

CS 683 Emerging Technologies
Fall Semester, 2004
Doc 30 One-to-Many & HQL
Contents

References.....	2
Many-to-One.....	3
Classes.....	3
Tables.....	4
Mapping Files.....	5
Some Mapping Options.....	8
Class.....	8
id.....	9
types.....	10
Generate Class.....	11
Property.....	12
Mapping Collections.....	13
Collection Mappings.....	16
Many-to-one.....	17
Variations.....	22
Queries - HQL.....	25
Querying a Class.....	27
Controlling Number of Hits.....	28
Getting One Result.....	29
Aliases.....	30
Restricting Queries.....	31
Logical Operators.....	32
Ordering Query Results.....	33
Properties of Properties.....	34

Copyright ©, All rights reserved. 2004 SDSU & Roger Whitney,
5500 Campanile Drive, San Diego, CA 92182-7700 USA.
OpenContent (<http://www.opencontent.org/opl.shtml>) license
defines the copyright on this document

References

Hibernate Reference 2.1.6

Online html version

http://www.hibernate.org/hib_docs/reference/en/html/

Other versions (including Chinese) at:

<http://www.hibernate.org/5.html>

Hibernate in Action, Bauer & King

Many-to-One Classes

```
public class Person {  
    String firstName;  
    String lastName;  
    Set addresses;  
    long id;
```

```
public class EmailAddress {  
    String userName;  
    String host;  
    long id;
```

Tables People

ID	FIRST_NAME	LAST_NAME
1	Roger	Whitney
2	Leland	Beck

ID	USER_NAME	HOST	PERSON_ID
1	whitney	cs.sdsu.edu	1
2	whitney	rohan.sdsu.edu	1
3	beck	cs.sdsu.edu	2
4	whitney	math.sdsu.edu	1

```
CREATE TABLE PEOPLE (
  ID serial NOT NULL ,
  FIRST_NAME varchar(50) NULL ,
  LAST_NAME varchar(50) NULL ,
  CONSTRAINT PEOPLE_PK PRIMARY KEY (id),
  CONSTRAINT PEOPLE_UNIQ UNIQUE (id))
```

```
CREATE TABLE EMAIL_ADDRESSES (
  USER_NAME varchar(50) NULL ,
  HOST varchar(50) NULL ,
  ID serial NOT NULL ,
  PERSON_ID int4 NULL ,
  CONSTRAINT EMAIL_ADDRESSES_PK PRIMARY KEY (id),
  CONSTRAINT EMAIL_ADDRESSES_UNIQ UNIQUE (id))
```

Mapping Files

EmailAddress.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >

<hibernate-mapping package="cs683">
  <class
    name="EmailAddress"
    table="EMAIL_ADDRESSES" >
    <id
      name="id"
      type="long"
      column="id" >
      <generator class="increment"/>
    </id>
    <property
      name="userName"
      column="user_name"
      type="string"
      not-null="false"
      length="50" />
    <property
      name="host"
      column="host"
      type="string"
      not-null="false"
      length="50" />
    </class>
</hibernate-mapping>
```

Person.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd" >

<hibernate-mapping package="cs683">
  <class
    name="Person"
    table="people"
  >
    <id
      name="id"
      type="long"
      column="id"
    >
      <generator class="increment"/>
    </id>

    <set name="addresses"
      inverse="false"
      cascade="save-update"
      table="EMAIL_ADDRESSES">
      <key column="person-id"/>

      <one-to-many class="cs683.EmailAddress" />
    </set>
```

Person.hbm.xml Continued

```
<property
  name="lastName"
  column="last_name"
  type="string"
  not-null="false"
  length="50"
/>
<property
  name="firstName"
  column="first_name"
  type="string"
  not-null="false"
  length="50"
/>

</class>
</hibernate-mapping>
```

