CHUYÊN ĐỀ JAVA

HIBERNATE MAPPING

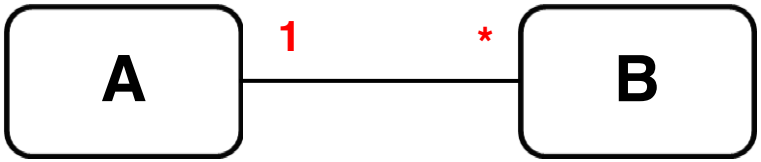
MANY-TO-ONE

Nguyễn Hoàng Anh
Email: nhanh@fit.hcmus.edu.vn
hoanganhis@gmail.com
ĐH KHTN, 2011

Nội dung trình bày

- Many to one
- Lazy Initialization
- Fetch
- Cascade

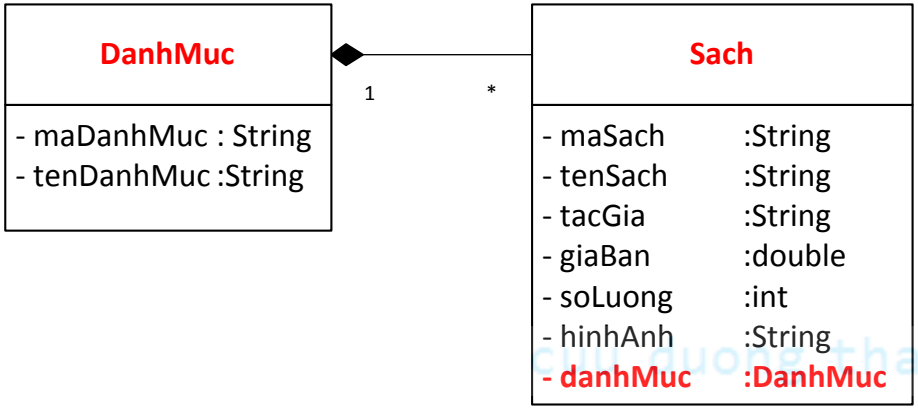
Mối quan hệ nhiều – một (many-to-one)

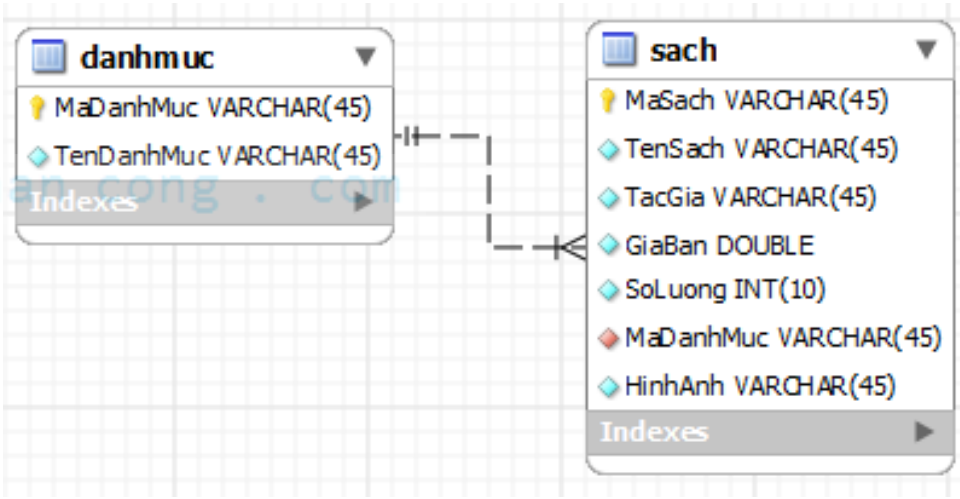


Mối quan hệ nhiều – một (many-to-one)

