Project Cookbook

Contains info for

- Part 0, Dev Environment
- Part 1, GIT and tools

Part 0 - JEE Setup

Note: JEE is not a typo, it is the new name for what the prof calls J2EE.

NOTE ON JAVA FOR WINDOWS

- Be sure to install **JDK 1.8 u144 64 bit**.
- **NOTE** that Java.com now defaults you to 32 bit. The easiest way is the windows installer: https://www.eclipse.org/downloads/download.php?file=/oomph/epp/oxygen/R/eclipse-inst-win64.exe
- For **Mac,** you're fine; Mac is always 64 bit

Eclipse IDE

1. Download and install the JEE IDE (Oxygen) from
    a. Windows: https://www.eclipse.org/downloads/download.php?file=/oomph/epp/oxygen/R/eclipse-inst-win64.exe
    b. Other OS: https://eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/oxygenr
2. Takes a long time. Have a coffee.
3. The installer will also update all your components, make sure to do this to get the latest versions so we're all in sync.

4. For those new to Eclipse, you need Eclipse IDE for Java EE Developers

5. This one particularly burried link is useful if you are new to Eclipse: http://help.eclipse.org/oxygen/nav/1
6. Personally, I use `New Project | Dynamic Web Project` to get started, but there's other ways

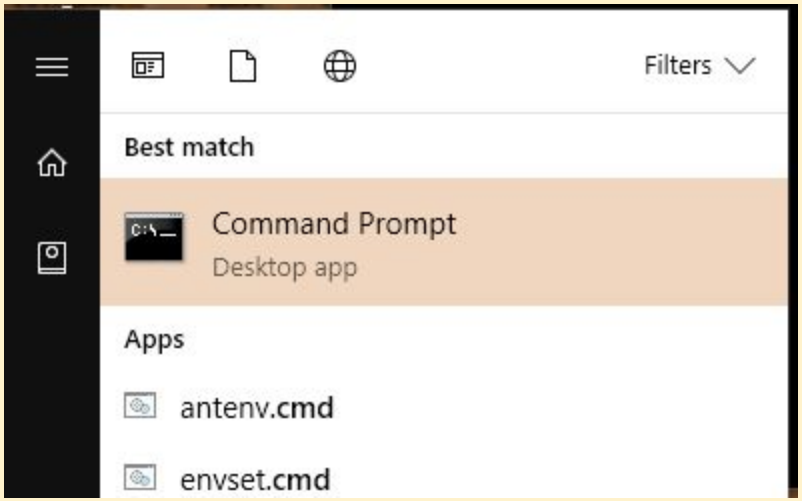
Tomcat

1. NOTE: Tomcat has a web server (Apache) built in. You will not need to install Apache, IIS, or any other web server to use it.
2. Required version: **8.5.20**.
3. Download and unzip/install: http://tomcat.apache.org/download-80.cgi#8.5.20
4. Starting / stopping Tomcat on PC/Mac is different; see http://crunchify.com/how-to-start-stop-apache-tomcat-server-via-command-line-setup-as-windows-service/ for info.
5. Tomcat uses port 8080; once running go to http://localhost:8080 and you should see the default home page.
6. Tomcat also user port 8443 for SSL.
7. You can change them to the standard ports 80 & 443 via Tomcats `/conf/server.xml` file.

8. Setting up SSL is lengthy / complex. If it's new to you, baby step it. My masters thesis was on Digital Certificates and I'm happy to help.
9. You will NEED a SELF-SIGNED certificate
10. This link has a detailed information: https://tomcat.apache.org/tomcat-8.0-doc/ssl-howto.html but this link has the best step-by-step: https://www.mulesoft.com/tcat/tomcat-ssl
11. Note that depending on your browser security settings, you may have trouble loading the SSL pages because the self-signed certificate is invalid. Click ignore or the equivalent; you're safe.
12. The deployment environment will have a legitimate certificate.

