REin
REliable, INtelligent & Scalable Systems

互联网应用开发技术

Web Application Development

第8课
WEB后端-SPRING JPA

Episode Eight

Spring JPA

陈昊鹏

chen-hp@sjtu.edu.cn

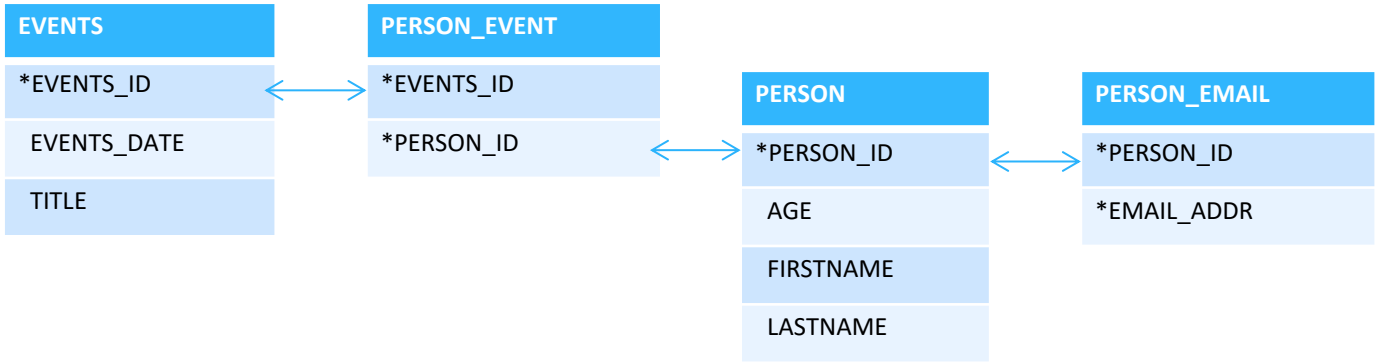
Overview

- Spring Data JPA
  - Relationship Mapping

- Structure of web project

Database Schema

REliable, INtelligent & Scalable Systems



application.properties


```
spring.datasource.url=jdbc:mysql://localhost/test
spring.datasource.username=root
spring.datasource.password=reins2011!
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

Entities: Event


```
@Entity
@Table(name = "events", schema = "test", catalog = "")
@JsonIgnoreProperties(value = {"handler","hibernateLazyInitializer","fieldHandler"})
@JsonIdentityInfo(
      generator = ObjectIdGenerators.PropertyGenerator.class,
      property = "eventId")
public class Event {
   private int eventId;
   private String title;
   private Timestamp eventDate;

   @Id
   @Column(name = "EVENT_ID")
   @GeneratedValue(strategy = IDENTITY)
   public int getEventId() {       return eventId;   }
   public void setEventId(int eventId) {       this.eventId = eventId;   }

   @Basic
   @Column(name = "title")
   public String getTitle() {       return title;   }
   public void setTitle(String title) {       this.title = title;   }

   @Basic
   @Column(name = "EVENT_DATE")
   public Timestamp getEventDate() {       return eventDate;   }
   public void setEventDate(Timestamp eventDate) {       this.eventDate = eventDate;   }
```

Entities: Event


```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Event that = (Event) o;

        if (eventId != that.eventId) return false;
        if (title != null ? !title.equals(that.title) : that.title != null) return false;
        if (eventDate != null ? !eventDate.equals(that.eventDate) : that.eventDate != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = eventId;
        result = 31 * result + (title != null ? title.hashCode() : 0);
        result = 31 * result + (eventDate != null ? eventDate.hashCode() : 0);
        return result;
    }

    private List<Person> participants;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(name = "PERSON_EVENT",joinColumns = @JoinColumn(name = "EVENT_ID"),
            inverseJoinColumns = @JoinColumn(name = "PERSON_ID"))
    public List<Person> getParticipants() {       return participants;    }
    public void setParticipants(List<Person> participants) {       this.participants = participants;    }
}
```

Entities: Person


```
@Entity
@Table(name = "persons", schema = "test", catalog = "")
@JsonIgnoreProperties(value = {"handler","hibernateLazyInitializer","fieldHandler"})
@JsonIdentityInfo(
      generator = ObjectIdGenerators.PropertyGenerator.class,
      property = "personId")
public class Person {
   private int personId;
   private Integer age;
   private String firstname;
   private String lastname;

   @Id
   @Column(name = "PERSON_ID")
   public int getPersonId() {      return personId;   }
   public void setPersonId(int personId) {      this.personId = personId;   }

   @Basic
   @Column(name = "age")
   public Integer getAge() {      return age;   }
   public void setAge(Integer age) {      this.age = age;   }

