互联网应用开发技术

Web Application Development

第8课 WEB后端-访问关系型数据库1

Episode Eight

Access to RDBMS With JPA 1

Web Application Development

陈昊鹏

chen-hp@sjtu.edu.cn

Overview

- Java Persistence API
  - Overview of the Java Persistence API
  - Entities
  - Entity Inheritance
  - Managing Entities
  - Querying Entities
  - Database Schema Creation
  - Further Information about Persistence

JPA

- The Java Persistence API
  - provides Java developers with an object/relational mapping facility for managing relational data in Java applications.

- Java Persistence consists of four areas:
  - The Java Persistence API
  - The query language
  - The Java Persistence Criteria API
  - Object/relational mapping metadata

What is Object/Relational Mapping?

- Persistence
  - Hibernate ORM is concerned with helping your application to achieve persistence.
  - So what is persistence? Persistence simply means that we would like our application's data to outlive the applications process.
  - In Java terms, we would like the state of (some of) our objects to live beyond the scope of the JVM so that the same state is available later.

- Relational Databases
  - Specifically, Hibernate ORM is concerned with data persistence as it applies to relational databases (RDBMS).
  - Suffice it to say that RDBMS remain a very popular persistence mechanism and will so for the foreseeable future.

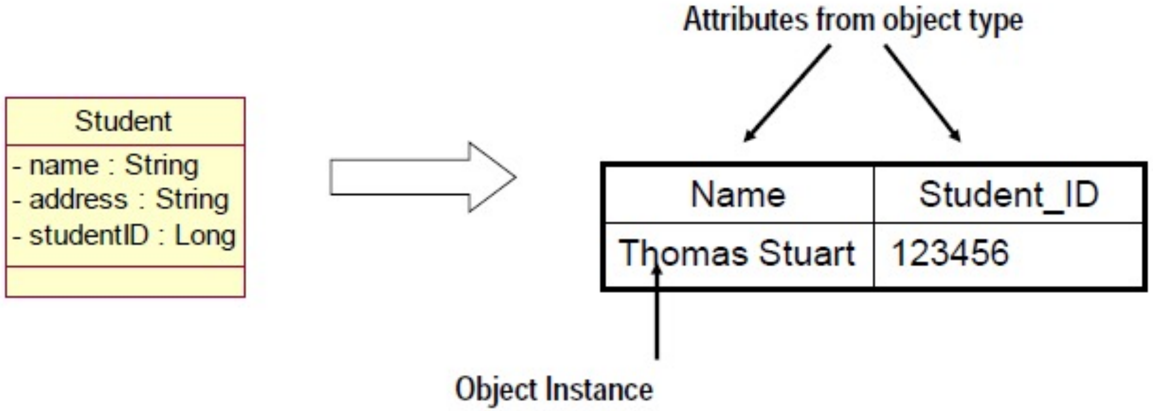
What is Object/Relational Mapping?

- The term Object/Relational Mapping refers to
  - the technique of mapping data from an object model representation to a relational data model representation (and visa versa)

- Hibernate is an Object/Relational Mapping solution for Java environments.

Mapping Persistent Classes to Tables

- In a relational database
  - Every row is regarded as an object
  - A column in a table is equivalent to a persistent attribute of a class



What is Object/Relational Mapping?

- The Object-Relational Impedance Mismatch
  - Object models and relational models do not work very well together.
    - RDBMSs represent data in a tabular format (a spreadsheet is a good visualization for those not familiar with RDBMSs),
    - whereas object-oriented languages, such as Java, represent it as an interconnected graph of objects.
  - Granularity
  - Subtypes (inheritance)
  - Identity
  - Associations
  - Data navigation

Entities

- Entities
  - An entity is a lightweight persistence domain object.
  - Typically, an entity represents a table in a relational database, and each entity instance corresponds to a row in that table.
  - The primary programming artifact of an entity is the entity class, although entities can use helper classes.
- The persistent state of an entity
  - is represented through either persistent fields or persistent properties.
  - These fields or properties use object/relational mapping annotations to map the entities and entity relationships to the relational data in the underlying data store.

