

```
1 =====pom.xml=====
2 <?xml version="1.0" encoding="UTF-8"?>
3 <project xmlns="http://maven.apache.org/POM/4.0.0"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6       https://maven.apache.org/xsd/maven-4.0.0.xsd">
7   <modelVersion>4.0.0</modelVersion>
8   <parent>
9     <groupId>org.springframework.boot</groupId>
10    <artifactId>spring-boot-starter-parent</artifactId>
11    <version>2.3.0.RELEASE</version>
12    <relativePath/> <!-- lookup parent from repository -->
13  </parent>
14  <groupId>com.ameya</groupId>
15  <artifactId>004-UserInfo-api-JPA-Hibernate</artifactId>
16  <version>0.0.1-SNAPSHOT</version>
17  <name>004-UserInfo-api-JPA-Hibernate</name>
18  <description>Demo project for Spring Boot</description>
19  <properties>
20    <java.version>1.8</java.version>
21  </properties>
22  <dependencies>
23    <dependency>
24      <groupId>org.springframework.boot</groupId>
25      <artifactId>spring-boot-starter-data-jpa</artifactId>
26    </dependency>
27    <dependency>
28      <groupId>org.springframework.boot</groupId>
29      <artifactId>spring-boot-starter-web</artifactId>
30    </dependency>
31    <dependency>
32      <groupId>mysql</groupId>
33      <artifactId>mysql-connector-java</artifactId>
34      <scope>runtime</scope>
35    </dependency>
36    <dependency>
37      <groupId>org.springframework.boot</groupId>
38      <artifactId>spring-boot-starter-test</artifactId>
39      <scope>test</scope>
40      <exclusions>
41        <exclusion>
42          <groupId>org.junit.vintage</groupId>
43          <artifactId>junit-vintage-engine</artifactId>
44        </exclusion>
45      </exclusions>
46    </dependency>
47  </dependencies>
```

```

47
48     <build>
49         <plugins>
50             <plugin>
51                 <groupId>org.springframework.boot</groupId>
52                 <artifactId>spring-boot-maven-plugin</artifactId>
53             </plugin>
54         </plugins>
55     </build>
56
57 </project>
58 =====application.properties=====
59
60 server.port=9001
61 logging.level.web=trace
62
63 # Database
64 db.url: jdbc:mysql://localhost:3306/sapientdb
65 db.driver: com.mysql.cj.jdbc.Driver
66 db.username: root
67 db.password: root
68
69 # Hibernate
70
71 hibernate.dialect: org.hibernate.dialect.MySQL5Dialect
72 hibernate.show_sql: true
73 hibernate.hbm2ddl.auto: update
74 entitymanager.pkgsScan: com
75
76 =====HibernateConfiguration.java=====
77
78 package com.ameya.config;
79
80 import java.util.Properties;
81
82 import javax.sql.DataSource;
83
84 import org.springframework.beans.factory.annotation.Value;
85 import org.springframework.context.annotation.Bean;
86 import org.springframework.context.annotation.Configuration;
87 import org.springframework.jdbc.datasource.DriverManagerDataSource;
88 import org.springframework.orm.hibernate5.HibernateTransactionManager;
89 import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
90 import org.springframework.transaction.annotation.EnableTransactionManagement;
91
92 @Configuration
93 @EnableTransactionManagement

