

Tagger Cat's Installation and Getting Started Guide



Tagger Cat is distributed as a ZIP file. Technically speaking, the installation is simply just unzipping it to your preferred file directory. The installation steps detailed in this document are for configuring your Eclipse and Hibernate environments and for creating your first Tagger Cat application.

The following procedures have been tested on the My SQL Database only. If you are installing on another database, you will need to at least change the connection string in the Hibernate mapping file, as well as install the corresponding JDBC driver.

Pre Installation Steps and Third Party Components:

- 1) Unzip the Tagger Cat Buildn_n.zip to a folder of your choice. We will use C:\TaggerCat for this document.
- 2) Install a JDK 1.6.x or newer. Tagger Cat will not work on JDK 1.4.x. or JDK 1.5.x.
- 3) Install the **Hibernate Tools** 3.2.4.x or later into Eclipse

<http://tools.hibernate.org/>

- 4) Download and install the Hibernate Core 3.3.1. GA

We install Hibernate to C:\dev\hibernate-3.3.1.GA

Tagger Cat uses this latest version of Hibernate and not the one installed internally with the Hibernate Tools IDE plugins.

- 5) Install Tomcat Version 6.x or newer. Tagger Cat will not work on Tomcat 5.5.x.

<http://tomcat.apache.org/>

We install Tomcat to C:\tomcat-6.0.18.

We recommend using the zip version of the install; and not the Windows installer. We don't recommend installing Tomcat as a Window's services for the development machines.

Make sure that you don't install Tomcat on a path that includes a space in the folder names.

- 6) Add an admin user to your C:\ tomcat-6.0.18\conf\tomcat-users.xml file.

```
<user username="admin" password="changeMe" roles="admin,manager"/>
```

Replace "changeMe" with a password of your choice. Later on you'll need to make the corresponding change in the your project's tomcat.properties file.

- 7) Install MySQL 5.x or later.

<http://dev.mysql.com/downloads/>

- 8) You will also need a database administration tool. Some good ones we've use are Navicat, SQLyog, and TOAD for MySQL.
- 9) Set the following environment variables:
 - a. JAVA_HOME your JDK install folder
 - b. CATALINA_OPTS -ms256m -mx256m -XX:PermSize=128m
 - c. JPDA_ADDRESS 8000
 - d. JPDA_TRANSPORT dt_socket
- 10) We prefer to start and stop Tomcat as a separate process, and not from within Eclipse.

Start Tomcat from **C:\tomcat-6.0.18\bin\startup.bat**. You will want to create a Window's shortcut to this file; as well as a second one for stopping Tomcat that calls the **C:\tomcat-6.0.18\bin\shutdown.bat** file.

Other recommended tools:

- a) The FireFox browser – <http://www.mozilla.com/en-US/firefox/>

We don't recommend using IE as your development browser; mostly because there are really good development plugins available for FireFox.

- b) The FireBug extension for FireFox – this plugin is an invaluable tool for CSS development, JavaScript debugging, and AJAX debugging.
- c) 7-Zip - <http://www.7-zip.org/download.html>

The 7-Zip utility knows how to deal with Java jar files better than WinZip does.

Installation Steps:

Creating and setting up the database

- 1) Create a new physical database for you application (or use an existing one). For this document we will use the name "application1".
- 2) Run the database script to create the metadata system tables:
C:\TaggerCat\SQLScripts\MySQL\metadata.sql
- 3) Run the database script to create the administrator group and user.
C:\TaggerCat\SQLScripts\createAdminUser.sql

Configure your Eclipse IDE

- 1) Configure your Installed JREs. From the main menu select Window→Preferences→Java select Installed JREs, and configure a JRE 1.6.x or newer.
- 2) Check your Compiler Compliance setting. Go to Window→Preferences→Java and then select Compiler. Make sure the Compiler compliance level is 1.6

If wanted, you can also create a new Server reference to run Tomcat from within Eclipse. But, you'll need to make some changes to the generated build script, etc. to support this.

Create your first Tagger Cat Application

We recommend that you use the Stub Application builder to create the shell (or the stub) of your Tagger Cat applications. The Stub Application builder is a GUI front end to an ANT script. The Ant script copies all the necessary files from an APPLICATION template to the specified destination folder.

The Application template is in the **TaggerCatStubAppBuilder/StubAppTemplate** folder and it is processed by the build.xml file in the same folder. As you get more familiar with the Tagger Cat, you will want to create and maintain your own versions of the Application Template.