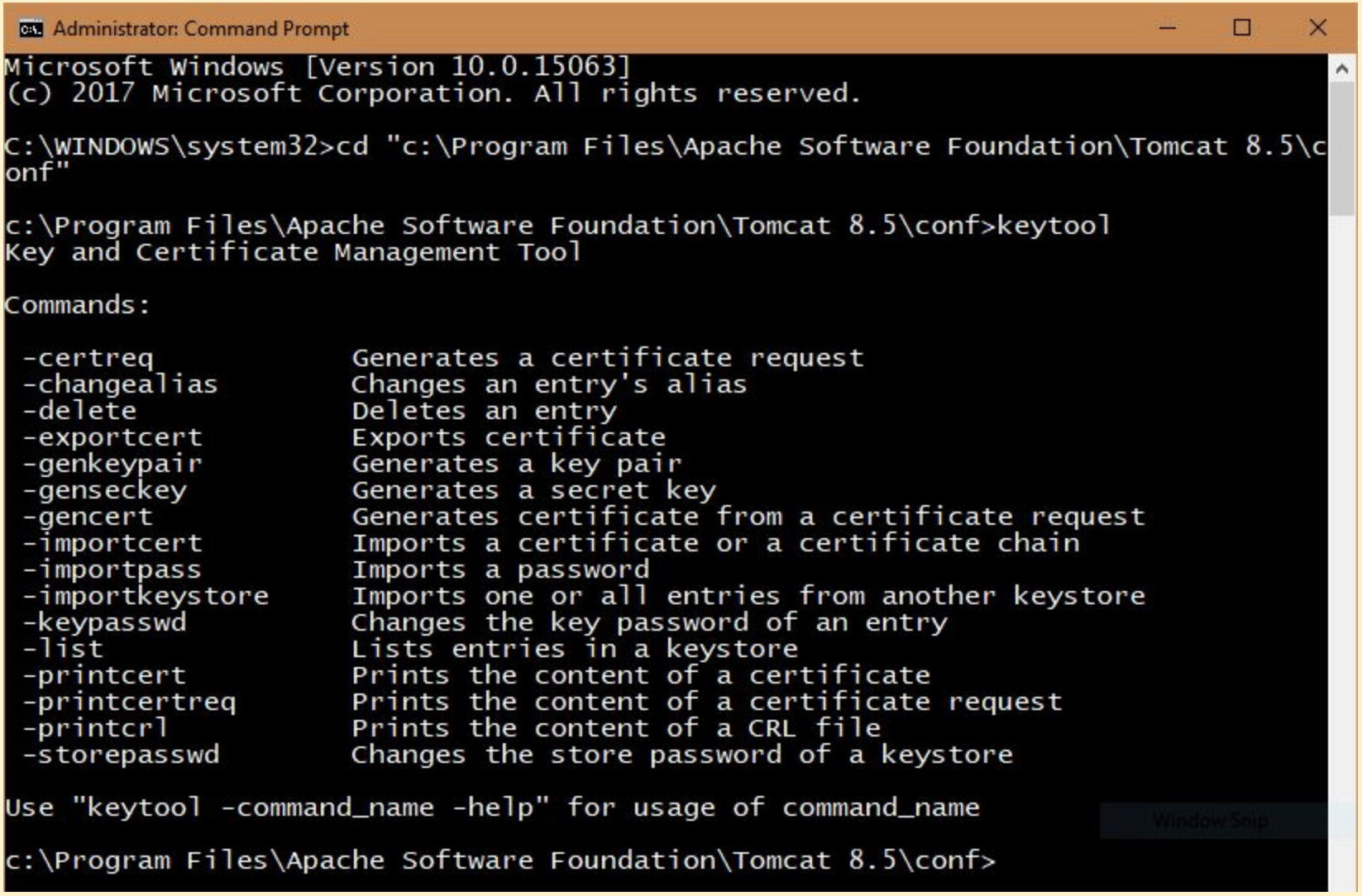
Additional Notes on SSL Setup

13. To make changes to the Tomcat directories, you must open a CMD shell in Administrative mode. Go to **START | CMD** but instead of clicking CMD, highlight it and hit **CTRL-SHIFT-ENTER**