- Trong ứng dụng BookOnline
  - Mỗi đầu sách thuộc về một danh mục
  - Mỗi danh mục có thể có nhiều đầu sách
- Mối quan hệ hướng từ đầu sách đến danh mục được gọi là mối quan hệ nhiều – một (many-to-one)
- Nếu chỉ có mối quan hệ hướng từ sách đến danh mục gọi là mối quan hệ một chiều (unidirectional association)
- Nếu mối quan hệ hướng cả từ sách đến danh mục và ngược lại gọi là mối quan hệ hai chiều (bidirectional association)

BookOnline



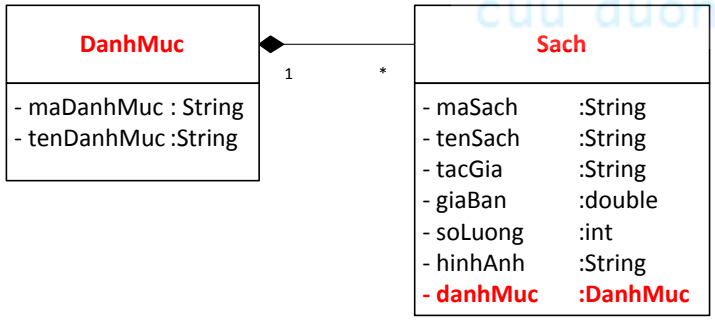


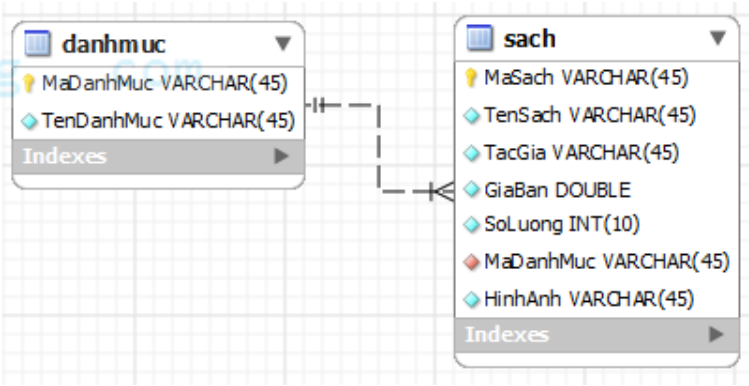
DanhMuc POJO

```
1  package pojo;
2  public class DanhMuc  implements java.io.Serializable {
3
4       private String maDanhMuc;
5       private String tenDanhMuc;
6
7      public DanhMuc() {
8      }
9
10     public DanhMuc(String maDanhMuc, String tenDanhMuc) {
11         this.maDanhMuc = maDanhMuc;
12         this.tenDanhMuc = tenDanhMuc;
13     }
14     //Getters & Setters
15
16 }
```

DanhMuc.hbm.xml

```
1  <hibernate-mapping>
2    <class catalog="bookonline" name="pojo.DanhMuc"
3           table="danhmuc">
4      <id name="maDanhMuc" type="string">
5        <column length="45" name="MaDanhMuc"/>
6        <generator class="assigned"/>
7      </id>
8      <property name="tenDanhMuc" type="string">
9        <column length="45" name="TenDanhMuc" not-null="true"/>
10     </property>
11   </class>
12 </hibernate-mapping>
```

