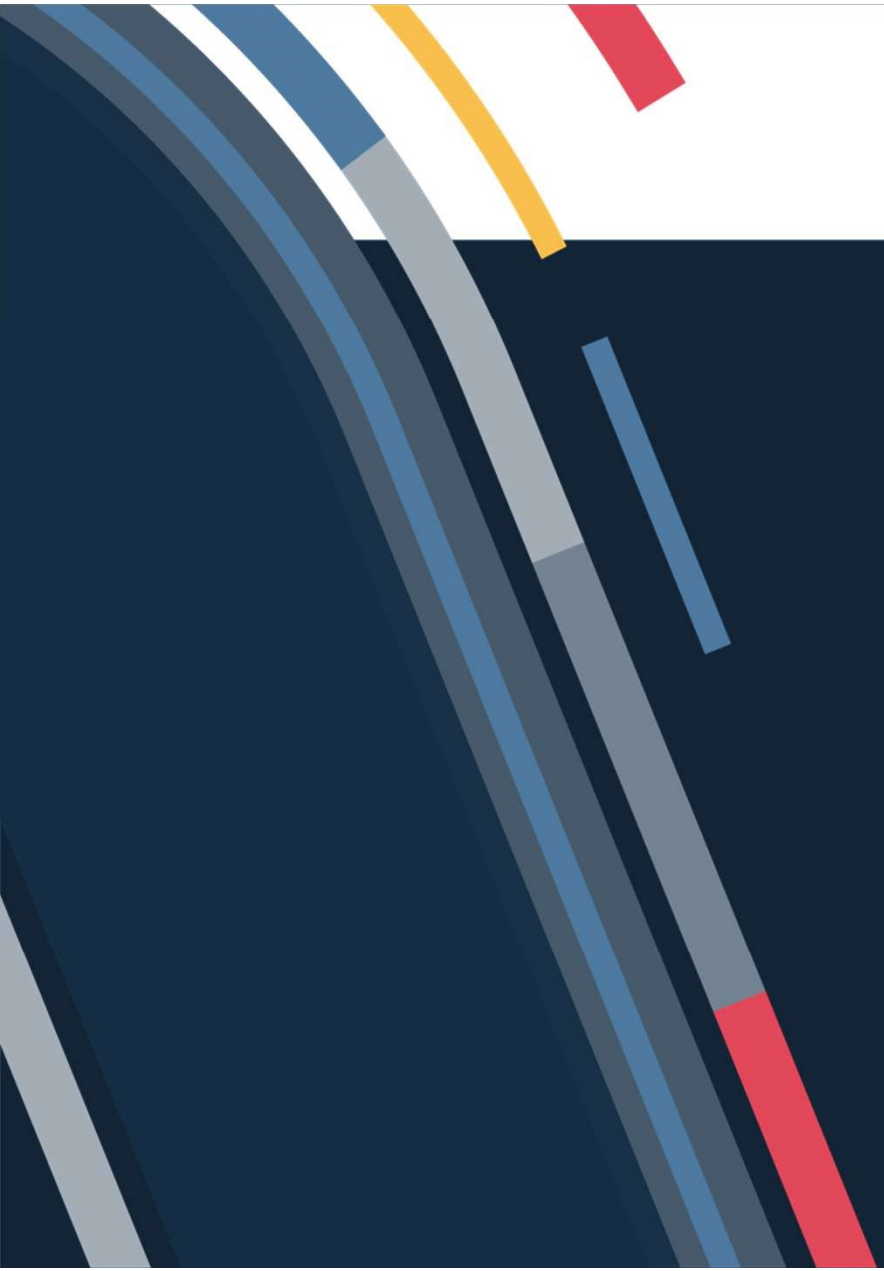


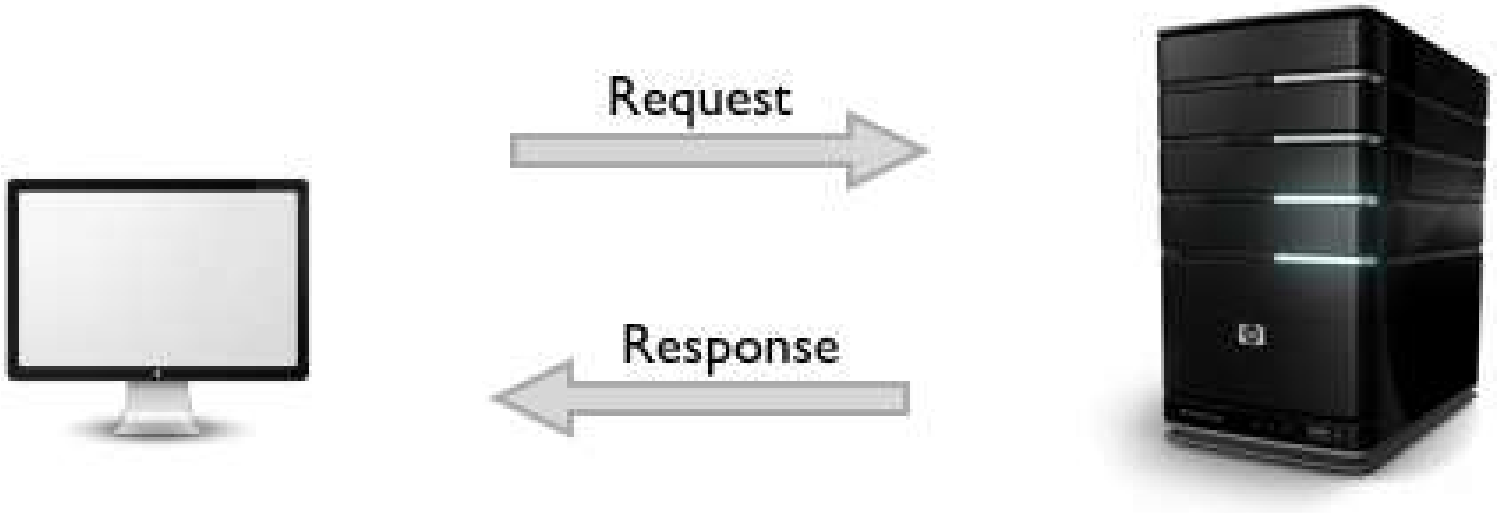
Focused forward

Netcracker

