GettingStartedWithCocoonAndHibernate

Why would you want to do this?

brief outline of the benefits of using an object relational mapping tool, like for Hibernate or OJB

The main benefit of an O/R mapping tool is that it enables you to deal with objects instead of database tables. Suppose you are building a web application based on Cocoon alone. You have a database of articles that is supposed to be navigated by users. The pure cocoon / flowscript MVC approach would be something like:

```
// ***** Get data from SQL Database (Model Layer) *****

var db_conn = Database.getConnection("name");
var s = db_conn.createStatement();
var r = s.executeQuery("SELECT * FROM articles;");

var articles = new Array();
var ri = r.iterator();
while(article = ri.next())
   articles[articles.length] = new Article( article.getString("Name"), article.getInt("Price"), ... ) )

// ***** Process data (Control Layer) *****

/* Do some processing, e.g. add images, comments etc. to the articles in the array or using CForms */

// ***** Send data to view (View Layer) *****

cocoon.sendPage("article-view",{articles:articles});
```

We love flowscript, don't we? In the above example, you can see that both the Control and the View layer are object oriented. However, the Model layer is a relational database, which makes conversion between M and C necessary - the typical kind of boring, repetitive and error-prone coding presented above. Now comes the same code fragment using Hibernate:

```
// ***** Get data from Hibernate (Model Layer) *****

var hs = openHibernateSession();
var articles = hs.find("from org.test.Article");

// ***** Process data (Control Layer) *****

/* Do some processing, e.g. using CForms to modify some articles or using CForms  */

// ***** Send data to view (View Layer) *****

cocoon.sendPage("article-view",{articles:articles});
```

Well this obviously makes your life easier. A Hibernate Session can be thought of as the object-oriented equivalent to a JDBC connection. As you can also see, Hibernate has its own - yet SQL-similar - query language. The fundamental difference is that Hibernate queries yield **objects** instead of database rows, so there is no further conversion necessary. You can directly process the result vector and pass it on to the View.

This is not only true for "flat" beans with simple attributes, it also holds for complex beans with collection attributes, sets, maps, vectors etc - and that is the point where it would get really messy to construct these "by hand" from a relational DB using flowscript. That part is now done by Hibernate.

The actual mapping between POJO Objects and the database is done by Hibernate. To accomplish this, it uses so-called mapping files. These can - up to a certain degree of complexity - also be automatically generated from Java code using tools that come with Hibernate. Even if you create these files by hand, it is still easier and more structured than writing endless flowscript code.

Summary

Hibernate makes the Model layer of an MVC application compatible with OO-based V and C layers. A much higher degree of abstraction from DB details is the result.

Again an outline of the main benefits:

- You don't have to write endless SQL access code anymore. The O/R mapping tool generates all necessary SQL access code behind the scenes and hands you the results in POJO format.
- You can manipulate your database (read, insert, delete) using POJO's and nothing but POJO's.
- Hibernate comes with transparent connection pooling (DBCP) and query caching (EHCache), only a matter of configuration.

What must you know before you start?

Be aware that you will not be ready to write Hibernate-based applications in 5 minutes. You are about to venture into a complex topic. Sit back, get a cup of tea and prepare for some days of reading and learning. Fortunately, the Hibernate framework is excellently documented (see http://www.hibernate.org/). The following skills are mandatory:

Skills and depth of knowledge required

- You need to be proficient in Java, have basic SQL knowledge in case you want to do troubleshooting.
- You should know how to translate your application domain into a solid database model.
- Understand the "lazy loading" concept that most O/R mapping tools use by default. This generally means that nothing is retrieved from the database until you really ask for it. http://www.hibernate.org/162.html. Also see below for notes on lazy collection initialization.

Make sure that you have understood the last point - if not, read the Hibernate documentation and then come back here. Otherwise you will only get frustrated with the task of integrating Hibernate into cocoon.

It must be made clear at this point that all documentation on this wiki focuses on the particular issue of integrating Hibernate into cocoon, which is a challenging but rewarding task. Rewarding, since you will learn some things about Cocoon's internal structure you might not have known before.

Skills and depth of knowledge that could be useful

What software is required ?

It is assumed that you have already downloaded, compiled and deployed the latest version of Cocoon. In addition you will need the software listed below.

Hibernate

What other software is also useful?

Random query monitoring tool

It is useful to see what queries your mapping tool is actually generating under the hood. With Hibernate you can configure this in the configuration file. The SQL generated will then be displayed in STDOUT. This is done as follows:

```
hibernate.show_sql=true
```

Alternatively most databases also offer query logging. See, for example:

- *Please add links to on-line docs (e.g. for mySQL)*
- When using mysql, pass the following command line parameter to mysqld or safe_mysqld:

--log=/location/of/log/file

This will log all queries sent to mysql. Beware, this file can grow HUGE. It is certainly not recommended to leave it turned on for a long time.

Middlegen

Middlegen extracts Hibernate mapping files from your database. It also correctly generates the mapping relations between tables if you set up foreign key constraints.