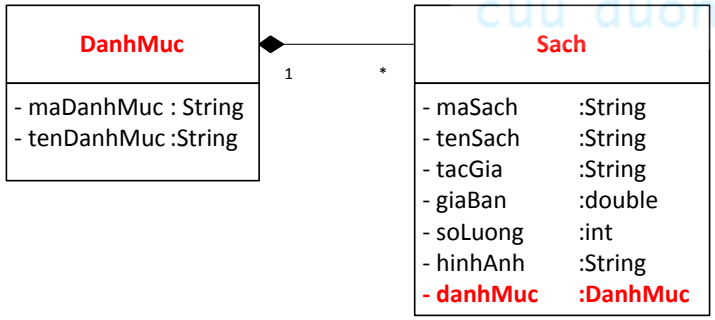


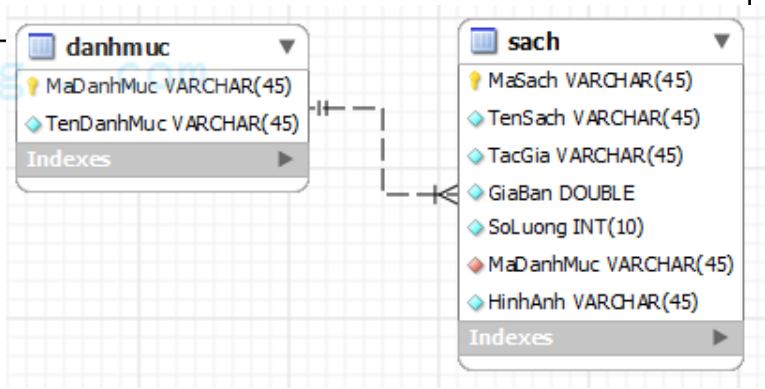
Sach POJO

```
1  package pojo;
2  public class Sach  implements java.io.Serializable {
3       private String maSach;
4       private DanhMuc danhMuc;
5       private String tenSach;
6       private String tacGia;
7       private double giaBan;
8       private int soLuong;
9       private String hinhAnh;
10
11      //Constructors
12
13      //Getters & Setters
14
15
16
```

Sach.hbm.xml

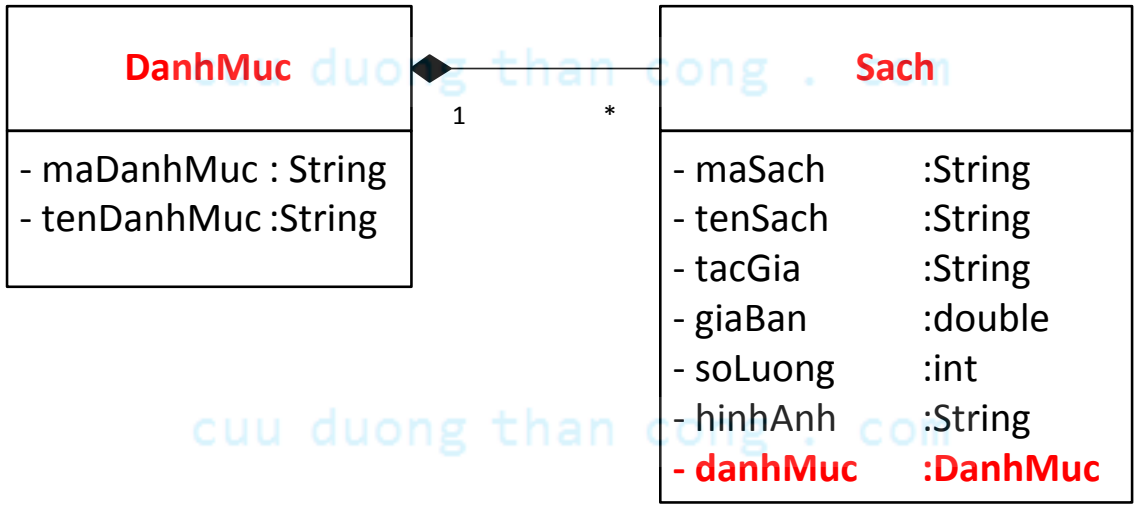
```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        <id name="maSach" type="string">
4           <column length="45" name="MaSach"/>
5           <generator class="assigned"/>
6        </id>
7        . . .
8        <many-to-one class="pojo.DanhMuc" name="danhMuc"
9                     fetch="select">
10          <column length="45" name="MaDanhMuc" not-null="true"/>
11       </many-to-one>
12    </class>
13 </hibernate-mapping>
```



SachDAO – lấy thông tin sách

- Viết phương thức lấy thông tin sách dựa vào mã sách cùng với danh mục mà nó thuộc về