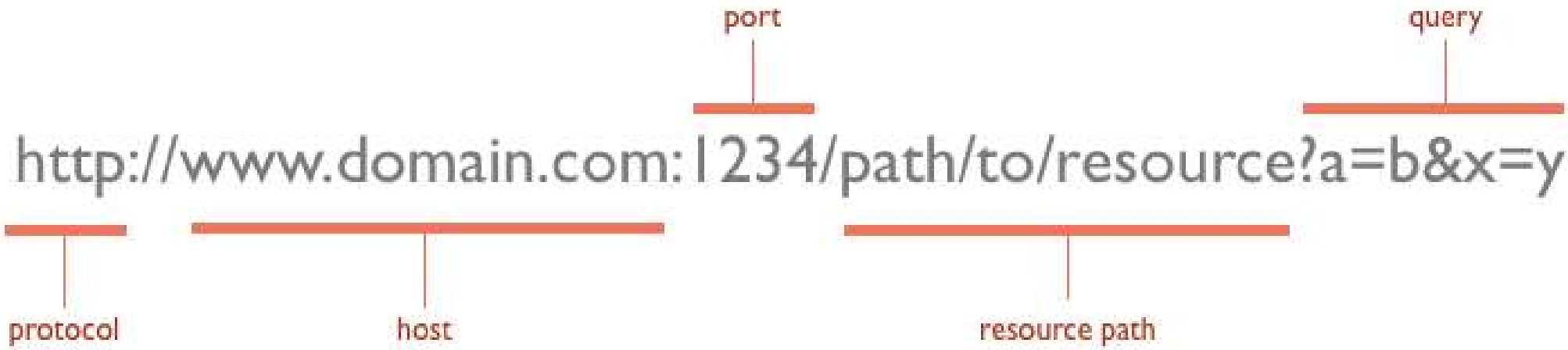
# Lecture 7. HTTP. JSON. REST Spring Data JPA. Spring Boot.