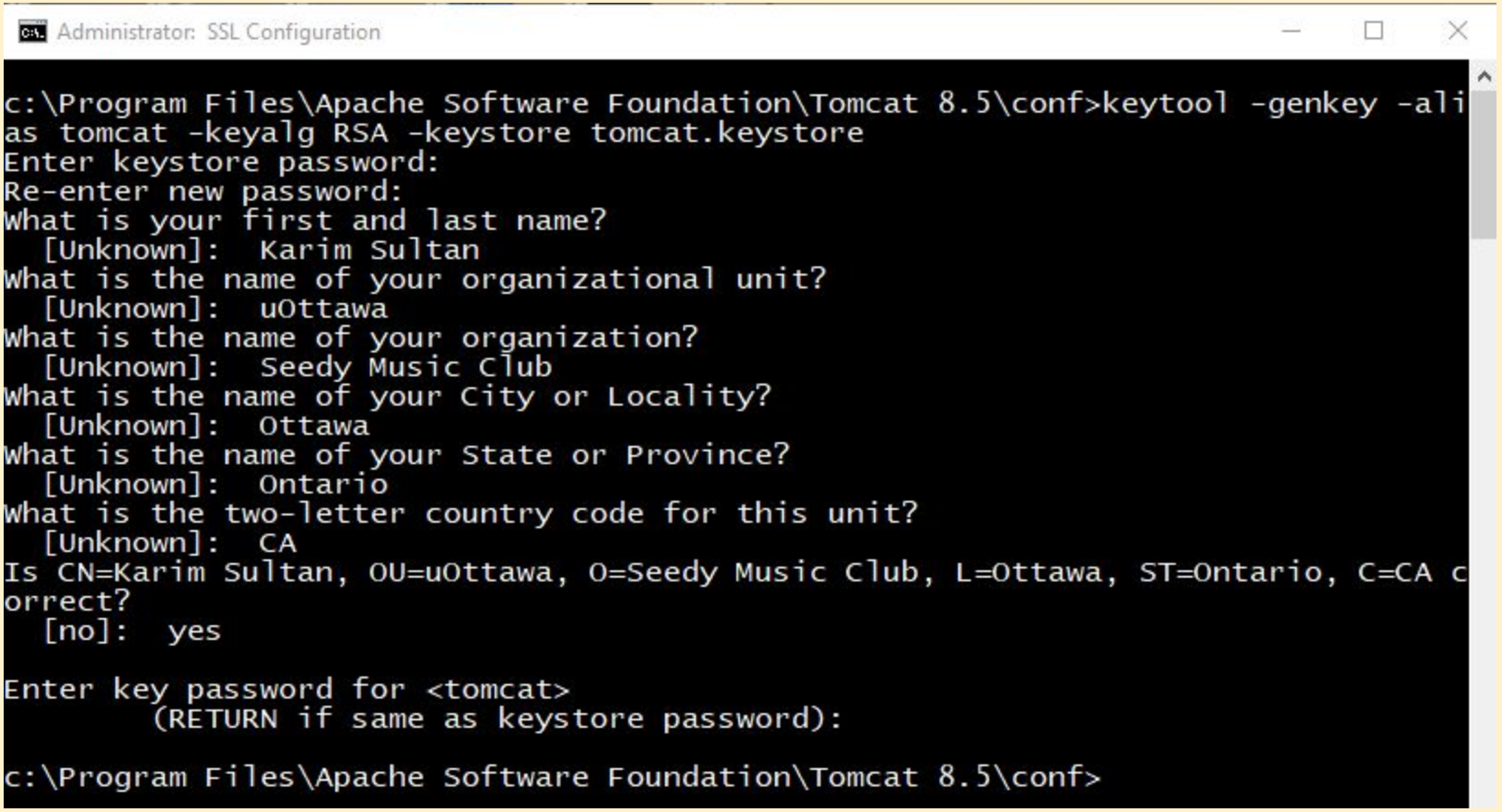
14. You will be prompted to approve running it as Administrator, say yes

15. Go to the Tomcat **conf/** directory and make sure you can run the **keytool** app



16. Enter the following command; you will need to make a password when prompted. Make sure that at the end, you say “yes” to reuse the same password, otherwise Tomcat may have issues (old bug). **This generates a self-signed certificate for**

Tomcat to use.



17. THERE ARE MULTIPLE WAYS TO DO THIS NEXT PART. I am showing how to do it in Eclipse.
    a. From the project explorer, open **Servers | Tomcat v8.5 Server… | server.xml**
    b. Uncomment the first `<Connector port="8443"...> … </Connector>` block (the SSL connector block - not the 8080 block) and replace it with below value:

```
<Connector
      protocol="org.apache.coyote.http11.Http11NioProtocol"
       port="8443"
      maxThreads="150"
      scheme="https"
      secure="true"
      SSLEnabled="true"
      keystoreFile="c:\program files\apache software foundation\tomcat8.5\conf\tomcat.keystore"
      keystorePass="password"
      clientAuth="false"
      sslProtocol="TLS"
/>
```

18. Run a jsp to make sure the JSP is working normally under port 8080. Then change the address to **`https:`**`//localhost`**`:8443`**`/your-jsp.jsp`
19. You will get a certificate error. **This is expected.** Just continue. Your JSP will come up under SSL!