Entities

REliable, INtelligent & Scalable Systems

- Persistent Fields and Properties in Entity Classes
  - The persistent state of an entity can be accessed through either the entity's instance variables or properties.

- Persistent Fields
  - If the entity class uses persistent fields, the Persistence runtime accesses entity-class instance variables directly.

- Persistent Properties
  - For every persistent property of the entity, there is a getter method getProperty and setter method setProperty. :
  - `Type getProperty()`
  - `void setProperty(Type type)`

Hibernate – first class

REliable, INtelligent & Scalable Systems

- Event.class

```
@Entity
@Table( name = “EVENTS” )
public class Event {
   private Long id;
   private String title;
   private Date date;

   public Event() {    // this form used by Hibernate    }
   public Event(String title, Date date) {
      // for application use, to create new events
      this.title = title;
      this.date = date;
   }

   @Id
   @GeneratedValue(generator=“increment”)
   @GenericGenerator(name=“increment”, strategy = “increment”)
   public Long getId() {    return id;    }
   private void setId(Long id) {     this.id = id;    }

   @Temporal(TemporalType.TIMESTAMP)
   @Column(name = “EVENT_DATE”)
   public Date getDate() {    return date;    }
   public void setDate(Date date) {     this.date = date;    }

   public String getTitle() {     return title;    }
   public void setTitle(String title) {    this.title = title;    }
}
```

Hibernate – configuration

- hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

  <session-factory>

    <!-- Database connection settings -->
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/ormsample</property>
    <property name="connection.username">root</property>
    <property name="connection.password">reins2011!</property>

    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>

    <!-- JDBC connection pool (use the built-in) -->
    <property name="connection.pool_size">1</property>
```

Hibernate – configuration

- hibernate.cfg.xml

```
        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>

        <!-- Names the annotated entity class -->
        <mapping class="org.reins.orm.entity.Event"/>

    </session-factory>

</hibernate-configuration>
```

Helper class

- HibernateUtil.java

```
public class HibernateUtil {
   private static final SessionFactory sessionFactory = buildSessionFactory();

   private static SessionFactory buildSessionFactory() {
      // A SessionFactory is set up once for an application!
      final StandardServiceRegistry registry = new StandardServiceRegistryBuilder()
            .configure() // configures settings from hibernate.cfg.xml
            .build();
      try {
         return new MetadataSources(registry).buildMetadata().buildSessionFactory();
      } catch (Exception e) {
         // The registry would be destroyed by the SessionFactory, but we had trouble building the SessionFactory
         // so destroy it manually.
         StandardServiceRegistryBuilder.destroy(registry);
         throw new ExceptionInInitializerError(e);
      }

   }

   public static SessionFactory getSessionFactory() {
      return sessionFactory;
   }
}
```

Servlet - storing object

REliable, INtelligent & Scalable Systems

- EventServlet

```
@WebServlet("/EventServlet")
public class EventServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

   public EventServlet() {
      super();
   }

   protected void processRequest(HttpServletRequest request,
         HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
         out.println("<html lang=\"en\">");
         out.println("<head>");
         out.println("<title>Servlet UserServlet</title>");
         out.println("</head>");
         out.println("<body>");
```

Servlet - storing object

- EventServlet

```
String title = (String) request.getParameter("title");
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String datestr = (String) request.getParameter("date");
Date date=sdf.parse(datestr);