It can be obtained from http://boss.bekk.no/boss/middlegen/, check also http://middlegen.codehaus.org/ or http://www.hibernate.org/98.html

hbm2java

hbm2java is a hibernate ant task that can generate java classes from hibernate mapping files. It can be obtained from the hibernate distribution http://www.hibernate.org

---

How to Install and Configure the Software

Hibernate

Hibernate itself can be deployed according to the installation procedure found at http://www.hibernate.org/.

Hibernate Under Cocoon

See CocoonAndHibernateTutorial for instructions on deploying Hibernate in the cocoon webapp. This covers registering Hibernate with the Avalon Framework and using cocoon.xconf datasources for connection pooling.

If you want to set up Hibernate under Cocoon, it is recommended that you setup a servlet filter for handling the disposal of Hibernate sessions. This ensures that your Hibernate session sticks around until **after** your view was rendered.

This technique is described in more detail in the "Hibernate in Action" book (http://www.manning.com/bauer).

An example filter along with some explanation can also be found at CocoonAndHibernateTutorial.

Other Installation Procedures

needed before getting started are ...

Baby Sits: A Really Simple Table

Objective

The objective of this part is to allow a user to perform an ACID (add/create/insert/delete) operation on a single table. The table, while simple, contains a range of data types that will demonstrate how data is formatted in the front end (by Cocoon) and handled on the back end (by Hibernate).

Step 1: Do this

Step 2: Do this

Step 3: Do this

Troubleshooting

Baby Crawls: Working with Mutiple Tables

Objective

The objective of this part is to allow a user to perform an ACID (add/create/insert/delete) operation on multiple tables in a single operation.

Step 1: Do this

Step 2: Do this

Step 3: Do this

Troubleshooting

Baby Stands: Working with Master-Child Tables

Objective

The objective of this part is to allow a user to perform an ACID (add/create/insert/delete) operation on related tables in a single operation. The "child" table is linked to the "master" table via a common key. This will demonstrate how rows of data can displayed and updated at the same time.

Step 1: Do this

Step 2: Do this

Step 3: Do this

Troubleshooting

Baby Stumbles: Adding Logic for Data Pre- and Post- Processing

Objective

The objective of this part is to show how data passed to and from Hibernate can be evaluated and/or processed in order to apply appropriate business logic.

Step 1: Do this

Step 2: Do this

Step 3: Do this

Troubleshooting

Baby Walks: Next Steps to Take

Performance Tuning and Optimum Configuration

Lazy Collection Initialization

Lazy Collection Initialization is an extremely important contribution to decent performance of your Hibernate application.

By default, when fetching an object from the database, Hibernate makes sure that every other object reachable via getter methods is also available.

Suppose you have an article object. Each article has a vector of related articles so users can comfortably browse to your store. The following java snippet illustrates this:

```
import java.util.List

public class Article{
  ...
  private List relatedArticles;
  ...

  public List getRelatedArticles()
  {
     return relatedArticles;
  }
}
```

The corresponding fragment of your mapping file setup straight forward might look like this:

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

<hibernate-mapping>
  <class name="Article" table="Article">
    <id name="id" type="long" column="id">
      <generator class="native" />
    </id>

    ....

    <list name="relatedArticles" table="Article_Relations">
      <key column="articleId" />
      <index column="rank" />
      <many-to-many column="relatedId"
      class="Article" />
    </list>

    ...

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

Now suppose you have five Articles in your database, say A,B,C,D,E. A links to B and D while B links to C and D links to E. If you use hibebernate to fetch Article A from the database, it will also fetch *all other Articles,* since you could theoretically navigate to all of them using getter methods, i.e. you could reference Article E by typing:

```
Article E = A.getRelatedArticles().get(1).getRelatedArticles().get(0);
```

This can be useful if you know that you are going to use all or most articles anyway. But in almost every case where you do not need to access your entire database at once (which is almost every use case I could imagine) it will be useless and lead to dramatic performance loss.

The solution is called Lazy Collection Initialization. This Hibernate feature basically leaves all elements in Lists, Maps and other collections uninitialized until the element is actually accessed via `get()` or similar methods. To turn it on, simply set the attribute "lazy" of the respective collection to "true":

```
<list name="relatedArticles" table="Article_Relations" lazy="true">
```

If you, like me, programmed a whole webapp without knowing about this feature, try it out and see that it will work wonders on overall performance 🙂 Now, when fetching article A, it will only fetch article A. If, and only if, you acess article B via the relatedArticles List, it will connect to the DB again and fetch it dynamically. For this to work, ensure your Hibernate Session is left open until you have entirely finished working with your business objects.

You should decide carefully where to use lazy initialization and where not; it is not always beneficial, since sometimes you might have a collection whose elements are almost always accessed. Query monitoring is useful to find out whether you have set up lazy initialization optimally.

Query Caching

Rich Application Interface

The adoption of a RAI such Laszlo (http://www.laszlosystems.com/) will enable you to create a graphically-rich front-end for your web application. Please refer to Getting Started with Cocoon and Hibernate and Laszlo.

Item

Item

References

Essential Books and Articles

Other Wiki Pages

Related Reading

- [URL] Article name
- [URL] Article name
- [URL] Article name

Downloads

The following files can be downloaded:

- filename #1: *brief description and purpose*
- filename #2: *brief description and purpose*
- filename #2: *brief description and purpose*