20. The last part is to make Eclipse automatically reroute to HTTPS instead of you having to do it manually.
21. If you want your server to automatically redirect to SSL `https` when a user goes to the `http` portion, you need to add this to the bottom of the `web.xml` file (**Eclipse | Servers | Tomcat…| web.xml**) file (just before `</web-app>`)

```
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Protected Context</web-resource-name>
            <url-pattern>/*</url-pattern>
```

```
        </web-resource-collection>
        <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
```

And presto, **automatic SSL**!

MySQL

1. So many versions… You want the **MySQL Community Server 5.7.19** from https://dev.mysql.com/downloads/mysql/
2. Another long DL and install process… have another coffee (it's our lifeblood).
3. Setup the superuser account (process varies by OS, part of installer)
4. Start MySQL as a service.
5. Create your database (schema) either through shell or workbench
6. Download the sample schema as `schema.sql` from https://uottawa.brightspace.com/content/enforced/47965-CSI5380[F]_20179/csfiles/home_dir/Project/cd.sql?ou=47965
7. Enter the MySQL shell and type "`source schema.sql`" (depending on OS, you may need to use \source) **OR** go into MySQL Workbench, `File | Run SQL Script`.
8. Validate it worked by typing "`SELECT * FROM CD`"

JDBC

1. NOTE: Some versions of the MySQL installer provide the JDBC driver automatically; **mine didn't (Linux).**
2. You will need **Connector/J 5.1.44** from: https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.44.zip
3. Ignore most of the zip archive content: you will only need the JAR file: **mysql-connector-java-5.1.44-bin.jar**
4. Put it in the Tomcat `lib/` directory.

5. It will now be available to all webapps in Tomcat.
6. **OPTIONAL:** These steps will let you reference the DB by name via tags in your JSP or via resource lookup in your Servlet for testing purposes. You don't need to do these steps if you don't want to.
7. In your webapp's `/META-INF/context.xml` file, add (create the file if you don't have one):

```
<Resource name="jdbc/[database]"
          auth="Container"
          type="javax.sql.DataSource"
          maxTotal="100" maxIdle="30" maxWaitMillis="10000"
          username="[username]"
          password="[password]"
          driverClassName="com.mysql.jdbc.Driver"
          url="jdbc:mysql://localhost:3306/seedy_db"
/>
```

7. In your webapp's /WEB-INF/web.xml file add:

```
<resource-ref>
    <description>DB Connection</description>
    <res-ref-name>jdbc/[database]</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

8. These extra steps let you access the database via **tags** in your JSP page, for example:

