

CS 683 Emerging Technologies
Fall Semester, 2004
Doc 31 Joins & Many-to-One
Contents

References	2
Joins	3
Inner Join (or just Join)	4
Outer Join	5
Many-to-One Bidirectional	8
Classes	8
Tables	9
Mapings	10
Methods	13
Write, Update & Delete Cascading	21
Some Useful Session Methods	22
Logging – log4j	23

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

Log4j Documentation, <http://logging.apache.org/log4j/docs/>

Joins

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck
3	Carl	Eckberg

Email_Addresses

id	user_name	host	person_id
1	beck	cs.sdsu.edu	2
2	whitney	cs.sdsu.edu	1
3	whitney	rohan.sdsu.edu	1
4	foo	rohan.sdsu.edu	

The tables have a column in common as `email_addresses.person_id` refers to `people.id`. So we can create a new table by joining the two tables together on that column

Inner Join (or just Join)

Only uses entries linked in two tables

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu

```
select
  first_name, last_name, user_name, host
from
  people, email_addresses
where
  people.id = email_addresses.person_id;
```

or equivalently

```
select
  first_name, last_name, user_name, host
from
  people inner join email_addresses
on
  (people.id = email_addresses.person_id);
```

Outer Join

Uses all entries from a table

Left Outer Join

Use all entries from the left table

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu
Carl	Eckberg		

```
select
  first_name, last_name, user_name, host
from
  people left outer join email_addresses
on
  (people.id = email_addresses.person_id);
```

Right Outer Join

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu
		foo	rohan.sdsu.edu

Use all entries from the right table

```
select
  first_name, last_name, user_name, host
from
  people right outer join email_addresses
on
  (people.id = email_addresses.person_id);
```

A right outer join B & B left outer join A

The following two statements are equivalent

```
select
  first_name, last_name, user_name, host
from
  people right outer join email_addresses
on
  (people.id = email_addresses.person_id);
```

```
select
  first_name, last_name, user_name, host
from
  email_addresses left outer join people
on
  (people.id = email_addresses.person_id);
```

Many-to-One Bidirectional Classes

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

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

Tables

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck
3	Carl	Eckberg

Email_Addresses

id	user_name	host	person_id
1	beck	cs.sdsu.edu	2
2	whitney	cs.sdsu.edu	1
3	whitney	rohan.sdsu.edu	1
4	foo	rohan.sdsu.edu	

```
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))
```

Mapings

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" />
    <many-to-one name="owner" column="person_id" class="Person"
not-null="true">
    </many-to-one>
  </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="all"
      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>
```

No changes from Document 30

Methods

Write

```
static void sampleWrite() throws MappingException, HibernateException,
    Exception
{
    Session session = getHibernateSession();
    Transaction save = null;
    try
    {
        save = session.beginTransaction();
        Person newPerson = new Person("Susan", "Many");

        EmailAddress bar = new EmailAddress("foo", "bar.aol.com");
        EmailAddress cat = new EmailAddress("catwoman", "gmail.com");
        newPerson.addAddress(cat);
        newPerson.addAddress(bar);
        session.save(bar);
        session.save(cat);
        session.save(newPerson);
        save.commit();
    }
    catch (Exception problem)
    {
        if (save != null)
            save.rollback();
        throw problem;
    }
    finally
    {
        session.close();
    }
}
```

Set all the links before saving

Read

static void sampleRead() throws MappingException, HibernateException,
Exception

```
{  
String query = "select p from Person p join p.addresses address where  
address.host like '%gmail%'"  
Session session = getHibernateSession();  
Query getGmailUser =  
session.createQuery(query);  
List result = getGmailUser.list();  
System.out.println("Number of People: " + result.size());  
  
Object first =result.get(0);  
System.out.println( first);  
session.close();  
}
```

Select

Select indicates which type of object to return

```
select
  address
from
  Person p join p.addresses address
where
  address host like '%gmail%'
```

```
select
  p
from
  Person p join p.addresses address
where
  address host like '%gmail%'
```

Select without a Join

```
select p from Person p where p.lastName = 'Beck'
```

Joins

Hibernate supports

- Inner join
- Left outer join
- Right outer join

Inner join

Join defaults to inner

The following are equivalent

```
select
  p
from
  Person p join p.addresses address
```

```
select
  p
from
  Person p inner join p.addresses address
```

Outer Joins

The left join will get Eckberg

```
select
  p
from
  Person p left join p.addresses address
where
  address host like '%gmail%' or p.lastName like 'E%'
```

The right join will not get Eckberg

```
select
  p
from
  Person p right join p.addresses address
where
  address host like '%gmail%' or p.lastName like 'E%'
```

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck
3	Carl	Eckberg

Email_Addresses

id	user_name	host	person_id
1	beck	cs.sdsu.edu	2
2	whitney	cs.sdsu.edu	1
3	whitney	rohan.sdsu.edu	1
4	foo	rohan.sdsu.edu	

Selecting Multiple Objects

```
select p, address from Person p join p.addresses address
```

```
Session session = getHibernateSession();
Query getGmailUser = session
    .createQuery("select p, address from Person p join p.addresses address");

