

```

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>
48
49  <build>
50    <plugins>
51      <plugin>
52        <groupId>org.springframework.boot</groupId>
53        <artifactId>spring-boot-maven-plugin</artifactId>
54      </plugin>
55    </plugins>
56  </build>
57
58 </project>
59 =====application.properties=====
60 server.port=9001
61 logging.level.web=trace
62
63 # Database
64 db.url: jdbc:mysql://localhost:3306/nineleapsdb
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 =====HibernateConfiguration.java=====
76 package com.ameya.config;
77
78 import java.util.Properties;
79
80 import javax.sql.DataSource;
81
82 import org.springframework.beans.factory.annotation.Value;
83 import org.springframework.context.annotation.Bean;
84 import org.springframework.context.annotation.Configuration;
85 import org.springframework.jdbc.datasource.DriverManagerDataSource;
86 import org.springframework.orm.hibernate5.HibernateTransactionManager;
87 import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
88 import org.springframework.transaction.annotation.EnableTransactionManagement;
89
90
91 @Configuration
92 @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 package com.ameya;
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 =====UserInfo.java=====
156 package com.ameya.models;
157
158 import javax.persistence.Column;
159 import javax.persistence.Entity;
160 import javax.persistence.GeneratedValue;
161 import javax.persistence.GenerationType;
162 import javax.persistence.Id;
163 import javax.persistence.Table;
164
165 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
166
167 @Entity
168 @Table(name="aj_userinfo")
169 @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
170 public class UserInfo {
171
172     @Id
173     @GeneratedValue(strategy=GenerationType.AUTO)
174     private int id;
175     @Column(name="fullname")
176     private String fullName;
177     @Column(name="country")
178     private String country;
179     public UserInfo() {
180
181     }
182     public UserInfo(int id, String fullName, String country) {
183         super();
184         this.id = id;
185         this.fullName = fullName;
186         this.country = country;
187     }
188     public int getId() {
189         return id;
190     }
191     public void setId(int id) {
192         this.id = id;
193     }
194     public String getFullName() {
195         return fullName;
196     }
197     public void setFullName(String fullName) {
198         this.fullName = fullName;
199     }
200     public String getCountry() {
201         return country;
202     }
203     public void setCountry(String country) {
204         this.country = country;
205     }
206     @Override
207     public String toString() {
208         return "UserInfo [id=" + id + ", fullName=" + fullName + ", country=" +
209             country + "]";
210     }
211 }
212 =====UserInfoDAO.java=====
213 =====
214 package com.ameya.daos;
215

```

```
216 import java.util.List;
217
218 import com.ameya.models.UserInfo;
219
220 public interface UserInfoDAO {
221     void addUser(UserInfo userInfo);
222     List<UserInfo> getAllUserInfo();
223     UserInfo findById(int id);
224     UserInfo findByIdQuery(int id);
225     UserInfo update(UserInfo userInfo, int id);
226     void delete(int id);
227 }
228 ======UserInfoDAOImpl.java=====
229 =====
230 package com.ameya.daos.impl;
231
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 UserInfo");
276         List<Integer> cntList=cntQuery.getResultList();
277         System.out.println("COUNT => "+cntList.get(0));
278         return users.get(0);
279     }
280
281     @Override
282     public UserInfo update(UserInfo userInfo, int id) {
283         Session session=sessionFactory.getCurrentSession();
284         UserInfo existingUserInfo=(UserInfo)session.load(UserInfo.class, id);
285         existingUserInfo.setFullName(userInfo.getFullName());
286         existingUserInfo.setCountry(userInfo.getCountry());
287 }
```

```

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  Query<Emp> query=session.createQuery("from Emp");//here persistent class name is
301  Emp
302  List list=query.list();
303
304  Query<Emp> query=session.createQuery("from Emp");
305  query.setFirstResult(5);
306  query.setMaxResult(10);
307  List list=query.list();//will return the records from 5 to 10th number