```
<sql:query var="rs" dataSource="jdbc/[database]">
   SELECT DISTINCT category FROM CD
</sql:query>
```

9. It also allows you to reference it by named resource in your servlet and not have to know the actual DB name and credentials as you can load them from the web.xml instead. Of course, you can hard code that info into your servlet, but then it won't be portable between your dev environment and the deployment environment.

Tag Libraries

1. Tags are reusable "snippets" for common commands that you can use in JSP. View them as java functions that you can call directly from the JSP source. Even if you don't use these tags, some of us will so will need to install them:
2. Download the Apache Tag Libraries **1.2.5** into the Tomcat `lib/` directory:
3. Impl: http://apache.mirror.vexxhost.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-impl-1.2.5.jar
4. Spec: http://apache.mirror.vexxhost.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-spec-1.2.5.jar
5. EL: http://apache.mirror.vexxhost.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-jstlel-1.2.5.jar
6. Compat: http://apache.mirror.vexxhost.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-compat-1.2.5.jar

Testing it All

1. You'll need to write a servlet and map its URL context; it has to load the categories from the CD table via SQL over Connector/J and output HTML results via SSL. Software's the easy part!

****

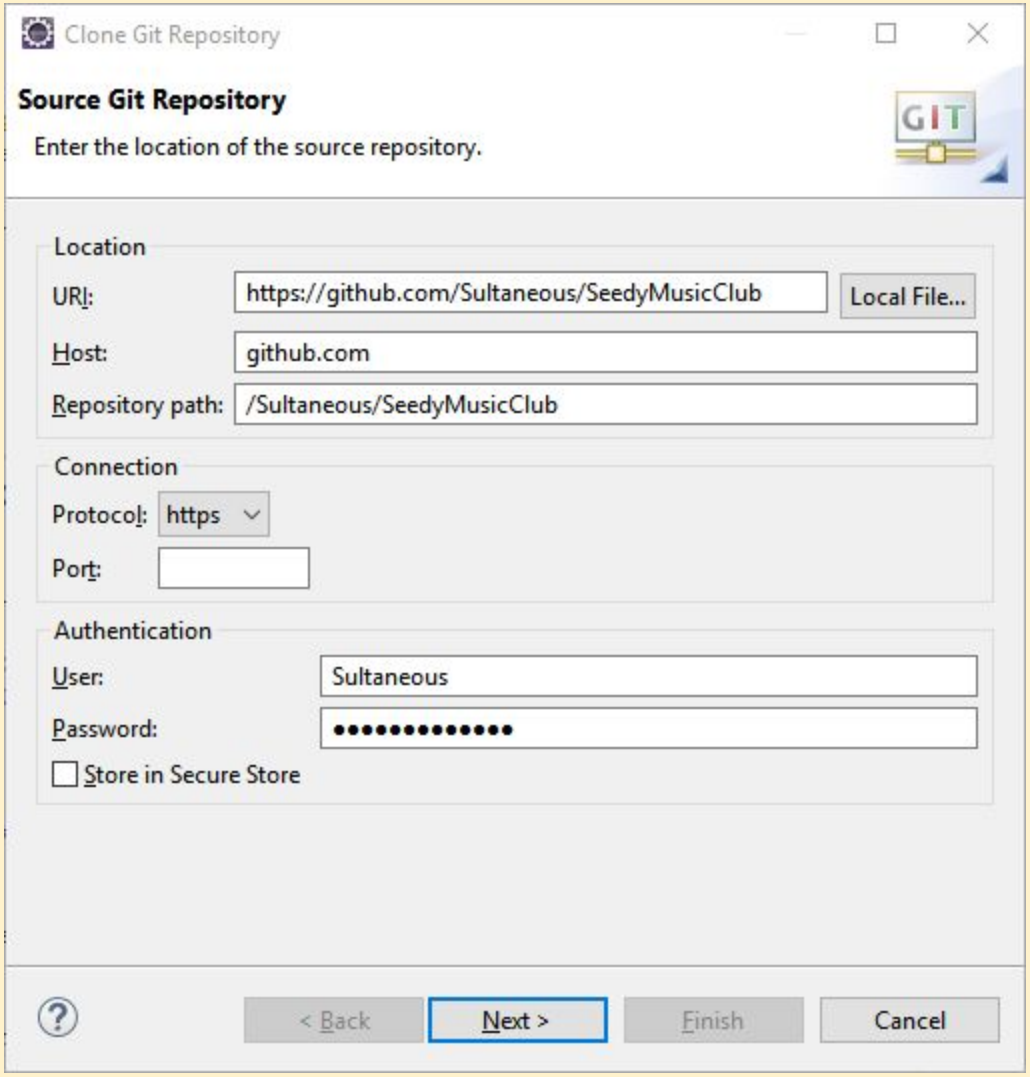