Alexei Khudnitsky

# HTTP



# URL


port
query
http://www.domain.com:1234/path/to/resource?a=b&x=y
protocol
host
resource path

# HTTP Методы


SAFE METHODS
NO ACTION ON SERVER
GET
HTTP/1.1 MUST IMPLEMENT THIS METHOD
HEAD
INSPECT RESOURCE HEADERS
MESSAGE WITH
BODY
SEND DATA TO SERVER
PUT
DEPOSIT DATA ON SERVER – INVERSE OF GET
POST
SEND INPUT DATA FOR PROCESSING
PATCH
PARTIALLY MODIFY A RESOURCE
TRACE
ECHO BACK RECEIVED MESSAGE
OPTIONS
SERVER CAPABILITIES
DELETE
DELETE A RESOURCE – NOT GUARANTEED

# Коды состояния

1. **1xx: Информационные сообщения**
2. **2xx: Сообщения об успехе**
3. **3xx: Перенаправление**
4. **4xx: Клиентские ошибки**
5. **5xx: Ошибки сервера**

# JSON

- JavaScript-объекты { ... } или
- Массивы [ ... ] или
- Значения одного из типов:
  - строки в двойных кавычках,
  - число,
  - логическое значение true/false,
  - null.

```
{
   "firstName": "Иван",
   "lastName": "Иванов",
   "address": {
       "streetAddress": "Московское ш., 101, кв.101",
       "city": "Ленинград",
       "postalCode": "101101"
   },
   "phoneNumbers": [
       "812 123-1234",
       "916 123-4567"
   ]
}
```

# Что такое REST?

GET /books/ — получить список всех книг
GET /books/3/ — получить книгу номер 3
POST /books/ — добавить книгу (данные в теле запроса)
PUT /books/3 – изменить книгу (данные в теле запроса)
DELETE /books/3 – удалить книгу

