HowToSetupEclipseCallistoJetty

NOTE: This is outdated information that applies primarily to Tapestry 4.

How to set up Eclipse/Callisto/Jetty for Developing Tapestry

Download and extract all of the following to a directory (I use `C:\java`)
*`apache-ant-1.6.5`
*`hivemind.1.1.1`
*`tapestry-4.0.2`
*`jetty-5.1.10`

Edit your system environment PATH variable to include `c:\java\apache-ant-1.6.5\bin`

Open `tapestry-4.0.2/config/build.properties` and change the hivebuild.dir to the following:

```
hivebuild.dir=C:/java/hivemind-1.1.1/hivebuild
```

From a terminal window, cd to the `c:\java\tapestry-4.0.2\framework` directory and type the following:

```
C:\java\tapestry-4.0.2\framework>ant compile-dependencies
```

This will download Tapestry's supporting libraries..

Download the latest Eclipse http://www.eclipse.org (I'm using 3.2)

Install the Callisto plugins for eclipse

*After installing eclipse, go to Help -> Software Updates -> Find and Install...
*Search for new features to install
*Check "Callisto Discovery Site"
*When presented with all the different modules, check everything except what you're absolutely sure you don't need (C/C++, charting, for example)

Install Jetty Launcher plugin

*Go to Help -> Software Updates -> Find and Install...
*Search for new features to install
*Click "New Remote Site", enter a name, use http://jettylauncher.sourceforge.net/updates as the URL

(Use Subversion)

*SSH to your linux server, install svn via RPM, debian packages, etc..
*`svnadmin create /usr/local/repository`
*`chown mylogin:mylogin /usr/local/repository -R`

*Install subclipse (same way as jettylauncher) using http://subclipse.tigris.org/update_1.2.x

*After installing subclipse, Go to Window -> Preferences... then Team -> SVN. Change "SVN interface" to JavaSVN. Unless you really like screwing around with pub/private keys this is a good way for you to save your ssh username/pass for accessing the repository. (took me awhile to find this)
*Go to Window -> Open Perspective -> Other..., select "SVN Repository Exploring"
*Right click on the "SVN Repository" view, Click "New -> Repository Location...", enter `svn+ssh://mylogin@myserver.com/usr/local/repository`
*Right click on your new repository location, click "Create new remote folder", give it a project name(it's not an eclipse project yet), such as `myapp`
*Right click on `myapp` and create three more folders under it called "tags", "branches", and "trunk" (Mini-subversion-lesson: by convention, most people work out of the trunk folder, when they make a release, they copy the trunk to `tags/release-1.0` for example and that remains a frozen release. Learn more at http://subversion.tigris.org/)
*Right click on `trunk` and click "Checkout..."
*Make sure to "Check out as a project configured using the New Project Wizard", click Finish
*Read on...

Note: Later when you want to deploy your app from the repository, you'll do first do an svn checkout(one time only) such as `svn co file:///usr/local/repository/myapp/trunk/context/ /home/myapp/www/` and every time you want to update your live version (after commiting changes from eclipse) simply do an `svn update /home/myapp/www/` and then restart your jetty server. This is a great way to save yourself because you can very easily roll back to an earlier revision on the server if there was a problem.

Note: Edit the XML filetypes so that *.page, *.application, and *.jwc files open up in the XML editor

From the New project wizard...

*Create a "Dynamic Web Project", Next
*Name it "myapp", Next
*adjust the java version to 5.0, Next
*I use `context` for the Content Directory, Finish
*(ignore its warning about project files)
*You should be in the java perspective now, Right-click your new project in the Project Explorer, Click Build Path -> Configure Build Path...
*Click the Source tab and change Default output folder to say `myapp/context/WEB-INF/classes`
*Click the Libraries tab, then "Add Library...", select "User Library", then click the "User Libraries..." button then click "New..."
*Enter `JETTY5_LIB` as the name, OK
*Now click "Add JARs..." for the following

```
jetty-5.1.10/*.jar
jetty-5.1.10/lib/*.jar
jetty-5.1.10/extra/lib/*.jar
jetty-5.1.10/extra/ext/*.jar
```