List result = getGmailUser.list();
System.out.println("Number of People: " + result.size());

for (int k = 0; k < result.size(); k++)
{
    Object[] resultArray = (Object[]) result.get(k);
    Person personResult = (Person) resultArray[0];
    EmailAddress addressResult = (EmailAddress) resultArray[1];
    System.out.println("" + personResult + " " + addressResult);
}

session.close();
```

Update

```
static void sampleUpdate() throws MappingException,  
HibernateException,  
Exception  
{  
    Session session = getHibernateSession();  
    Transaction update = session.beginTransaction();  
    Query getGmailUser = session.createQuery(  
        "select p from Person p join p.addresses address where  
        address.host='gmail.com'");  
  
    List result = getGmailUser.list();  
    Person gmailUser = (Person) result.get(0);  
  
    gmailUser.setFirstName("Roger");  
    Iterator listEmails = gmailUser.getAddresses().iterator();  
    while (listEmails.hasNext())  
    {  
        EmailAddress anAddress =(EmailAddress) listEmails.next();  
        if (anAddress.getHost().equals("gmail.com"))  
            anAddress.setHost("gmail.google.com");  
    }  
    update.commit();  
    session.close();  
}
```

Delete

```
static void sampleDelete() throws MappingException, HibernateException,
    Exception
{
    Session session = getHibernateSession();
    Transaction deleter = null;
    try
    {
        deleter = session.beginTransaction();
        Query getGmailUser = session.createQuery(
            "select p from Person p join p.addresses address where
            address.host like '%gmail%'");
        Person result = (Person) getGmailUser.uniqueResult();
        session.delete(result);
        deleter.commit();
    }
    catch (Exception problem)
    {
        if (deleter != null) deleter.rollback();
        throw problem;
    }
    finally
    {
        session.close();
    }
}
```

Does this delete the email addresses?

Write, Update & Delete Cascading

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

Values for Cascade

all – cascade write, update & delete

none – no cascading

save-update – cascade only update & save

delete – cascade only deletes

Some Useful Session Methods

Hibernate API http://www.hibernate.org/hib_docs/api/

Session API

http://www.hibernate.org/hib_docs/api/net/sf/hibernate/Session.html

`int delete(String query)` throws `HibernateException`
Delete all objects returned by the query

`public Iterator iterate(String query)` throws `HibernateException`
Execute a query and return the results in an iterator.

`public void saveOrUpdate(Object object)`
throws `HibernateException`

Either `save()` or `update()` the given instance, depending upon the value of its identifier property

Logging – log4j

Main Web site <http://logging.apache.org/log4j/docs/>

Defines 5 levels of logging (from weak to strong)

- Debug
- Info
- Warn
- Error
- Fatal

Logging methods

- void debug(Object message)
- void debug(Object message, Throwable t)
- void info(Object message)
- void info(Object message, Throwable t)
- void warn(Object message)
- void warn(Object message, Throwable t)
- void error(Object message)
- void error(Object message, Throwable t)
- void fatal(Object message)
- void fatal(Object message, Throwable t)

Can set

- Which level of log messages are recorded
- Where log message is recorded
- Normally done in configuration file

DEBUG < INFO < WARN < ERROR < FATAL

Example

```
import org.apache.log4j.Logger;

public class Foo
{
    static Logger log = Logger.getLogger(Foo.class);

    public static void main(String[] args) throws Exception
    {
        log.info("Starting Foo.main" + Foo.class);
        new Foo().sampleMethod();
        log.info("End Foo.main");
    }

    public void sampleMethod() throws Exception
    {
        int x = 0;
        int y = 1;
        try
        {
            int z = y/x;
        }
        catch (Exception e)
        {
            log.fatal("In Sample",e);
            throw e;
        }
    }
}
```

Configuration File - log4j.properties

```
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE}
%5p %c{1}:%L - %m%n
```

```
log4j.rootLogger=off, stdout
log4j.logger.cs683=error
```

log4j.properties file placed in a directory in the classpath

Can read file manually

Set log levels per package

Loggers inherit values from parent logger

RootLogger is ancestor of all loggers

Order of logging levels

ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF

Hibernate log4j.properties

```
#### direct log messages to stdout ####
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p
%c{1}:%L - %m%n

#### direct messages to file hibernate.log ####
#log4j.appender.file=org.apache.log4j.FileAppender
#log4j.appender.file.File=hibernate.log
#log4j.appender.file.layout=org.apache.log4j.PatternLayout
#log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p
#c{1}:%L - %m%n

#### set log levels - for more verbose logging change 'info' to 'debug' ####

log4j.rootLogger=off, stdout

log4j.logger.net.sf.hibernate=off

#### log just the SQL
log4j.logger.net.sf.hibernate.SQL=debug

#### log JDBC bind parameters ####
log4j.logger.net.sf.hibernate.type=off

#### log schema export/update ####
log4j.logger.net.sf.hibernate.tool.hbm2ddl=debug

#### log cache activity ####
#log4j.logger.net.sf.hibernate.cache=debug
```

Handling Errors

```
static void sampleDelete() throws MappingException, HibernateException,
    Exception
{
    log.debug("Enter delete");
    Session session = getHibernateSession();
    Transaction deleter = null;
    try
    {
        deleter = session.beginTransaction();
        session.delete("select p from Person p join p.addresses address
            where address.host like '%gmail%'");
        deleter.commit();
    }
    catch (Exception problem)
    {
        if (deleter != null)
            deleter.rollback();
        log.error("Delete problem", problem);
        throw problem;
    }
    finally
    {
        session.close();
    }
    log.debug("Exit delete");
}
```