Part 1 - Tools

Git

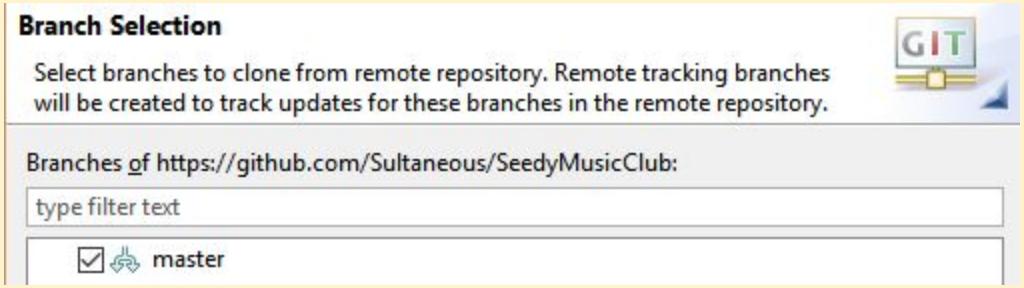
- Git is our source code versioning / sharing tool so that we all use the same source code. It works like this:
    - Update your source to what is on the server;
    - Make your changes to JSP, java, configuration, etc… files;
    - Commit and push the changed files back to the server.
    - Sometimes, you will need to resolve merge conflicts
- First, create a **github** account on www.github.com
- Second, do the Hello World tutorial, it is simple but teaches all you need to know about Git.

Git on Eclipse

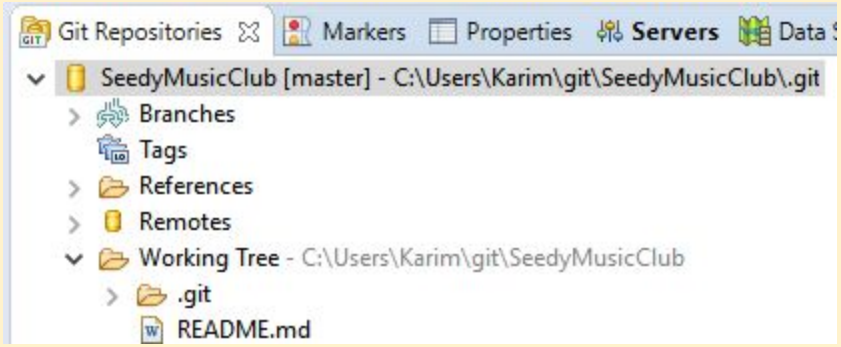
- So easy to work with, but first you need to add the **Git Repositories** tab to Eclipse:
    - Window | Show View | Other… | Git | Git Repository
- Now go to this URL: https://github.com/Sultaneous/SeedyMusicClub
- From your browser address field, drag and drop the URL onto Eclipse.
- You will get this view:



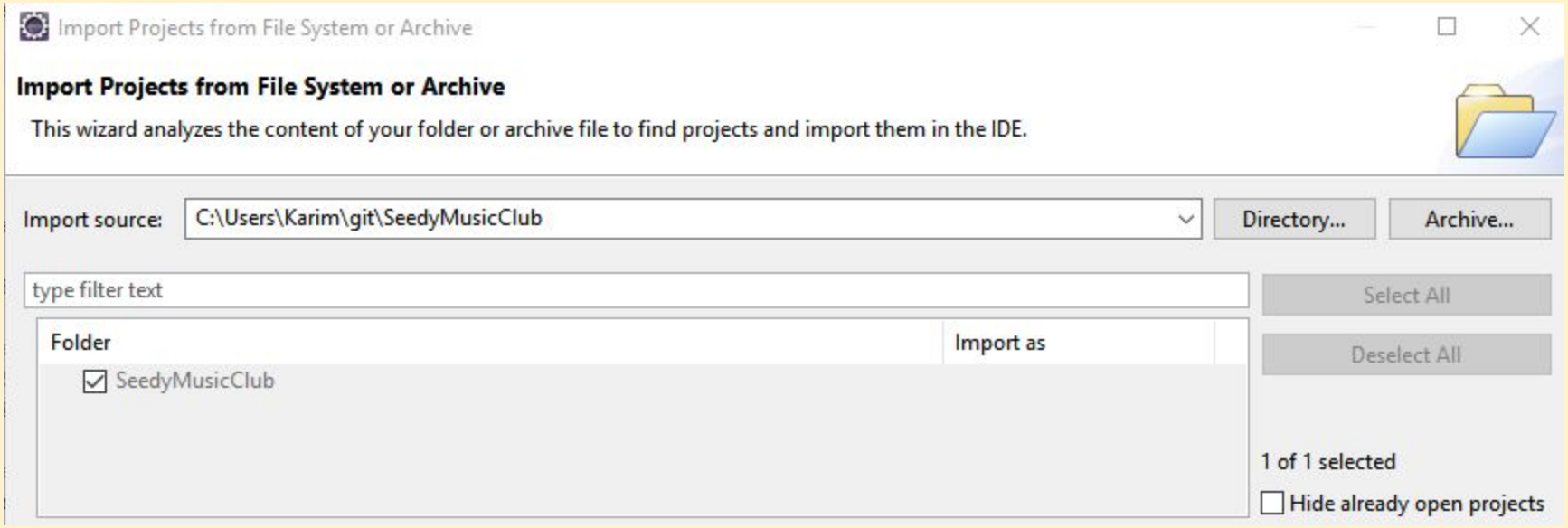
- You need to add your user/password info for your Github account
- On the next screen, make sure the **Master** branch is selected:



- Hit **Next** and then **Finish**
- Now you need to create an Eclipse **project** from the repository. Go to the **Git Repositories** Tab in Eclipse:



- Right click and select **Import Project…**



- NOTE: Your import source will be the local copy of the gir repository that you made in the first few steps