```

```
93 public class HibernateConfiguration {
94
95     @Value("${db.driver}")
96     private String DB_DRIVER;
97     @Value("${db.password}")
98     private String DB_PASSWORD;
99     @Value("${db.url}")
100    private String DB_URL;
101    @Value("${db.username}")
102    private String DB_USERNAME;
103    @Value("${hibernate.dialect}")
104    private String HIBERNATE_DIALECT;
105    @Value("${hibernate.show_sql}")
106    private String HIBERNATE_SHOW_SQL;
107    @Value("${hibernate.hbm2ddl.auto}")
108    private String HBM2DDL_AUTO;
109    @Value("${entitymanager.pkgsScan}")
110    private String PACKAGES_TO_SCAN;
111
112    @Bean
113    public DataSource dataSource() {
114        DriverManagerDataSource dataSource=new DriverManagerDataSource();
115        dataSource.setDriverClassName(DB_DRIVER);
116        dataSource.setUrl(DB_URL);
117        dataSource.setUsername(DB_USERNAME);
118        dataSource.setPassword(DB_PASSWORD);
119        return dataSource;
120    }
121    @Bean
122    public LocalSessionFactoryBean sessionFactory() {
123        LocalSessionFactoryBean sessionFactory=new LocalSessionFactoryBean();
124        sessionFactory.setDataSource(dataSource());
125        sessionFactory.setPackagesToScan(PACKAGES_TO_SCAN);
126        Properties hibernateProps=new Properties();
127        hibernateProps.put("hibernate.dialect", HIBERNATE_DIALECT);
128        hibernateProps.put("hibernate.show_sql", HIBERNATE_SHOW_SQL);
129        hibernateProps.put("hibernate.hbm2ddl.auto", HBM2DDL_AUTO);
130        sessionFactory.setHibernateProperties(hibernateProps);
131        return sessionFactory;
132    }
133    @Bean
134    public HibernateTransactionManager transactionManager() {
135        HibernateTransactionManager txManager=new HibernateTransactionManager();
136        txManager.setSessionFactory(sessionFactory().getObject());
137        return txManager;
138    }
139
140 }
```

```

141 =====Application.java=====
142 =====
143
144 import org.springframework.boot.SpringApplication;
145 import org.springframework.boot.autoconfigure.SpringBootApplication;
146 import org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAutoConfiguration;
147
148 @SpringBootApplication(exclude = HibernateJpaAutoConfiguration.class)
149 public class Application {
150
151     public static void main(String[] args) {
152         SpringApplication.run(Application.class, args);
153     }
154
155 }
156 =====UserInfo.java=====
157 =====
158
159 package com.ameya.models;
160
161 import javax.persistence.Column;
162 import javax.persistence.Entity;
163 import javax.persistence.GeneratedValue;
164 import javax.persistence.GenerationType;
165 import javax.persistence.Id;
166 import javax.persistence.Table;
167
168 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
169
170 @Entity
171 @Table(name="aj_userinfo")
172 @JsonIgnoreProperties({"hibernateLazyInitializer","handler"})
173 public class UserInfo {
174
175     @Id
176     @GeneratedValue(strategy=GenerationType.AUTO)
177     private int id;
178     @Column(name="fullname")
179     private String fullName;
180     @Column(name="country")
181     private String country;
182     public UserInfo() {
183
184     }
185     public UserInfo(int id, String fullName, String country) {
186         super();
187         this.id = id;
188         this.fullName = fullName;

```

```

187         this.country = country;
188     }
189     public int getId() {
190         return id;
191     }
192     public void setId(int id) {
193         this.id = id;
194     }
195     public String getFullName() {
196         return fullName;
197     }
198     public void setFullName(String fullName) {
199         this.fullName = fullName;
200     }
201     public String getCountry() {
202         return country;
203     }
204     public void setCountry(String country) {
205         this.country = country;
206     }
207     @Override
208     public String toString() {
209         return "UserInfo [id=" + id + ", fullName=" + fullName + ", country=" +
                country + "]";
210     }
211
212 }
213 =====UserInfoDAO.java=====
214 =====
215 package com.ameya.daos;
216
217 import java.util.List;
218
219 import com.ameya.models.UserInfo;
220
221 public interface UserInfoDAO {
222     void addUser(UserInfo userInfo);
223     List<UserInfo>getAllUserInfo();
224     UserInfo findById(int id);
225     UserInfo findByIdQuery(int id);
226     UserInfo update(UserInfo userInfo,int id);
227     void delete(int id);
228 }
229 =====UserInfoDAOImpl.java=====
230 =====
231 package com.ameya.daos.impl;

