Working with Large: Objects (LOBs)

===:

LOBs (files)

======

BLOB

(Binary LOB)

audio file, video file,

image file, pdf and etc..

CLOB

(Character LOB)

V

Text file,Rich Text file

CSV file and etc...

=>All most all major DB s/w are supporting Large Object by giving different data types =>oracle gives BLOB, CLOB Data types

=>mysql gives BLOB, tinyText,medium Text, large text

tinyBLOB,mediumBLOB

data types..

=>Only in standalone Apps the Large objects (files) will be saved directly in Db table cols

eg: standalone matrimony Apps, standalone job portal Apps and etc...

=> In web applications, distributed Apps the LOBs (files) will saved in Server machine file System and their address paths will be written to DB table cols as String values.

(To avoid perforence issue with Db s/w)

eg:: matrimony web applications, job portal App, social networking app and etc.. (shaadi.com) (Naukri.com)

(fb.com)

=>In some web applicaitons or Distributed Apps the LOBS (files) will be saved in seperate

In standalone Apps

job seeker_info gracle)

jsid(n) jsname(vc2) jsqlfy(vc2) photo(blob) resume(clob)

CDN(Content Delivery NetWork) server, DMS(Document Management System) server, NAS (Network Attached Storage), S3 Bucket servers and etc.. and we get token ids from those server representing LOBS (files) insertion and these token ids will be inserted BB table cols as String values.

eg:: matrimony web applications, job portal App, social networking app and etc..

=>The Entity class should have byte[] and char[] type properties represeting BLOB,CLOB type cols of the db table

101

raja

Hyd

name:raja age:30

skills:: java, oralce

Files are directly saved in DB table cols

In web applications or distributed Apps

====================================

Server machine file system

E:\store

job_seeker_info pacle db table) jsId(n) jsName(vc2) sqlfy(vc2)

101

102

raja ravi

B.Tech

Btech

|----> raja_photo.gif |---->raja_resume.txt |---> ravi_photo.gif |---->ravi_resume.txt

(The uploaded files are saved here)

=> In ORM f/ws (like hibernate) we take entity class as Serializable class for two reasons (Entity class implementing java.io.Serializable(I)) (a) Now a days, we are making entity class itself as the Model class(Java Bean) to carry data across

the multiple layers like (contoller-serivce - repo /DAO class) ..Some times we need to send

same data over the network from current project to another project.. So it need to take Entity class as Serialziable class (Flipkart sending Card Details to PayPall over the network)

(b) The ORM f/w like hibernate supports Disk caching in second level caching i.e it uses hard disk memory through serialization while performing second Level caching..For This Entity classes should be taken as Serializable.

class

datatype =>In Entity @Lob on byte[] property makes spring data jpa/ORM f/w giving BLOB col in db table =>In Entity@Lob on char[] property makes spring data jpa/ORM f/w giving CLOB col in db table data type => The boolean property data (true/false) of Entity class obj will be stored in Db table as number col value (0/1)

Example App

================

Entity class

@Entity

O for false

1 for true

use

In Oracle,mysql DB s/w there is no boolean data type support directly.. So numberic col holding 0 or 1.

O for flase

1 for true

Serialization is the process of converting the object data into

bits and bytes so that they can written to a file or over the network

photo_path(vc2) resume_path(vc2)

e:\store\raja_photo.gif e:\store\raja_resume.txt

e:\store\ravi_photo.gif

e:\store\ravi_resume.txt

[Here we are saving only the address paths of the files as the String values]