Using a Source Controlled Folder

We recommend that you use a *Source Controlled Folder* for your project, and that this same folder is used as the document base for your application in a Tomcat context. In other words, we don't recommend that you develop (or deploy) your application to Tomcat's **webapps** folder *during the development phase* of your project.

Where the term: *Source Controlled Folder* means the folder structure that you manage (check in – out etc.) with your version control system. Typically, this folder will be somewhere on a dev path in your source control system's repository structure. The Tomcat **webapps** folder is not appropriate for this.

Therefore, you will typically use the Stub Application builder to create the web application outside Tomcat's **webapps** folder. The Stub Application builder will create a context in Tomcat and set the document root for that context to your application's folder (typically a Source Controlled Folder).

We find that deploying applications outside of the Tomcat webapps folder *for production* also has a number of benefits.

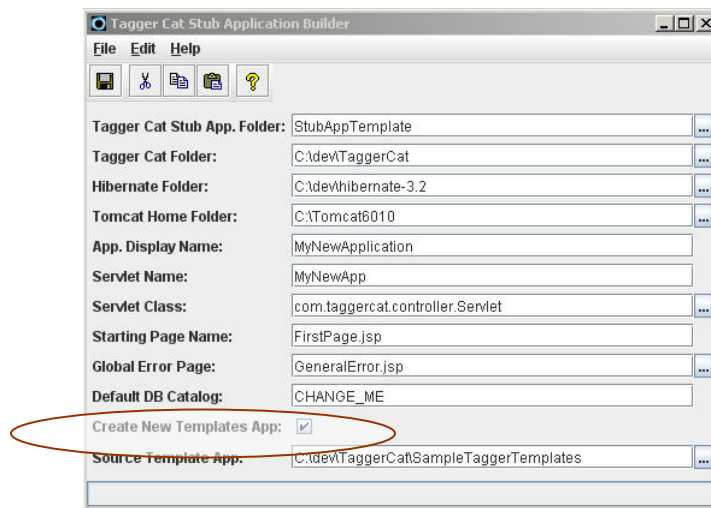
Run the Stub Application Builder

- 1) Configure the tomcat.properties file. Open the /TaggerCatStubAppBuilder/StubAppTemplate/tomcat.properties file and change the

following properties to match your system (don't change the tomcat.home=/@TOMCAT_HOME@).

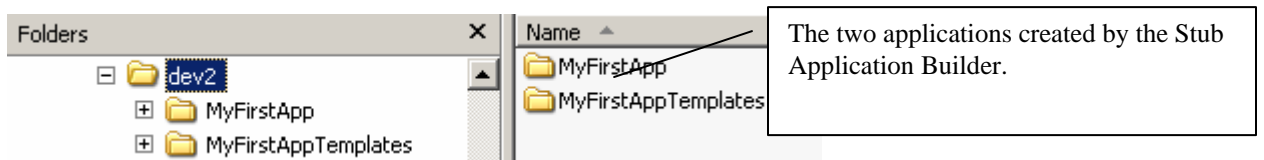
```
tomcat.server=localhost
tomcat.manager.url=http://${tomcat.server}:8080/manager
tomcat.username=admin
tomcat.password=dontShare
tomcat.home=/@TOMCAT_HOME@
```

- 2) From the StubApplicationBuilder project of your workspace (or from C:\TaggerCat\StubApplicationBuilder) run the CreateStubApp.cmd file.
- 3) Fill in the prompted information: watch the demo here?



- 4) From the file menu, select Save, and specify your Source Controlled Folder such as C:\dev\MyFirstApp. Note that if the selected directory name does not exist, it will be created.

If you have selected the Create New Templates App: option (and we highly recommend that you do), then the Stub Application Builder will create a second WEB application named: *YourAppNameTemplates*.



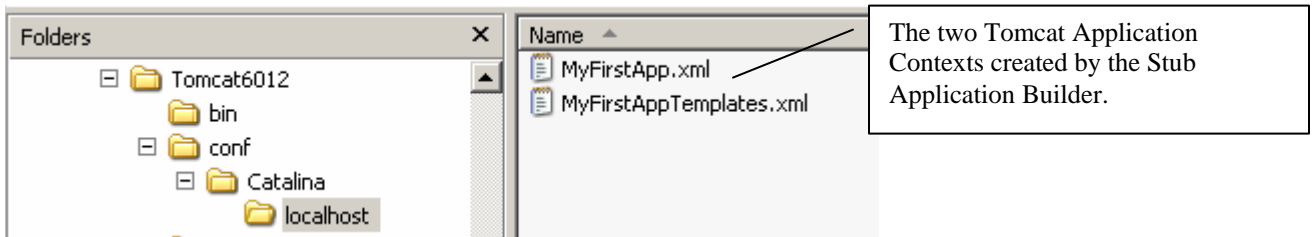
The Templates Application is separate web application that provides the web-services to the JSP Tagger to read your application's metadata, as well as it provides the templates used by the JSP Tagger.