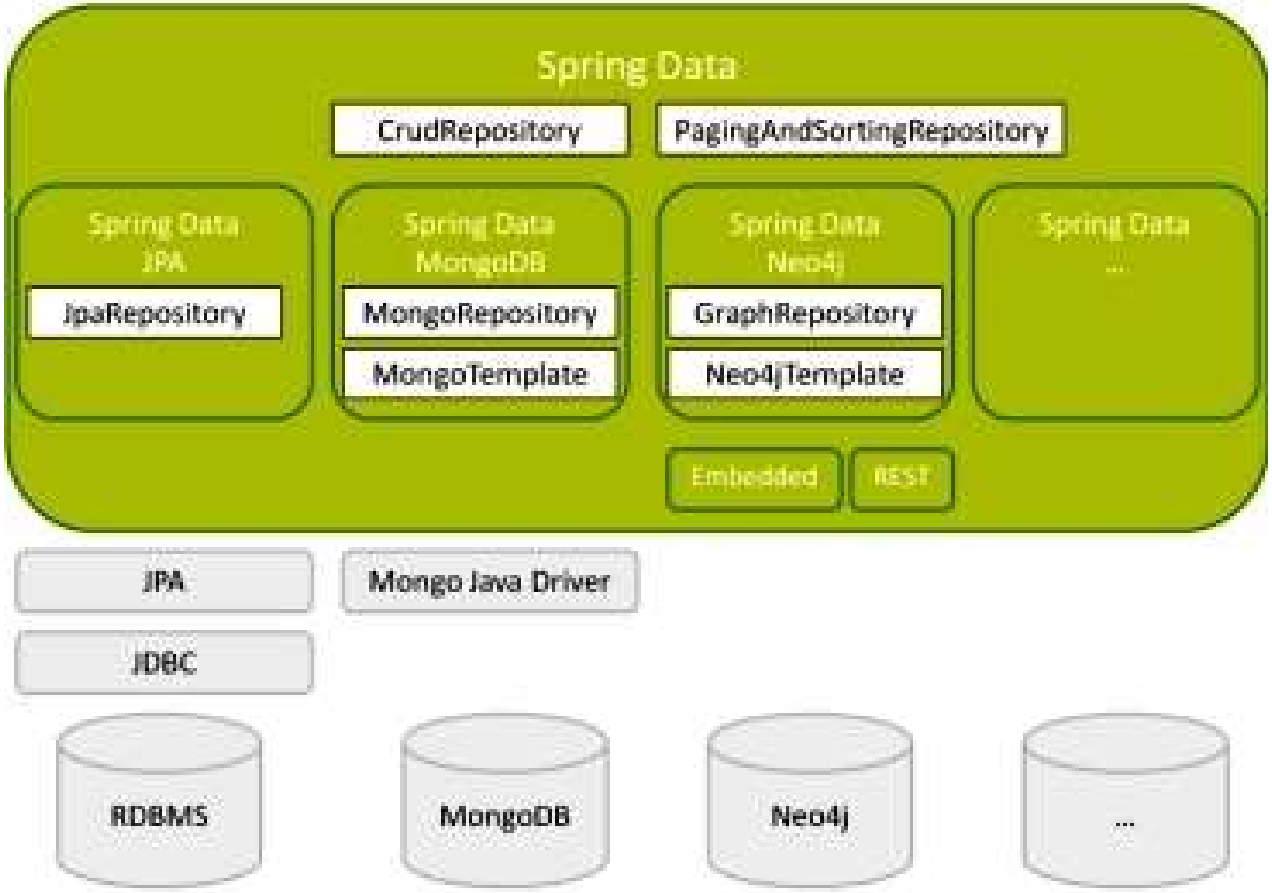
# Spring Data JPA

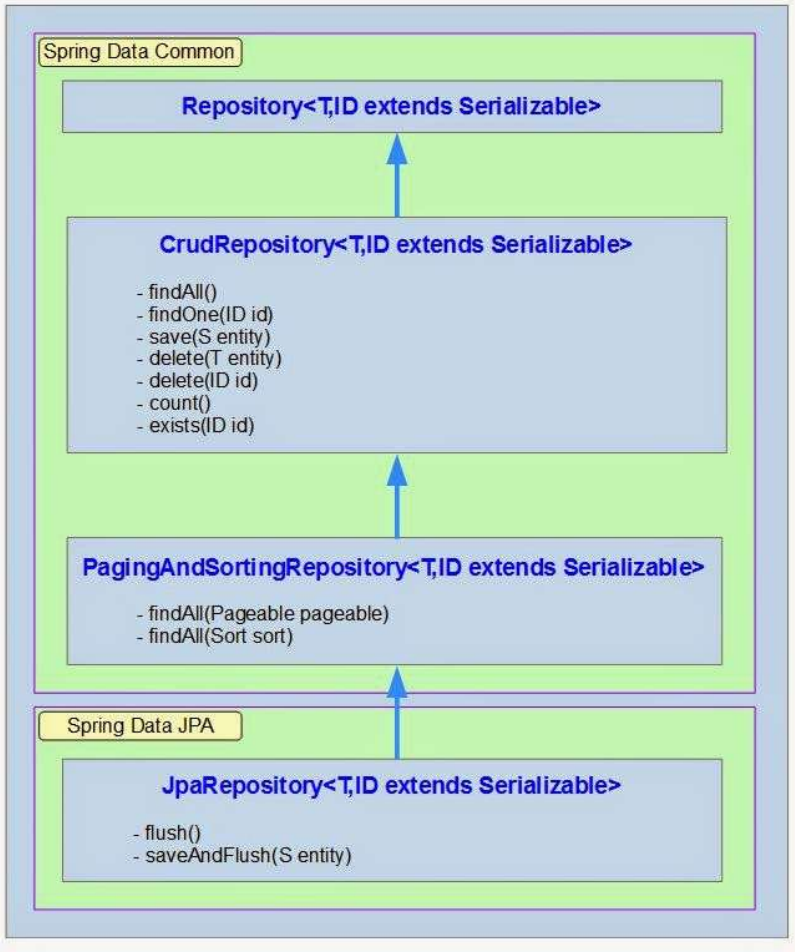
Spring Data JPA

# Преимущества Spring Data JPA

- Имеет в себе реализацию множества CRUD-операций
- Имеет реализации для нескольких СУБД
- Query builder mechanism
- Возможность создавать named queries

