Versioning and TimeStamping Features

==

Versioning :: It keeps track of how many times the entity object is loaded and modified using hibernate logics directly or indirectly

=>For this we need to add @Version annotation on the top of numberic property which internally creates one numberic col in Db table whose value will be incremented by 1 for every update operation on the object. (record) card usecases :: a) keeping caller tune change count b) keeping track DOB change count in aadhar and etc..

==========

of

is

..

TimeStamping :: Keeps track of when the Object is saved and when the object lastly updated it will maintain both date and time values of when the object is saved and when the object is lastly updated. For this we need to use @Creation Timestamp and @UpdateTimestamp_on java.time.LocalDateTime type properties of an Entity class.

usecases :: useful to keep track of when Bank account is opended and lastly operated

useful to keep track of when when flipkart/gmail/.. account opened and lastly operated

note:: To implement these features in JDBC ; we need to put lots of efforts in all angels like working with triggers, event management and etc... In JPA- hibernate these are part Eco System(Built-in features)

boot

step1) create Spring starter Project using gradle adding the following starters a) spring data jpa b)lombok c) oracle driver

Impotant JPA Generators

IDENTITY ---> works only in MYSQL where it uses autoincrement constraint to generate

id value using lastValue+1 formulae SEQUENCE ---> uses the specified details to create the sequence in Oracle DB s/w

AUTO ---> uses different generators in different Db s/ws.. uses IDENTITY in MYSQL uses SEQUENCE in Oracle

✓Lombok

Spring Data JPA

Available:

MySQL Driver

Type to search dependencies

▸ Developer Tools

▸ Google Cloud Platform

Selected:

X Lombok

X Spring Data JPA

X Oracle Driver

step2) Add the following entries in application.properties file

```
#jdbc properties (for oracle)
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=tiger
spring.datasource.hikari.maximum-pool-size=100
spring.datasource.hikari.minimum-idle=10
spring.datasource.hikari.keepalive-time=100000
spring.jpa.datasource-platform=org.hibernate.dialect.Oracle 10gDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto-update
```

step3) create the following packages

```
//CallerTuneInfo.iava
package com.nt.entity;
import java.time.LocalDateTime;
import org.hibernate.annotations. Creation Timestamp;
import org.hibernate.annotations. UpdateTimestamp;
```

Oracle Driver

```
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.persistence.Version;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
@Entity
@Table(name="CALLER_TUNE_INFO")
@Data
@RequiredArgsConstructor
@NoArgsConstructor
public class CallerTuneInfo {
@Id
```

@GeneratedValue(strategy = GenerationType.AUTO) private Integer tuneId;

@Column(length = 20)

@NonNull

private String tuneName;

@Column(length = 20)

@NonNull

private String movieName;

@Version

@Column(name = "UPDATE_COUNT") private Integer updatedCount;

@Column(name = "SERVICE_OPTED_ON") @CreationTimestamp @Column(insertrue) private LocalDateTime serviceOptedOn;

For Versioning feature

For Time stamping feature

@Column(name = "LASTLY_UPDATED_ON") @UpdateTimestamp @Column(insertfalse) private LocalDateTime lastlyUpdatedOn;

For Time stamping feature

}

step4) Develop the Repository Interface

// ICallerTuneInfoRepository.java

package com.nt.repository;

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

import com.nt.entity. CallerTuneInfo;

public interface ICallerTuneInfoRepository extends JpaRepository<CallerTuneInfo, Integer> {

}

step5) develop service Interface and service Impl class

service Interface

package com.nt.service;

import java.util.Optional;

import com.nt.entity. CallerTuneInfo;

public interface ICallerTuneMgmtService {

public String saveCallerTuneInfo(CallerTuneInfo info);

public String updateTuneInfoById(Integer id, String tuneName, String movieName); public CallerTuneInfo showCallerTuneDetailsById(Integer id);

}

Service Impl class

//service Impl class

package com.nt.service;

import java.util.Optional;