After the Stub Application Builder has run, you will have a new **Eclipse project folder** named **MyFirstApp** that includes a Stub Tagger Cat WEB application. The WEB application context root is in the *WebContent* folder within the application.

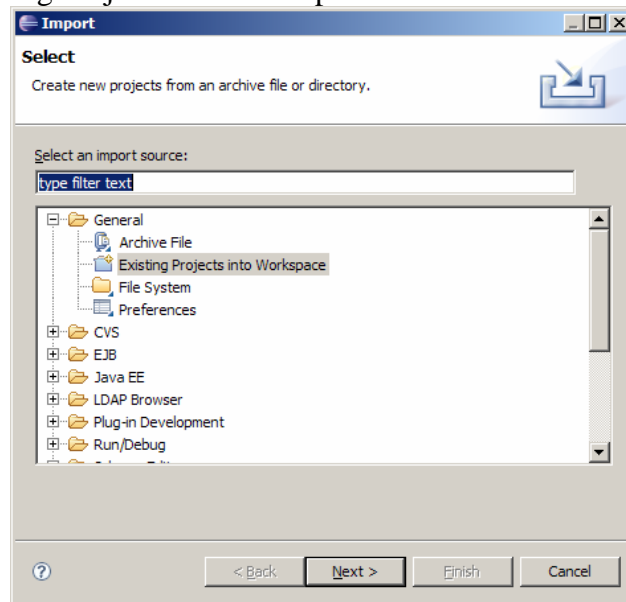
Note that the Stub App Builder will report a message such as:

Created Context File: C:\tomcat-6.0.18/conf/Catalina/localhost/MyFirstApp.xml

and also a message about creating a new context for the corresponding Tagger Cat Templates application context.

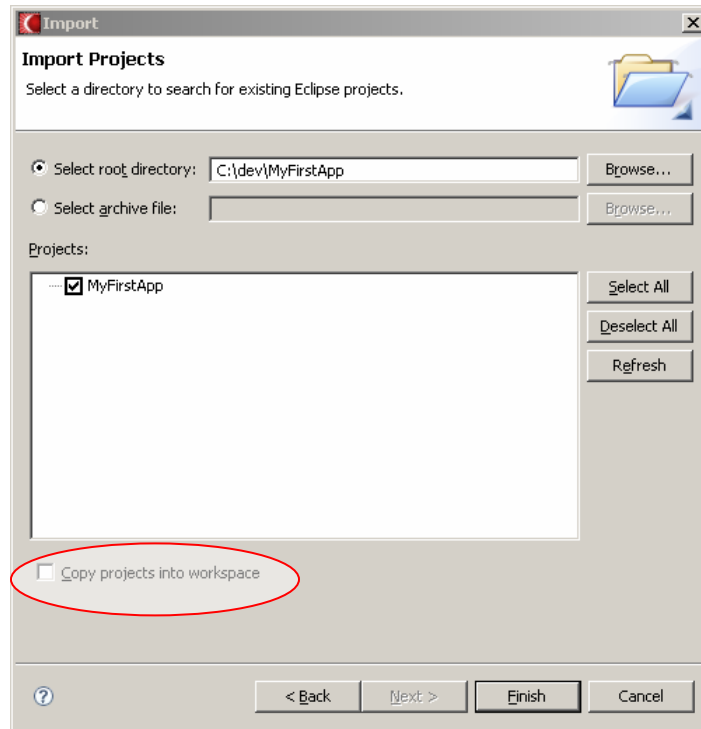


- 5) Import your new project into the Eclipse workspace. Select the menu File→Import and then General→Existing Projects into Workspace

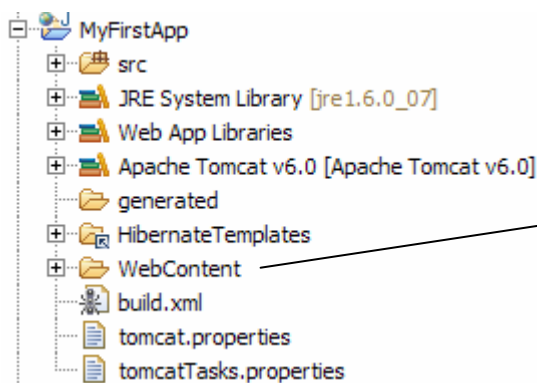


Since you have built the application to your Source Controlled Folder, you don't want to copy it into your workspace.