SachDAO – lấy thông tin sách

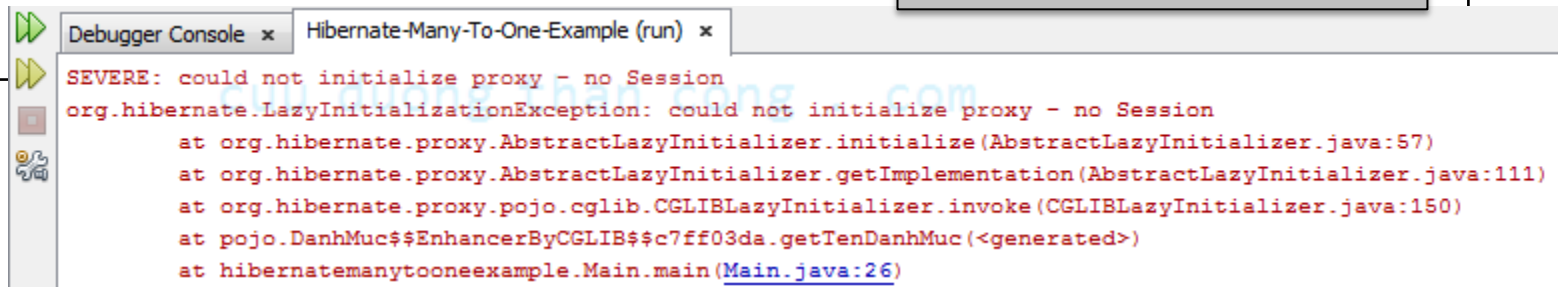
```
1  public class SachDAO {
2    public static Sach layThongTinSach(String maSach) {
3          Sach sach = null;
4          Session session = HibernateUtil.getSessionFactory()
5                                                  .openSession();
6          try {
7              sach = (Sach) session.get(Sach.class, maSach);
8          } catch (HibernateException ex) {
9              //Log the exception
10             System.err.println(ex);
11         } finally {
12             session.close();
13         }
14         return sach;
15     }
16 }
```

SachDAO – lấy thông tin sách

```
1  public class Main {
2
3      public static void main(String[] args) {
4
5          Sach sach = SachDAO.layThongTinSach("S001");
6          if (sach != null) {
7              DanhMuc dm = sach.getDanhMuc();
8              System.out.println(dm.getMaDanhMuc());
9              System.out.println(dm.getTenDanhMuc());
10         }
11     }
12 }
13
```

Mã danh mục được nạp

Mã danh mục:DM001
Exception xảy ra



SachDAO – lấy thông tin sách

```
1  public class SachDAO {
2    public static Sach layThongTinSach(String maSach) {
3          Sach sach = null;
4          Session session = HibernateUtil.getSessionFactory()
5                                                  .openSession();
6          try {
7              sach = (Sach) session.get(Sach.class, maSach);
8
9              DanhMuc dm=sach.getDanhMuc();
10             System.out.println(dm.getMaDanhMuc());
11             System.out.println(dm.getTenDanhMuc());
12         } catch (HibernateException ex) {
13             //Log the exception
14             System.err.println(ex);
15         } finally {session.close();}
16         return sach;
17     }
18 }
```