   @Basic
   @Column(name = "firstname")
   public String getFirstname() {      return firstname;   }
   public void setFirstname(String firstname) {      this.firstname = firstname;   }

   @Basic
   @Column(name = "lastname")
   public String getLastname() {      return lastname;   }
   public void setLastname(String lastname) {      this.lastname = lastname;   }
```

Entities: Person


```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Person person = (Person) o;

        if (personId != person.personId) return false;
        if (age != null ? !age.equals(person.age) : person.age != null) return false;
        if (firstname != null ? !firstname.equals(person.firstname) : person.firstname != null) return false;
        if (lastname != null ? !lastname.equals(person.lastname) : person.lastname != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = personId;
        result = 31 * result + (age != null ? age.hashCode() : 0);
        result = 31 * result + (firstname != null ? firstname.hashCode() : 0);
        result = 31 * result + (lastname != null ? lastname.hashCode() : 0);
        return result;
    }

    private List<Event> activities;

    @ManyToMany(fetch = FetchType.LAZY,mappedBy = "participants")
    public List<Event> getActivities() {       return activities;    }
    public void setActivities(List<Event> activities) {       this.activities = activities;    }

    private List<String> emails = new ArrayList<String>();

    @ElementCollection(fetch = FetchType.EAGER)
    @CollectionTable( name="PERSON_EMAIL",
            joinColumns = { @JoinColumn(name = "PersonId", referencedColumnName = "PERSON_ID")})
    @Column(name="EMAIL_ADDRESS")
    public List<String> getEmails() {       return emails;    }
    public void setEmails(List<String> emails) {       this.emails = emails;    }

}
```

Repositories

REliable, INtelligent & Scalable Systems

public interface EventRepository **extends** JpaRepository<Event, Integer>{ }
public interface PersonRepository **extends** JpaRepository<Person, Integer>{ }

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
void	**deleteAllInBatch**() Deletes all entities in a batch call.
void	**deleteInBatch**(**Iterable**<**T**> entities) Deletes the given entities in a batch which means it will create a single **Query**.
List<**T**>	**findAll**()
<S extends **T**> **List**<S>	**findAll**(**Example**<S> example)
<S extends **T**> **List**<S>	**findAll**(**Example**<S> example, **Sort** sort)
List<**T**>	**findAll**(**Sort** sort)
List<**T**>	**findAllById**(**Iterable**<**ID**> ids)
void	**flush**() Flushes all pending changes to the database.
T	**getOne**(**ID** id) Returns a reference to the entity with the given identifier.
<S extends **T**> **List**<S>	**saveAll**(**Iterable**<S> entities)
<S extends **T**> S	**saveAndFlush**(S entity) Saves an entity and flushes changes instantly.

DAO & DAO Implementation


```
public interface EventDao {
   Event findOne(Integer id);
}

@Repository
public class EventDaoImpl implements EventDao {
      @Autowired
      private EventRepository eventRepository;


      @Override
      public Event findOne(Integer id) {
         return eventRepository.getOne(id);
      }
}

public interface PersonDao {
   Person findOne(Integer id);
}

@Repository
public class PersonDaoImpl implements PersonDao {
      @Autowired
      private PersonRepository personRepository;


      @Override
      public Person findOne(Integer id) {
         return personRepository.getOne(id);
      }
}
```

Services & Service Implementation

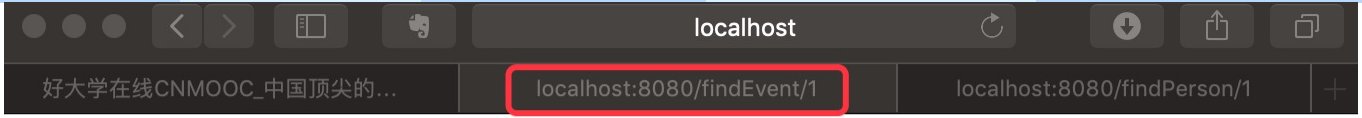

```
public interface EventService {
   Event findEventById(Integer id);
}

@Service
public class EventServiceImpl implements EventService {

   @Autowired
   private EventDao eventDao;

   @Override
   public Event findEventById(Integer id){
      return eventDao.findOne(id);
   }
}

public interface PersonService {
   Person findEventById(Integer id);
}

@Service
public class PersonServiceImpl implements PersonService {

   @Autowired
   private PersonDao personDao;