Therefore, **do not** select the Copy projects into workspace option. The new Tomcat context created by the Stub Application Builder points to the original location, and will not be automatically updated to point to a workspace folder.



Your new Eclipse web project within will have a structure similar to this:



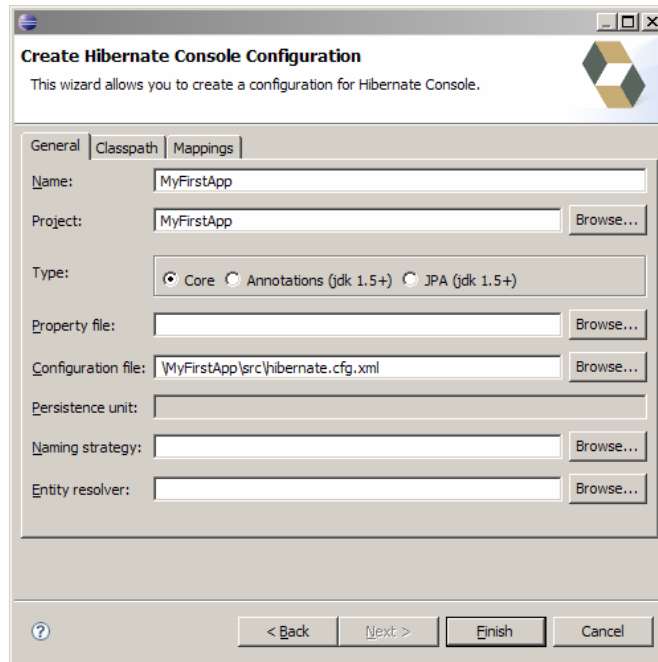
This is the context root of your **WEB** application. The docbase attribute in your Tomcat context file points here.

- 6) Verify that the linked folder named **HibernateTemplates** contains Hibernate's template folders and files. If needed, re-create this linked folder to point to \TaggerCat\HibernateTemplates.

Create a new Hibernate Console Configuration

- 1) Right click on your project folder, and then select New→Other→Hibernate→Hibernate Console Configuration

You will get a dialog similar to this:



Make sure that the Configuration file: entry is specified correctly.

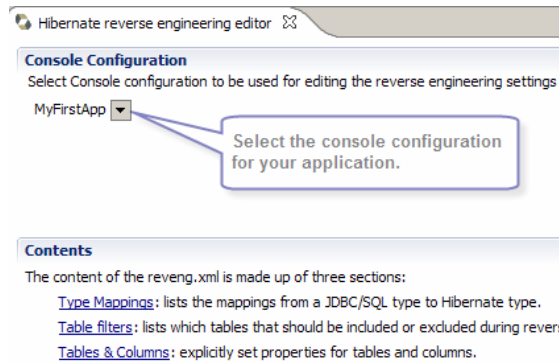
Reverse Engineer your Physical Database into Hibernate Mapping files.

*There are numerous approaches for building and mapping your Hibernate classes. We use the reverse engineering approach from the physical database. Using this approach is a personal preference and the following section documents this approach. We have not used other approaches, and therefore it is up to you to implement them if wanted. Also, we always design our physical schema where ALL tables have an auto-generated **id** property.*

- 1) Edit your database connection properties in the **src/hibernate.cfg.xml** file. Specifically, set the properties shown in bold:

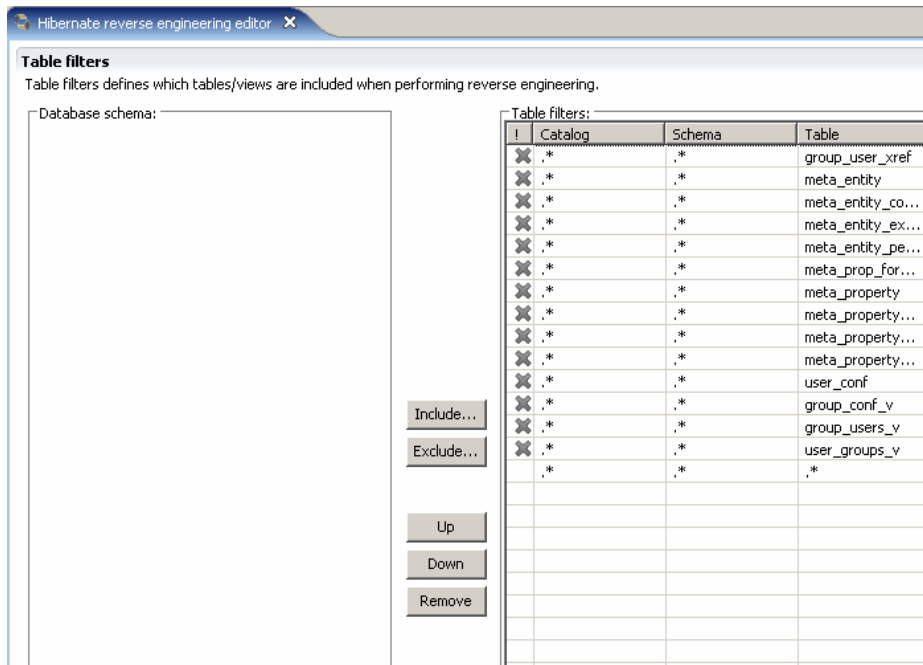
```
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/application1</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">changeme</property>
<property name="hibernate.default_catalog">test</property>
```

