

CS 683 Emerging Technologies
Fall Semester, 2004
Doc 32 Mapping Collections
Contents

References	2
Mapping Collections	3
Bag	4
Tables.....	4
List	7
Tables.....	7
Arrays	10
Tables.....	10
Sorted Collection.....	13
Tables.....	13
Map	19
Tables.....	19
Glorp Dictionary Mapping	23
Classes.....	24

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>

Glorp Tutorial, Whitney

Mapping Collections

Types of Collections

- Set
- Bag
- List
- Maps
- Arrays

Bag

Allows repeats of elements

Order is not important

Hibernate uses List to implement a bag

Order of elements added to the database is not preserved

Example is the Person-EmailAddresses of doc 31

Tables

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck

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

People.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>

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

    <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>
```

People Class

```
package cs683;
import java.util.*;

public class Person {
    String firstName;
    String lastName;
    List addresses = new ArrayList();
    long id;

    public Person () { super(); }

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

    public List getAddresses() { return addresses; }
    public void setAddresses(List addresses) { this.addresses = addresses; }

    public void addAddress(EmailAddress newAddress) {
        addresses.add( newAddress);
        newAddress.setOwner(this);
    }

    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; }
}
```

List

ArrayList & Vector are lists

Order of elements in the list are preserved

Tables

People

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

Email_Addresses

id	user_name	host	person_id	Position
1	beck	cs.sdsu.edu	2	0
2	whitney	cs.sdsu.edu	1	0
3	whitney	rohan.sdsu.edu	1	1

People.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>

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

    properties same as last example

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

Person

```
package cs683;

import java.util.*;

public class Person {
    String firstName;
    String lastName;
    List addresses = new ArrayList();
    long id;

    public Person () {
        super();
    }

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

    public List getAddresses() {
        return addresses;
    }

    public void setAddresses(List addresses) {
        this.addresses = addresses;
    }

    public void addAddress(EmailAddress newAddress) {
        addresses.add( newAddress);
        newAddress.setOwner(this);
    }
}
```

rest same as last example

Arrays

Java arrays are not Collection classes

Tables

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck

Email_Addresses

id	user_name	host	person_id	Position
1	beck	cs.sdsu.edu	2	0
2	whitney	cs.sdsu.edu	1	0
3	whitney	rohan.sdsu.edu	1	1

People.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>

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

Rest is same as before

Person

```
package cs683;

public class Person {
    String firstName;
    String lastName;
    EmailAddress[] addresses = new EmailAddress[10];
    int endOfList;
    long id;

    public Person () {super(); }

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

    public EmailAddress[] getAddresses() {
        return addresses;
    }

    public void setAddresses(EmailAddress[] addresses) {
        this.addresses = addresses;
    }

    public void addAddress(EmailAddress newAddress) {
        addresses[endOfList++]= newAddress;
        newAddress.setOwner(this);
    }
}
```

rest is as previous examples

Sorted Collection

TreeMap and TreeSet are examples of sorted collections in Java

Since the collection orders the elements one does not need extra columns in the database

Tables People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck

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

People.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" sort="natural">
      <key column="person_id"/>

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

etc.

Options for Sort

```
<set name="addresses" inverse="false" cascade="all"  
    table="EMAIL_ADDRESSES" sort="natural">
```

Sort can be any of

- unsorted
- natural
- Name of a classes implementing java.util.Comparator

People

```
package cs683;

import java.util.TreeSet;

public class Person {
    String firstName;
    String lastName;
    SortedSet addresses = new TreeSet();
    long id;

    public Person () { super(); }

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

    public SortedSet getAddresses() {
        return addresses;
    }

    public void setAddresses(SortedSet addresses) {
        this.addresses = addresses;
    }

    public void addAddress(EmailAddress newAddress) {
        addresses.add( newAddress);
        newAddress.setOwner(this);
    }
}
etc
```

EmailAddress

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

    public Person getOwner()
    {
        return owner;
    }

    public int compareTo(Object anEmailAddress)
    {
        EmailAddress address = (EmailAddress)anEmailAddress;
        return (userName + host).compareTo(address.getUserName() +
            address.getHost());
    }
}
```

Set/get methods not shown

Comparator

Permits a sorted collection to

```
package cs683;

import java.util.Comparator;

public class EmailComparator implements Comparator
{
    public int compare(Object a, Object b)
    {
        EmailAddress first = (EmailAddress)a;
        EmailAddress last = (EmailAddress)b;
        return (first.host + first.userName).compareTo(last.host +
            last.userName);
    }
}

<set name="addresses" inverse="false" cascade="all"
    table="EMAIL_ADDRESSES"
    sort="cs683.EmailComparator">
    <key column="person_id"/>
```

Map

HashMap, Hashtable & TreeMap are instances of a map

Tables People

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

Email_Addresses

id	user_name	host	person_id	Alias
1	beck	cs.sdsu.edu	2	chair
2	whitney	cs.sdsu.edu	1	main
3	whitney	rohan.sdsu.edu	1	backup

People.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>

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

etc.