OK

Mã danh mục:DM001
Danh mục: Java

SachDAO – lấy thông tin sách

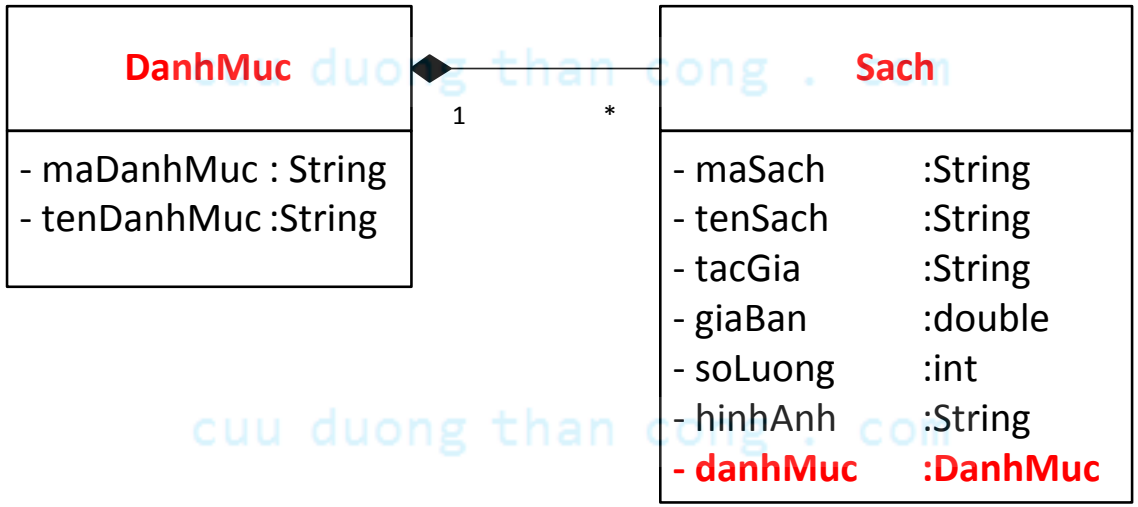
- Khi truy xuất các thuộc tính bên trong danh mục (ngoài mã danh mục) từ đối tượng sách bên trong session thì hoàn toàn được.
- Nhưng khi truy xuất các thuộc tính bên trong danh mục (ngoài mã danh mục) từ đối tượng sách bên ngoài session thì exception sẽ xảy ra.
- Trong Hibernate cơ chế này được gọi là Lazy Initialization

Lazy Initialization

- Trong Hibernate, Lazy Initialization giúp
  - Tránh các câu truy vấn cơ sở dữ liệu không cần thiết
  - Gia tăng hiệu suất thực thi
  - Lazy mặc định có giá trị là **true**

SachDAO – lấy thông tin sách

- Viết phương thức lấy thông tin sách dựa vào mã sách cùng với danh mục mà nó thuộc về



Cách 1 – lấy thông tin danh mục từ mã danh mục

```
1  public class Main {
2
3      public static void main(String[] args) {
4
5          Sach sach = SachDAO.layThongTinSach("S001");
6          if (sach != null) {
7            String maDanhMuc=sach.getDanhMuc().getMaDanhMuc();
8            DanhMuc dm = DanhMucDAO.layThongTinDanhMuc(maDanhMuc);
9            System.out.println(dm.getMaDanhMuc());
10           System.out.println(dm.getTenDanhMuc());
11         }
12     }
13 }
14
15
16
17
18
```

OK

Mã danh mục:DM001
Danh mục: Java

Cách 2 – Sử dụng Hibernate.initialize(Object obj)

```
1   public class SachDAO {
2     public static Sach layThongTinSach(String maSach) {
3           Sach sach = null;
4           Session session = HibernateUtil.getSessionFactory()
5                                                   .openSession();
6           try {
7               sach = (Sach) session.get(Sach.class, maSach);
8               Hibernate.initialize(sach.getDanhMuc());
9           } catch (HibernateException ex) {
10              //Log the exception
11              System.err.println(ex);
12          } finally {
13              session.close();
14          }
15          return sach;
16      }
    }
```

Cách 2 – Sử dụng Hibernate.initialize(Object obj)

```
1  public class Main {
2
3      public static void main(String[] args) {
4
5          Sach sach = SachDAO.layThongTinSach("S001");
6          if (sach != null) {
7            DanhMuc dm = sach.getDanhMuc();
8            System.out.println(dm.getMaDanhMuc());
9            System.out.println(dm.getTenDanhMuc());
10         }
11     }
12 }
13
14
15
16
17
18
```

Mã danh mục:DM001
Danh mục: Java

Cách 3: điều chỉnh thuộc tính lazy trong Sach.hbm.xml

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        <id name="maSach" type="string">
4           <column length="45" name="MaSach"/>
5           <generator class="assigned"/>
6        </id>
7        . . .
8        <many-to-one class="pojo.DanhMuc" name="danhMuc"
9                     fetch="select" lazy="false" >
10          <column length="45" name="MaDanhMuc" not-null="true"/>
11       </many-to-one>
12    </class>
13 </hibernate-mapping>
```

Cách 3: điều chỉnh thuộc tính lazy trong Sach.hbm.xml