- 2) Open the **src/hibernate.reveng.xml** file using the Hibernate Reverse Engineering editor.
- 3) You will be in the Hibernate Reverse Engineering Editor. Select the console configuration we set up in the pervious step.

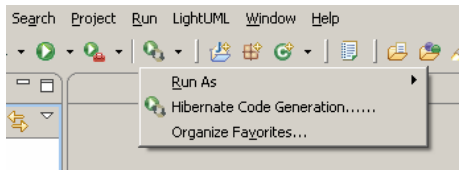


- 4) You will be in the Hibernate Reverse Engineering Editor. Select the console configuration we set up in the pervious step. Note that Tagger Cat's metadata and user configuration tables are excluded from the reverse engineering process. These mapping files and classes are already included in Tagger Cat's taggercat-metadata.jar file.

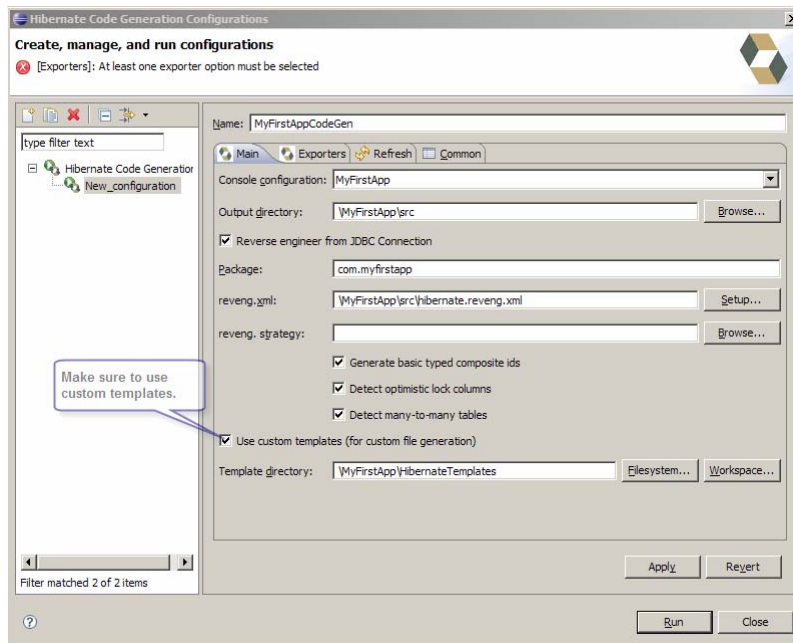
You can exclude any other tables, such as ones that you may have post-modified the generated code on.



- 5) The configuration of the **hibernate.reveng.xml** created by the Stub App Builder includes a wild card pattern to include all the other tables in the schema. Modify this to suite your needs. See the Hibernate Tools documentation for more details.
- 6) Create a Hibernate Code Generation Run Configuration. If the Hibernate Code Generation tool is not shown on your current Eclipse perspective, you can add it using Window→Window Perspective; select the Commands tab, and check the Hibernate Code Generation.
- 7) From the Toolbar select the Hibernate Run Configuration, and then the Hibernate code Generation... option as shown here.



