HIBERNATE ANNOTATIONS -BY MR.RAGHU

pom.xml

```
<properties>
	<maven.compiler.source>13</maven.compiler.source>
	<maven.compiler.target>13</maven.compiler.target>
</properties>
<dependencies>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>5.4.10.Final</version>
	</dependency>
	<dependency>
		<groupId>org.projectlombok</groupId>
		<artifactId>lombok</artifactId>
		<version>1.18.12</version>
		<scope>provided</scope>
	</dependency>

	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>5.1.46</version>
	</dependency>
</dependencies>
```

BASIC ANNOTATIONS:

@Entity
@Table(name="empt_tab")
@Id
@Column(name="eid")

PRIMARY KEY GENERATOR:

@GeneratedValue @GeneratedValue(strategy=GenerationType.AUTO)
@GeneratedValue(strategy=GenerationType.IDENTITY)
@GeneratedValue(strategy=GenerationType.SEQUENCE)

```
@GeneratedValue(strategy=GenerationType.TABLE)

@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="sample")
@SequenceGenerator(name="sample",sequenceName="emp_seq")

@GeneratedValue(generator="sample")
@GenericGenerator(name="sample",strategy="com.app.model.MyGen")
@GenericGenerator(name="sample",strategy="native")//identity,hilo,increment
```

DATE AND TIME:(java.util.DATE)

```
@Temporal(TemporalType.DATE)
private Date dateOne;
@Temporal(TemporalType.TIME)
private Date dateTwo;
@Temporal(TemporalType.TIMESTAMP)
private Date dateThree;
```

BLOB and CLOB:

```
@Lob
private byte[] image;
@Lob
private char[] doc;
```

VERSION OF OBJECT:

```
@Version
private int ver1;
@Version
private Date ver2;
```

LIST, SET AND MAP WITH PRIMITIVES:

```
@ElementCollection @CollectionTable(name="emp_dtls", //table
joinColumns=@JoinColumn(name="eidFk")) //key col
@Column(name="lst_data") //element col
private Set<String> details=new HashSet<String>(0);

@ElementCollection @CollectionTable(name="emp_data", //table
joinColumns=@JoinColumn(name="eidFk")) //key col
@OrderColumn(name="pos") //index col
@Column(name="prjs") //element col
private List<String> data=new ArrayList<String>(0);

@ElementCollection @CollectionTable(name="emp_models", //table
joinColumns=@JoinColumn(name="eidFk")) //key col
@MapKeyColumn(name="pos") //index col
@Column(name="model_data") //element col
private Map<Integer,String> models=new HashMap<Integer, String>();
```

COMPONENT MAPPING:

```
@Embeddable public class Address{
        @Column(name="hno")
        private int hno;
        @Column(name="loc")
        private String loc;
}

@Entity
class Employee {
@Embedded
@AttributeOverrides({
@AttributeOverride(name="hno",column=@Column(name="hno")),
@AttributeOverride(name="loc",column=@Column(name="location"))
})
        private Address addr=new Address();
}
```

INHERITANCE MAPPING:

TABLE PER CLASS HIERARCHY

```
@Entity
@Table(name="empt_tab")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="ob_type",discriminatorType=DiscriminatorType.STRING)
@DiscriminatorValue("EMP")
class Employee{
        @Id
        @Column(name="eid");
        private int empId;
        @Column(name="ename");
        private String empName;
}
@DiscriminatorValue("REG")
class RegEmployee exntends Employee {
        @Column(name="emp_prj");
        private String projId;
        @Column(name="emp_bouns");
        private double yearlyBouns;
}

@DiscriminatorValue("CNT")
class ContractEmployee extends Employee {
        @Column(name="emp_wrk_hrs"); private double workingHrs;
@Column(name="emp_shift_grade"); private String shiftGrade;
}
```