```

```
232 import java.util.List;
233
234 import org.hibernate.Session;
235 import org.hibernate.SessionFactory;
236 import org.hibernate.query.Query;
237 import org.springframework.beans.factory.annotation.Autowired;
238 import org.springframework.stereotype.Repository;
239
240 import com.ameya.daos.UserInfoDAO;
241 import com.ameya.models.UserInfo;
242
243 @Repository
244 public class UserInfoDAOImpl implements UserInfoDAO {
245
246     @Autowired
247     private SessionFactory sessionFactory;
248
249     @Override
250     public void addUser(UserInfo userInfo) {
251         Session session=sessionFactory.getCurrentSession();
252         session.save(userInfo);
253     }
254
255     @Override
256     public List<UserInfo> getAllUserInfo() {
257         Session session=sessionFactory.getCurrentSession();
258         List<UserInfo> list=session.createQuery("from UserInfo").list();
259         return list;
260     }
261
262     @Override
263     public UserInfo findById(int id) {
264         Session session=sessionFactory.getCurrentSession();
265         UserInfo userInfo=(UserInfo)session.get(UserInfo.class, id);
266         return userInfo;
267     }
268
269     @Override
270     public UserInfo findByIdQuery(int id) {
271         Session session=sessionFactory.getCurrentSession();
272         Query<UserInfo> query=session.createQuery("from UserInfo where id = :id");
273         query.setParameter("id", id);
274         List<UserInfo> users=query.getResultList();
275         Query<Integer> cntQuery=session.createQuery("select count(id) from
276         UserInfo");
277         List<Integer> cntList=cntQuery.getResultList();
278         System.out.println("COUNT => "+cntList.get(0));
```

```

279         return users.get(0);
280     }
281
282     @Override
283     public UserInfo update(UserInfo userInfo, int id) {
284         Session session=sessionFactory.getCurrentSession();
285         UserInfo existingUserInfo=(UserInfo)session.load(UserInfo.class, id);
286         existingUserInfo.setFullName(userInfo.getFullName());
287         existingUserInfo.setCountry(userInfo.getCountry());
288         return existingUserInfo;
289     }
290
291     @Override
292     public void delete(int id) {
293         Session session=sessionFactory.getCurrentSession();
294         UserInfo userInfo=findById(id);
295         session.delete(userInfo);
296     }
297 }
298
299 }
300
301 /*
302 Query<Emp> query=session.createQuery("from Emp");//here persistent class name is
    Emp
303 List list=query.list();
304
305 Query<Emp> query=session.createQuery("from Emp");
306 query.setFirstResult(5);
307 query.setMaxResult(10);
308 List list=query.list();//will return the records from 5 to 10th number
309
310 Query<Integer> q=session.createQuery("update User set name=:n where id=:i");
311 q.setParameter("n","Ameya Joshi");
312 q.setParameter("i",111);
313
314 int status=q.executeUpdate();
315
316 Query<Integer> query=session.createQuery("delete from Emp where id=100");
317 query.executeUpdate();
318
319 Query<Integer> q=session.createQuery("select sum(salary) from Emp");
320 List<Integer> list=q.list();
321
322 Query<Integer> q=session.createQuery("select max(salary) from Emp");
323
324 Query<Integer> q=session.createQuery("select min(salary) from Emp");
325

```

```

326 Query<Integer> q=session.createQuery("select count(id) from Emp");
327 List<Integer> cntList=q.getResultList();
328 System.out.println("COUNT =====> "+cntList.get(0));
329
330 */
331 =====UserInfoService.java=====
332 =====
333 package com.ameya.services;
334
335 import java.util.List;
336
337 import com.ameya.models.UserInfo;
338
339 public interface UserInfoService {
340     void createUser(UserInfo userInfo);
341     List<UserInfo> findAll();
342     UserInfo findById(int id);
343     UserInfo findByIdQuery(int id);
344     UserInfo updateUser(UserInfo userInfo,int id);
345     void deleteById(int id);
346 }
347 =====UserInfoServiceImpl.java=====
348 =====
349 package com.ameya.services.impl;
350
351 import java.util.List;
352
353 import org.springframework.beans.factory.annotation.Autowired;
354 import org.springframework.stereotype.Service;
355 import org.springframework.transaction.annotation.Transactional;
356
357 import com.ameya.daos.UserInfoDAO;
358 import com.ameya.models.UserInfo;
359 import com.ameya.services.UserInfoService;
360
361 @Service
362 @Transactional
363 public class UserInfoServiceImpl implements UserInfoService {
364
365     @Autowired
366     private UserInfoDAO userInfoDao;
367
368     @Override
369     public void createUser(UserInfo userInfo) {
370         userInfoDao.addUser(userInfo);
371     }

```



```

372     @Override
373     public List<UserInfo> findAll() {
374         return userInfoDao.getAllUserInfo();
375     }
376
377     @Override
378     public UserInfo findById(int id) {
379         return userInfoDao.findById(id);
380     }
381
382     @Override
383     public UserInfo findByIdQuery(int id) {
384         return userInfoDao.findByIdQuery(id);
385     }
386
387     @Override
388     public UserInfo updateUserInfo(UserInfo userInfo, int id) {
389         return userInfoDao.update(userInfo, id);
390     }
391
392     @Override
393     public void deleteById(int id) {
394         userInfoDao.delete(id);
395     }
396 }
397
398 }
399 =====UserInfoController.java=====
400
401 package com.ameya.controllers;
402
403 import java.util.List;
404
405 import org.springframework.beans.factory.annotation.Autowired;
406 import org.springframework.http.HttpHeaders;
407 import org.springframework.http.HttpStatus;
408 import org.springframework.http.MediaType;
409 import org.springframework.http.ResponseEntity;
410 import org.springframework.web.bind.annotation.DeleteMapping;
411 import org.springframework.web.bind.annotation.GetMapping;
412 import org.springframework.web.bind.annotation.PathVariable;
413 import org.springframework.web.bind.annotation.PostMapping;
414 import org.springframework.web.bind.annotation.PutMapping;
415 import org.springframework.web.bind.annotation.RequestBody;
416 import org.springframework.web.bind.annotation.RequestMapping;
417 import org.springframework.web.bind.annotation.RestController;
418