308
309  Query<Integer> q=session.createQuery("update User set name=:n where id=:i");
310  q.setParameter("n","Udit Kumar");
311  q.setParameter("i",111);
312
313  int status=q.executeUpdate();
314
315  Query<Integer> query=session.createQuery("delete from Emp where id=100");
316  query.executeUpdate();
317
318  Query<Integer> q=session.createQuery("select sum(salary) from Emp");
319  List<Integer> list=q.list();
320
321  Query<Integer> q=session.createQuery("select max(salary) from Emp");
322
323  Query<Integer> q=session.createQuery("select min(salary) from Emp");
324
325  Query<Integer> q=session.createQuery("select count(id) from Emp");
326  List<Integer> cntList=q.getResultList();
327  System.out.println("COUNT =====> "+cntList.get(0));
328
329 */
330 =====User信息服务.java=====
331 =====
332 package com.ameya.services;
333
334 import java.util.List;
335
336 import com.ameya.models.UserInfo;
337
338 public interface UserInfoService {
339     void createUser(UserInfo userInfo);
340     List<UserInfo> findAll();
341     UserInfo findById(int id);
342     UserInfo findByIdQuery(int id);
343     UserInfo updateUserInfo(UserInfo userInfo,int id);
344     void deleteById(int id);
345 }
346 =====User信息服务实现.java=====
347 package com.ameya.services.impl;
348
349 import java.util.List;
350
351 import org.springframework.beans.factory.annotation.Autowired;
352 import org.springframework.stereotype.Service;
353 import org.springframework.transaction.annotation.Transactional;
354
355 import com.ameya.daos.UserInfoDAO;
356 import com.ameya.models.UserInfo;
357 import com.ameya.services.UserInfoService;

```

```

358
359     @Service
360     @Transactional
361     public class UserInfoServiceImpl implements UserInfoService {
362
363         @Autowired
364         private UserInfoDAO userInfoDao;
365
366         @Override
367         public void createUser(UserInfo userInfo) {
368             userInfoDao.addUser(userInfo);
369         }
370
371         @Override
372         public List<UserInfo> findAll() {
373             return userInfoDao.getAllUserInfo();
374         }
375
376         @Override
377         public UserInfo findById(int id) {
378             return userInfoDao.findById(id);
379         }
380
381         @Override
382         public UserInfo findByIdQuery(int id) {
383             return userInfoDao.findByIdQuery(id);
384         }
385
386         @Override
387         public UserInfo updateUserInfo(UserInfo userInfo, int id) {
388             return userInfoDao.update(userInfo, id);
389         }
390
391         @Override
392         public void deleteById(int id) {
393             userInfoDao.delete(id);
394
395         }
396
397     }
398 ======UserInfoController.java=====
399 =
400 package com.ameya.controllers;
401
402 import java.util.List;
403
404 import org.springframework.beans.factory.annotation.Autowired;
405 import org.springframework.http.HttpHeaders;
406 import org.springframework.http.HttpStatus;
407 import org.springframework.http.MediaType;
408 import org.springframework.http.ResponseEntity;
409 import org.springframework.web.bind.annotation.DeleteMapping;
410 import org.springframework.web.bind.annotation.GetMapping;
411 import org.springframework.web.bind.annotation.PathVariable;
412 import org.springframework.web.bind.annotation.PostMapping;
413 import org.springframework.web.bind.annotation.PutMapping;
414 import org.springframework.web.bind.annotation.RequestBody;
415 import org.springframework.web.bind.annotation.RequestMapping;
416 import org.springframework.web.bind.annotation.RestController;
417 import org.springframework.web.util.UriComponentsBuilder;
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> getUserId(@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_VALUE)
438 public ResponseEntity<UserInfo> getUserByIdQuery(@PathVariable("id") int id){
439     UserInfo userInfo=userInfoService.findById(id);
440     if(userInfo==null) {
441         return new ResponseEntity<UserInfo>(HttpStatus.NOT_FOUND);
442     }
443     return new ResponseEntity<UserInfo>(userInfo,HttpStatus.OK);
444 }
445
446 @PostMapping(path="/create",headers="Accept=application/json")
447 public ResponseEntity<Void> createUser(@RequestBody UserInfo userInfo,
448                                         UriComponentsBuilder builder){