Session session = HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
Event t = new Event();
t.setDate(date);
t.setTitle(title);
session.save(t);
session.getTransaction().commit();

out.println("<FORM METHOD=POST ACTION=\"PersonServlet\">");
out.println("Event ID <INPUT TYPE=TEXT NAME=event SIZE=20 ><BR>");
out.println("Person ID <INPUT TYPE=TEXT NAME=person SIZE=20 >");
out.println("<P><INPUT TYPE=SUBMIT value=\"Next\">");
out.println("<h1>The event has been inserted!</h1><br>");
```

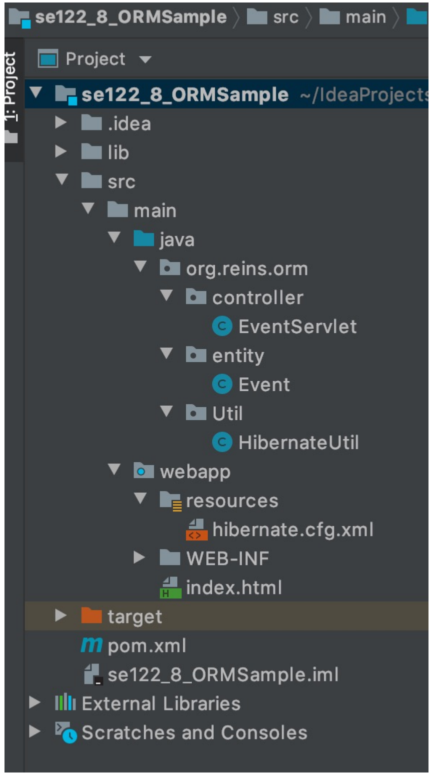
Servlet - loading object

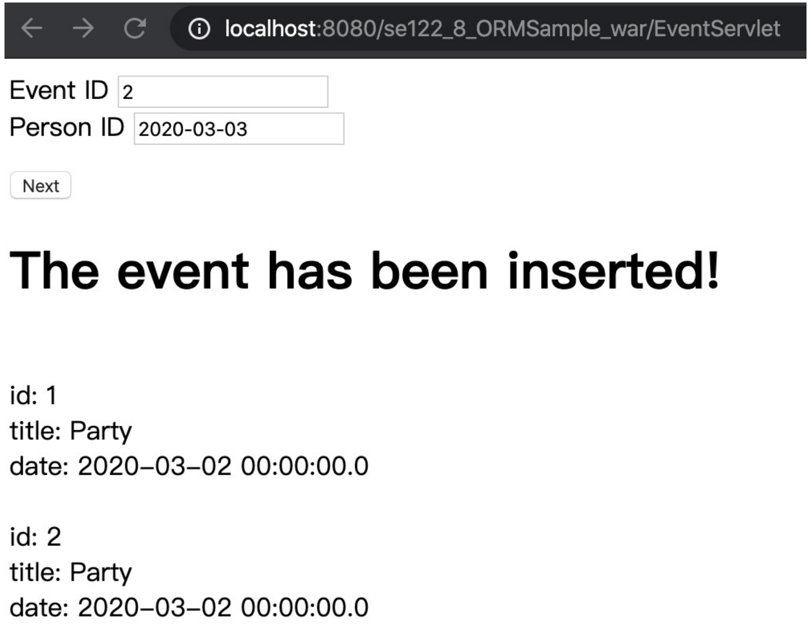
REliable, INtelligent & Scalable Systems

- EventServlet

```
          session = HibernateUtil.getSessionFactory().getCurrentSession();
          session.beginTransaction();
          List events = session.createQuery(“from Event”).list();
          session.getTransaction().commit();
          for (int i = 0; i < events.size(); i++) {
            Event theEvent = (Event) events.get(i);
            out.println(“id: ” + theEvent.getId() + “<br>” + “title: ”
                + theEvent.getTitle() + “<br>” + “date: ” + theEvent.getDate() + “<br><br>”);
          }

        out.println("</body>");
        out.println("</html>");

