Module Hibernate Mapping Files

A hibernate mapping bi-directionally maps data from MySQL to Java and vice versa. A mapping might look something like this:

Example Mapping File

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="org.openmrs.module.YourJavaClass">
    <class name="YourJavaClass" table="your_mysql_table">
        <id name="id" column="your_mysql_table_id">
            <generator class="native"/>
        </id>
        <property name="sampleInteger" type="java.lang.Integer" column="simple_integer"/>
        <property name="sampleString" type="java.lang.String" column="sample_string" length="255"/>
        <property name="uuid" type="java.lang.String" column="uuid" length="38" not-null="true" unique="true"/>
        <property name="dateCreated" type="java.util.Date" column="date_created" length="19" not-null="true"/>
        <many-to-one name="patient" class="org.openmrs.Patient" column="author_id" not-null="true"/>
        <one-to-one name="encounter" class="org.openmrs.Encounter" column="encounter_id" not-null="true"/>
                <one-to-many name="provider" class="org.openmrs.Provider" column="provider_id" not-null="true"/>
    </class>
</hibernate-mapping>
```

Creating an Object

- Every table you create in the database requires its own hibernate mapping file in **omod/src/main/resources**
- The name field must be the same name as your Java class
- The column name must be the same as it is defined in your liquibase.xml file

Common Fields

Fields	
package	The location of the java class file being mapped
class name	The name of the java class being mapped
table	The name of the MySQL table being mapped
id	The id of the object
column	the MySQL field relating to the defined variable

Common MySQL to Java translations

Fields	
name	The field defined by your java object
column	The field defined by your MySQL table
type	The data type of the data, integer, string, ect.
not-null	When set to be true the field is required when defining the object

Common MySQL relations

Relations are commonly translated from MySQL foreign keys to Java classes.

Fields		Java class definition
one-to-one	Your object relates to one of the defined objects	Defined as the related object type
many-to-one	Multiple of your object can relate to one of another object	Defined as a list of the related object type
one-to-many	Your object can contain multiple of the related objects	Defined as a list of the related object type
class	Similar to type, this field defines the object type that is being related	

Note: These associations are defined and created by your liquibase.xml file first, so you will need to reference those to determine the relation type.

There should be a mapping file for each entity that your module/hibernate is maintaining. If you have two objects, then make two classes called FormEntryQueue and FormEntryArchive, then make FormEntryQueue.hbm.xml and FormEntryArchive.hbm.xml and put them both in the resources folder.

Deploying your mapping

If you are using the maven layout, this element is automatically populated with all found *.hbm.xml files.

If using the pre-maven layout (ant) then you will need to list off each of your hbm.xml files in that element.

If you are building using ant add the following lines to the bottom of your config.xml

config.xml

```
<mappingFiles>
        YourObject.hbm.xml
</mappingFiles>
```