Some Mapping Options Class

```
<class
  name="ClassName"
  table="tableName"

  mutable="true|false"      (default: true)
  dynamic-update="true|false" (default: false)
  dynamic-insert="true|false" (default: false)
  where="arbitrary sql where condition"
  batch-size="N"
  lazy="true|false"
/>
```

http://www.hibernate.org/hib_docs/reference/en/html/mapping.html#mapping-declaration-class

id

```
<id
  name="propertyName"
  type="typename"
  column="column_name"

  unsaved-value="any|none|null|id_value"      (default: null)
  access="field|property|ClassName">        (default: property)

  <generator class="generatorClass"/>
</id>
```

http://www.hibernate.org/hib_docs/reference/en/html/mapping.html#mapping-declaration-id

types

- integer, long, short, float, double, character, byte, boolean
Java primitives or wrapper classes to appropriate SQL types
- string
java.lang.String to VARCHAR (or Oracle VARCHAR2)
- date, time, timestamp
java.util.Date to SQL types DATE, TIME and TIMESTAMP
- calendar, calendar_date
java.util.Calendar to SQL types TIMESTAMP and DATE
- big_decimal
java.math.BigDecimal to NUMERIC (or Oracle NUMBER)
- locale, timezone, currency
java.util.Locale, java.util.TimeZone and java.util.Currency to VARCHAR (or Oracle VARCHAR2)
- binary
byte arrays to an appropriate SQL binary type.
- text
Maps long Java strings to a SQL CLOB or TEXT type.
- serializable
Maps serializable Java types to an appropriate SQL binary type

http://www.hibernate.org/hib_docs/reference/en/html/mapping.html#mapping-types

Generate Class

- native

Picks identity, sequence or hilo depending upon the underlying database.

- increment

Generates unique identifiers

- identity

Identity columns in DB2, MySQL, MS SQL Server, Sybase and HypersonicSQL

- sequence

Sequence in DB2, PostgreSQL, Oracle, SAP DB, McKoi

- hilo

uses a hi/lo algorithm to efficiently generate identifiers

- uuid.hex 128-bit UUID algorithm

- assigned

The application to assign an identifier

- foreign

Uses the identifier of another associated object. Usually used in conjunction with a <one-to-one> primary key association.

Property

```
<property
  name="propertyName"

  column="column_name"      (default: property name)
  type="typename"
  update="true|false"      (default: true)
  insert="true|false"      (default: true)
  formula="arbitrary SQL expression"
  access="field|property|ClassName" (default: property)
/>
```

update – should property be included in updates

insert – should property be included in inserts

formula – used to compute a value rather than reading it

access – how Hibernate accesses the value in the object

http://www.hibernate.org/hib_docs/reference/en/html/mapping.html#mapping-declaration-property

Mapping Collections

Hibernate supports:

- Set
- Bag
- List
- Maps

Java Collection Information

API

<http://java.sun.com/j2se/1.4.2/docs/api/java/util/package-summary.html>

Overview

<http://java.sun.com/j2se/1.4.2/docs/guide/collections/index.html>

Set

Collection that does not allow duplicate entries

No order on the elements

Hibernate has own Set implementation

You can use any class that implements Set interface

`java.util.HashSet`

`java.util.TreeSet`

Bag

Collection that allows duplicate entries

No order on the elements

Hibernate has own Bag implementation

You can use any class that with the `java.util.List` interface

Hibernate does not maintain original order of elements

List Mapping

Collection that maintains order

Requires additional column in table

Map Mapping

Map is a collection indexed by keys (Dictionary)

Requires additional column in table for keys

Collection Mappings

```
<map
  name="propertyName"

  table="table_name"           (default: property name)
  schema="schema_name"
  lazy="true|false"           (default: false)
  inverse="true|false"        (default: false)
  cascade="all|none|save-update|delete|all-delete-orphan"
    (default: none)
  sort="unsorted|natural|comparatorClass"
  order-by="column_name asc|desc"
  where="arbitrary sql where condition"
  outer-join="true|false|auto"
  batch-size="N"               (default: 1)
  access="field|property|ClassName" (default: property)
>

  <key .... />
  <index .... />
  <element .... />
</map>
```

http://www.hibernate.org/hib_docs/reference/en/html/collections.html#collections-mapping

Many-to-one

```
<many-to-one
  name="propertyName"
  column="column_name"
  class="ClassName"
  cascade="all|none|save-update|delete"
  outer-join="true|false|auto"
  update="true|false"
  insert="true|false"
  property-ref="propertyNameFromAssociatedClass"
  access="field|property|ClassName"
  unique="true|false"
/>
```

