

Dette er bufferen til Google for

https://www.ibm.com/developerworks/mydeveloperworks/blogs/johnsrandommusings/entry/eclipse_maven_m2eclipse_and_pydev_we_re_having_a_webapp_party116?lang=pt_br. Det er et øyeblikksbilde over siden slik den var 27. sep 2011 12:31:36 GMT. [Den gjeldende siden](#) kan ha endret seg siden da. [Les mer](#)

Disse søkeordene er fremhevet: **pydev add tomcat**

[Tekstversjon](#)

- [Procurar nos Blogs](#)
- [Meu Blog](#)
- [Minhas Atualizações](#)

[Ajuda](#)

[Este Blog](#) ▾ Procura

[Procura](#)



LATEST TRENDS: [java](#) [serviceability](#) [troubleshooting](#) [websphere](#)

Eclipse, Maven, m2Eclipse, and PyDev - we're having a webapp party!

| Marcações: [webapp](#) [jython](#) [eclipse](#) [java](#) [maven](#) [m2eclipse](#) [pydev](#) | Comentários (2) | Visitas (1,948) [Classificações](#) ⁰

I decided on Monday to investigate how one would write a webapp using some jython code. After a bit of searching on the interwebs I found some tutorials that confirmed that it could be done. So I set out to make it happen using my new favorite development rig: Eclipse Galileo 3.5.1 (with m2eclipse and **pydev**) and maven. I've really taken a liking to using maven for dependency management since I had a sizable collection of random Java libraries laying around in my projects directory. Additionally, I like that I can use maven to do all the build, test, and deploy operations at the command line as well as in Eclipse. Score! So anyway back to the task at hand...

I spent the greater part of Wednesday trying to hack out the project using different methods to get things all working right in Eclipse. Some permutations got everything in order pretty well except Eclipse would not code-assist on the Java libs; other times code-assist would work fine but the project would not deploy to **Tomcat** because Eclipse disagreed with me that the project was, in fact, a web app. These instructions assume that you are familiar with Maven, m2eclipse, eclipse, Java, and Jython. In the end, this is the order that made things work for me :

1. Create a new Maven project using the **maven-archetype-webapp** archetype.
2. Now, do a **maven run** using the goals **clean eclipse:eclipse** and be sure to **add** the property **wtpversion** with a value of **2.0** (The documentation says valid value is 1.5 and make no mention of 2.0. My eclipse reports v3.0 for the WTP feature but I did not test it with 3.0 - so it might work, might not...). Refresh the project using Eclipse's **refresh** option; you may be able to do the m2eclipse command **Update Project Configuration** in place of the **refresh** but that's not the order I did it. You'll know it worked because the project view will now show the Deployment Descriptor node in the project view.
3. After Eclipse is done creating the project, I set out creating the maven folder structure: creating **main/java** and **main/test** under the src directory. **Add** those new folders to the build path as source folders
4. Then I created my java package name and created a simple servlet class. Since the maven archetype already created a simple JSP for us, there's no need for us to create one. Note that you'll need to **add** the **javax.servlet.servlet-api** dependency in order to compile the servlet. Also, be sure you set the scope of this new dependency to **provided** since the server we use will provide support for servlets in its run-time.
5. Next I altered the build path of the project and got rid of the J2SE 1.4 JRE that maven tacked on and replaced it with my IBM Java 6 JRE I typically use (IBM Java SR6 Linux-32bit)
6. From here you should be able to run the simple web app using **tomcat** or jetty or whatever server you wish.
7. Next, **add PyDev** project support by right-clicking the project and selecting **PyDev / Set as PyDev project**. You might have to wait a bit while Eclipse and **PyDev** sort things out.
8. Next, edit the project properties and set **PyDev - Interpreter/Grammar** to use **jython**. Hit apply.
9. Next, edit the **PyDev - PYTHONPATH** page adding the existing **src/main/webapp** folder to the **Source Folders** list. And since we're going to be writing a Jython version of an HttpServlet we need to include the JAR file maven downloaded earlier in step 4 from our maven repo. Mine was located in **<HOME>/m2/repository/javax/servlet/servlet-api/2.5/servlet-api-2.5.jar**. Now your Jython editor will be able to help you code with java objects. Hit OK on the project properties dialog.
10. Create a new **PyDev Module** in your **src/main/webapp** folder. When you're done you should be looking at the Jython code in the editor
11. Name the class correctly (it defaults to **MyClass**). "Correctly" means name the class the same as the module file name you specified in step 10. Also, notice that the new Jython class extends the **object** type? Change that to **HttpServlet** and **add** this line just before the class definition: **from javax.servlet.http import HttpServlet** (Eclipse should help you out a bit here). Lastly, you can remove the **__init__** constructor.
12. Now **add** a doGet(self, request, response) method and do something interesting in it.
13. Almost there, in order for us to use Jython as a servlet we have to setup a dispatcher servlet to help us get requests to our Jython code. Do this by adding a new servlet definition to the web.xml file:

```
<servlet>
  <servlet-name>PyServlet</servlet-name>
  <display-name>Jython dispatcher servlet</display-name>
  <servlet-class>org.python.util.PyServlet</servlet-class>
</servlet>
```