      } catch(Exception e){
        e.printStackTrace();
      }
      finally {
          out.close();
      }
  }
```

First Sample





Event ID 2
Person ID 2020-03-03

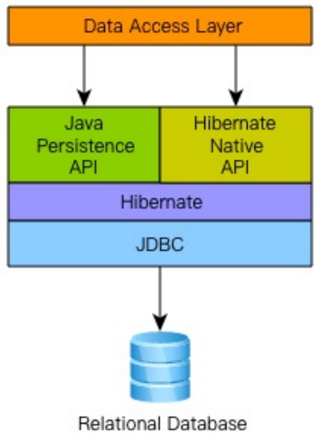
Next

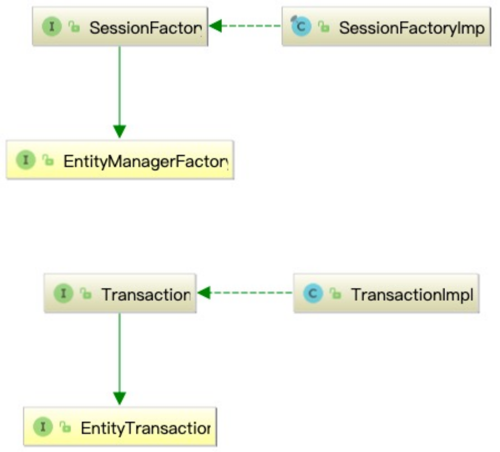
The event has been inserted!

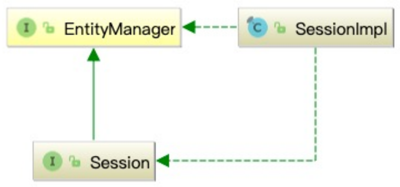
```
id: 1
title: Party
date: 2020–03–02 00:00:00.0

id: 2
title: Party
date: 2020–03–02 00:00:00.0
```

Hibernate

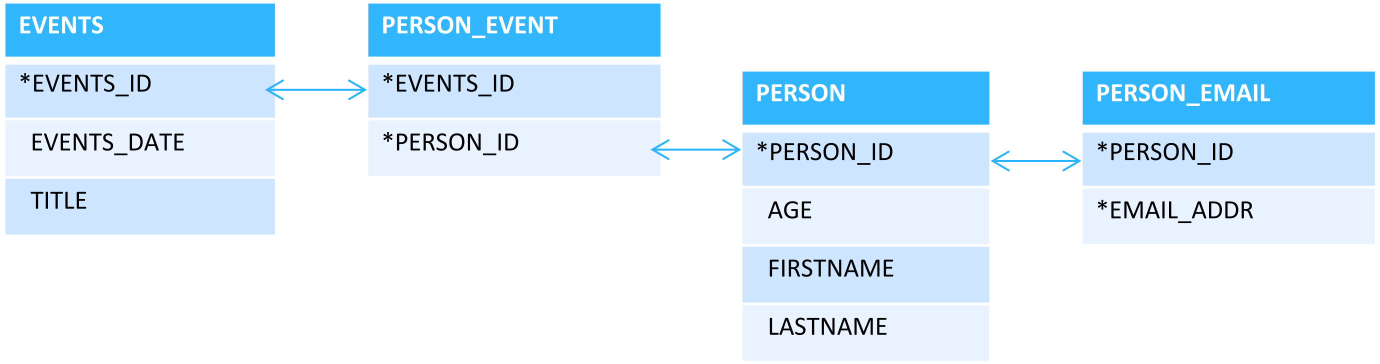
- Overview







Database Schema



Entity - Event

REliable, INtelligent & Scalable Systems

- Event.java

```
@Entity
@Table(name = "EVENTS")
public class Event {
   private Long id;
   private String title;
   private Date date;

   private Set<Person> participants = new HashSet<>();

   @ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE}, fetch = FetchType.EAGER)
   @JoinTable(name = "PERSON_EVENT",
         joinColumns = @JoinColumn(name = "event_id", referencedColumnName = "id"),
         inverseJoinColumns = @JoinColumn(name = "person_id", referencedColumnName = "id"))

   public Set<Person> getParticipants() {
      return participants;
   }

   public void setParticipants(Set<Person> participants) {
      this.participants = participants;
   }