```

```

419 import com.ameya.models.UserInfo;
420 import com.ameya.services.UserInfoService;
421
422 @RestController
423 @RequestMapping("/userinfo")
424 public class UserInfoController {
425     @Autowired
426     private UserInfoService userInfoService;
427
428     @GetMapping(path="/{id}", produces=MediaType.APPLICATION_JSON_VALUE)
429     public ResponseEntity<UserInfo> getUserById(@PathVariable("id") int id){
430         UserInfo userInfo=userInfoService.findById(id);
431         if(userInfo==null) {
432             return new ResponseEntity<UserInfo>(HttpStatus.NOT_FOUND);
433         }
434         return new ResponseEntity<UserInfo>(userInfo,HttpStatus.OK);
435     }
436
437     @GetMapping(path="/byquery/{id}", produces=MediaType.APPLICATION_JSON_VAL
438     UE)
439     public ResponseEntity<UserInfo> getUserByIdQuery(@PathVariable("id") int id){
440         UserInfo userInfo=userInfoService.findById(id);
441         if(userInfo==null) {
442             return new ResponseEntity<UserInfo>(HttpStatus.NOT_FOUND);
443         }
444         return new ResponseEntity<UserInfo>(userInfo,HttpStatus.OK);
445     }
446
447     @PostMapping(path="/create", headers="Accept=application/json")
448     public ResponseEntity<Void> createUser(@RequestBody UserInfo userInfo,
449         UriComponentsBuilder builder){
450         userInfoService.createUser(userInfo);
451         HttpHeaders headers=new HttpHeaders();
452
453         headers.setLocation(builder.path("/userinfo/{id}").buildAndExpand(userInfo.getId
454         ()).toUri());
455         return new ResponseEntity<Void>(headers,HttpStatus.CREATED);
456     }
457
458     @GetMapping("/getall")
459     public List<UserInfo> getAllUserInfo(){
460         return userInfoService.findAll();
461     }
462
463     @PutMapping(path="/update/{id}")
464     public ResponseEntity<String> updateUserInfo(@RequestBody UserInfo
465     currentUserInfo,@PathVariable("id") int id){

```

```

462     UserInfo userInfo=userInfoService.findById(id);
463     if(userInfo==null) {
464         return new ResponseEntity<String>("User Not
           Found",HttpStatus.NOT_FOUND);
465     }
466     userInfoService.updateUserInfo(currentUserInfo, id);
467     return new ResponseEntity<String>("User Updated",HttpStatus.OK);
468 }
469
470 @DeleteMapping(path="/delete/{id}")
471 public ResponseEntity<String> deleteUserInfo(@PathVariable("id") int id){
472     UserInfo userInfo=userInfoService.findById(id);
473     if(userInfo==null) {
474         return new ResponseEntity<String>("User Not
           Found",HttpStatus.NOT_FOUND);
475     }
476     userInfoService.deleteById(id);
477     return new ResponseEntity<String>("User Deleted",HttpStatus.NO_CONTENT);
478 }
479 }
480 =====
481
482
483 http://localhost:9001/userinfo/create
484 {
485     "fullName": "Ameya Joshi",
486     "country": "Germany"
487 }
488 http://localhost:9001/userinfo/delete/4
489 http://localhost:9001/userinfo/update/3
490 http://localhost:9001/userinfo/3
491 http://localhost:9001/userinfo/byquery/3
492 http://localhost:9001/userinfo/getall
493
494 ++++++Relationships
495 ++++++
496
497 =====Employee.java=====
498
499
500 import javax.persistence.CascadeType;
501 import javax.persistence.Column;
502 import javax.persistence.Entity;
503 import javax.persistence.GeneratedValue;
504 import javax.persistence.GenerationType;

```

```

505 import javax.persistence.Id;
506 import javax.persistence.OneToOne;
507 import javax.persistence.PrimaryKeyJoinColumn;
508 import javax.persistence.Table;
509
510 @Entity
511 @Table(name="aj_employee")
512 public class Employee {
513
514     @Id
515     @GeneratedValue(strategy=GenerationType.AUTO)
516     @Column(name="id")
517     private int id;
518     @Column(name="name")
519     private String name;
520     @OneToOne(targetEntity=Address.class, cascade = CascadeType.ALL)
521     private Address address;
522     public Employee() {
523         super();
524         // TODO Auto-generated constructor stub
525     }
526     public int getId() {
527         return id;
528     }
529     public void setId(int id) {
530         this.id = id;
531     }
532     public String getName() {
533         return name;
534     }
535     public void setName(String name) {
536         this.name = name;
537     }
538     public Address getAddress() {
539         //address.setEmpId(getId());
540         return address;
541     }
542     public void setAddress(Address address) {
543         this.address = address;
544     }
545 }
546
547
548 =====Address.java=====
549
549 package com.ameya.models;
550
551 import javax.persistence.Entity;

