Digital Digressions by Stuart Sierra

From programming to everything else

Run Your Own Maven Repository With Nothing but an FTP Server

I hope I've demonstrated in the last few posts that Maven is pretty cool, not so scary. But the public Maven repositories sometimes leave a bit to be desired. They don't have entries for every possible library, and occasionally they have incorrect dependencies or other metadata. Also, the process of adding new libraries to the central repositories is somewhat involved.

Maybe you want to depend on a project that isn't in the public repos. Or maybe you want to publish development snapshots of your own projects. In either case, you need your own Maven repository.

Fortunately, running your own Maven repository is dirt simple. All you need is a web server where you can upload files. Maven understands FTP, SCP, WebDAV, and more exotic protocols like Subversion. Here I'm going to describe the simplest one, plain old FTP. If you have a web site on a cheap, shared web host, odds are you can use FTP to manage the files on the server.

Step 1: Get Some Web Space

I'm assuming you have a web site somewhere. Lets say it's at http://www.example.net/. Furthermore, let's say you can publish files on this web site by uploading them to the FTP server *ftp.example.net*. Your FTP user name is *samiam* with the password *greeneggs*. You put the files for your web site in the directory */home/samiam/public_html* on the server.

Using your favorite FTP client, create a directory named *maven2* inside your web site directory (*public_html* in our example).

Congratulations, you just created a Maven 2 repository!

Check that you can visit http://www.example.net/maven2 in a web browser. You should see a directory listing; it's empty, because we haven't added any files yet.

Step 2: Configure Your Project Deployment

Now you're ready to deploy a project to your Maven repository. If you've been following along with my Maven blog posts, you know that each Maven project has a project description file named *pom.xml*.

Open up your awesome software project and edit the *pom.xml* file. You're going to add two new sections, ending up with a file that looks like this:

```
<project ...>
  ...
  <groupId>net.example</groupId>
  <artifactId>awesome</artifactId>
  <name>The Awesome Library</name>
  <version>1.0-SNAPSHOT</version>
  ...

  <build>
    ...
    <extensions>
      <extension>
        <groupId>org.apache.maven.wagon</groupId>
        <artifactId>wagon-ftp</artifactId>
        <version>1.0-alpha-6</version>
      </extension>
    </extensions>
    ...
  </build>

  ...
  <distributionManagement>
    <repository>
      <id>example-ftp</id>
      <url>ftp://ftp.example.net/home/samiam/public_html/maven2</url>
    </repository>
```

```
  </distributionManagement>
</project>
```

The <extensions> section loads the Maven plugin that handles FTP uploads. The <distributionManagement> section tells Maven where to publish the project. Here we specified a <url> with the full URL path to our maven2 directory. The <id> tag in the <repository> is a name that you choose — just remember it for the next step.

Step 3: Configure Your Server Credentials

Remember that the *pom.xml* will be part of the public distribution of your project. It's OK to put the name of the FTP server there, but you wouldn't want to include private information like your user name and password. Those go in a special Maven configuration file called *settings.xml* that you keep private.

You can find *settings.xml* in your personal Maven cache directory. On Unix-like systems, it should be at *~/.m2/settings.xml*. You may have to create the file if it doesn't already exist. Here's what it should contain:

```
<settings xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <servers>
    <server>
      <id>example-ftp</id>
      <username>samiam</username>
      <password>greeneggs</password>
    </server>
  </servers>
</settings>
```

The <id> is the same as in our *pom.xml*. The user name and password are the credentials for your FTP server.

Step 4: Deploy!

Now all you have to do is run this command in your project directory:

```
mvn deploy
```

Maven builds your project and uploads it to your public repository. That's all there is to it!

Take a look with your web browser at http://www.example.net/maven2/net/example/awesome/1.0-SNAPSHOT and you'll see the JAR and POM files there.

Step 5: Tell the World

Now, anyone who wants to use your awesome library can just add a dependency to their pom.xml, like this:

```
...
<dependencies>
  ...
  <dependency>
    <groupId>net.example</groupId>
    <artifactId>awesome</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
  ...

</dependencies>
...

<repositories>
  ...
  <repository>
    <id>example-net</id>
    <name>The Example Repository</name>
    <url>http://www.example.net/maven2</url>
  </repository>
```

```
    ...

</repositories>
...
```

The <repository> section is necessary, since your repository is not on the list of "central" repositories that Maven searches by default.

In general, if your project has a stable release that is widely used, then it's worth the effort to get it into the central Maven repository. This is easier to do when you already have your own personal repository to point to. Note that the central repository does not accept SNAPSHOT releases, nor does it allow any changes to a release after it has been uploaded.

 Stuart / September 8, 2009 / Programming / Maven

7 thoughts on "Run Your Own Maven Repository With Nothing but an FTP Server"

Seba

October 17, 2009 at 12:39 am

Is there a way to hide the distributionManagement stuff? maybe putting it on some other pom. I wouldn't like to show the details of the deploy path in a public pom.

 **Stuart** ▲

October 17, 2009 at 10:49 am

I don't know. You could put it in a parent POM, but that still has to be deployed to a public repository. Maybe ask the Maven mailing list.

Anil

April 17, 2010 at 6:38 pm

Thanks. Works perfect. And migration between the server is swift.

The best part is, a five dollars host gator account serves excellent :).

Only complaint i have so far is unlike artifactory, the eclipse maven is unable to pull older version to suggest while editing the pom file.

Pingback: Hosting Maven Repos with Github | cemerick

Pingback: Hosting Maven Repos on Github | cemerick

Pingback: Dependency Management - Digital Digressions by Stuart Sierra

---

Mani

August 23, 2012 at 8:47 am

Hi Stuart,

What you mentioned is works perfectly. Now we want to use ftp repository instead of http from the server location. is there a way to do it...? can you please help in this. How maven can handle the ftp type repository instead of http...?

Comments are closed.