```
1  public class SachDAO {
2    public static Sach layThongTinSach(String maSach) {
3          Sach sach = null;
4          Session session = HibernateUtil.getSessionFactory()
5                                                  .openSession();
6          try {
7              sach = (Sach) session.get(Sach.class, maSach);
8          } catch (HibernateException ex) {
9              //Log the exception
10             System.err.println(ex);
11         } finally {
12             session.close();
13         }
14         return sach;
15     }
16 }
```

Cách 3: điều chỉnh thuộc tính lazy trong Sach.hbm.xml

```
1  public class Main {
2
3      public static void main(String[] args) {
4
5          Sach sach = SachDAO.layThongTinSach("S001");
6          if (sach != null) {
7            DanhMuc dm = sach.getDanhMuc();
8            System.out.println(dm.getMaDanhMuc());
9            System.out.println(dm.getTenDanhMuc());
10         }
11     }
12 }
```

Mã danh mục:DM001
Danh mục: Java

Fetch

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        . . .
4        <many-to-one class="pojo.DanhMuc" name="danhMuc"
5                     fetch="select" lazy="false" >
6           <column length="45" name="MaDanhMuc" not-null="true"/>
7        </many-to-one>
8     </class>
9  </hibernate-mapping>
```


- Việc tắt lazy initialization cùng với cơ chế nạp đối tượng sách và danh mục theo cơ chế fetch = "select" , khi đó theo cách này 2 câu lệnh select được phát sinh để truy vấn lấy thông tin đối tượng sách và đối tượng danh mục.

- Điều này có thể không hiệu quả bởi vì cần truy xuất vào cơ sở dữ liệu và thực hiện truy vấn hai lần.

Fetch

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        . . .
4        <many-to-one class="pojo.DanhMuc" name="danhMuc"
5                     fetch="join" lazy="false" >
6           <column length="45" name="MaDanhMuc" not-null="true"/>
7        </many-to-one>
8     </class>
9  </hibernate-mapping>
```


- Thay cơ chế fetch = “select” thành fetch = “join”, khi đó theo cách này 1 câu lệnh select được phát sinh duy nhất bằng cách kết bảng để truy vấn lấy thông tin cho đối tượng sách và đối tượng danh mục.
- fetch = “join” sẽ hiệu quả bởi vì chỉ cần truy xuất vào cơ sở dữ liệu và thực hiện truy vấn một lần.

Fetch

```
1  public class SachDAO {
2      public static Sach layThongTinSach(String maSach) {
3          Sach sach = null;
4          Session session = HibernateUtil.getSessionFactory()
5                                          .openSession();
6          try {
7              String hql="select s from Sach s
8                                  where s.maSach=:maSach";
9              Query query=session.createQuery(hql);
10             query.setString("maSach", maSach);
11             sach = (Sach) query.uniqueResult();
12         } catch (HibernateException ex) {
13             System.err.println(ex);
14         } finally {session.close();}
15         return sach;
16     }
17 }
18
19
```

- Trong trường hợp này 2 câu lệnh select vẫn được phát sinh để truy vấn lấy thông tin đối tượng sách và đối tượng danh mục.
- Bởi vì câu truy vấn HQL đều sẽ được thông dịch trực tiếp ra câu lệnh SQL

Fetch

```
1  public class SachDAO {
2      public static Sach layThongTinSach(String maSach) {
3          Sach sach = null;
4          Session session = HibernateUtil.getSessionFactory()
5                                          .openSession();
6          try {
7              String hql="select s
8                          from Sach s left join fetch s.danhMuc
9                          where s.maSach=:maSach";
10             Query query=session.createQuery(hql);
11             query.setString("maSach", maSach);
12             sach = (Sach) query.uniqueResult();
13         } catch (HibernateException ex) {
14             System.err.println(ex);
15         } finally { session.close();}
16         return sach;
17     }
18 }
```

Fetch

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        . . .
4        <many-to-one class="pojo.DanhMuc" name="danhMuc"
5                     fetch="join" lazy="false" >
6           <column length="45" name="MaDanhMuc" not-null="true"/>
7        </many-to-one>
8     </class>
9  </hibernate-mapping>
```