- 8) Create a new Hibernate Code Generation run configuration as shown in the following window:



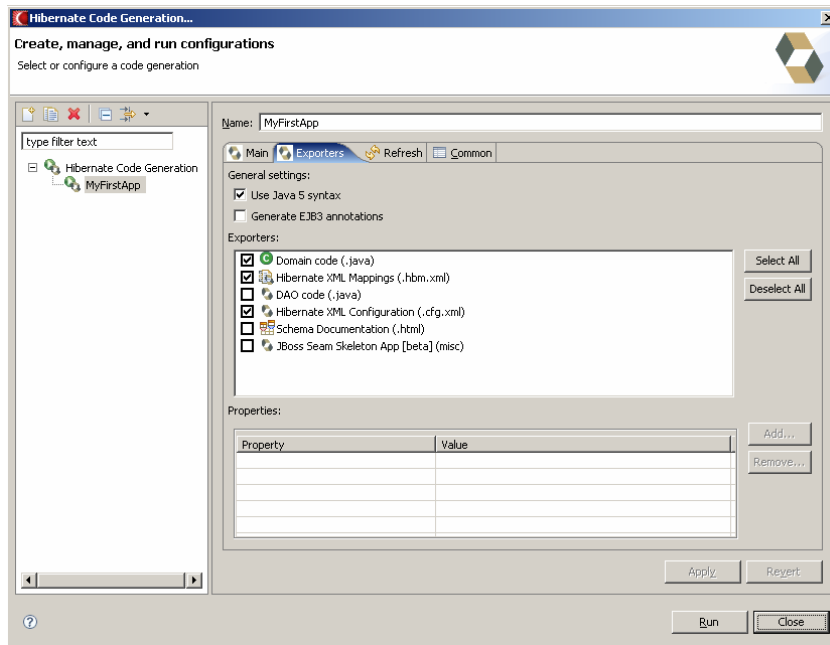
Noteworthy items:

- a. We have created the run configuration prefixed with the same name as our application, and a suffix of “CodeGen”. This is not required; but we find it a useful convention.
- b. We have selected the Hibernate console configuration for our application.
- c. We have chosen to generate directly to our application’s source folder by setting the Output directory to \MyFirstApp\src. The risk here is that you will clobber post-

modified source code or mapping files. The practice you can use to avoid clobbering post-modified files is to exclude them in your reveng configuration file.

Alternatively, you could set your Output directory to `\MyFirstApp\generated` and selective copy the generated `*.java` and `*.hbm.xml` files to your source folder. We use this approach in our own development.

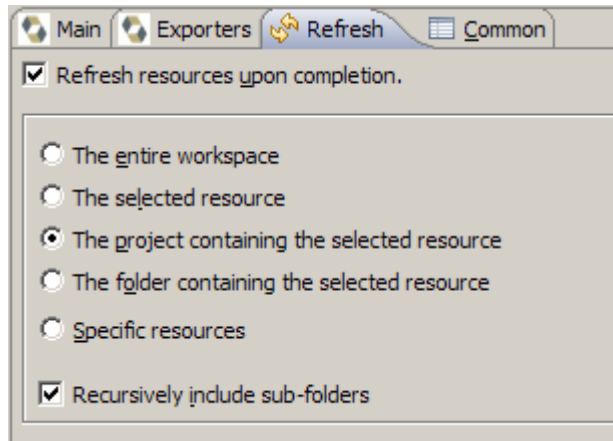
- d. We have set the generated package name to be: `com.myfirstapp`; use the package name of your preference.
 - e. We have specified our reverse engineering file from our project. **This is an important step!**
 - f. We have specified the use of Tagger Cat's custom Hibernate templates to be from `\MyFirstApp\HibernateTemplates` folder. This is important so that Tagger Cat's entity base class is used for all generated entities.
- 9) Select the options from the Exporters tab:



Noteworthy items:

- a. We have selected the Domain code, Hibernate Mapping files, and the Hibernate XML Configuration file.
- b. The Tagger Cat does not use DAO.

10) Select the Refresh tab, and make the setting shown here.

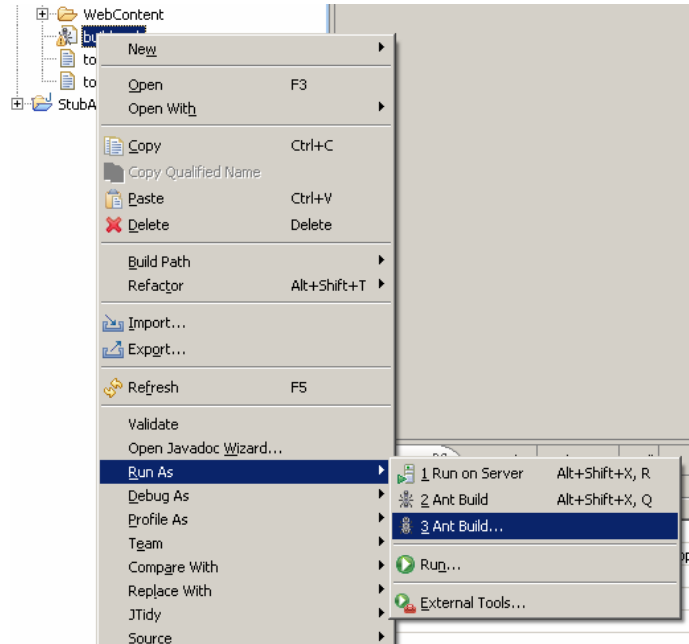


11) Select the Common tab, and unselect the **Launch in Background** option.