TABLE PER SUB CLASS

```
@Entity @Table(name="emp")
@Inheritance(strategy=InheritanceType.JOINED)
```

```
class Employee{
	@Id
	@Column(name="eid");
	private int empId;
	@Column(name="ename");
	private String empName;
}

@Entity @Table(name="reg_emp")
@PrimaryKeyJoinColumn(name="eidFk")
class RegEmployee exntends Employee {
	@Column(name="emp_prj");
	 private String projId;
	@Column(name="emp_bouns");
	private double yearlyBouns;
}

@Entity @Table(name="cnt_emp")
@PrimaryKeyJoinColumn(name="eidFk")
class ContractEmployee extends Employee {
	@Column(name="emp_wrk_hrs");
	private double workingHrs;
	@Column(name="emp_shift_grade");
	private String shiftGrade;
}
```

TABLE PER CONCRETE CLASS

```
@Entity
@Table(name="emp")
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS)
class Employee{
	@Id
	@Column(name="eid");
	private int empId;
	@Column(name="ename");
	private String empName;
}

@Entity
@Table(name="reg_emp")
class RegEmployee exntends Employee {
	@Column(name="emp_prj");
	private String projId;
	@Column(name="emp_bouns");
	private double yearlyBouns;
}
@Entity
@Table(name="cnt_emp")
class ContractEmployee extends Employee {
        @Column(name="emp_wrk_hrs");
        private double workingHrs;
        @Column(name="emp_shift_grade");
        private String shiftGrade;
}
```

ASSOCIATION MAPPING:

Many-To-One and One-To-One (Employee ----<> Address HAS-A)

```
@Entity
@Table(name="addrs_tab")
 public class Address {
        @Id @Column(name="aid")
        private int addrId;
        @Column(name="loc")
        private String loc;
        @OneToMany(mappedBy="addr")
        private List<Employee> emp=new ArrayList<Employee>(0);
}
```

```
@Entity
@Table(name="empt_tab")
class Employee{
        @Id
        @Column(name="eid");
        private int empId;

        @ManyToOne(fetch=FetchType.EAGER,cascade=CascadeType.ALL)
        @JoinColumn(name="aidFk",unique=true/false)
        private Address addr=new Address();

}
```

Many-To-Many

```
@Entity @Table(name="addrs_tab")

public class Address {
        @Id
        @Column(name="aid")
        @GeneratedValue private int addrId;
        @Column(name="loc") private String loc;

        @ManyToMany(mappedBy="addr")
        private List<Employee> emp=new ArrayList<Employee>(0);
}
```

```
@Entity
@Table(name="empt_tab")
 class Employee{
        @Id
        @Column(name="eid");
        private int empId;
        @ManyToMany(cascade=CascadeType.ALL,fetch=FetchType.EAGER)
        @JoinTable(name="emp_addr", joinColumns=@JoinColumn(name="eidFk"),
        inverseJoinColumns=@JoinColumn(name="aidFk"))
        private List<Address> addr=new ArrayList<Address>(0);

}
```

One-To-Many

```
@Entity
@Table(name="addrs_tab")
public class Address {

        @Id
        @Column(name="aid")
        @GeneratedValue private int addrId;
        @Column(name="loc") private String loc;

    @ManyToOne(mappedBy="addr")
    private Employee emp;

}

@Entity @Table(name="empt_tab") class Employee{
@Id @Column(name="eid"); private int empId;

        @OneToMany(cascade=CascadeType.ALL,fetch=FetchType.EAGER)
        @JoinColumn(name="eidFk")
        private List<Address> addr=new ArrayList<Address>(0);
}
```

BAG AND IDBAG

Bag:

```
@ElementCollection
@CollectionTable(name="emp_data", //table
joinColumns=@JoinColumn(name="eidFk")) //key col
@Column(name="prjs") //element col
private List<String> data=new ArrayList<String>(0);
```