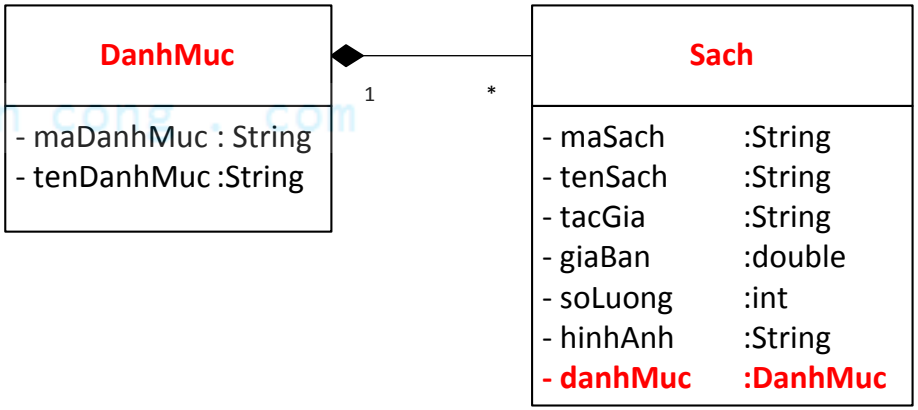

- Sử dụng truy vấn left join trong HQL để nạp thông tin cho các đối tượng lazy (ví dụ danh mục trong sách)
- Do đó các đối tượng lazy này có thể được truy xuất bên ngoài session
- Cách này thường chọn để tăng tốc độ thực thi và tùy trường hợp có nạp hay không nạp thông tin cho đối tượng lazy.

Cascade

- Cascade thường dùng
  - none
  - delete
  - save-update

Cascade : none

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        . . .
4        <many-to-one class="pojo.DanhMuc" name="danhMuc"
5                     fetch="join" lazy="false" cascade="none">
6           <column length="45" name="MaDanhMuc" not-null="true"/>
7        </many-to-one>
8     </class>
9  </hibernate-mapping>
```

Cascade : none

```
1  public class SachDAO {
2      public static boolean themSach(Sach sach) {
3          Session session = HibernateUtil.getSessionFactory()
4                                                  .openSession();
5          if (SachDAO.layThongTinSach(sach.getMaSach()) != null) {
6              return false;
7          }
8          boolean kq = true;
9          Transaction transaction = null;
10         try {
11             transaction = session.beginTransaction();
12             session.save(sach);
13             transaction.commit();
14         } catch (HibernateException ex) {
15             transaction.rollback();
16             System.err.println(ex); kq = false;
17         } finally {session.close();}
18         return kq;
19     }
```

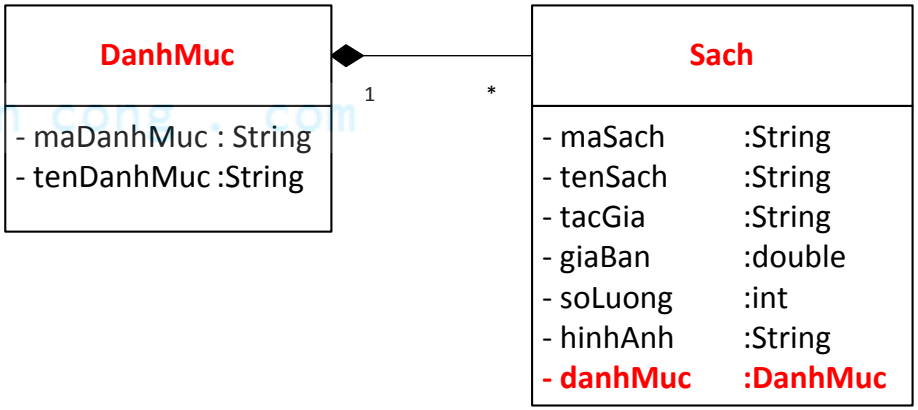
Cascade : none