*Click through all the "OK" buttons to get out of there
*When back at the java perspective delete that pesky `build` folder it had created
*Now we need to add Tapestry, Hivemind and supporting libraries so go to your filesystem and copy your downloaded jars and paste them to the `context/WEB-INF/lib` dir in eclipse. add the following:

```
C:\java\tapestry-4.0.2\*.jar
C:\java\tapestry-4.0.2\ext-package\lib\*.jar
```

*Now edit your `context/WEB-INF/web.xml` file to look like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">
        <display-name>
        udosquares</display-name>
        <servlet>
                <servlet-name>Udosquares</servlet-name>
                <servlet-class>
                        org.apache.tapestry.ApplicationServlet
                </servlet-class>
                <load-on-startup>1</load-on-startup>
        </servlet>

        <servlet-mapping>
                <servlet-name>Udosquares</servlet-name>
                <url-pattern>/app</url-pattern>
        </servlet-mapping>

</web-app>
```

- And create a new file named `context/WEB-INF/Udosquares.application` and paste the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Apache Software Foundation//Tapestry Specification 4.0//EN"
        "http://jakarta.apache.org/tapestry/dtd/Tapestry_4_0.dtd">
<application>

    <description>Udosquares Application File</description>

        <meta key="org.apache.tapestry.messages-encoding" value="UTF-8"/>
        <meta key="org.apache.tapestry.accepted-locales" value="en"/>

        <meta key="org.apache.tapestry.page-class-packages" value="com.udosquares.pages"/>
        <meta key="org.apache.tapestry.component-class-packages" value="com.udosquares.components"/>
</application>
```

*Now create a a file called `context/Home.html` just put some starter text in there:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>I love Tapestry</title>
</head>
<body>
Tapestry is awesome
</body>
</html>
```

Now for a tricky part that kinda threw me for a loop the first time.
*First things first, let's make sure JettyLauncher buttons are showing. so go to Window -> Customize Perspective..., click the Commands tab and check the "Jetty Launcher Actions" option. Doing so should add another button on your toolbar which is used for restarting or stopping the Jetty server.
*Now we need be able to start the Jetty server, so go to Run -> Run... and right-click "Jetty Web" to create a new jetty launch configuration
*From the Main tab, select your Jetty home directory, specify `context` as the "webapp root dir" and select the project it goes with
*__IMPORTANT__: On the classpath tab, remove the "Udosquares (default classpath)" otherwise you get a fatal error about how hivemind is duplicated.
*Browse to http://localhost:8080/app in FIREFOX now you should see "Tapestry is awesome"
*Tapestry creates pools for its page objects so changes made to Home.html won't be seen until jetty is restarted. You can now edit the file and restart Jetty by clicking the JettyLauncher button on the toolbar. Alternatively you could add an argument to the launch configuration (Run -> Run...) to pass VM arguments which says `-Dorg.apache.tapestry.disable-caching=true` this will allow the .html pages to be reloaded every time you access them, which can help with development. Problem is, after awhile Tapestry fills up with too many page objects and slows down. use this command only in development NEVER in production.
*Create a package called `com.udosquares.pages` and create a new class under it called "Home". Paste the following

```
package com.udosquares.pages;

import org.apache.tapestry.annotations.InitialValue;
import org.apache.tapestry.html.BasePage;

public abstract class Home extends BasePage {

        @InitialValue("literal:mike")
        public abstract void setName(String name);

}
```

*Now modify the Home.html again and paste this new text to it:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>I love Tapestry</title>
</head>
<body>
My name is <span jwcid="@Insert" value="ognl:name" />.
</body>
</html>
```

*Restart Jetty and browse again to http://localhost:8080/app
*You should see "My name is mike."
*Notice a couple things here - the Home class is abstract and tapestry recognizes abstract setters (setName is our only one in this instance), when tapestry runs, it it will create a page class which will also have a getter (automatically made based on the one setter provided). You could also use just a getter and tapestry will create a setter automatically as well...sometimes you want both because you're submitting a form or something...you'll learn.
*Notice also that using the @Insert component we're giving an ognl expression as a value. Here, ognl is calling "getName" because we have given it "name" if we had specified value="ognl:person.name" it would have called "getPerson().getName()" on our page object.

more later?? any pros are welcome to update this..after messing around a lot with other app servers and different setups I found this to be the most productive for tapestry work.