Prepared by Radha V Krishna

SPRING BOOT AND DATABASE

WAYS TO CONNECT SB TO DB

// connect to RDBMS

- Using JdbcTemplate
- Using JPA
- Using JPARepository
- //connect to Mongodb
  - Using MongoRepository

```
@SpringBootApplication
public class SpringBootJdbcExample implements
CommandLineRunner {

    @Autowired
    JdbcTemplate jdbcTemplate;

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

    @Override
    public void run(String... args) throws Exception {

        String sql = "INSERT INTO books (title, author, price)
VALUES (?, ?, ?)";
        int result = jdbcTemplate.update(sql, "Head First Java",
"Kathy Sierra", 18.55f);

        if (result > 0) {
            System.out.println("Insert successfully.");
        }
    }
}
```

```
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupI
            <artifactId>spring-boot-starter-jdbc</artifact
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
```

```
void testUpdate() {
    String sql = "UPDATE books SET price=? WHERE title=?";
    Object[] params = {9.99f, "Effective Java"};
    int result = jdbcTemplate.update(sql, params);

    if (result > 0) {
        System.out.println("Update successfully.");
    }
}
```

```
void testListAll() {
    String sql = "SELECT * FROM books";

    List<Book> listBooks = jdbcTemplate.query(sql,
            BeanPropertyRowMapper.newInstance(Book.class));

    for (Book book : listBooks) {
        System.out.println(book);
    }
}
```

```
//h2 database

spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=user
spring.datasource.password=password
spring.jpa.database-
platform=org.hibernate.dialect.H2Dialect

// Mysql

spring.datasource.url=jdbc:mysql://localhost:3306/training
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5Dialect
#spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
 # validate #create – creates,updates or validates the
table.
spring.datasource.initialization-mode=always
```

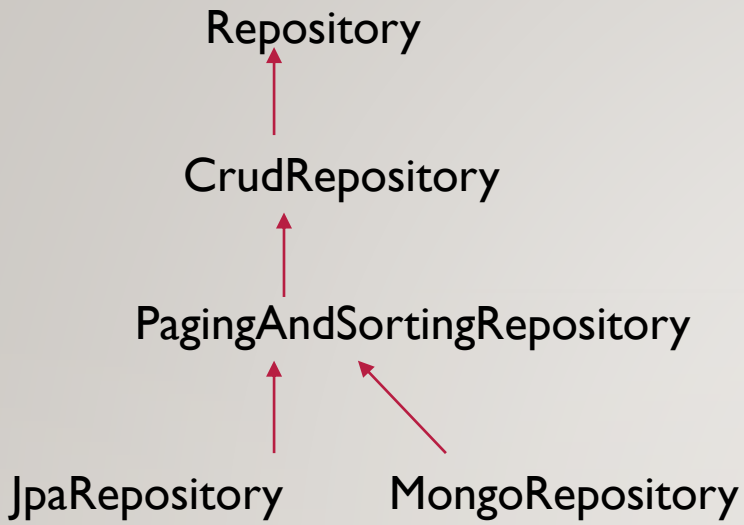
```
//Mongodb

spring.data.mongodb.uri=mongodb://user1:user1@l
ocalhost:27017/training

spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=training
spring.data.mongodb.username=user1
spring.data.mongodb.password=user1
```

Using JPA Repositories for RDBMS


```
@Entity  // for RDBMS
@Table(name = "books")
@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
@Data
public class Book {

@Id     // Primary Identifier
//@Column(length = 100)
@GeneratedValue(strategy = GenerationType.AUTO)
private long isbn;  // native assigned sequence

@Column(name = "title",nullable = true)
private String title;
private double price;
private long stock;
private String category;


}
```

For NoSQL

```
@AllArgsConstructor
@NoArgsConstructor
@Data
@Document("customers") // for noSQL
public class Customer {
@Id
private long id;
private String name;
private String email;


}
```




```
public interface BookRepo extends
JpaRepository<Book, Long>{}
```

```
@Repository
public interface CustomerRepo extends
MongoRepository<Customer, Long> {


}
```

// queries and DML operations


```
@Repository
public interface BookRepo extends JpaRepository<Book, Long>{

List<Book> findByTitle(String title);
List<Book> findByPrice(double price);
List<Book> findByTitleAndPrice(String title,double price);

@Query("select b from Book b where b.title like :pattern")
List<Book> getBooksTitleLike(@Param("pattern") String pattern);

@Transactional
@Modifying
@Query("update Book b set b.stock=b.stock+:stock where b.title = :title")
int updateBookStockTitle(@Param("title") String title,@Param("stock") long
newStock);
}
```

Reference : https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

Class Relations


```
public class Account {

@Id
@GeneratedValue(strategy =
GenerationType.AUTO)
@Column(name="account_no")
private long accountNo;

private double balance;

@Column(name="account_type")
private String accountType;
private LocalDate doa;

@ManyToOne()
@JoinColumn(name = "custId")
private Customer customer;
}
```

```
@Entity
public class Customer {

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name="cust_id")
private long custId;

private String name;
private String username;

//@Temporal(TemporalType.DATE)
private LocalDate dob;


@OneToMany(fetch = FetchType.EAGER,cascade =
CascadeType.ALL)
private List<Account> accounts;
```

Field	Type	Null	Key	Default	Extra
account_no	bigint	NO	PRI	NULL	auto_increment
doa	date	YES		NULL	
balance	double	YES		NULL	
account_type	text	YES		NULL	
cust_id	bigint	YES	MUL	NULL	

Field	Type	Null	Key	Default	Extra
cust_id	bigint	NO	PRI	NULL	auto_increment
dob	date	YES		NULL	
name	varchar(100)	NO		NULL	
username	text	YES		NULL	

//in MongoDb Repo


```
@Query("{id :?0}")
//SQL Equivalent : SELECT * FROM BOOK WHERE ID=?

// JPA   @Query(select Book b where id=:id)

        Optional<Book> getBookById(Integer id);

        @Query("{price : {$lt: ?0}}")                          // SQL Equivalent : SELECT * FROM BOOK where pages<?
        //@Query("{ pages : { $gte: ?0 } }")                   // SQL Equivalent : SELECT * FROM BOOK where pages>=?
        //@Query("{ pages : ?0 }")                              // SQL Equivalent : SELECT * FROM BOOK where pages=?
        List<Book> getBooksByPages(Integer pages);

        @Query("{author : ?0}")                                // SQL Equivalent : SELECT * FROM BOOK where author = ?
        List<Book> getBooksByAuthor(String author);
@Query("{author: ?0, price: ?1}")
```

Reference : https://docs.spring.io/spring-data/mongodb/docs/1.2.0.RELEASE/reference/html/mongo.repositories.html