

WoodyHibernateAndFlow

20031109: Updated for Cocoon CVS 2.1.3 (8 nov 2003) and for Hibernate 2.1b6

20031113: Updated the attachment. Removed the classes already supplied by Woody.

1. Read [UsingHibernateToMakeYourJavaBeansPersistent](#) if the instructions below go too fast.
2. Add the Hibernate jars from the Hibernate distribution to `WEB-INF/lib`. Copy only the jars that are not yet present. And don't forget the Hibernate jar itself (it is not in the Hibernate lib directory)
3. Copy the content of `WEB-INF` of the zip file (see attachment) into your Cocoon `WEB-INF`.
4. Merge the line in `_cocoon.xconf` into your real `cocoon.xconf` (this adds the Avalon component that creates the Hibernate session)
5. Edit `hibernate.properties` in `WEB-INF/classes`. Insert your login credentials for your database
6. Create tables in your database with the `mysql.sql` script. (Hibernate has support for many different databases. Change the SQL dialect in `hibernate.properties` and edit this script if don't use mysql but something else.
7. Copy the content (the woody map in the zip file) somewhere in your cocoon directory.
8. Restart Tomcat. Near the end of the debug info in the console, there should be a line `Hibernate initialize` called. If you don't see this line, one of the points above went wrong.
9. Try `http://...../woody/`

This is how it works:

- The Woody Java beans `Contact`, `Sex`, and `Form2Bean` are not modified.
- The sitemap of the Woody example is not modified.
- Two classes are added (The interface of an Avalon component `PersistenceFactory` and its implementation `HibernateFactory`). This component creates the Hibernate session.
- The flowscript `binding_example.js` is changed a little. The function `form2bean` tries to load the `Form2Bean` object from disk with a hard-coded email address as the key. If it fails, it creates a new instance and saves it to disk.
- The real beef is in the file `WEB-INF/classes/org/apache/cocoon/woody/samples/Form2Bean.hbm.xml`. This configuration file tells Hibernate how to do the real work: hide all details about the one-to-many relation (contacts that are added to the bean) for the developer. When you play with the example and look into the database, you can see that contacts are added with an unique key that is generated by Hibernate (the `id` field; don't edit it, as the Woody intructions tell you) and a foreign key (the email address). In a real-world application, you almost always will need the `lazy` and `inverse` attributes. But in this case, I wanted to stay as close as possible to the original classes of the Woody sample.

Hugo Burn

Attachment: [Hibernate_Woody_Flow.zip](#)