Person Class

```
package cs683;

import java.util.HashSet;
import java.util.Set;

public class Person {
    String firstName;
    String lastName;
    Set addresses = new HashSet();
    long id;

    public Person () { super(); }

    public Person(String first, String last) {
        firstName = first;
        lastName = last;
    }

    public String getLastName() { return lastName; }
    public String getFirstName() { return firstName; }

    public void setFirstName( String name) { firstName = name; }
    public void setLastName( String name) {      lastName = name; }

    public long getId() { return id; }
    public void setId(long l) {id = l; }

    public String toString() {      return firstName + " " + lastName + id; }
    public Set getAddresses() { return addresses; }
    public void setAddresses(Set addresses) { this.addresses = addresses; }
    public void addAddress(EmailAddress newAddress) {
        addresses.add( newAddress);
    }
}
```

EmailAddress class

```
package cs683;

public class EmailAddress {
    String userName;
    String host;
    long id;

    public EmailAddress() { }

    public EmailAddress(String user, String host) {
        userName = user;
        this.host = host;
    }

    public String getHost() { return host; }

    public void setHost(String host) { this.host = host; }

    public long getId() { return id; }

    public void setId(long id) { this.id = id; }

    public String getUserName() { return userName; }

    public void setUserName(String userName) {
        this.userName = userName;
    }
}
```

Main

```
package cs683;

import java.util.List;
import java.util.Set;

import net.sf.hibernate.HibernateException;
import net.sf.hibernate.MappingException;
import net.sf.hibernate.Query;
import net.sf.hibernate.Session;
import net.sf.hibernate.SessionFactory;
import net.sf.hibernate.Transaction;
import net.sf.hibernate.cfg.Configuration;

public class Main {
    public static void main(String[] args) throws Exception {
        sampleWrite();
        sampleRead();
    }

    static Session getHibernateSession() throws MappingException,
        HibernateException, Exception
    {
        Configuration config = new Configuration();
        config.addClass(cs683.Person.class);
        config.addClass(cs683.EmailAddress.class);

        SessionFactory sessions = config.buildSessionFactory();
        Session session = sessions.openSession();
        return session;
    }
}
```

Main Continued

```
static void sampleWrite() throws MappingException, HibernateException,
    Exception {
    Session session = getHibernateSession();
    Transaction save = session.beginTransaction();
    Person newPerson = new Person("Susan", "LittleBit");
    session.save(newPerson);
    EmailAddress bar = new EmailAddress("foo", "bar.aol.com");
    session.save(bar);
    EmailAddress cat = new EmailAddress("catwoman", "gmail.com");
    session.save(cat);
    newPerson.addAddress(cat);
    newPerson.addAddress(bar);
    save.commit();
    session.close();
}
```

```
static void sampleRead() throws MappingException, HibernateException,
    Exception {
    Session session = getHibernateSession();
    Query getPerson =
        session.createQuery(" from Person p where p.lastName = 'LittleBit'");
    List result = getPerson.list();
    System.out.println("Number of Objects: " + result.size());
    System.out.println(result.get(0));
    Set addresses = ((Person) result.get(0)).getAddresses();
    System.out.println(addresses.getClass());
    session.close();
}
}
```

Variations Cascading Saves

Person contains a collection of email addresses

When saving a person to also save the email addresses

- Cascade must be on

```
<set name="addresses" inverse="false" cascade="save-update"  
      table="EMAIL_ADDRESSES">  
  <key column="person_id"/>  
  
  <one-to-many class="cs683.EmailAddress" />  
</set>
```

- Email Address must be registered with the session

```
Person newPerson = new Person("Susan", "LittleBit");  
session.save(newPerson);  
EmailAddress bar = new EmailAddress("foo", "bar.aol.com");  
session.save(bar);  
EmailAddress cat = new EmailAddress("catwoman", "gmail.com");  
session.save(cat);  
newPerson.addAddress(cat);  
newPerson.addAddress(bar);
```

Lazy Fetching

Only read email addresses from database when they are needed

```
<set name="addresses"  
  inverse="false"  
  cascade="save-update"  
  lazy=true  
  table="EMAIL_ADDRESSES">  
  <key column="person_id"/>
```

```
static void sampleRead() throws MappingException, HibernateException,  
    Exception {  
    Session session = getHibernateSession();  
    Query getPerson =  
        session.createQuery(" from Person p where p.lastName = 'LittleBit'");  
    List result = getPerson.list();  
    System.out.println("Number of Objects: " + result.size());  
    System.out.println(result.get(0));  
  
    //Next line reads EmailAddress from database  
    Set addresses = ((Person) result.get(0)).getAddresses();  
    System.out.println(addresses.getClass());  
    session.close();  
}
```