```

```
552 import javax.persistence.GeneratedValue;
553 import javax.persistence.GenerationType;
554 import javax.persistence.Id;
555 import javax.persistence.OneToOne;
556 import javax.persistence.PrimaryKeyJoinColumn;
557 import javax.persistence.Table;
558
559 import org.hibernate.annotations.GenericGenerator;
560 import org.hibernate.annotations.Parameter;
561
562 @Entity
563 @Table(name="aj_address")
564 public class Address {
565
566     @Id
567     @GeneratedValue(generator="gen")
568     @GenericGenerator(name="gen", strategy="foreign",
569         parameters=@Parameter(name="property",value="employee"))
570     private int id;
571
572     @OneToOne(targetEntity = Employee.class)
573     @PrimaryKeyJoinColumn
574     private Employee employee;
575     private String address;
576     private String country;
577     public Address() {
578         super();
579         // TODO Auto-generated constructor stub
580     }
581     public int getId() {
582         return id;
583     }
584     public void setId(int id) {
585         this.id = id;
586     }
587
588
589     public Employee getEmployee() {
590         return employee;
591     }
592     public void setEmployee(Employee employee) {
593         this.employee = employee;
594     }
595     public String getAddress() {
596         return address;
597     }
598     public void setAddress(String address) {
599         this.address = address;
```

```

600     }
601     public String getCountry() {
602         return country;
603     }
604     public void setCountry(String country) {
605         this.country = country;
606     }
607
608 }
609 ===== + Tables Created as below +
610 =====
611 CREATE TABLE `aj_employee` (
612     `id` INT(11) NOT NULL,
613     `name` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
614     `address_id` INT(11) NULL DEFAULT NULL,
615     PRIMARY KEY (`id`) USING BTREE,
616     INDEX `FKey4gc58msqeklskis35o7f81w` (`address_id`) USING BTREE,
617     CONSTRAINT `FKey4gc58msqeklskis35o7f81w` FOREIGN KEY (`address_id`)
618     REFERENCES `sapientdb`.`aj_address` (`id`) ON UPDATE RESTRICT ON
619     DELETE RESTRICT
620 )
621 COLLATE='latin1_swedish_ci'
622 ENGINE=InnoDB
623 ;
624
625 CREATE TABLE `aj_address` (
626     `id` INT(11) NOT NULL,
627     `address` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
628     `country` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
629     PRIMARY KEY (`id`) USING BTREE
630 )
631 COLLATE='latin1_swedish_ci'
632 ENGINE=InnoDB
633 ;
634 =====
635 =====
636
637 =====Group.java=====
638 ==
639 package com.ameya.models;
640
641 import java.util.List;
642
643 import javax.persistence.CascadeType;
644 import javax.persistence.Entity;
645 import javax.persistence.FetchType;
646 import javax.persistence.GeneratedValue;

```

```

643 import javax.persistence.GenerationType;
644 import javax.persistence.Id;
645 import javax.persistence.JoinColumn;
646 import javax.persistence.OneToOne;
647 import javax.persistence.OrderColumn;
648 import javax.persistence.Table;
649
650 @Entity
651 @Table(name="aj_group")
652 public class Group {
653     @Id
654     @GeneratedValue(strategy=GenerationType.AUTO)
655     private int id;
656     private String name;
657     @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
658     @OrderColumn(name="listIdx")
659     @JoinColumn(name="gr_id")
660     private List<Story> stories;
661     public Group() {}
662     public int getId() {
663         return id;
664     }
665     public void setId(int id) {
666         this.id = id;
667     }
668     public String getName() {
669         return name;
670     }
671     public void setName(String name) {
672         this.name = name;
673     }
674     public List<Story> getStories() {
675         return stories;
676     }
677     public void setStories(List<Story> stories) {
678         this.stories = stories;
679     }
680 }
681 =====Story.java=====
682 package com.ameya.models;
683
684 import javax.persistence.Entity;
685 import javax.persistence.GeneratedValue;
686 import javax.persistence.GenerationType;
687 import javax.persistence.Id;
688 import javax.persistence.JoinColumn;
689 import javax.persistence.ManyToOne;
690 import javax.persistence.Table;