}
```

Entity - Person

REliable, INtelligent & Scalable Systems

- Person.java

```
@Entity
@Table(name = "PERSONS")
public class Person {
   private Long id;
   private int age;
   private String firstname;
   private String lastname;
   private List<Event> events = new ArrayList<>();
   private List<String> emailAddresses = new ArrayList();


   public Person() {
      // this form used by Hibernate
   }

   @Id
   @GeneratedValue(generator = "increment")
   @GenericGenerator(name = "increment", strategy = "increment")
   public Long getId() {
      return id;
   }
   private void setId(Long id) {
      this.id = id;
   }
```

Entity - Person

- Person.java

```java
public int getAge() {
    return age;
}

private void setAge(int age) {
    this.age = age;
}

public String getFirstname() {
    return firstname;
}

private void setFirstname(String firstname) {
    this.firstname = firstname;
}

public String getLastname() {
    return lastname;
}

private void setLastname(String lastname) {
    this.lastname = lastname;
}

@ManyToMany(mappedBy = "participants",
            fetch = FetchType.EAGER)
public List<Event> getEvents() {
    return events;
}

public void setEvents(List<Event> events) {
    this.events = events;
}

@ElementCollection
@CollectionTable( name="PERSON_EMAIL",
    joinColumns = {
        @JoinColumn(name = "PERSON_ID",
                    referencedColumnName = "id")})
@Column(name="EMAIL_ADDRESS")
public List<String> getEmailAddresses() {
  return emailAddresses;
}
public void setEmailAddresses(List<String> emailAddresses) {
  this.emailAddresses = emailAddresses;
}

}
```

Servlet - PersonServlet

- PersonServlet.java

```
@WebServlet("/PersonServlet")
public class PersonServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

   public PersonServlet() {
      super();
   }

   protected void processRequest(HttpServletRequest request,
         HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
         out.println("<html lang=\"en\">");
         out.println("<head>");
         out.println("<title>Servlet PersonServlet</title>");
         out.println("</head>");
         out.println("<body>");

         String event = (String) request.getParameter("event");
         String person = (String) request.getParameter("person");
         long eventId = Long.valueOf(event);
         long personId = Long.valueOf(person);
```

Servlet - PersonServlet

- PersonServlet.java

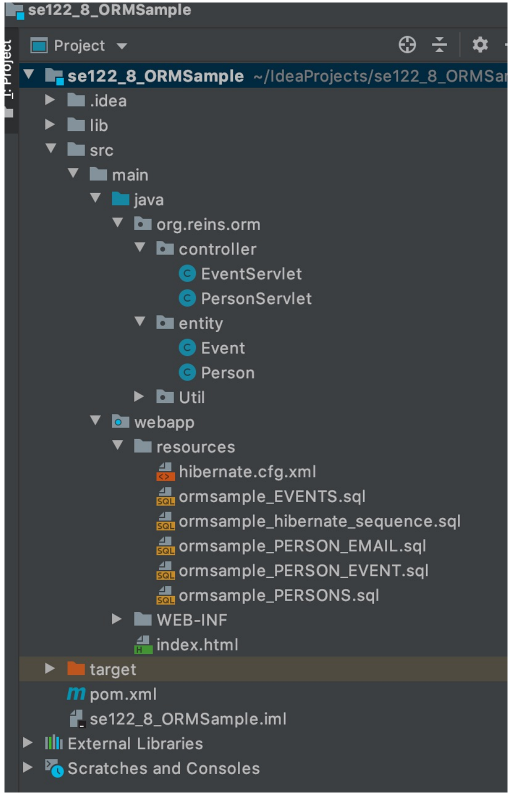
```
Session session = HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
Person aPerson = session.get(Person.class, personId);
Event anEvent = session.get(Event.class, eventId);

aPerson.getEvents().add(anEvent);
anEvent.getParticipants().add(aPerson);
aPerson.getEmailAddresses().add("new@new.com");
session.update(aPerson);
session.getTransaction().commit();

session = HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
Set participants = anEvent.getParticipants();
Iterator iter = participants.iterator();
while(iter.hasNext()){
   Person thePerson = (Person)iter.next();
   out.println("Participant: " + thePerson.getFirstname() + " " + thePerson.getLastname() + "<br><br>");
}
session.getTransaction().commit();
```

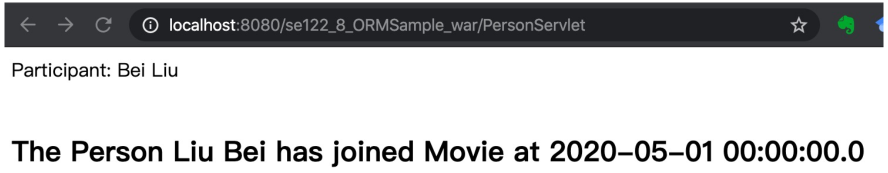
Servlet - PersonServlet

- PersonServlet.java

```
            String ss = "<h2> The Person " + aPerson.getLastname() + " " + aPerson.getFirstname()
                    + " has joined " + anEvent.getTitle() + " at " + anEvent.getDate();
            out.println(ss);
          out.println("</body>");
          out.println("</html>");

        } catch(Exception e){
          e.printStackTrace();
        }
        finally {
            out.close();
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
```

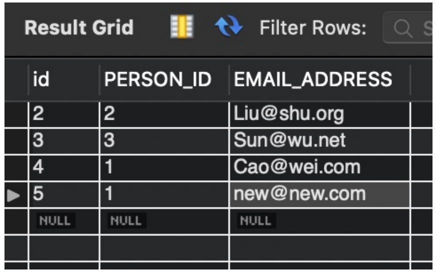
Run the Sample


```
se122_8_ORMSample
Project
▼ se122_8_ORMSample ~/IdeaProjects/se122_8_ORMSa
  ▶ .idea
  ▶ lib
  ▼ src
    ▼ main
      ▼ java
        ▼ org.reins.orm
          ▼ controller
              EventServlet
              PersonServlet
          ▼ entity
              Event
              Person
          ▶ Util
      ▼ webapp
        ▼ resources
            hibernate.cfg.xml
            ormsample_EVENTS.sql
            ormsample_hibernate_sequence.sql
            ormsample_PERSON_EMAIL.sql
            ormsample_PERSON_EVENT.sql
            ormsample_PERSONS.sql
        ▶ WEB-INF
          index.html
  ▶ target
    pom.xml
    se122_8_ORMSample.iml
▶ External Libraries
▶ Scratches and Consoles
```

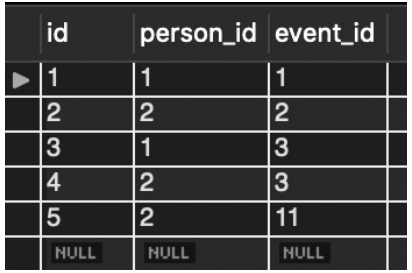

localhost:8080/se122_8_ORMSample_war/PersonServlet

Participant: Bei Liu

The Person Liu Bei has joined Movie at 2020-05-01 00:00:00.0


Result Grid | Filter Rows:

id	PERSON_ID	EMAIL_ADDRESS
2	2	Liu@shu.org
3	3	Sun@wu.net
4	1	Cao@wei.com
5	1	new@new.com
NULL	NULL	NULL



id	person_id	event_id
1	1	1
2	2	2
3	1	3
4	2	3
5	2	11
NULL	NULL	NULL

Basic API

- SessionFactory (org.hibernate.SessionFactory)
- Session (org.hibernate.Session)
- Persistent objects and collections
- Transient and detached objects and collections
- Transaction (org.hibernate.Transaction)(Optional)

Entity

- An entity is a regular Java object (aka POJO) which will be persisted by Hibernate.