All the files and folders of computer together is called FileSystem

```
}
@Data
@AllArgsConstructor
@NoArgsConstructor
@RequiredArgsConstructor
public class MarriageSeeker implements Serializable {
@Id
@GeneratedValue
private Long id;
@NonNull
@Column(length=20)
private String name; @NonNull
@Column(length=20)
private String addrs; @NonNull
@Lob
private byte[] photo;
@NonNull
private LocalDateTime dob;
@NonNull
@Lob
private char[] biodata;
@NonNull
private boolean indian;
```

applicaiton.properties

```
#DataSource cfg
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql:///ntspbms714db
spring.datasource.username=root
spring.datasource.password=root
#spring.datasource.driver-class-name-oracle.jdbc.driver.Oracle Driver
#spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
```

```
#spring.datasource.username=system
#spring.datasource.password=manager
#JPA-Hiberante properties
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
#spring.jpa.database-platform=org.hibernate.dialect.Oracle 10gDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto-update
# other possible values create,validate,create-drop
```

service interface

```
public interface IMatrimonySerivceMgmt {
}
public String registerMarriageSeeker(MarriageSeeker seeker); public Optional<MarriageSeeker>
searchSeekerById(Long id);
```

service Impl class

```
@Service("matrimonyService")
public class MartrimonyServiceImpl implements IMatrimonySerivceMgmt { @Autowired
private IMarriageSeekerRepo msRepo;
```

Runner class

```
@Override
public String registerMarriageSeeker(MarriageSeeker seeker) {
return "Marriage Seeker Info is saved with id value "+msRepo.save(seeker).getId();
}
@Override
public Optional<MarriageSeeker> searchSeekerById(Long id) {
return msRepo.findById(id);
}
}
@Component
public class LOBsTestRunner implements CommandLineRunner { @Autowired
private IMatrimonySerivceMgmt service;
@Override
public void run(String... args) throws Exception { /*Scanner sc=new Scanner(System.in);
System.out.println("Enter person name::");
String name=sc.next();
System.out.println("Enter person address::");
String addrs=sc.next();
System.out.println("Enter Person Photo file complete path::");
String photoPath=sc.next().replace("?","");
System.out.println("Enter Person biodata file complete path::");
String biodata Path=sc.next().replace("?","");
System.out.println("Is the Person Indian? ");
boolean indian-sc.nextBoolean();
//prepare byte[] representing photo file content InputStream is=new FileInputStream(photoPath);
byte[] photoData=new byte[is.available()];
photo Data=is.readAllBytes();
//prepare char[] representing biddata file content
File file=new File(biodata Path);
Reader reader-new FileReader(file);
char bioDataContent[]=new char[(int) file.length()]; reader.read(bioDataContent);
//prepare Entity class obj
MarriageSeeker seeker=new MarriageSeeker(name, addrs, photo Data,
}
}
LocalDateTime.of(1990,11,23,12,45), bioDataContent, indian);
==========");
System.out.println(service.registerMarriageSeeker(seeker));*/
Optional<MarriageSeeker> opt-service.searchSeekerById(4L);
System.out.println(":
if(opt.isPresent()) {
MarriageSeeker seeker-opt.get();
System.out.println(seeker.getId()+" "+seeker.getName()+" "+seeker.getAddrs()+" "+seeker.isIndian()); OutputStream
os=new FileOutputStream("retrieve_photo.gfif");
os.write(seeker.getPhoto());
os.flush();
Writer writer=new FileWriter("retrieve_biodata.txt");
writer.write(seeker.getBiodata());
writer.flush();
os.close();
writer.close();
System.out.println("LOBS are retrieved");
else {
System.out.println("Records not found");
}
```

Working with Date values using java 8 Date and Time api (JODA date and time API)

===

In java 8

JODA is third party company name (joda.org) whose

api is used by jdk s/w for date and time api operations

LocalTime: To set time values and to get current time LocalDate : To set date values and to get current date LocalDateTime :: To set date and time values and to get current date and time

In oracle db s/w

[illegible]

date data type --> To store date values

timestamp data type --> to store date and time values

note: time data type is not given

In mysql Db s/w

date data type, datetime data type

note:: Storing date,time values in db table cols as the String values is not a good option becoz we can not perform arithmetic opeations on the String date, time values

with

timestamp date type.. time data type

=> Earlier (before java8) we used work java.sql.Date, java.sql.Timestamp and etc.. classes to deal with date, time values.. From Java8, we can use JODA Date -time apis.

