Date : 13/10/2020
Spring Boot 7AM
Mr. RAGHU

PDF docs:
https://www.mediafire.com/file/w5x9w5vcmkwkkdv/RaghuSirNareshITJavaPdfs.zip/file
Github:
https://github.com/javabyraghu

*)Working with Collections:-(3) List, Set, Map
 Every Collection is taken as one child table in Daatabase, by data JPA,
 if we define a collection variable in model class.

*)We have 3 Collections support in Data JPA, those are again devided into 2 types

Index Based Collection	Non-Index Based Collections
List	Set
Map	

=> For every collection variable one table gets created with min 2 columns
  and max 3 columns. (key Column, Index Column, Element Column)

```
  key column     == (FK)link column to parent table Primary Key | Join Column
  index column   == position of data in collection
  element column == collection actual data
```

-JPA Annotations for Element Collections---
a) @ElementCollection: This annotation must apply at Collection variable level.
  It will create one DB table for Collection variable.

b) @CollectionTable(name="",joinColumns = @JoinColumn(name=""))
  It is optional, this is used to provide child_table name and Key Column(Join column)
  name.

--Default, if no annotation is applied---
  parentModelclassName_CollectionVariableName --> child table name
  parentModelClassName_PKColumNname ---> child table Join Column(key column)

----

c) @Column(name="") : It is used to provide element column name. (optional)

d) @OrderColumn(name=""): It is applied for index only for List Type

e) @MapKeyColumn(name="") : It is applied for index only for Map Type

========================code==
=

1. Model class

```
package in.nareshit.raghu.model;

import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.persistence.CollectionTable;
import javax.persistence.Column;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.MapKeyColumn;
import javax.persistence.OrderColumn;
import javax.persistence.Table;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="emptab")
public class Employee {
	@Id
	@Column(name="eid")
	private Integer empId;
	@Column(name="ename")
	private String empName;
	@Column(name="esal")
	private Double empSal;

	@ElementCollection
	@CollectionTable(name="empprjstab",
	joinColumns = @JoinColumn(name="eid")) //key column
	@Column(name="prj") //element column
	@OrderColumn(name="pos") //index column
	private List<String> empPrjs;


	@ElementCollection
	@CollectionTable(name="emptasktab",
	joinColumns = @JoinColumn(name="eid"))//key column
	@Column(name="task") // element column
	private Set<String> empTaks;


	@ElementCollection
	@CollectionTable(name="empmodulestab",
	joinColumns = @JoinColumn(name="eid")) //key column
	@Column(name="module") //element column
	@MapKeyColumn(name="pos") // index column
	private Map<Integer,String> empModules;


}
```

2. Repository Interface

```
package in.nareshit.raghu.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Employee;

public interface EmployeeRepository
	extends JpaRepository<Employee, Integer> {

}
```

3. Runner class

```
package in.nareshit.raghu.runner;
```

```
import java.util.List;
import java.util.Map;
import java.util.Set;

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

import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.repo.EmployeeRepository;

@Component
public class EmployeeTestRunner implements CommandLineRunner {
	@Autowired
	private EmployeeRepository repo;

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

		repo.save(
				new Employee(
						10, "A", 2.2,
						List.of("P1","P2"),
						Set.of("T1","T2"),
						Map.of(101, "M1",102,"M2")
						)
				);
		System.out.println("____DONE____");
	}

}
```

4. application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot7am
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=create
```
