CS 683 Emerging Technologies
Fall Semester, 2004
Doc 33 Database & Web
Contents

Cocoon & Hibernate
Person – Address Example
Classes

```
package cs683;

public class Address {
   String street;
   String city;
   long id;

   public String getCity() {
      return city;
   }

   public void setCity(String city) {
      this.city = city;
   }
   public long getId() {
      return id;
   }
   public void setId(long id) {
      this.id = id;
   }
   public String getStreet() {
      return street;
   }
   public void setStreet(String street) {
      this.street = street;
   }
}
```

Person

```
package cs683;

public class Person {
   String firstName;
   String lastName;
   Address address;
   long id;

   public Person () { }

   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 ; }

   public Address getAddress() { return address; }
   public void setAddress(Address address) { this.address = address; }
}
```

DbAccessor

```
package cs683;

import java.util.List;
import net.sf.hibernate.HibernateException;
import net.sf.hibernate.Session;
import net.sf.hibernate.SessionFactory;
import net.sf.hibernate.cfg.Configuration;

public class DbAccessor
   {
   Session dbAccess;

   public String hello() { return "hi there"; }

   public void close() throws HibernateException { dbAccess.close(); }

   public List performRead(String query) throws Exception
      {
      if (dbAccess == null)
         dbAccess= getHibernateSession();
      return dbAccess.createQuery(query).list();
      }

   public Session getHibernateSession() throws Exception
      {
      Configuration config = new Configuration();
      config.addClass(cs683.Person.class);
      config.addClass(cs683.Address.class);
      SessionFactory sessions = config.buildSessionFactory();
      Session session = sessions.openSession();
      return session;
      }
   }
```

Mapping Files
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>

      <many-to-one name="address" class="cs683.Address" column="address_id"
unique="true" cascade="all"  />

      <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>
```

Address.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="Address"
      table="addresses"
   >
      <id
         name="id"
         type="long"
         column="id"
      >
         <generator class="increment"/>
      </id>

      <property
         name="street"
         column="street"
         type="string"
         not-null="false"
         length="50"
      />
      <property
         name="city"
         column="city"
         type="string"
         not-null="false"
         length="50"
      />

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

Using Hibernate in Cocoon

cocoon/ is the cocoon directory

Hibernate jar files

- Jdbc driver jar
- hibernate-2.1/hibernate2.jar
- hibernate-2.1/lib/*.jar (all 30 jar files)

Place them in

cocoon/WEB-INF/lib

The following jar files are already part of Cocoon

- xml-apis.jar
- log4j-1.2.8.jar

and the ant jar files are not needed.

Your Class & hbm & Properities files

Place them in

- cocoon/WEB-INF/classes

So we have

- cocoon/WEB-INF/classes/hibernate.properties
- cocoon/WEB-INF/classes/log4j.properties
- cocoon/WEB-INF/classes/cs683/Address.hbm.xml
- cocoon/WEB-INF/classes/cs683/Person.hbm.xml
- cocoon/WEB-INF/classes/cs683/Address.class
- cocoon/WEB-INF/classes/cs683/DbAccessor.class
- cocoon/WEB-INF/classes/cs683/Person.class

Cocoon Files

Directory for cocoon files:

- cocoon/courses

Files

- cocoon/courses/sitemap.xmap
- cocoon/courses/documents/selection.vm
- cocoon/courses/flow/example.js

sitemap.xmap

```
<?xml version="1.0" encoding="UTF-8"?>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">

<map:components>
   <map:generators default="file">
      <!-- in this example we use JXTemplateGenerator to insert
            Flow variables in page content -->
      <map:generator label="content,data" logger="sitemap.generator.jx"
                        name="jx"
src="org.apache.cocoon.generation.JXTemplateGenerator"/>
   </map:generators>
   <map:transformers default="xslt"/>
   <map:serializers default="html"/>
   <map:matchers default="wildcard"/>
   <map:selectors default="browser">
      <map:selector name="exception"
src="org.apache.cocoon.selection.XPathExceptionSelector">
         <exception name="invalid-continuation"

class="org.apache.cocoon.components.flow.InvalidContinuationException"/>
         <exception class="java.lang.Throwable" unroll="true"/>
      </map:selector>
   </map:selectors>
   <map:actions/>
   <map:pipes default="caching"/>
</map:components>