We must also **add** a servlet mapping definition:

```
<servlet-mapping>
  <servlet-name>PyServlet</servlet-name>
```

EXPLORE THIS BLOG

You're currently viewing: Eclipse, Maven, m2Eclipse, and **PyDev** - we're having a web party!

Previous Post: [Recent interview...](#)

Next Post: [A sample application...](#)

View all Posts: [At this blogs home page.](#)

RELATED POSTS

[A sample application sporting JPA and the WebSphere eXtreme Scale JPALoader](#)

[Please welcome Kevin to the blogosphere developerWorks!](#)

[Disabling your signal handler thread?](#)

[New developerWorks article on WebSphere eXtreme Scale Key Performance Indicators](#)

[Memory-based eviction and garbage collection algorithms](#)

[Logging smarter...](#)

[\(Really\) Simple WXS!](#)

[WebSphere Application Server and XOR encoding](#)

[Discover additional related res](#)

POPULAR TAGS

[core-group](#) [database](#) [dcs](#) [developerworks](#) [extreme-scale](#) [hamanager](#) [java](#) [jdk](#) [jvm](#) [log4](#) [orb](#) [performance](#) [serviceabil](#) [tracing](#) [troubleshooting](#) [virtual-enterprise](#) [websphere](#) [websphere_extreme_scale](#) [websphere_virtual_enterprise](#) [w](#) [wxs](#)

[View more tags at blog home](#)

RECENT TWEETS

@IBM_AppServer link's changed <http://t.co/0WKuxrGD>. Time to up Thx! about 4 hours ago

RT @IBM_AppServer: How do yo properly renew a certificate using iKeyman utility? <http://t.co/l6o8UC> Lo! @NickBunn @PhiSig2229 at hours ago

@PhiSig2229 @NickBunn Could help but laugh at that. Been there done that. Things never change, about 4 hours ago

[Follow @jpapejr on](#)

FOLLOWING THIS BLOG

```
<url-pattern>*.py</url-pattern>
</servlet-mapping>
```

14. Finally, **add** the **org.python jython** dependency to your project and execute a **maven run** with the goals **compile tomcat:run**. This command will instruct maven to download and execute a **tomcat** container with your project deployed. It will automatically utilize the dependencies you have associated with your project (i.e. put the **org.python jython** jar in the **tomcat** runtime path so your servlet will function. To reach your Jython servlet just use **hostname:port/context/myservletname.py**

That's it!

Lessons Learned

1. Maven likes it's directory structure, so don't try to change it around to suit you.
2. Don't try to bend Maven to your will. You'll lose and be sore about it. Seek to coexist.
3. Using the embedded Jetty server via the org.mortbay.jetty jetty-maven-plugin over the **tomcat** plugin seems to be an easier option because jetty is able to detect code changes and show you the results without restarting (your mileage may vary)
4. Maven hints about using JUnit in each project you create with it. Testing is a good idea. Use it.

Github Sample

You can now grab a clone of my sample project from github - [git://github.com/jpapejr/maven.jython.web.sample.git](https://github.com/jpapejr/maven.jython.web.sample.git)

Like Be the first of your friends to like this.



Adicionar um Comentário Recomendar esta Entrada Mais Ações

Comentários (2)



1 [N29X_Rebecca_Peltz](#) comentou às [Link permanente](#) [Classificações](#) ⁰

Can you provide links to any tutorials or docs you used in setting up PyServlet. Have you set it up with WebSphere?



2 [JohnPape](#) comentou às [Link permanente](#) [Classificações](#) ⁰

Unfortunately, I don't have the tutorial I used to get the JythonServlet.py file written. Fortunately, it's very simple! As for trying it out on WebSphere. I just booted up a WAS v7 server and dropped in the maven produced WAR file and it works great!

Adicionar um Comentário

[Anterior](#) | [Principal](#) | [Avancar](#)

[Alimentação para Entradas de Blog](#) | [Alimentação para Comentários de Blog](#) | [Alimentação para Comentários desta Entrada](#)