DB table in mysql DB s/w

(No need of creating this

db table manullly will be

generated dynamically

by spring data jpa)

Table Name: employee_date_time Charset/Collation: utf8mb4

Schema: ntspbms616db

utf8mb4_0900_ai_ci

Engine:

InnoDB

Comments:

Column Name

Datatype

INT

eno

◇ desg

VARCHAR(255)

◇ dob

DATETIME(6)

◇ doj

DATE

◇ ename

VARCHAR(255)

◇ toj

TIME

PK

NN

0000000

0000000

U

90000000

L

U

0000000"

UQ B UN

ZF

30000000

0000000

○○

*0000000

AI G

Default/Expression

U NULL

U NULL

NULL

U

L NULL

U

NULL

application.properties

```
===================
```

#DataSource cfg

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql:///ntspbms714db

spring.datasource.username=root

spring.datasource.password=root

# JPA-Hiberante properties

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto-update

# other possible values create,validate,create-drop
```

```
//Employee_Date_Time.java

package com.nt.entity;

i@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

@RequiredArgsConstructor

public class Employee_Date_Time {

@GeneratedValue(strategy = GenerationType.AUTO)

@Id

private Integer eno;

@NonNull

private String ename;

@NonNull

private String desg;

@NonNull

private LocalDateTime dob;

@NonNull

private LocalTime toj;

@NonNull

private LocalDate doj;
```

service interface

```
public interface IEmployeeMgmtService {

public String saveEmployee(Employee_Date_Time dateTime);

public List<Employee_Date_Time> getAllEmployees();

public List<Integer> showEmployeeAgesByDesg(String desg);

}
```

Repository Interface

```
}
//service Impl class
=====================
>ublic interface EmployeeDateTimeRepository extends JpaRepository<Employee_Date_Time, Integer> {
@Query(nativeQuery = true, value="SELECT YEAR(CURRENT_TIMESTAMP)-YEAR(DOB) FROM
EMPLOYEE_DATE_TIME WHERE DESG=:job")
public List<Integer> getEmployeeAgesByDesg(String job);
@Component
public class DateTimeTestRunner implements Command Line Runner {
@Autowired
private IEmployeeMgmtService service;
@Override
public void run(String... args) throws Exception {
@Service("empService")
public class EmployeeMgmtServiceImpl implements IEmployeeMgmtService {
@Autowired
private Employee DateTimeRepository empRepo;
@Override
public String saveEmployee (Employee_Date_Time dateTime) {
int idVal-empRepo.save(dateTime).getEno();
return "Employee Object is saved with the id Value ::"+idVal;
}
@Override
public List<Employee_Date_Time> getAllEmployees() {
return empRepo.findAll();
}
@Override
public List<Integer> showEmployeeAgesByDesg(String desg) {
return empRepo.getEmployeeAgesByDesg(desg);
}
try {
//save the object
Employee_Date_Time emp-new Employee_Date_Time( "raja", "clerk",
LocalDateTime.of(1990, 10, 20, 11, 34),
LocalTime.of(17,45),
LocalDate.of(2020, 10, 30));
String result-service.saveEmployee(emp);
```

```
System.out.println(result);
}
catch (Exception e) {
e.printStackTrace();
}
System.out.println("+++++++++++++++++++++++++++++++++++++");
try {
service.getAllEmployees().forEach(System.out::println);
catch (Exception e) {
e.printStackTrace();
}
try {
service.showEmployeeAgesByDesg("clerk").forEach(System.out::println);
}
catch(Exception e) {
e.printStackTrace();
}
```

Assignment:: Find out the age of the Customer by using given customer id ?

```
}//run(-)
}//class
```