MAHARISHI · UNIVERSITY OF MANAGEMENT · ENGAGE THE MANAGING INTELLIGENCE OF NATURAL LAW

CS544 EA

Hibernate

Access

Access

- Hibernate can either:
  - Use the getters / setters methods
  - Or use reflection to get/set directly

- Which one it uses depends on where you place your **@Id** annotation

Field Access

- Examples so far always used field access
  - @Id placed directly on the field
  - Hibernate uses reflection to directly get/set field
  - All other annotations also **have to be** on the fields

```
@Entity
public class Customer {
      @Id
      @GeneratedValue
      private Long id;
      @Column(name="first", length=45, nullable=false)
      private String firstName;
      @Column(name="last", length=60, nullable=true)
      private String lastName;

      ...
}
```

@Id on the field indicates
Field Access

Property Access

- Place @Id on a getter for Property Access
  - All other annotations also **have to be on the getters**

```
@Entity
public class Customer2 {
      private Long id;
      private String firstName;
      private String lastName;
      private Date birthDate;
      private String temp;
      private String biography;

      @Id
      @GeneratedValue
      public Long getId() { return id; }
      @Column(name = "first", length = 45, nullable = false)
      public String getFirstName() { return firstName; }
      @Column(name = "last", length = 60, nullable = true)
      public String getLastName() { return lastName; }
      @Temporal(TemporalType.TIMESTAMP)
      public Date getBirthDate() { return birthDate; }
      @Transient
      public String getTemp() { return temp; }
      @Lob
      public String getBiography() { return biography; }
```

@Id on the getter indicates
Property Access
Hibernate will use
methods to get and set

Changing Access

- You can change access for individual fields

```
@Entity
public class Customer {
        @Id
        @GeneratedValue
        private Long id;
        @Access(AccessType.PROPERTY)
        private String firstName;
        private String lastName;

        ...
}
```

Everything will be accessed through field except firstName will use getters / setters

```
@Entity
public class Customer2 {
        private Long id;
        private String firstName;
        private String lastName;

        @Id
        @GeneratedValue
        public Long getId() { return id; }
        @Access(AccessType.FIELD)
        public String getFirstName() { return firstName; }
        public String getLastName() { return lastName; }

        ...
}
```

Everything will be accessed through methods (property) except firstName will use the field directly

Reflection

- Here is an example of how reflection works:

```
package cs544.hibernate01.data;

import java.lang.reflect.Field;

public class TestReflection {
      public static void main(String[] args)
                  throws NoSuchFieldException, SecurityException,
                        IllegalArgumentException, IllegalAccessException {

            Customer c = new Customer("George", "Simpson");
            // Possible NoSuchFieldException
            Field f = c.getClass().getDeclaredField("firstName");
            f.setAccessible(true);
            // Possible IllegalAccessException
            String fieldData = (String) f.get(c);
            System.out.println(fieldData);
      }
}
```

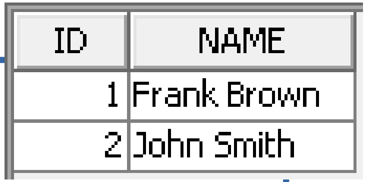
Encapsulation

- Using **reflection breaks** the OO principle of encapsulation
  - Property access hides implementation details
  - Next slide shows an example of hiding the implementation details

```
@Entity
public class Person {
        private Long id;
        private String firstname;
        private String lastname;

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

        public String getName() { return firstname + " " + lastname; }
        public void setName(String name) {
          StringTokenizer st = new StringTokenizer(name);
          firstname = st.nextToken();
          lastname = st.nextToken();
        }

        @Transient
        public String getFirstname() { return firstname; }
        public void setFirstname(String firstname) { this.firstname = firstname; }

        @Transient
        public String getLastname() { return lastname; }
        public void setLastname(String lastname) { this.lastname = lastname; }
}
```

Encapsulation used to map
3 instance variables to
2 database columns

Table: PERSON



ID	NAME
1	Frank Brown
2	John Smith