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.nt.entity. CallerTuneInfo;
import com.nt.repository.ICallerTuneInfoRepository;
@Service("callerTuneService")
public class CallerTuneMgmtServiceImpl implements ICallerTuneMgmtService {
@Autowired
private ICallerTuneInfoRepository callerTuneRepo;
@Override
public String saveCallerTuneInfo(CallerTuneInfo info) {
Integer idVal-callerTuneRepo.save(info).getTuneId();
return "CallerTune is saved with the id Value::"+idVal;
}
@Override
public String updateTuneInfoById(Integer id, String tuneName, String movieName) {
Optional<CallerTuneInfo> opt=callerTuneRepo.findById(id);
if(opt.isPresent()) {
CallerTuneInfo info=opt.get();
info.setTuneName(tuneName);
info.setMovieName(movieName);
CallerTuneInfo tune=callerTuneRepo.save(info);
return "Object is updated for "+tune.getUpdatedCoun()+"times ... lastly modified on:"
+tune.getLastlyUpdatedOn()+" .... created on ::"+tune.getServiceOptedOn();
}
else {
return "CallerTuneService is not found";
}
}
@Override
public CallerTuneInfo showCallerTune DetailsById(Integer id) {
return callerTuneRepo.findById(id).orElse Throw(()->new IllegalArgumentException("caller tune not found"));
}
}
```

Runner class

```
==============
@Component
public class VersioningAndTimeStampingTest implements CommandLineRunner {
```

```
@Autowired
private ICallerTuneMgmtService service;
}
@Override
public void run(String... args) throws Exception {
/* try{
CallerTuneInfo info=new CallerTuneInfo("oo antava mama","puspha"); System.out.println(ser
.saveCallerTuneInfo(info));
}
catch(Exception e) {
e.printStackTrace();
}
System.out.println("++++++++
+++++++++");*/
try{
System.out.println(service.updateTuneInfoById(1,"joome jo pathan1","pathan"));
System.out.println(service.showCallerTune DetailsById(1));
}
catch(Exception e) {
e.printStackTrace();
}
System.out.println("++++++++++++
++++++++++++");
try{
System.out.println(service.updateTuneInfoById(1,"Natu Natu1","RRR"));
System.out.println(service.showCallerTune DetailsById(1));
}
catch(Exception e) {
e.printStackTrace();
}
}
```

Welcome Page

con1

CALLER TUNE_INFO

Columns Data Model | Constraints | Grants Statistics Triggers | Flashback |Dependencies Details Partitions | Indexes | SQL + EX Sort.. Filter:

TUNE_ID

LASTLY_UPDATED_ON

127-02-23 08:42:29.199305000 PM RRR

MOVIE_NAMESERVICE_OPTED_ON TUNE_NAME UPDATE_COUNT 27-02-23 08:40:08.537714000 PM Natu Natu1

4

8277

In real projects where O-R Mapping persistence logics are kept, we design Entity classes having two types of properites

a) Data properties (these properties carry actual Business data of the application)

b) Metata Properties (These properites carry more additional information about the records)

