Hibernate-Value Types and Embedding Objects

Lets first understand what is a Value Type.
Suppose in your entity class you have an object as a member variable which has its own attribute but which does not have a meaning unless it has a context with the entity class.
For example In our entity class UserDetails suppose we have Address which does not have a single attribute like String but it has some more attributes like,street,city,state,pincode.

@Embeddable and @Embedded

In above case how to save the data.
The most common impletmentation is not to have seperate table for Address but to include seperate coloumn for each attribute in the UserDetails table itself.
So create a separate class Address.java ,add the attributes to it,generate the getters and setters for it.

```
package com.technicalstack.dto;

import javax.persistence.Embeddable;

@Embeddable
public class Address {
 private String street;
 private String city;
 private String state;
 private String pincode;

 public String getStreet() {
  return street;
 }
 public void setStreet(String street) {
  this.street = street;
 }
 public String getCity() {
  return city;
 }
 public void setCity(String city) {
  this.city = city;
 }
 public String getState() {
  return state;
 }
 public void setState(String state) {
  this.state = state;
 }
 public String getPincode() {
  return pincode;
 }
 public void setPincode(String pincode) {
  this.pincode = pincode;
 }
```

```
}
```

In UserDetails.java.add Address attribute add generate its getters and setters

```
 @Column (name="USER_NAME")
 private String name;

 @Embedded
 private Address address;

 public Address getAddress() {
  return address;
 }
```

The way to tell hibernate not to create a new table for this class is by using @Embeddable annotation on the Address class.
so add the attribute Address in your UserDetails class and then annotate it with @Embedded

```
package com.technicalstack.dto;

import javax.persistence.GeneratedValue;
import javax.persistence.Id;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateTest {

 public static void main(String[] args) {
  UserDetails user1 = new UserDetails();
 // user1.setUserId(1);  as we are using @Id @GeneratedValue so we don't need to set it.
  user1.setName("Shailesh");

  Address add = new Address();
  add.setCity("Pune");
  add.setState("Maharashtra");
  add.setPincode("440011");
  add.setStreet("MG-Road");
  user1.setAddress(add);
  SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
  Session session = sessionFactory.openSession();
  session.beginTransaction();

  session.save(user1);

  session.getTransaction().commit();
  session.close();

  }

}
```

so we will make necessary changes as required in the code and then run it.
Output:

Hibernate: create sequence hibernate_sequence start 1 increment 1
Hibernate: create table USER_DETAILS (userId int4 not null, city varchar(255), pincode varchar(255), state varchar(255), street varchar(255), USER_NAME varchar(255), primary key (userId))
Sep 24, 2016 4:50:10 PM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000230: Schema export complete
Hibernate: select nextval ('hibernate_sequence')
Hibernate: insert into USER_DETAILS (city, pincode, state, street, USER_NAME, userId) values (?, ?, ?, ?, ?, ?)