Beware of Lazy Fetching

```
static void sampleRead() throws MappingException, HibernateException,
    Exception {
    Session session = getHibernateSession();
    Query getPerson =
        session.createQuery(" from Person p where p.lastName = 'LittleBit'");
    List result = getPerson.list();
    session.close();

    System.out.println("Number of Objects: " + result.size());
    System.out.println(result.get(0));

    Set addresses = ((Person) result.get(0)).getAddresses();
    System.out.println(addresses.getClass());
}
```

If lazy=false the above works fine

If lazy=true the above does not work

Queries - HQL

Hibernate Query Language

Docs

http://www.hibernate.org/hib_docs/reference/en/html/query_hql.html

Hibern8ide

Java program to test queries

```
Session session = getHibernateSession();
```

```
String queryString = " from Person p where p.lastName = 'Whitney' ";
```

```
Query getPerson = session.createQuery( queryString);
```

```
List result = getPerson.list();
```

```
session.close();
```

Examples of queries will just show the queryString

Examples will run in Hibern8ide as given

Running Hibern8ide

```
public class Main
{
    public static void main(String[] args) throws Exception
    {
        Hibern8IDE.main(args);
    }
}
```

Classpath needs all the normal Hibernate jar files plus:

- hibern8ide/hibern8ide.jar
- hibern8ide/lib/*.jar

where hibern8ide is part of the hibernate extensions download

Querying a Class

```
from Person  
from EmailAddress
```

```
from java.lang.Object
```

```
from x
```

- Returns all object of type x from database
- Includes objects of subclasses of x

Controlling Number of Hits

```
Session session = getHibernateSession();
```

```
String queryString = " from Person p where p.lastName = 'Whitney' ";
```

```
Query getPerson = session.createQuery( queryString);
```

```
getPerson.setFirstResult(5);
```

```
getPerson.setMaxResults(20);
```

```
List result = getPerson.list();
```

```
session.close();
```

`setFirstResult(5)`

the first result returned is the fifth one found in the database

`setMaxResults(20);`

Return at most 20 results

Getting One Result

```
Session session = getHibernateSession();
```

```
String queryString = " from Person p where p.lastName = 'Whitney' ";
```

```
Query getPerson = session.createQuery( queryString);
```

```
getPerson.setMaxResults(1);
```

```
Person whitney = (Person) getPerson.uniqueResult();
```

```
session.close();
```

Don't need a list if know we will get one result

Aliases

from Person p where p.lastName = 'Whitney'

from Person aPerson where person.lastName = 'Whitney'

from Person as aPerson where person.lastName = 'Whitney'

p & aPerson are aliases for Person object in rest of query

Restricting Queries

from Person p where p.lastName = 'Whitney'

from Person p where p.lastName like 'W%'

from Person p where p.lastName not like 'W%'

from Person p where p.lastName in ('Whitney', 'Tester')

from Person p where p.age between 21 and 50

from Person p where p.age < 21

from Person p where p.lastName is null

from Person p where p.lastName is not null

like wildcard characters

- % matches any number of characters
- _ matches a single character

Basic SQL operators supported

=	<>	<
>	>=	<=
between	not between	in
not in	is null	is not null
+	-	*
/		upper()

Logical Operators

```
from Person p
  where p.lastName like 'W%'
  and p.firstName in ('Roger', 'Paul')
```

```
from Person p
  where (p.lastName like 'W%' and p.firstName in ('Roger', 'Paul') )
  or p.lastName = 'Chen'
```

“and” “or” and “()” can be used to combine expressions

Ordering Query Results

from Person p order by p.lastName

from Person p order by p.lastName asc

from Person p order by p.lastName desc

from Person p order by p.lastName asc, p.firstName

from Person p where p.lastName = 'Whitney' order by p.firstName

Properties of Properties

from People p where p.address.city = 'La Jolla'

from Person p where p.addresses.size > 1