449     userInfoService.createUser(userInfo);
450     HttpHeaders headers=new HttpHeaders();
451
452     headers.setLocation(builder.path("/userinfo/{id}").buildAndExpand(userInfo.getId())
453                         .toUri());
454     return new ResponseEntity<Void>(headers,HttpStatus.CREATED);
455 }
456
457 @GetMapping("/getall")
458 public List<UserInfo> getAllUserInfo(){
459     return userInfoService.findAll();
460 }
461
462 @PutMapping(path="/update/{id}")
463 public ResponseEntity<String> updateUserInfo(@RequestBody UserInfo
464                                                 currentUserInfo,@PathVariable("id") int id){
465     UserInfo userInfo=userInfoService.findById(id);
466     if(userInfo==null) {
467         return new ResponseEntity<String>("User Not Found",HttpStatus.NOT_FOUND);
468     }
469     userInfoService.updateUserInfo(currentUserInfo, id);
470     return new ResponseEntity<String>("User Updated",HttpStatus.OK);
471 }
472
473 @DeleteMapping(path="/delete/{id}")
474 public ResponseEntity<String> deleteUserInfo(@PathVariable("id") int id){
475     UserInfo userInfo=userInfoService.findById(id);
476     if(userInfo==null) {
477         return new ResponseEntity<String>("User Not Found",HttpStatus.NOT_FOUND);
478     }
479     userInfoService.deleteById(id);
480     return new ResponseEntity<String>("User Deleted",HttpStatus.NO_CONTENT);
481 }
482 =====
483
484 http://localhost:9001/userinfo/create
485 {
486     "fullName": "Sanjay Joshi",
487     "country": "Germany"
488 }
489 http://localhost:9001/userinfo/delete/4
490 http://localhost:9001/userinfo/update/3
491 http://localhost:9001/userinfo/3
492 http://localhost:9001/userinfo/getall
493 ++++++RelationShips
494 ++++++
495 ++++++
496 =====Employee.java=====
497 ==

```

```
497 package com.ameya.models;
498
499 import javax.persistence.CascadeType;
500 import javax.persistence.Column;
501 import javax.persistence.Entity;
502 import javax.persistence.GeneratedValue;
503 import javax.persistence.GenerationType;
504 import javax.persistence.Id;
505 import javax.persistence.OneToOne;
506 import javax.persistence.PrimaryKeyJoinColumn;
507 import javax.persistence.Table;
508
509 @Entity
510 @Table(name="aj_employee")
511 public class Employee {
512
513     @Id
514     @GeneratedValue(strategy=GenerationType.AUTO)
515     @Column(name="id")
516     @PrimaryKeyJoinColumn
517     private int id;
518
519     @Column(name="name")
520     private String name;
521
522     @OneToOne(targetEntity = Address.class, cascade = CascadeType.ALL)
523     private Address address;
524
525     public Employee() {
526
527         }
528
529     public int getId() {
530         return id;
531     }
532
533     public void setId(int id) {
534         this.id = id;
535     }
536
537     public String getName() {
538         return name;
539     }
540
541     public void setName(String name) {
542         this.name = name;
543     }
544
545     public Address getAddress() {
546         address.setEmpId(getId());
547         return address;
548     }
549
550     public void setAddress(Address address) {
551         this.address = address;
552     }
553
554 }
555 =====Address.java=====
556 package com.ameya.models;
557
558 import javax.persistence.Entity;
559 import javax.persistence.GeneratedValue;
560 import javax.persistence.GenerationType;
561 import javax.persistence.Id;
562 import javax.persistence.OneToOne;
563 import javax.persistence.Table;
564
565 import org.hibernate.annotations.GenericGenerator;
566 import org.hibernate.annotations.Parameter;
567
568 @Entity
569 @Table(name="aj_address")
570 public class Address {
571
572     @Id
573     @GeneratedValue(strategy=GenerationType.AUTO)
574     private int id;
575
576     @GeneratedValue(generator="gen")
577     @GenericGenerator(name="gen", strategy="foreign",
578     parameters=@Parameter(name="property", value="employee"))
579 }
```

```

570     private int empId;
571     private String address;
572     private String country;
573     @OneToOne(targetEntity = Employee.class)
574     private Employee employee;
575     public Address() {
576
577     }
578     public int getId() {
579         return id;
580     }
581     public void setId(int id) {
582         this.id = id;
583     }
584     public int getEmpId() {
585         return empId;
586     }
587     public void setEmpId(int empId) {