# Repositories



# Query builder mechanism

| Ключевое слово | Пример |
|---|---|
| And | findByFirstNameAndLastName |
| Or | findByIdOrLastName |
| Between | findByBirthdayBetween |
| LessThan | findByAgeLessThan |
| GreaterThan | findByAgeGreaterThan |
| Containing | findByLastNameContaining |
| Not | findByLastNameNot |

# Entities mapping

```
@Entity
public class Flight implements Serializable {
    Long id;

    @Id
    public Long getId() { return id; }

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

```
@Entity
@Table(name="tbl_sky")
public class Sky implements Serializable {
    ...
}
```

# Generating the identifier property

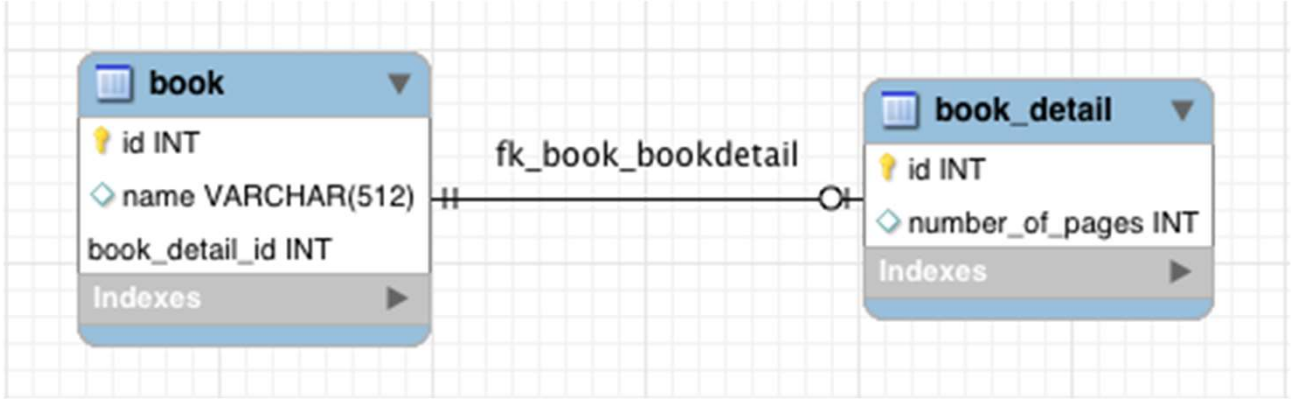
- AUTO - either identity column, sequence or table depending on the underlying DB
- TABLE - table holding the id
- IDENTITY - identity column
- SEQUENCE - sequence
- identity copy - the identity is copied from another entity

```
@Id @GeneratedValue(strategy=GenerationType.IDENTITY)
public Long getId() { ... }
```

# Relationship mapping

- One-to-one
- One-to-many
- Many-to-many

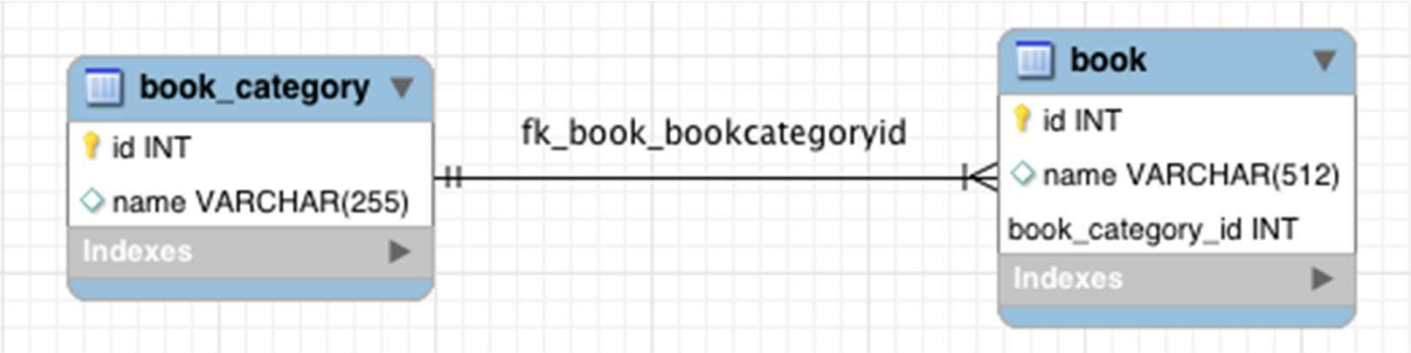
# One-to-one relationship mapping