IdBag

```
@GenericGenerator(name="sample",strategy="increment")
@Entity
@Table(name="emp_tab")
class Employee{
        @ElementCollection
        @CollectionTable(name="emp_data", //table
        joinColumns=@JoinColumn(name="eidFk")) //key col
        @CollectionId(
                columns=@Column(name="unqPos"),
                 generator = "sample",
                type = @Type(type="long"))
        @Column(name="prjs") //element col
        private List<String> data=new ArrayList<String>(0);
}
```

NAMED QUERIES:

HQL EXAMPLE :

```
@NamedQueries(
        @NamedQuery(name="getemp",
                query="from com.app.model.Employee where empId=?")
)
@Entity
@Table(name="empt_tab")
class Employee{
}
```

NATIVE SQL EXAMPLE:

```
@NamedNativeQueries({
        @NamedNativeQuery( name = " getemps ",
        query = "select * from Employee eid = :eidCode", resultClass = Employee.class
        )
})
@Entity
@Table(name="emp_tab")
class Employee {}
```

Test class:

```
Query q=ses.getNamedQuery("getemp"); q.setParameter(0, 3);
List<Employee> list=q.list();
```

SECONDARY TABLE:

```
@Entity
@Table(name="emp_tab")
```

```
@SecondaryTables(
        @SecondaryTable(name = "emp_child",
                pkJoinColumns=@PrimaryKeyJoinColumn(
                name="eidFk",referencedColumnName="eid")
        )
)
class Employee {
        @Id
        @Column(name="eid")
        int empId;
        @Column(name="ename",table="emp_child")
        String empName;
        @Column(name="esal")
        double empSal
}
```

VALIDATIONS:

```
@NotNull(message="Employee name must not be null")
@Size(min=3,max=6,message="Employee Name must be in 3-6 chars")
@Pattern(regexp="SAT[A-Z]*")
private String empName;

@Min(value=3,message="EmpSal minimum nuber is 3")
@Max(4)
@Column(name="esal")
private double empSal;

@AssertTrue
private boolean isEnabled;
@AssertFalse
private boolean isFinished;
@Past
@NotNull
private Date date1;
@Future
private Date date2;
```

Dynamic Query Annotations

```
@DynamicUpdate
@DyanmicInsert
```

TRANSIENT COLUMN

(avoid Mapping to DB Column)

```
@Transient
```

Natural Identity for PK

```
@NaturalId
```

Cache Management

@Cache

Hibernate BootStrap class:-

```
package in.nit.util;
import java.util.Properties;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.cfg.Environment;

import in.nit.model.Employee;

public class HibernateUtil {

      private static SessionFactory sf=null;

      static {
            try {
                  //1. Properties object using Environment
                  Properties p=new Properties();
                  p.put(Environment.DRIVER, "com.mysql.jdbc.Driver");
                  p.put(Environment.URL,
"jdbc:mysql://localhost:3306/hibernate");
                  p.put(Environment.USER, "root");
                  p.put(Environment.PASS, "root");

                  p.put(Environment.DIALECT,
"org.hibernate.dialect.MySQL55Dialect");
                  p.put(Environment.SHOW_SQL, true);
                  p.put(Environment.FORMAT_SQL,true);
                  p.put(Environment.HBM2DDL_AUTO,"update");

                  //2. Convert into Hibernate Object format
                  Configuration cfg=new Configuration();

                  //3. Load Properties into Configuration
                  cfg.setProperties(p);

                  //4. Provide entity details to cfg
                  cfg.addAnnotatedClass(Employee.class);
                  //cfg.addAnnotatedClass(Employee.class);

                  //5. ServiceRegistery
                  StandardServiceRegistry register=
                                new StandardServiceRegistryBuilder()
                                .applySettings(cfg.getProperties())
                                .build();

                    //6. SessionFactory
                    sf=cfg.buildSessionFactory(register);

            } catch (Exception e) {
                    e.printStackTrace();
            }
    }

    public static SessionFactory getSf() {
            return sf;
    }
}
```

FB Group: https://www.facebook.com/groups/thejavatemple