HIBERNATE ANNOTATIONS -[by RAGHU]

pom.xml

```
<dependencies>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>4.3.4.Final</version>
	</dependency>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-validator</artifactId>
		<version>4.3.2.Final</version>
	</dependency>
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>5.1.6</version>
	</dependency>
</dependencies>
```

1. BASIC ANNOTATIONS:

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

2. 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
```

3. DATE AND TIME:(java.util.DATE)

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

4. BLOB and CLOB:

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

5. VERSION OF OBJECT:

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

6. 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>();
```

7. 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();
}
```

8.INHERITANCE MAPPING:

i. 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;
}
```

ii. 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;
}
```

iii. 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;
}
```

9.ASSOCIATION MAPPING:

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

Box Means : Bi-Directional (optional one)

```
@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();

}
```

ii. 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);

}
```

iii. 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);
}
```

10. 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);

}
```

11. NAMED QUERIES: HQL EXAMPLE

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

12. 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();
```

13. 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
}
```

14. 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;
```