Integrating MySQL into Java Applets

A guide create signed Java Applets with Database Connectivity

Wayne Weibel, Bastian Tenbergen

wayne@wayneweibel.net| bastian@tenbergen.org

http://www.wayneweibel.net | http://www.tenbergen.org

Human-Computer Interaction MA Program

Department of Psychology

Department of Computer Science

State University of New York, College at Oswego

Oswego, NY, USA

1. Abstract.

This tutorial discusses how to go about integrating MySQL into Java Applets. The process is rather straight forward, but there are some things to pay attention to. This document is an attempt to compile all available information on the Internet and combining them into one resource and to straighten out some of the contradictory (and sometimes false) information available. Also, some of the tutorials and resources on the Internet are somewhat outdated or overly complicated. Hence, this tutorial proposes a method that is known to work at least for me in my projects and is as simple as possible.

2. Table of Content.

3. Preamble.

This tutorial is my attempt to summarize my findings in search the Internet on how to do so, along with massive findings that I discovered myself in integrating MySQL into Java Applets. I neither claim this document to be correct, nor flawlessly correct. If you notice any glitches, please contact me for revisions.

4. Introduction.

Why this tutorial? As you may or may not know, Java Applets run in a so-called sandbox – a simplified environment in the browser that makes executed foreign code safe and harmless. This sandbox has a very strict security policy and does not allow Java Applets to perform Input/Output operations such as reading and writing files to disk or utilizing the network interface on the client machine. Yet, it is possible to do so, but it requires signing the JAR file, the Applet lives in. A different common issue with Web distributed JAR content is the correct set-up of classpaths and dependencies. Information on this topic are rather sparse and not beginner-friendly. This tutorial suggests a work-around, which may work for many projects.

5. Assumptions.

I will assume that you have an understanding of how to do create Java Applets in first place. I will assume that you have read other tutorials (which you should find by google'ing [4]) that explain to you how Applets differ from applications and how to implement them. You will most likely want to try and develop your Applet within Eclipse, so I will assume that you have done just that and everything is working just fine. Another assumption is that you have access to some web server, and that you can use the world's largest knowledge base [4] to find out how to deploy Applets to an HTML document. Lastly, I will assume that you are familiar with manipulating Zip files – either using some graphical tool or the command-line.

6. Integrating MySQL into Java Applets.

In this little tutorial, we will deploy a simple MySQL enhanced Java Applet to an HTML page. This is universally, completely independent from your application, and only depends on the Applet code itself. Therefore, we will keep this tutorial as generic as possible.

6.1. The Short Version. Like always, this is for the impatient.

Do the following steps and it will work (assuming your Applet is running just fine in Java's Appletviewer and you are done developing for now):

1. Remove all MySQL driver resources from the classpath. Yes, you heard right.
2. Create a JAR file for your Applet.
3. Copy the MySQL Driver binaries from the MySQL Driver JAR into the root folder of the Applet JAR.
4. Create a certificate in you local keystore.
5. Export the certificate.
6. Sign the Applet JAR that contains the MySQL Driver binaries.
   The certificate will be valid for 6 months only.
7. Deploy the Applet JAR to a HTML page hosted on a web server.
8. Stir, serve, and enjoy - you're done :-)

Now for the detailed stuff.

6.2. Necessary Software.

Of course, you will need the MySQL Driver binaries for Java [7]. Install it, by adding it to your classpath. I strongly recommend to use some form of Integrated Development Environment (IDE) for your development. Develop your Applet and test it in the IDE's (ideally, Eclipse [4], version 3.4 or higher, also known as Ganymede [5]) hosted mode using Java's Appletviewer. You will also need a database you can use - get WAMP [6], as it comes with all you need for that. MySQL, a good free database server and some PHP-based management software. Make sure MySQL in WAMP is running when you deploy. Your MySQL server must be reachable from the Internet to be of any value for you in production. You will also need a Zip Utility, like WinZip, PowerZip, WinRar or whatever. *NIX flavor operating systems typically come with a cool command-line tool that you ought to be familiar with.

6.3. Remove Driver from Classpath.

This step is optional. Since you already have written your Applet, and all you want to do is deploy it, I think it is safe to assume that you have your MySQL Driver set up in your project's classpath. This may be either a global one, being valid for all your projects or even your entire development machine, or this may be just for this one project. The latter method is how Eclipse likes to do it. To make things easier when you create the JAR file, you may want to remove the MySQL Driver JAR from the classpath. In Eclipse, simply right-click on your project, navigate to `Properties` in the context-menu and find `Java Built` on the left hand side. Access that menu and follow the screen instructions to remove the JAR from your project classpath.

6.4 Create Applet JAR File.

Now you are ready to build the JAR file that will host your Applet. Certainly, one can run Applets as a collection of class files in a folder structure from a web server, but this won't allow you to sign the JAR. Right-click on your project in Eclipse, then go to `Export...`. Under `Java`, find the entry `JAR file...` and follow the instructions of the wizard. You may select whether or not you would like to include source files at your convenience, but when it asks you if you would like to include dependencies and resources, make sure to **not specify** the MySQL Driver – we will do this in the next step. Continue with the wizard. At the appropriate step, it will ask you if you would like to provide a Manifest file. Make sure to advise the wizard to auto-generate one on your behalf. This is necessary, as otherwise signing the JAR later on might fail. You may close your IDE now – we won't need it anymore.