588         this.empId = empId;
589     }
590     public String getAddress() {
591         return address;
592     }
593     public void setAddress(String address) {
594         this.address = address;
595     }
596     public String getCountry() {
597         return country;
598     }
599     public void setCountry(String country) {
600         this.country = country;
601     }
602     public Employee getEmployee() {
603         return employee;
604     }
605     public void setEmployee(Employee employee) {
606         this.employee = employee;
607     }
608 }
609 =====Group.java=====
610 package com.ameya.models;
611
612 import java.util.List;
613
614 import javax.persistence.CascadeType;
615 import javax.persistence.Entity;
616 import javax.persistence.FetchType;
617 import javax.persistence.GeneratedValue;
618 import javax.persistence.GenerationType;
619 import javax.persistence.Id;
620 import javax.persistence.JoinColumn;
621 import javax.persistence.OneToMany;
622 import javax.persistence.OrderColumn;
623 import javax.persistence.Table;
624
625 @Entity
626 @Table(name="aj_group")
627 public class Group {
628
629     @Id
630     @GeneratedValue(strategy=GenerationType.AUTO)
631     private int id;
632     private String name;
633     @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
634     @OrderColumn(name="listIdx")
635     @JoinColumn(name="gr_id")
636     private List<Story> stories;
637     public Group() {
638
639     }
640     public int getId() {
641         return id;
642     }

```

```
643     }
644     public void setId(int id) {
645         this.id = id;
646     }
647     public String getName() {
648         return name;
649     }
650     public void setName(String name) {
651         this.name = name;
652     }
653     public List<Story> getStories() {
654         return stories;
655     }
656     public void setStories(List<Story> stories) {
657         this.stories = stories;
658     }
659
660 }
661 =====Story.java=====
662 package com.ameya.models;
663
664 import javax.persistence.Entity;
665 import javax.persistence.GeneratedValue;
666 import javax.persistence.GenerationType;
667 import javax.persistence.Id;
668 import javax.persistence.JoinColumn;
669 import javax.persistence.ManyToOne;
670 import javax.persistence.Table;
671
672 @Entity
673 @Table(name="aj_story")
674 public class Story {
675
676     @Id
677     @GeneratedValue(strategy=GenerationType.AUTO)
678     private int id;
679     private String info;
680     @ManyToOne
681     @JoinColumn(name="gr_id")
682     private Group group;
683     public Story() {
684
685     }
686     public int getId() {
687         return id;
688     }
689     public void setId(int id) {
690         this.id = id;
691     }
692     public String getInfo() {
693         return info;
694     }
695     public void setInfo(String info) {
696         this.info = info;
697     }
698     public Group getGroup() {
699         return group;
700     }
701     public void setGroup(Group group) {
702         this.group = group;
703     }
704 }
705
706 }
707 =====Author.java=====
708 package com.ameya.models;
709
710 import java.util.Set;
711
712 import javax.persistence.CascadeType;
713 import javax.persistence.Entity;
714 import javax.persistence.GeneratedValue;
715 import javax.persistence.GenerationType;
```

```
716 import javax.persistence.Id;
717 import javax.persistence.JoinColumn;
718 import javax.persistence.JoinTable;
719 import javax.persistence.ManyToMany;
720 import javax.persistence.Table;
721
722 @Entity
723 @Table(name="aj_authors")
724 public class Author {
725     @Id
726     @GeneratedValue(strategy=GenerationType.AUTO)
727     private int id;
728     private String authorName;
729     @ManyToMany(cascade=CascadeType.ALL)
730     @JoinTable(
731         name="aj_author_book",
732         joinColumns= {@JoinColumn(name="author_id")},
733         inverseJoinColumns= {@JoinColumn(name="book_id")})
734     )
735     private Set<Book> books;
736     public Author() {
737
738     }
739     public int getId() {
740         return id;
741     }
742     public void setId(int id) {
743         this.id = id;
744     }
745     public String getAuthorName() {
746         return authorName;
747     }
748     public void setAuthorName(String authorName) {
749         this.authorName = authorName;