```

```

691
692 @Entity
693 @Table(name="aj_story")
694 public class Story {
695     @Id
696     @GeneratedValue(strategy=GenerationType.AUTO)
697     private int id;
698     private String info;
699     @ManyToOne
700     @JoinColumn(name="gr_id")
701     private Group group;
702     public Story() {}
703     public int getId() {
704         return id;
705     }
706     public void setId(int id) {
707         this.id = id;
708     }
709     public String getInfo() {
710         return info;
711     }
712     public void setInfo(String info) {
713         this.info = info;
714     }
715     public Group getGroup() {
716         return group;
717     }
718     public void setGroup(Group group) {
719         this.group = group;
720     }
721 }
722 =====+ Tables created as below
723 +=====
724 CREATE TABLE `aj_group` (
725     `id` INT(11) NOT NULL,
726     `name` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
727     PRIMARY KEY (`id`) USING BTREE
728 )
729 COLLATE='latin1_swedish_ci'
730 ENGINE=InnoDB
731 ;
732 CREATE TABLE `aj_story` (
733     `id` INT(11) NOT NULL,
734     `info` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
735     `gr_id` INT(11) NULL DEFAULT NULL,
736     `listIdx` INT(11) NULL DEFAULT NULL,
737     PRIMARY KEY (`id`) USING BTREE,

```



```

738 INDEX `FKmjkud1gmd4b6d02xnskmx2xwc` (`gr_id`) USING BTREE,
739 CONSTRAINT `FKmjkud1gmd4b6d02xnskmx2xwc` FOREIGN KEY (`gr_id`)
REFERENCES `sapientdb`.`aj_group` (`id`) ON UPDATE RESTRICT ON DELETE
RESTRICT
740 )
741 COLLATE='latin1_swedish_ci'
742 ENGINE=InnoDB
743 ;
744 =====
745 =====Author.java=====
746 package com.ameya.models;
747
748 import java.util.Set;
749
750 import javax.persistence.CascadeType;
751 import javax.persistence.Entity;
752 import javax.persistence.GeneratedValue;
753 import javax.persistence.GenerationType;
754 import javax.persistence.Id;
755 import javax.persistence.JoinColumn;
756 import javax.persistence.JoinTable;
757 import javax.persistence.ManyToMany;
758 import javax.persistence.Table;
759
760 @Entity
761 @Table(name="aj_authors")
762 public class Author {
763     @Id
764     @GeneratedValue(strategy=GenerationType.AUTO)
765     private int id;
766     private String authorName;
767     @ManyToMany(cascade=CascadeType.ALL)
768     @JoinTable(
769         name="aj_author_book",
770         joinColumns= {@JoinColumn(name="author_id")},
771         inverseJoinColumns= {@JoinColumn(name="book_id")}
772     )
773     private Set<Book> books;
774     public Author() {}
775     public int getId() {
776         return id;
777     }
778     public void setId(int id) {
779         this.id = id;
780     }
781     public String getAuthorName() {

```

```

782         return authorName;
783     }
784     public void setAuthorName(String authorName) {
785         this.authorName = authorName;
786     }
787     public Set<Book> getBooks() {
788         return books;
789     }
790     public void setBooks(Set<Book> books) {
791         this.books = books;
792     }
793 }
794 =====Book.java=====
795 package com.ameya.models;
796
797 import java.util.Set;
798
799 import javax.persistence.CascadeType;
800 import javax.persistence.Entity;
801 import javax.persistence.GeneratedValue;
802 import javax.persistence.GenerationType;
803 import javax.persistence.Id;
804 import javax.persistence.JoinColumn;
805 import javax.persistence.JoinTable;
806 import javax.persistence.ManyToMany;
807 import javax.persistence.Table;
808
809 @Entity
810 @Table(name="aj_books")
811 public class Book {
812     @Id
813     @GeneratedValue(strategy = GenerationType.AUTO)
814     private int id;
815     private String bookName;
816     @ManyToMany(cascade = CascadeType.ALL)
817     @JoinTable(
818         name="aj_author_book",
819         joinColumns = {@JoinColumn(name="book_id")},
820         inverseJoinColumns = {@JoinColumn(name="author_id")})
821     )
822     private Set<Author> authors;
823     public Book() {}
824     public int getId() {
825         return id;
826     }
827     public void setId(int id) {
828         this.id = id;
829     }