```
1  public class Main {
2
3       public static void main(String[] args) {
4          Sach sach=new Sach();
5          sach.setMaSach("S027");
6          sach.setTenSach("Hibernate 3");
7          sach.setGiaBan(200000);
8          sach.setSoLuong(1000);
9          sach.setHinhAnh("images/hibernate3.jpg");
10         sach.setTacGia("Nguyễn Hoàng Anh");
11         DanhMuc dm = new DanhMuc("DM013", "Java2");
12         sach.setDanhMuc(dm);
13         boolean kq=SachDAO.themSach(sach);
14      }
15
16 }
```


- Thêm sách thất bại khi
  - + Danh mục chưa tồn tại trong cơ sở dữ liệu
  - + Mã sách đã tồn tại trong cơ sở dữ liệu

Cascade : save-update

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        . . .
4        <many-to-one class="pojo.DanhMuc" name="danhMuc"
5            fetch="join" lazy="false" cascade="save-update">
6           <column length="45" name="MaDanhMuc" not-null="true"/>
7        </many-to-one>
8     </class>
9  </hibernate-mapping>
```

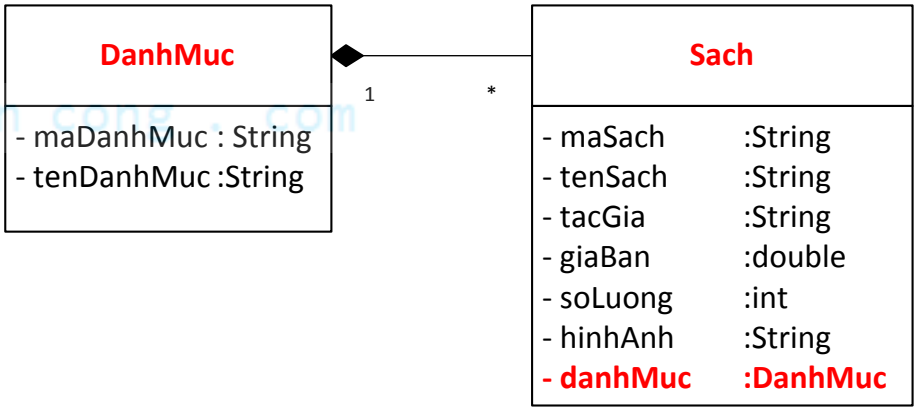
Cascade : save-update

```
1  public class Main {
2
3       public static void main(String[] args) {
4          Sach sach=new Sach();
5          sach.setMaSach("S027");
6          sach.setTenSach("Hibernate 3");
7          sach.setGiaBan(200000);
8          sach.setSoLuong(1000);
9          sach.setHinhAnh("images/hibernate3.jpg");
10         sach.setTacGia("Nguyễn Hoàng Anh");
11         DanhMuc dm = new DanhMuc("DM013", "Java2");
12         sach.setDanhMuc(dm);
13         boolean kq=SachDAO.themSach(sach);
14      }
15
16 }
```

- Nếu danh mục chưa tồn tại, hibernate sẽ tạo danh mục vào cơ sở dữ liệu

Cascade : save-update

```
1  <hibernate-mapping>
2     <class catalog="bookonline" name="pojo.Sach" table="sach">
3        . . .
4        <many-to-one class="pojo.DanhMuc" name="danhMuc"
5          fetch="join" lazy="false" cascade="save-update, delete">
6           <column length="45" name="MaDanhMuc" not-null="true"/>
7        </many-to-one>
8     </class>
9  </hibernate-mapping>
```

Tài liệu tham khảo

- Nguyễn Hoàng Anh, Tập bài giảng và video môn chuyên đề Java, 2010
- Gary Mak, Tập hướng dẫn từng bước Hibernate, 2006

HỎI VÀ ĐÁP