```java
@Entity
public class Book {
  private int id;
  private String name;
  private BookDetail bookDetail;

  ...

  @OneToOne(cascade = CascadeType.ALL)
  @JoinColumn(name = "book_detail_id")
  public BookDetail getBookDetail() {
    return bookDetail;
  }
}
```

```java
@Entity
@Table(name = "book_detail")
public class BookDetail {
  private Integer id;
  private Integer numberOfPages;
  private Book book;

  ...

  @OneToOne(mappedBy = "bookDetail")
  public Book getBook() {
    return book;
  }
}
```

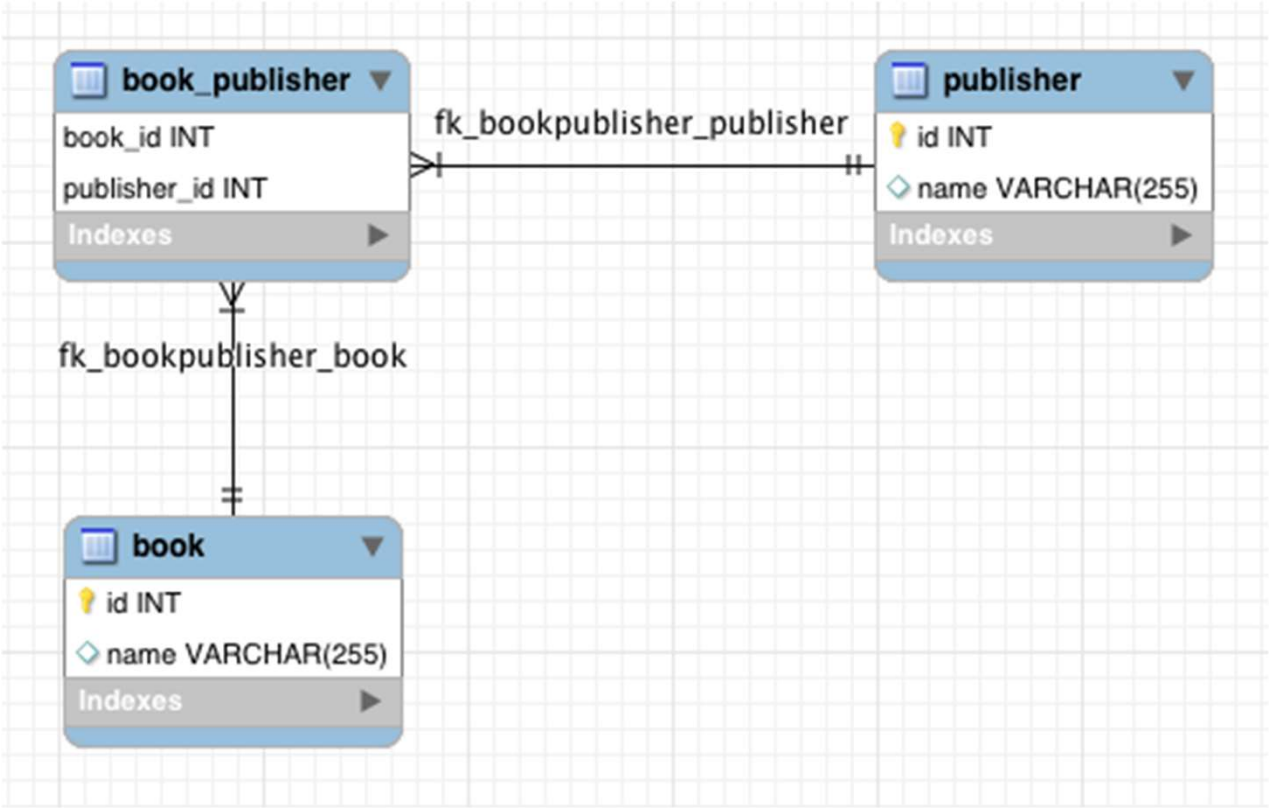
# One-to-many relationship mapping