```
//Doctor.java
package com.nt.entity;
import java.time.LocalDateTime;
import org.hibernate.annotations.Creation Timestamp;
import org.hibernate.annotations.UpdateTimestamp;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.persistence.Transient;
import jakarta.persistence.Version;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
@Entity
@Table(name="JPA_DOCTOR_VER_TS")
@Data
@NoArgsConstructor
@AllArgsConstructor
@RequiredArgsConstructor
public class Doctor {
@Column(name="DOCTOR_ID")
@GeneratedValue(strategy = GenerationType.AUTO) //generated id values like 1,2,52 and etc..
@Id
private Integer did;
```

@NonNull

@Column(name="DOCTOR_NAME",length = 20)

private String dname;

Data

private String addrs;

properties @Column(name="DOCTOR_EXPERT", length = 20)

@Column(name="DOCTOR_ADDRS", length = 20)

In Service Interface

@NonNull

public interface IDoctorMgmtService {

public String registerDoctor(Doctor doctor);

public String modify DoctorFee(int id, double hikePercent); public Doctor showDoctorById(int id);

@NonNull

private String expert;

@Column(name="DOCTOR_FEE")

@NonNull

//@Transient

private Double fee;

MetaData

//MetaData properites

@Version private Integer updateCount;

Properties @Creation Timestamp

@Column(updatable = false,insertable = true) private LocalDateTime registeredOn; @UpdateTimestamp

@Column(insertable = false, updatable = true)

private LocalDateTime lastlyUpdatedOn; @Column(length = 30)

private String createdBy;

@Column(length = 30)

private String updatedBy;

@Column(length = 10) private String active_SW;

}

In Service Impl class

package com.nt.service; import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.data.domain.Example;

import org.springframework.data.domain.Sort;

import org.springframework.stereotype.Service;

```
import com.nt.entity.Doctor;
}
//VersioningAndTimeStampingRunner.java
package com.nt.runners;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.Command LineRunner;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Component;
import com.nt.BootJpaProj06VersioningAndTimeStamping;
import com.nt.entity.Doctor;
import com.nt.repository.IDoctorRepository;
import com.nt.service.IDoctorMgmtService;
@Component
import com.nt.repository.IDoctorRepository;
@Service("docService")
@Autowired
public class Doctor MgmtServiceImpl implements IDoctorMgmtService {
private IDoctorRepository docRepo;
@Override
public String register Doctor(Doctor doctor) {
doctor.setCreated By(System.getProperty("user.name"));
doctor.setActive_SW("active");
int idVal=docRepo.save(doctor).getDid();
return "Doctor obj is saved with id values:"+idVal;
}
@Override
public String modify DoctorFee(int id, double hike Percent) {
//Load object
if(opt.isPresent()) {
Optional<Doctor> opt=docRepo.findById(id);
Doctor doctor=opt.get();
doctor.setFee(doctor.getFee()+ doctor.getFee()*hikePercent/100.0);
doctor.setUpdatedBy(System.getProperty("user.name"));
docRepo.save(doctor);
return id+" doctor fee is updated";
```

```
}
return id+" doctor is not found for updation";
}
@Override
public Doctor showDoctorById(int id) {
return docRepo.findById(id).orElse Throw(()->new IllegalArgumentException("Invalid Id"));
}
}//class
public class VersioningAndTimeStamping TestRunner implements CommandLineRunner {
@Autowired
private IDoctor MgmtService docService;
@Override
public void run(String... args) throws Exception {
/*
try {
Doctor doctor=new Doctor("Suresh", "delhi", "cardio", 700.0);
String msg=docService.registerDoctor(doctor);
System.out.println(msg);
Doctor doc=docService.showDoctorById(1);
System.out.println("update count ::"+doc.getUpdateCount()+"inserted On:"+doc.getRegistered On());
}
catch(Exception e) {
e.printStackTrace();
}*/
try {
String msg=docService.modify DoctorFee(1, 20.0);
System.out.println(msg);
}//main
}//class
-
Doctor doc=docService.showDoctorById(1);
System.out.println("update count::"+doc.getUpdateCount()+", inserted On "+doc.getRegistered On()+", lastly
updated On"+doc.getLastlyUpdatedOn());
}
catch (Exception e) {
e.printStackTrace();
}
```

Result Grid

Filter Rows:

doctor_id

active_sw

doctor_addrs

1

active

delhi

NULL

NULL

NULL

Edit:

created_by

Nataraz

Suresh

NULL

NULL

Export/Import:

Wrap Cell Content: A

doctor_name doctor_expert

doctor_fee

lastly_updated_on

registered_on

cardio

NULL

840

2025-05-02 18:32:47.71... 2025-05-02 18:31:01.771513 1

update_count updated_by Nataraz

NULL

NULL

NULL

NULL

NULL