750     }
751     public Set<Book> getBooks() {
752         return books;
753     }
754     public void setBooks(Set<Book> books) {
755         this.books = books;
756     }
757
758 }
759 =====Book.java=====
760 package com.ameya.models;
761
762 import java.util.Set;
763
764 import javax.persistence.CascadeType;
765 import javax.persistence.Entity;
766 import javax.persistence.GeneratedValue;
767 import javax.persistence.GenerationType;
768 import javax.persistence.Id;
769 import javax.persistence.JoinColumn;
770 import javax.persistence.JoinTable;
771 import javax.persistence.ManyToMany;
772 import javax.persistence.Table;
773
774 @Entity
775 @Table(name="aj_books")
776 public class Book {
777     @Id
778     @GeneratedValue(strategy = GenerationType.AUTO)
779     private int id;
780     private String bookName;
781     @ManyToMany(cascade = CascadeType.ALL)
782     @JoinTable(
783         name="aj_author_book",
784         joinColumns = {@JoinColumn(name="book_id")},
785         inverseJoinColumns = {@JoinColumn(name="author_id")})
786     )
787     private Set<Author> authors;
788     public Book() {
```

```

789
790     }
791     public int getId() {
792         return id;
793     }
794     public void setId(int id) {
795         this.id = id;
796     }
797     public String getBookName() {
798         return bookName;
799     }
800     public void setBookName(String bookName) {
801         this.bookName = bookName;
802     }
803     public Set<Author> getAuthors() {
804         return authors;
805     }
806     public void setAuthors(Set<Author> authors) {
807         this.authors = authors;
808     }
809 }
810 =====RelationShipDAOImpl.java=====
811 =====
812 package com.ameya.daos.impl;
813
814 import java.io.Serializable;
815 import java.util.ArrayList;
816 import java.util.HashSet;
817 import java.util.List;
818 import java.util.Set;
819
820 import org.hibernate.Session;
821 import org.hibernate.SessionFactory;
822 import org.hibernate.Transaction;
823 import org.hibernate.query.Query;
824 import org.springframework.beans.factory.annotation.Autowired;
825 import org.springframework.stereotype.Repository;
826
827 import com.ameya.models.Address;
828 import com.ameya.models.Author;
829 import com.ameya.models.Book;
830 import com.ameya.models.Employee;
831 import com.ameya.models.Group;
832 import com.ameya.models.Story;
833
834 @Repository
835 public class RelationShipDAOImpl {
836
837     @Autowired
838     private SessionFactory sessionFactory;
839     public void testOneToOne() {
840         Session sess=sessionFactory.openSession();
841         Transaction tr=sess.beginTransaction();
842         Employee emp=null;
843         Address a1=null;
844
845         emp=new Employee();
846         emp.setName("Ameya Joshi");
847         a1=new Address();
848         a1.setAddress("Kothrud, Pune");
849         a1.setCountry("India");
850
851         emp.setAddress(a1);
852         a1.setEmployee(emp);
853
854         sess.save(emp);
855         sess.flush();
856         Query<Employee> query=sess.createQuery("from Employee");
857         List<Employee> list=query.list();
858         for(Employee e : list) {
859             System.out.println(e.getId()+" : "+e.getName()+" : "+
860 e.getAddress().getAddress()+" : "+e.getAddress().getCountry());

```

```

861     }
862     tr.commit();
863     sess.close();
864 }
865 public void testOneToMany() {
866     Session sess=sessionFactory.openSession();
867     Transaction tr=sess.beginTransaction();
868     Group group=new Group();
869     group.setName("SPORTS");
870     ArrayList<Story> stories=new ArrayList<>();
871
872     Story s1=new Story();
873     s1.setInfo("The Allegations- Life Of a Player");
874     stories.add(s1);
875     Story s2=new Story();
876     s2.setInfo("The Cancer Of match Fixing");
877     stories.add(s2);
878     Story s3=new Story();
879     s3.setInfo("The Master Blaster - Sachin");
880     stories.add(s3);
881
882     group.setStories(stories);
883
884     Serializable id=sess.save(group);
885     sess.flush();
886     Group g=(Group) sess.load(Group.class, id);
887     System.out.println("GROUP ID :: "+g.getId()+" GROUP NAME :: "+g.getName());
888     List<Story> groupStories=g.getStories();
889     System.out.println("STORIES :: ");
890     for(Story story : groupStories) {