   @Override
   public Person findEventById(Integer id){
      return personDao.findOne(id);
   }
}
```

Controllers

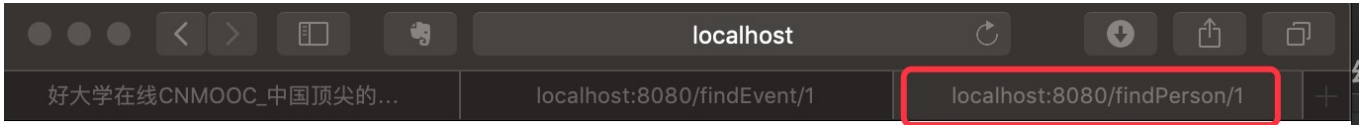

```
@RestController
public class EventController {

    @Autowired
    private EventService eventService;

    @GetMapping(value = "/findEvent/{id}")
    public Event findEvent(@PathVariable("id") Integer id) {
        System.out.println("Searching Event: " + id);
        return eventService.findEventById(id);
    }
}
```

```
@RestController
public class PersonController {

    @Autowired
    private PersonService personService;

    @GetMapping(value = "/findPerson/{id}")
    public Person findPerson(@PathVariable("id") Integer id) {
        System.out.println("Searching Person: " + id);
        return personService.findEventById(id);
    }
}
```

Results



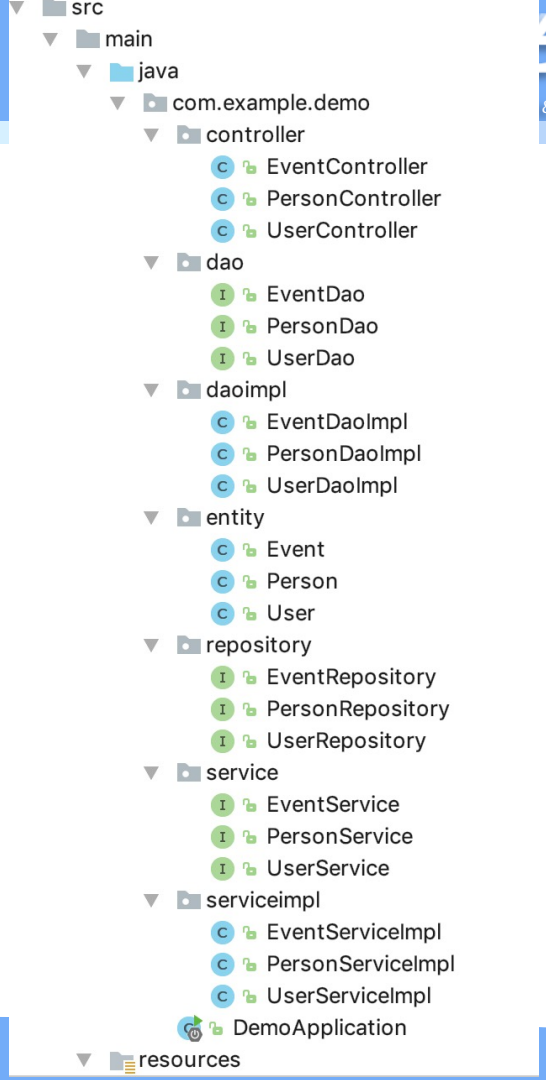

```
{"eventId":1,"title":"party","eventDate":"2017-04-
20T05:00:00.000+0000","participants":
[{"personId":1,"age":47,"firstname":"Cao","lastname":"Cao","activities":[1,
{"eventId":5,"title":"class","eventDate":"2017-04-
19T05:00:00.000+0000","participants":[1]},
{"eventId":11,"title":"Avengers4","eventDate":"2019-04-
26T05:00:00.000+0000","participants":[1]}]},
{"personId":3,"age":26,"firstname":"Liang","lastname":"Zhuge","activities":[1]}]}
```


```
{"personId":1,"age":47,"firstname":"Cao","lastname":"Cao","activities":
[{"eventId":1,"title":"party","eventDate":"2017-04-
20T05:00:00.000+0000","participants":[1,
{"personId":3,"age":26,"firstname":"Liang","lastname":"Zhuge","activities":
[1],"emails":[]}]},{"eventId":5,"title":"class","eventDate":"2017-04-
19T05:00:00.000+0000","participants":[1]},
{"eventId":11,"title":"Avengers4","eventDate":"2019-04-
26T05:00:00.000+0000","participants":[1]}],"emails":["new@new.com"]}
```

Architecture

- Layered Architecture
  - Separation of Interface and Implementation
  - Entity – Auto mapped from database schema
  - Repository – Extended from existing lib class
  - Dao – Your own access control logic
  - Service – Business logic
  - Controller – Dispatch requests
  - IoC/DI – Independent of implementation



References

- Spring Document - Object Relational Mapping (ORM) Data Access
  - https://docs.spring.io/spring/docs/5.2.4.RELEASE/spring-framework-reference/data-access.html#orm

- *Web开发技术*
- *Web Application Development*

Thank You!