<map:views/>
<map:resources/>
<map:action-sets/>

<map:flow language="javascript">
   <map:script src="flow/example.js"/>
</map:flow>
```

Sitemap continued

```
<map:pipelines>
   <map:component-configurations>
      <global-variables/>
   </map:component-configurations>

   <map:pipeline>
      <map:match pattern="*.html">
         <map:call function="{1}"/>
      </map:match>

      <map:match pattern="*.vm">
         <map:generate type="velocity" src="documents/{1}.vm"/>
         <map:serialize type="html"/>
      </map:match>

      <map:match pattern="*.kont">
         <map:call continuation="{1}"/>
      </map:match>

      <map:handle-errors>
         <map:select type="exception">
            <map:when test="invalid-continuation">
               <map:generate src="documents/invalidContinuation.html"/>
               <map:serialize type="xhtml"/>
            </map:when>
         </map:select>
      </map:handle-errors>
   </map:pipeline>

</map:pipelines>

</map:sitemap>
```

example.js

```
importClass(Packages.cs683.DbAccessor);

function selection() {

   var test = new DbAccessor();
   var people = test.performRead("select p from Person p ");
   cocoon.sendPage( "selection.vm", {list:people});
}
```

Don’t forget the Packages prefix in the import

selection.vm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
   <meta http-equiv="content-type" content="text/html; charset=iso-8859-1"
/>
   <title>Selection</title>
</head>
<body>
Here is the data you requested.
<ul>
#foreach( $person in $list )
<li>$person.getFirstName() $person.lastName, $person.address.city</li>
#end
</ul>
</body>
</html>
```

$person.lastName calls the getter method

$person.address.city is same as $person.getAddress().getCity()

Dynamic Compilation

Changes to

- selection.vm
- example.js

Are dynamically updated

Dynamically Compiling Java Code

http://cocoon.apache.org/2.1/userdocs/flow/java.html#Dynamic+Compilation

I was not able to get Java code to compile dynamically

Deployment

When you deploy put the java classes in a jar file

Place the following in a jar file

- hibernate.properties
- log4j.properties
- cs683/Address.hbm.xml
- cs683/Person.hbm.xml
- cs683/Address.class
- cs683/DbAccessor.class
- cs683/Person.class

Place the jar file in cocoon/WEB-INF/lib

Glorp & Seaside

Setup

- Load Glorp
- Load Seaside

Seaside classes may need to import Glorp namespace

A Session with Database Access

```
Smalltalk defineClass: #DatabaseSession
   superclass: #{Seaside.WASession}
   indexedType: #none
   private: false
   instanceVariableNames: 'dbSession '
   classInstanceVariableNames: ''
   imports: '
         Glorp.*
         Seaside.*
         '
   category: 'Cs683Assignment4'
```

Class methods

```
new
   ^super new initialize
```

Instance methods

```
initialize
   | login accessor |
   login := (Login new)
            database: PostgreSQLPlatform new;
            username: 'cs683whitney';
            password: 'secret';
            connectString: 'bismarck.sdsu.edu_cs683whitney'.
   accessor := DatabaseAccessor forLogin: login.
   accessor
      logging: false;
      login.
   dbSession := GlorpSession new.
   dbSession system: (Assignment4Descriptor forPlatform: login
database).
   dbSession accessor: accessor

close
   dbSession accessor logout

dbSession
   ^dbSession
```

Using the Session

```
Smalltalk defineClass: #SimpleCourseDataDisplay
   superclass: #{Seaside.WAComponent}
   indexedType: #none
   private: false
   instanceVariableNames: 'instructor course '
   classInstanceVariableNames: ''
   imports: '
         Seaside.*
         '
   category: 'Cs683Assignment4'
```

Class Methods

```
canBeRoot
   ^ true
```

Instance Methods

```
renderContentOn: html
   | instructors |
   instructors := self fetchInstructors.
   html list: instructors
      do: [:each | html  text: each name]

fetchInstructors
   ^self session dbSession readManyOf: Instructor orderBy: [:each |
each name]
```