People

```
package cs683;
```

```
import java.util.HashMap;
```

```
public class Person {  
    String firstName;  
    String lastName;  
    Map addresses = new HashMap();  
    long id;
```

```
    public Person () { super(); }
```

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

```
    public Map getAddresses() { return addresses; }  
    public void setAddresses(Map addresses) {  
        this.addresses = addresses;  
    }
```

```
    public void addAddress(String index, EmailAddress newAddress)  
    {  
        addresses.put(index, newAddress);  
        newAddress.setOwner(this);  
    }
```

etc

Sample Write

static void sampleWrite() throws MappingException, HibernateException,
Exception

```
{  
    Session session = getHibernateSession();  
    Transaction save = null;  
    try  
    {  
        save = session.beginTransaction();  
        Person newPerson = new Person("Roger", "Whitney");  
  
        EmailAddress a = new EmailAddress("whitney", "cs.sdsu.edu");  
        EmailAddress b = new EmailAddress("whitney", "rohan.sdsu.edu");  
        newPerson.addAddress("main", a);  
        newPerson.addAddress("backup", b);  
        session.save(a);  
        session.save(b);  
        session.save(newPerson);  
        newPerson = new Person("Leland", "Beck");  
        a = new EmailAddress("beck", "cs.sdsu.edu");  
        newPerson.addAddress("chair", a);  
        session.save(a);  
        session.save(newPerson);  
        newPerson = new Person("Carl", "Eckberg");  
        session.save(newPerson);  
        save.commit();  
    }  
    catch (Exception problem)
```

etc.

Glorp Dictionary Mapping Example – Address Book

Address book contains dictionary of People

Tables People

id	first_name	last_name
1	Leland	Beck
2	Roger	Whitney

Address_Book

id	title
1	Work
2	Dance

Address_Book_Links

person_id	address_book_id	person_key
1	1	Chair
2	1	OO
1	1	Security
2	2	Email Person

Classes

AddressBook

```
Smalltalk defineClass: #AddressBook
  superclass: #{Core.Object}
  indexedType: #none
  private: false
  instanceVariableNames: 'id title entries '
  classInstanceVariableNames: "
  imports: "
  category: 'GlorpExperiments'
```

AddressBook class methods

```
title: aString
  ^super new setTitle: aString
```

AddressBook instance methods

```
at: aString
  ^entries at: aString

at: aString put: aPerson
  entries at: aString put: aPerson

title
  ^title

setTitle: aString
  title := aString.
  entries := Dictionary new.
```

Person

```
Smalltalk defineClass: #Person
  superclass: #{Core.Object}
  indexedType: #none
  private: false
  instanceVariableNames: 'firstName lastName id '
  classInstanceVariableNames: "
  imports: "
  category: 'GlorpExperiments'
```

add needed accessor methods

Mapping

```
Smalltalk defineClass: #GlorpTutorialDescriptor
  superclass: #{Glorp.DescriptorSystem}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: "
  imports: 'Glorp.*'
  category: 'GlorpExperiments'
```

allTableNames

```
^#('PEOPLE' 'ADDRESS_BOOK' 'ADDRESS_BOOK_LINKS')
```

constructAllClasses

```
^(super constructAllClasses)
  add: Person;
  add: AddressBook;
  yourself
```

classModelForAddressBook: aClassModel

```
aClassModel newAttributeNamed: #id.
aClassModel newAttributeNamed: #title.
aClassModel newAttributeNamed: #entries collection: Dictionary of:
Person.
```

classModelForPerson: aClassModel

```
aClassModel newAttributeNamed: #lastName.
aClassModel newAttributeNamed: #firstName.
aClassModel newAttributeNamed: #id.
```

```
descriptorForAddressBook: aDescriptor
| table linkTable |
table := self tableNamed: 'ADDRESS_BOOK'.
linkTable := self tableNamed: 'ADDRESS_BOOK_LINKS'.
aDescriptor table: table.
aDescriptor addMapping: (DirectMapping from: #id to: (table fieldNamed:
'id')).
aDescriptor addMapping:
(DirectMapping from: #title to: (table fieldNamed: 'title')).
aDescriptor addMapping: ((BasicDictionaryMapping new)
attributeName: #entries;
referenceClass: Person;
keyField: (linkTable fieldNamed: 'person_key');
relevantLinkTableFields:
(Array with: (linkTable fieldNamed: 'person_id'))))
```

```
descriptorForPerson: aDescriptor
| personTable |
personTable := self tableNamed: 'PEOPLE'.
aDescriptor table: personTable.
(aDescriptor newMapping: DirectMapping)
from: #id to: (personTable fieldNamed: 'id').
(aDescriptor newMapping: DirectMapping)
from: #firstName to: (personTable fieldNamed: 'first_name').
(aDescriptor newMapping: DirectMapping)
from: #lastName to: (personTable fieldNamed: 'last_name').
```

tableForADDRESS_BOOK: aTable

(aTable createFieldNamed: 'title' type: (platform varChar: 50)).

(aTable createFieldNamed: 'id' type: (platform sequence)) bePrimaryKey.

tableForADDRESS_BOOK_LINKS: aTable

| personId bookId |

personId := aTable createFieldNamed: 'person_id' type: platform int4.

aTable

addForeignKeyFrom: personId

to: ((self tableNamed: 'PEOPLE') fieldNamed: 'id').

bookId := aTable createFieldNamed: 'address_book_id' type: (platform varChar: 50).

aTable

addForeignKeyFrom: bookId

to: ((self tableNamed: 'ADDRESS_BOOK') fieldNamed: 'id').

aTable createFieldNamed: 'person_key' type: (platform varChar: 30).

tableForPEOPLE: aTable

(aTable createFieldNamed: 'first_name' type: (platform varChar: 50)).

(aTable createFieldNamed: 'last_name' type: (platform varChar: 50)).

(aTable createFieldNamed: 'id' type: (platform sequence)) bePrimaryKey.

Adding an AddressBook to the database.

session inUnitOfWorkDo:

[addresses := AddressBook title: 'work'.

addresses

at: 'musicMan' put: (Person first: 'Sam' last: 'Hinton');

at: 'author' put: (Person first: 'Martin' last: 'Fowler');

at: 'self' put: (Person first: 'Roger' last: 'Whitney').

session register: addresses].