```

```

830     public String getBookName() {
831         return bookName;
832     }
833     public void setBookName(String bookName) {
834         this.bookName = bookName;
835     }
836     public Set<Author> getAuthors() {
837         return authors;
838     }
839     public void setAuthors(Set<Author> authors) {
840         this.authors = authors;
841     }
842 }
843 =====+ Tables created as below
844 +=====
845 CREATE TABLE `aj_authors` (
846     `id` INT(11) NOT NULL,
847     `authorName` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
848     PRIMARY KEY (`id`) USING BTREE
849 )
850 COLLATE='latin1_swedish_ci'
851 ENGINE=InnoDB
852 ;
853 CREATE TABLE `aj_books` (
854     `id` INT(11) NOT NULL,
855     `bookName` VARCHAR(255) NULL DEFAULT NULL COLLATE 'latin1_swedish_ci',
856     PRIMARY KEY (`id`) USING BTREE
857 )
858 COLLATE='latin1_swedish_ci'
859 ENGINE=InnoDB
860 ;
861 CREATE TABLE `aj_author_book` (
862     `book_id` INT(11) NOT NULL,
863     `author_id` INT(11) NOT NULL,
864     PRIMARY KEY (`author_id`, `book_id`) USING BTREE,
865     INDEX `FK2yv8pkohjaj7gijby3mrvup2p` (`book_id`) USING BTREE,
866     CONSTRAINT `FK2yv8pkohjaj7gijby3mrvup2p` FOREIGN KEY (`book_id`)
867     REFERENCES `sapientdb`.`aj_books` (`id`) ON UPDATE RESTRICT ON DELETE
868     RESTRICT,
869     CONSTRAINT `FKafyj8djplfr8j5g2wkollcge` FOREIGN KEY (`author_id`)
870     REFERENCES `sapientdb`.`aj_authors` (`id`) ON UPDATE RESTRICT ON
871     DELETE RESTRICT
872 )
873 COLLATE='latin1_swedish_ci'
874 ENGINE=InnoDB
875 ;

```

```

873 =====RelationShipDAOImpl.java=====
874 package com.ameya.daos.impl;
875
876 import java.io.Serializable;
877 import java.util.ArrayList;
878 import java.util.HashSet;
879 import java.util.List;
880 import java.util.Set;
881
882 import org.hibernate.Session;
883 import org.hibernate.SessionFactory;
884 import org.hibernate.Transaction;
885 import org.hibernate.query.Query;
886 import org.springframework.beans.factory.annotation.Autowired;
887 import org.springframework.stereotype.Repository;
888
889 import com.ameya.models.Address;
890 import com.ameya.models.Author;
891 import com.ameya.models.Book;
892 import com.ameya.models.Employee;
893 import com.ameya.models.Group;
894 import com.ameya.models.Story;
895
896 @Repository
897 public class RelationShipDAOImpl {
898
899     @Autowired
900     private SessionFactory sessionFactory;
901     public void testOneToOne() {
902
903         Session sess=sessionFactory.openSession();
904         Transaction tr=sess.beginTransaction();
905         Employee emp=null;
906         Address a1=null;
907         emp=new Employee();
908         emp.setName("Ameya Joshi");
909         a1=new Address();
910         a1.setAddress("Kothrud, Pune");
911         a1.setCountry("India");
912         emp.setAddress(a1);
913         a1.setEmployee(emp);
914         sess.save(emp);
915         sess.flush();
916         Query<Employee> query=sess.createQuery("from Employee");
917         List<Employee> list=query.list();
918         for(Employee e : list) {
919             System.out.println(e.getId()+" : "+e.getName()+" :

```

```

        "+e.getAddress().getAddress()+" : "+e.getAddress().getCountry());
920     }
921     Query<Address> query1=sess.createQuery("from Address");
922     List<Address> list1=query1.list();
923     for(Address e : list1) {
924         System.out.println(e.getAddress()+" : "+e.getCountry()+" :
        "+e.getEmployee().getId()+" : "+e.getEmployee().getName());
925     }
926     tr.commit();
927     sess.close();
928 }
929 public void testOneToMany() {
930     Session sess=sessionFactory.openSession();
931     Transaction tr=sess.beginTransaction();
932     Group group=new Group();
933     group.setName("SPORTS");
934     ArrayList<Story> stories=new ArrayList<>();
935
936     Story s1=new Story();
937     s1.setInfo("The Allegations- Life Of a Player");
938     stories.add(s1);
939     Story s2=new Story();
940     s2.setInfo("The Cancer Of match Fixing");
941     stories.add(s2);
942     Story s3=new Story();
943     s3.setInfo("The Master Blaster - Sachin");
944     stories.add(s3);
945
946     group.setStories(stories);
947
948     Serializable id=sess.save(group);
949     sess.flush();
950     Group g=(Group) sess.load(Group.class, id);
951     System.out.println("GROUP ID :: "+g.getId()+" GROUP NAME ::
        "+g.getName());
952     List<Story> groupStories=g.getStories();
953     System.out.println("STORIES :: ");
954     for(Story story : groupStories) {
955         System.out.println(story.getId()+" : "+story.getInfo());
956     }
957     tr.commit();
958     sess.close();
959 }
960 public void testManyToOne() {
961     Session sess=sessionFactory.openSession();
962     Story story =(Story)sess.load(Story.class, 5);
963     System.out.println(story.getId()+" : "+story.getInfo());
964     Group group=story.getGroup();