```
@Entity
public class Flight implements Serializable {
  Long id;

  @Id
  public Long getId() { return id; }
  public void setId(Long id) { this.id = id; }
}
```

Entity

```
@Entity
@Table(name="TBL_FLIGHT",
        schema="AIR_COMMAND",
        uniqueConstraints=
                @UniqueConstraint(
        name="flight_number",
                 columnNames={"comp_prefix","flight_number"} ) )
public class Flight implements Serializable {

  @Column(name="comp_prefix")
  public String getCompagnyPrefix() {
   return companyPrefix;
  }

  @Column(name="flight_number")
  public String getNumber() { return number; }
}
```

Entity

name="ClassName“
table="tableName“
discriminator-value="discriminator_value“
mutable="true|false“
schema="owner“
catalog="catalog“
proxy="ProxyInterface“
dynamic-update="true|false“
dynamic-insert="true|false“
select-before-update="true|false“
polymorphism="implicit|explicit“
where="arbitrary sql where condition
persister="PersisterClass“
batch-size="N“
optimistic-lock="none|version|dirty|all“
lazy="true|false"
entity-name="EntityName"
check="arbitrary sql check condition"
rowxml:id="rowid"
subselect="SQL expression"
abstract="true|false"
node="element-name"

Identifiers

```
@Entity
public class Person {
    @Id
    @GeneratedValue(generator = "increment")
    @GenericGenerator(name = "increment", strategy = "increment")
    Integer getId() { … }
    …
}
```

Composite identifier

- id as a property using a component type

```
@Entity
class User {
  @EmbeddedId
  @AttributeOverride(
    name="firstName", column=@Column(name="fld_firstname")
  UserId id;
  Integer age;
}

@Embeddable
class UserId implements Serializable {
  String firstName;
  String lastName;
}
```

Composite identifier

- id as a property using a component type

```
@Entity
class Customer {
  @EmbeddedId
  CustomerId id;
  boolean preferredCustomer;

  @MapsId("userId")
  @JoinColumns({
    @JoinColumn(name="userfirstname_fk",
            referencedColumnName="firstName"),
    @JoinColumn(name="userlastname_fk",
            referencedColumnName="lastName")
  })
  @OneToOne
  User user;
}

@Embeddable
class CustomerId implements Serializable {
  UserID userId;
  String customerNumber;
  //implements equals and hashCode
}
```

Composite identifier

- Multiple id properties without identifier type

```
@Entity
class Customer implements Serializable {
  @Id
  @OneToOne
  @JoinColumns({
    @JoinColumn(name="userfirstname_fk",
                    referencedColumnName="firstName"),
    @JoinColumn(name="userlastname_fk",
                    referencedColumnName="lastName")
  })
  User user;

  @Id
  String customerNumber;
  boolean preferredCustomer;
  //implements equals and hashCode
}
```

Composite identifier

- Multiple id properties with a dedicated identifier type

```
@Entity
@IdClass(CustomerId.class)
class Customer implements Serializable {
  @Id
  @OneToOne
  @JoinColumns({
    @JoinColumn(name="userfirstname_fk", referencedColumnName="firstName"),
    @JoinColumn(name="userlastname_fk", referencedColumnName="lastName")
  })
  User user;

  @Id
  String customerNumber;
  boolean preferredCustomer;
}

class CustomerId implements Serializable {
   UserId user;
   String customerNumber
  //implements equals and hashCode
}
```

Identifier generator

- IDENTITY
- SEQUENCE (called seqhilo in Hibernate)
- TABLE (called MultipleHiLoPerTableGenerator in Hibernate)
- AUTO

```
@Entity
public class Customer {
   @Id
   @GeneratedValue
   Integer getId() { ... };
}

@Entity
public class Invoice {
   @Id
   @GeneratedValue(strategy=GenerationType.IDENTITY)
   Integer getId() { ... };
}
```

References

- Hibernate ORM - Getting started with Hibernate ORM
  - https://hibernate.org/orm/documentation/getting-started/
- Hibernate ORM - What is Object/Relational Mapping?
  - https://hibernate.org/orm/what-is-an-orm/
- The Java EE 8 Tutorial – Introduction to the Java Persistence API
  - https://javaee.github.io/tutorial/persistence-intro.html#BNBPZ

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

Thank You!