6.5 Copy MySQL binaries into Applet JAR.

Now for a somewhat unusual part – we are placing all binaries from a resource JAR into the new Applet JAR. The reason why we didn't do this in step 6.4, but are doing it independently is that Eclipse sometimes tries to de-compile class files, when it kind find the source, and this might result in problems with MySQL. Also, doing it the other way, i.e. finding MySQL Driver source files and compiling it yourself is quite an undertaking, which is unlikely to succeed – therefore, we do it the manual way.

In the filesystem of your development computer, find the MySQL Driver binary JAR that you

once downloaded and the Apple JAR that you have just created. Open both in your favorite Zip utility, I personally prefer WinRAR, or the command-line. In the MySQL Driver JAR, you will find two directories – `org` and `com`. Copy both of them into the Applet JAR into the root folder. Alternatively, you may unzip the MySQL Driver JAR and add the contents to your Applet JAR. Careful, however! Make sure not to overwrite the Manifest file if you use this method!

6.6 Create Certificate in local Keystore.

The directions in this section are based on a forum entry (http://forums.sun.com/thread.jspa?threadID=174214 , accessed December 2008). This method is one of the few universally working methods that I discovered and it is fairly easy.

Open a command-line window. Enter the command

```
keytool -genkey -keyalg rsa -alias yourKey
```

where `yourKey` is some alias under which you can refer to the key you will now create. Naturally, you may specify any key algorithm using the `-keyalg` argument, but RSA will work just fine and is as good as any. Answer the questions the tool will ask you for – don't worry. In the end, you may review the answers and make changes. Remember to remember your password well! Especially the one for the Keystore, as this is the master password for all keys.

6.7 Export Certificate.

This step is also optional. Exporting the key is however quite useful when you wanty to sign many things with the same key on different machines. In that case, you have to import the file you exported after this step (we will not cover that in this tutorial, though).

Still in command-line mode, enter the following command:

```
keytool -export -alias yourKey -file file.ext
```

where `yourKey` is the key alias you have specified in step 6.6 and `file.ext` is some file name with some extension. A convention is to use the alias of the key as the file name with the extension `.crt`. Hence, if your alias is `yourKey`, it is a good idea to call the file `yourKey.crt`.

When you hit enter, the key with the specified alias will be placed in the certificate file in the current working directory. You may have to move it into the same directory of your Applet JAR.

6.8 Sign the Applet JAR.

Now for the step that makes it all possible to run Applets with I/O permissions and MySQL: We are signing the Applet JAR. A word on this, however. In this tutorial, we use what is called a self-signed certificate. These may or may not be acceptable for every user, depending on their individual (or corporate) security policies regarding foreign interpreted code. After all, the code you have written could cause harm. Also, **the certificate will be valid for only 6 months** by default, which means that after 6 months, your users will receive a message, indicating that the certificate of the Applet has expired. When that happens, you will need to repeat this step again and re-sign your JAR.

To sign your Applet and MySQL Driver JAR, simply enter the following in command-line:

```
jarsigner Applet.jar yourKey
```

`Applet.jar` is the name of the Applet JAR that you have created in step 6.4 and that contains the MySQL Driver binaries from step 6.5. Note, that you will have to specify the same key from step 6.6 for what is called `yourKey` in this example. Also, you will be prompted for the Keystore password – I hope you still remember it. After this is done, you may verify the JAR:

```
jarsigner -verify -verbose -certs Applet.jar
```

And it should tell you some details about the JAR and let you know if an error is present.

6.9 Deploy Applet JAR to HTML Document on Web Server.

You are done. Well almost. You Applet is now ready to be run from within a web browser and for that, you of course need to upload your Applet to a server and integrate it to an HTML Document. How this is done is simple and easy and tutorials can be found plenty on the Internet. A short quick start shall be this, just for completeness:

Create an HTML Document and add the `<applet>` tag. Enter the main class of your Applet that extends `java.applet.Applet` in the `code` attribute and naturally the Applet JAR in the `archive` attribute. Somewhat like this:

```
<applet code="com.yourdomain.packagename.YourAppletWithMySQL.class"
                        name="Some name goes here"
                        archive="AppletWithMySQL.jar"
                        width="1000" height="700">
                        Your browser is not Java enabled.
</applet>
```

7. Summary.

Well, this is it. Your Applet is now available through the HTML Document on your web server and should work with your MySQL database. I hope you like this little tutorial and find it helpful in your next Applet programming venture. Let me know of any successes, failures, and whatnot.

As the other GWT tutorial, this document was made possible, in part, by many kind souls in various forums across the Internet that post answers to questions of others. Those people are way to many to mention them all, but all of them shall be thanked much.

8. References.

[1] Dewsbury, R. (2007). *Google Web Toolkit Applications.* Boston, MA: Addison-Wesley Professional.

[2] Geary, D. with Gordon, R. (2008). *Google Web Toolkit Solutions.* Boston, MA: Prentice Hall PTR.

[3] Google Search Engine, accessed October, 24th, 2008

http://www.google.com/

[4] The Eclipse Project, accessed October, 24th, 2008

http://www.eclipse.org/

[5] Eclipse IDE v3.4 "Ganymede", accessed October, 24th, 2008

http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/ganymede/SR1/eclipse-jee-ganymede-SR1-win32.zip

[6] WAMPServer – Apache, MySQL, PHP on Windows, accessed October, 24th, 2008

http://www.wampserver.com/en/

[7] MySQL Connector/J, accessed October, 24th, 2008

http://www.mysql.com/products/connector/j/

[8] Niederst, J. (1999). *Web Design in a Nutshell, 1st Edition.* O'Reilly.

[9] Silberschatz, A., Korth, H. F., Sudarshan, S. (2006). *Database System Concepts, 5th Edition.*

McGraw-Hill Higher Education.