```

```
965     System.out.println(group.getId()+" : "+group.getName());
966     sess.close();
967 }
968 public void testManyToMany() {
969     Session sess=sessionFactory.openSession();
970     Transaction tr=sess.beginTransaction();
971
972     Author a1=new Author();
973     a1.setAuthorName("Sachin");
974
975     Book b1=new Book();
976     b1.setBookName("My First Pakistan Tour");
977     Book b2=new Book();
978     b2.setBookName("My Cricket - My Life");
979     Book b3=new Book();
980     b3.setBookName("The Unforgettable Don - Don Bradman");
981     Book b4=new Book();
982     b4.setBookName("WC-2011 The Fantastic Experience");
983     Set<Book> books=new HashSet<>();
984     books.add(b1);
985     books.add(b2);
986     books.add(b3);
987     books.add(b4);
988
989     a1.setBooks(books);
990
991     Serializable authorId=sess.save(a1);
992     sess.flush();
993
994     Book b5=new Book();
995     b5.setBookName("The Wonderful Experiences From Cricketing Life");
996
997     Set<Author> authors=new HashSet<>();
998     Author a2=new Author();
999     a2.setAuthorName("Viru");
1000    Author a3=new Author();
1001    a3.setAuthorName("Zaheer");
1002    Author a4=new Author();
1003    a4.setAuthorName("Sunny G");
1004    authors.add(a2);
1005    authors.add(a3);
1006    authors.add(a4);
1007    authors.add(a1);
1008    b5.setAuthors(authors);
1009
1010    Serializable bookId=sess.save(b5);
1011
1012    Author a=(Author)sess.get(Author.class, authorId);
```

```

1013     System.out.println(a.getId()+" : "+a.getAuthorName());
1014     Set<Book> bookSet=a.getBooks();
1015     for(Book b : bookSet) {
1016         System.out.println(b.getId()+" : "+b.getBookName());
1017     }
1018     System.out.println("+++++++");
1019
1020     Book bk=(Book)sess.get(Book.class,bookId);
1021     System.out.println(bk.getId()+" : "+bk.getBookName());
1022     Set<Author> authorSet=bk.getAuthors();
1023     for(Author at : authorSet) {
1024         System.out.println(at.getId()+" : "+at.getAuthorName());
1025     }
1026
1027     tr.commit();
1028     sess.close();
1029 }
1030 }
1031 =====RelationshipServiceImpl.java=====
1032
1033 package com.ameya.services.impl;
1034
1035 import org.springframework.beans.factory.annotation.Autowired;
1036 import org.springframework.stereotype.Service;
1037
1038 import com.ameya.daos.impl.RelationshipDAOImpl;
1039
1040 @Service
1041 public class RelationshipServiceImpl {
1042
1043     @Autowired
1044     private RelationshipDAOImpl dao;
1045
1046     public void testOneToOne() {
1047         dao.testOneToOne();
1048     }
1049     public void testOneToMany() {
1050         dao.testOneToMany();
1051     }
1052     public void testManyToMany() {
1053         dao.testManyToMany();
1054     }
1055     public void testManyToOne() {
1056         dao.testManyToOne();
1057     }
1058 }
1059 =====RelationshipController.java=====
1060

```

```
1059 package com.ameya.controllers;
1060
1061 import org.springframework.beans.factory.annotation.Autowired;
1062 import org.springframework.web.bind.annotation.GetMapping;
1063 import org.springframework.web.bind.annotation.RequestMapping;
1064 import org.springframework.web.bind.annotation.RestController;
1065
1066 import com.ameya.services.impl.RelationshipServiceImpl;
1067
1068 @RestController
1069 @RequestMapping("/relations")
1070 public class RelationshipController {
1071
1072     @Autowired
1073     private RelationshipServiceImpl service;
1074     @GetMapping(path="/onetoone")
1075     public void testOneToOne() {
1076         service.testOneToOne();
1077     }
1078     @GetMapping(path="/onetomany")
1079     public void testOneToMany() {
1080         service.testOneToMany();
1081     }
1082     @GetMapping(path="/manytoone")
1083     public void testManyToOne() {
1084         service.testManyToOne();
1085     }
1086     @GetMapping(path="/manytomany")
1087     public void testManyToMany() {
1088         service.testManyToMany();
1089     }
1090 }
1091 =====
1092
1093 http://localhost:9001/relations/manytomany
1094
```