891         System.out.println(story.getId()+" : "+story.getInfo());
892     }
893     tr.commit();
894     sess.close();
895 }
896 public void testManyToOne() {
897     Session sess=sessionFactory.openSession();
898     Story story =(Story)sess.load(Story.class, 5);
899     System.out.println(story.getId()+" : "+story.getInfo());
900     Group group=story.getGroup();
901     System.out.println(group.getId()+" : "+group.getName());
902     sess.close();
903 }
904 public void testManyToMany() {
905     Session sess=sessionFactory.openSession();
906     Transaction tr=sess.beginTransaction();
907
908     Author a1=new Author();
909     a1.setAuthorName("Sachin");
910
911     Book b1=new Book();
912     b1.setBookName("My First Pakistan Tour");
913     Book b2=new Book();
914     b2.setBookName("My Cricket - My Life");
915     Book b3=new Book();
916     b3.setBookName("The Unforgettable Don - Don Bradman");
917     Book b4=new Book();
918     b4.setBookName("WC-2011 The Fantastic Experience");
919     Set<Book> books=new HashSet<>();
920     books.add(b1);
921     books.add(b2);
922     books.add(b3);
923     books.add(b4);
924
925     a1.setBooks(books);
926
927     Serializable authorId=sess.save(a1);
928     sess.flush();
929
930     Book b5=new Book();
931     b5.setBookName("The Wonderful Experiences From Cricketing Life");
932
933     Set<Author> authors=new HashSet<>();

```

```

934     Author a2=new Author();
935     a2.setAuthorName("Viru");
936     Author a3=new Author();
937     a3.setAuthorName("Zaheer");
938     Author a4=new Author();
939     a4.setAuthorName("Sunny G");
940     authors.add(a2);
941     authors.add(a3);
942     authors.add(a4);
943     authors.add(a1);
944     b5.setAuthors(authors);
945
946     Serializable bookId=sess.save(b5);
947
948     Author a=(Author)sess.get(Author.class, authorId);
949     System.out.println(a.getId()+" : "+a.getAuthorName());
950     Set<Book> bookSet=a.getBooks();
951     for(Book b : bookSet) {
952         System.out.println(b.getId()+" : "+b.getBookName());
953     }
954     System.out.println("+++++++" );
955
956     Book bk=(Book)sess.get(Book.class,bookId);
957     System.out.println(bk.getId()+" : "+bk.getBookName());
958     Set<Author> authorSet=bk.getAuthors();
959     for(Author at : authorSet) {
960         System.out.println(at.getId()+" : "+at.getAuthorName());
961     }
962
963     tr.commit();
964     sess.close();
965 }
966
967 =====RelationShipServiceImpl.java=====
968 =====
969 package com.ameya.services.impl;
970
971 import org.springframework.beans.factory.annotation.Autowired;
972 import org.springframework.stereotype.Service;
973
974 import com.ameya.daos.impl.RelationShipDAOImpl;
975
976 @Service
977 public class RelationShipServiceImpl {
978
979     @Autowired
980     private RelationShipDAOImpl dao;
981
982     public void testOneToOne() {
983         dao.testOneToOne();
984     }
985     public void testOneToMany() {
986         dao.testOneToMany();
987     }
988     public void testManyToOne() {
989         dao.testManyToOne();
990     }
991     public void testManyToMany() {
992         dao.testManyToMany();
993     }
994 =====RelationShipController.java=====
995 =====
996 package com.ameya.controllers;
997
998 import org.springframework.beans.factory.annotation.Autowired;
999 import org.springframework.web.bind.annotation.GetMapping;
1000 import org.springframework.web.bind.annotation.RequestMapping;
1001 import org.springframework.web.bind.annotation.RestController;
1002
1003 import com.ameya.services.impl.RelationShipServiceImpl;
1004
1005 @RestController

```

```
1005 @RequestMapping("/relations")
1006 public class RelationShipController {
1007
1008     @Autowired
1009     private RelationShipServiceImpl service;
1010     @GetMapping(path="/onetoone")
1011     public void testOneToOne() {
1012         service.testOneToOne();
1013     }
1014     @GetMapping(path="/onetomany")
1015     public void testOneToMany() {
1016         service.testOneToMany();
1017     }
1018     @GetMapping(path="/manytoone")
1019     public void testManyToOne() {
1020         service.testManyToOne();
1021     }
1022     @GetMapping(path="/manytomany")
1023     public void testManyToMany() {
1024         service.testManyToMany();
1025     }
1026 }
1027 =====
1028
1029 http://localhost:9001/relations/manytomany
1030
```