```java
@Entity
@Table(name = "book_category")
public class BookCategory {
  private int id;
  private String name;
  private Set<Book> books;

  @OneToMany(mappedBy = "bookCategory", cascade = CascadeType.ALL)
  public Set<Book> getBooks() {
    return books;
  }
}
```

```java
@Entity
public class Book{
  private int id;
  private String name;
  private BookCategory bookCategory;

  @ManyToOne
  @JoinColumn(name = "book_category_id")
  public BookCategory getBookCategory() {
    return bookCategory;
  }
}
```

# Many-to-many relationship mapping



# Many-to-many relationship mapping

```java
@Entity
public class Book{
    private int id;
    private String name;
    private Set<Publisher> publishers;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int getId() {
        return id;
    }

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "book_publisher",
            joinColumns = @JoinColumn(name = "book_id", referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(name = "publisher_id", referencedColumnName = "id"))
    public Set<Publisher> getPublishers() {
        return publishers;
    }
}
```

```java
@Entity
public class Publisher {
    private int id;
    private String name;
    private Set<Book> books;

    @ManyToMany(mappedBy = "publishers")
    public Set<Book> getBooks() {
        return books;
    }
}
```

# Fetch type

```
@OneToMany(fetch = FetchType.EAGER, mappedBy = "book")
private Set<Author> users;
```

```
@OneToMany(fetch = FetchType.LAZY, mappedBy = "book")
private Set<Author> users;
```

# Spring Boot


# Возможности

- Создание полноценных Spring приложений
- Встроенный Tomcat или Jetty (не требуется установки WAR файлов)
- Обеспечивает 'начальные' POMs для упрощения вашей Maven конфигурации
- Автоматическая конфигурирация Spring когда это возможно
- Конфигурация без генерации кода и без написания XML

# Dependencies

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.5.RELEASE</version>
</parent>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

# Application class

```
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class);
    }
}
```

# application.yml

```yaml
spring:
    profiles: test
name: test-YAML
environment: test
servers:
    - www.abc.test.com
    - www.xyz.test.com

---
spring:
    profiles: prod
name: prod-YAML
environment: production
servers:
    - www.abc.com
    - www.xyz.com
```

1.@Value("${envitonment}")
2.private String envitonment;

# Controller

```java
@Controller
@RequestMapping("books")
public class SimpleBookController {

    @GetMapping("/{id}", produces = "application/json")
    public @ResponseBody Book getBook(@PathVariable int id) {
        return findBookById(id);
    }

    private Book findBookById(int id) {
        // ...
    }
}
```

```java
@RestController
@RequestMapping("books-rest")
public class SimpleBookRestController {

    @GetMapping("/{id}", produces = "application/json")
    public Book getBook(@PathVariable int id) {
        return findBookById(id);
    }

    private Book findBookById(int id) {
        // ...
    }
}
```

# Литература


1. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.entity-persistence
2. https://docs.jboss.org/hibernate/annotations/3.5/reference/en/html/entity.html#entity-hibspec-entity
3. https://habr.com/post/215117/ (HTTP)
4. https://habr.com/post/38730/ (REST)
5. Spring Boot in action

# Пример

# Q&A

# Thank You