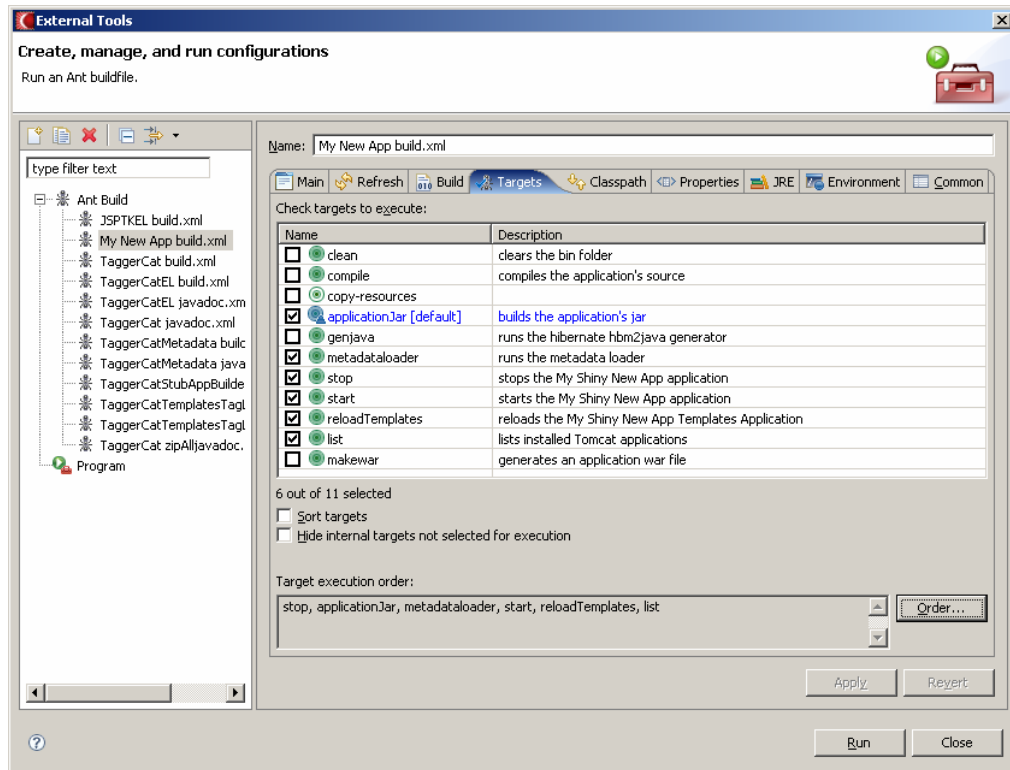
12) **Apply** and **run** this configuration. You can now re-run this command whenever your application's physical data model changes

Create a Ant Run Configuration for you new Application

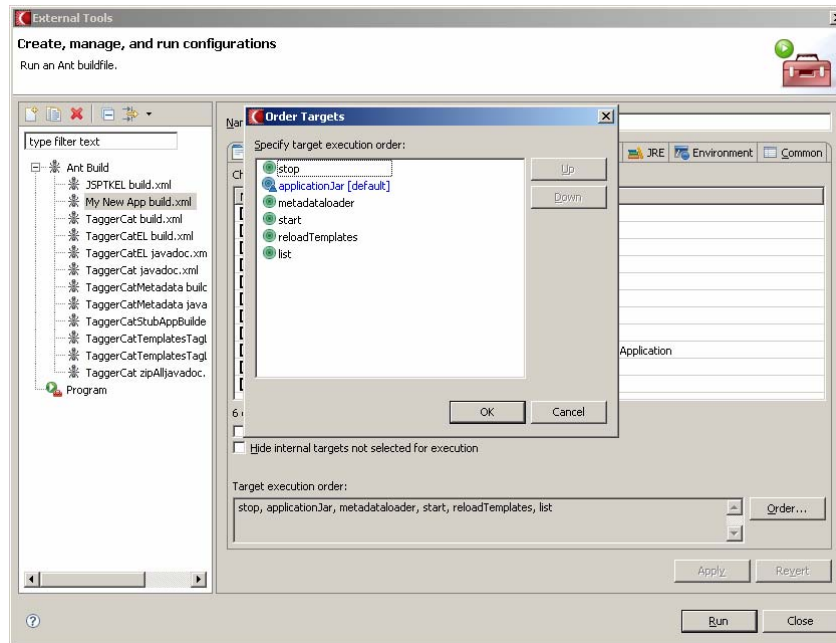
- 1) An ANT file for you application has already been created by the Stub Application Builder. Right click on the build.xml file in your application's project folder, and select Run As -> 3 Ant Build... as shown here.



- 2) Select the Targets tab, and select the Ant Build targets named: **applicationJar**, **metadataloader**, **stop**, **reloadTemplates**, and **start** as shown below.



- 3) Change the order of the Ant Targets to: **stop**, **applicationJar**, **metadataloader**, **reloadTemplates**, and then **start**.



These targets do the following:

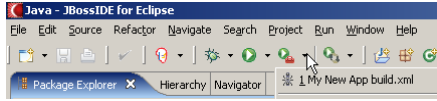
- a. Stops the application if it is currently running. This allows the application's jar file to be replaced.
- b. Builds the application's jar file. We sometimes refer to this as the application's "model" jar file. It contains your application persistent classes, and mapping files etc. Also, note that we are using the Eclipse to automatically build our project's class files. Therefore, we don't need to include the compile target.
- c. Runs the metadata loader. This process loads your application's persistent entity and property definitions in your application's metadata system tables.
- d. Reloads the Templates application, so that it also knows about your changed metadata.
- e. Starts the application running on Tomcat.

If you encounter the following error, then it is likely that you don't have the tomcat-home property set correctly in the tomcat.properties files.

[C:\dev\MyFirstApp\build.xml:24: taskdef class org.apache.catalina.ant.StopTask cannot be found](#)

You must use a Unix style path for this property; for example: /C:/tomcat-6.0.18.

- 4) You will now have an Ant Run Configuration that you can select from the Eclipse toolbar.



- 5) If your Tomcat server is running, you should be able to login to your application from <http://localhost:8080/MyFirstApp>. The user name and password are both "sa".

Next, please check out our demo and training videos available at <http://www.taggercat.com>

Importing the Tagger Cat Projects into your Workspace (Optional)

We recommend (but it entirely optional) is that you import the core Tagger Cat's projects into your workspace. You will be modifying the application template used by the Stub Application builder, as well as the template files as part of the Templates application. Additionally, having these projects imported into your workspace can make it more convenient for subsequent debugging of your application etc.

1. Create user libraries definitions:

You need to create user library definitions for Hibernate jars, for the Tomcat JSP related jars, and other third party jars. We provide an exported definition of the User Libraries that need to be configured. The file is `\TaggerCat\taggercat.userlibraries`.

Open this file in a text editor, and change the file paths to match your installation.

Note that this file contains references to the JavaDoc and source folders for some of the jars. These references can be updated if you have the source and Java Docs installed on your machine. Otherwise, you can ignore those references.

From the Eclipse File menu select `Window→Preferences` select `Java→Build Path→User Libraries`, Select the `Import...` button, and import the user libraries.

2. Import the Tagger Cat Projects

From the main menu, select `File→Import` and then `General→ Existing Projects into Workspace` Import the projects from `C:\TaggerCat` (you can choose to copy the project into the workspace or not)

TaggerCatMetadata	the metadata project
TaggerCatStubAppBuilder	the Stub Application builder
TaggerCatTemplatesTagLib	the Templates JSP Tag Library
TaggerCat	the core WEB application framework

Install the Tagger Cat Extension into Dreamweaver

The Tagger Cat Extension is packaged in the TaggerCat.mxp file. Open the Macromedia / Adobe extension manager to install this into Dreamweaver.

The TaggerCat extension is supported on Dreamweaver 8 and Dreamweaver CS3, CS4. Install the taggercat.mxp file Using the Adobe extension manager.

For the sample application, the template's URL will be: <http://localhost:8080/MyFirstAppTemplates/>

Adding the .tag file extension to Dreamweaver

Tagger Cat's sample application makes use of JSP 2.0 tag files. You will also likely want to use these, and other, JSP 2.0 tag files in your applications. The default installation of Dreamweaver does not recognize the .tag extension as a form of a .jsp file. Therefore, you will need to configure the .tag extension in Dreamweaver. The procedure for doing this is documented in the tech note: http://www.adobe.com/go/tn_16410