

Deploying MySQL Enhanced GWT Applications

A guide on deploying GWT Applications with MySQL Servlets to live dynamic web servers.

Bastian Tenbergen

bastian@tenbergen.org

<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 deploying Web Applications made with the Google Web Toolkit [5] that are enhanced by a database connection through a MySQL Servlet. 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 Also, some of the tutorials and resources on the Internet are somewhat outdated or overly complicated.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.

1. Abstract	2
2. Table of Content.	2
3. Preamble.	3
4. Introduction.	3
5. Assumptions.	3
6. Deploying MySQL enhanced GWT applications.	4
6.1 The Short Version.	4
6.2 Necessary Software.	5
6.3 Modify Web.xml for Servlet.	5
6.4 Create WAR File.	6
6.5 Install MySQL into Tomcat.	7
6.6 Install WAR file into Tomcat.	7
6.7 Launch Tomcat.	7
6.8 Verify success of Deployment.	8
7. Summary	9
8. Acknowledgements.	9
9. References	10

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 “taming” the beast of GWT and MySQL. I neither claim this document to be correct, nor flawlessly correct. If you notice any glitches, please contact me for revisions.

This document may be used, (re-)distributed, shared and made available in printed or electronic version for personal, educational, and scientific use without explicit permission. You may not gain any financial profit or any other profit from my work without explicit prior written permission. Prerequisite for any distribution and use of this document is that used in it's entirety, with copyright notices intact.

4. Introduction.

Why this tutorial? Isn't there enough info on this on the Intertubes? Well, yes. And no. Just like the first tutorial on GWT that I have written, this document aims at compiling all available information, getting rid of any contradictions and false information, and serving as one resource detailing the simplest possible way.

Recall: The actual database queries are handled by a **Servlet** and you need to make **RPC calls**, using a **callback handler** to have the client side GUI (as the GWT application is executed client side in the browser, not on the server!) talk to the database. The mission of this tutorial is hence to show the necessary steps needed to deploy MySQL enhanced GWT applications and explain what happens at each step.

5. Assumptions.

I will assume that you have an understanding of how to do GWT in first place. I will assume that you have read the other tutorial (which you should find at the same location you found this tutorial, or at least by [google'ing \[4\]](#)). You will most likely want to try and develop your GWT app in hosted mode within Eclipse, using Cypal Studio, so I will assume that you have done just that and everything is working just fine. Lastly, the assumption is that you have some dynamic web server, like Tomcat, at your disposal, and that you can use the [world's largest knowledge base \[4\]](#) to find out how to configure it.

6. Deploying MySQL enhanced GWT applications.

In this little tutorial, we will deploy a simple GWT application that uses a MySQL database connection that you have created using Eclipse and Cypal Studio. This is universally, completely independent from your application, and only depends on the Servlet, really. But for the more hands-on parts, we will take the little `LoginScreen` app from the other tutorial and its `MySQLConnection` Servlet as the living example.

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

Do the following steps and it will work (assuming your GWT application is running just fine in hosted mode and you are done developing for now):

1. Add your Servlet information to the `web.xml` file.
2. Create a WAR file containing your project and any resources.
3. Copy the MySQL Driver to the `$TOMCAT_HOME/common/lib` folder of your Tomcat.
4. Copy the WAR file from step 2 into the `$TOMCAT_HOME/webapps` folder of your Tomcat.
5. Start your Tomcat and watch it automatically deploy your GWT application.
6. Access your application via `http://servername:port/warfilename/`

Where `servername` is the URL of your server, `port` the port you told Tomcat to run on and `warfilename` is the file name of the WAR file you have copied into Tomcat's `webapps` folder.

7. In the hence displayed file listing, look for the HTML file that contains is named after the Java class that implements the `EntryPoint` interface, in this case `LoginScreen.html` and click on it.
8. Stir, serve, and enjoy - you're done :-)

Now for the detailed stuff.

6.2. Necessary Software.

Of course, you will need [GWT \[5\]](#). [Download it \[6\]](#) and install it, of course. I strongly recommend to use some form of Integrated Development Environment (IDE) for your development. Develop your Web app and test it in the IDE's (ideally, [Eclipse \[7\]](#), version 3.4 or higher, also known as [Ganymede \[8\]](#)) hosted mode using the integrated dynamic Web Server (most likely Tomcat Lite). I recommend [Cypal Studio \[9\]](#) for GWT integration with Eclipse, as it is free, and the documentation is just what you need, even though the [docs \[10\]](#) are somewhat sparse. Follow their how-to on getting started integrating Cypal Studio and making your first GWT App. You will also need a database you can use - get [WAMP \[11\]](#), 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. Get yourself a dynamic web server, ideally [Tomcat 5.5 \[16\]](#) or higher. Of course, your Tomcat and MySQL servers must be reachable from the Internet to be of any value for you in production.

6.3. Modify Web.xml for Servlet.

Cypal Studio will already have created the necessary folder structure and file system for your GWT application project. That's why you should follow the instructions on the GWT Documentation Page [10] or my other tutorial on GWT. In the folder structure, navigate to the `WebContent/WEB-INF` folder. In there, locate the file `web.xml` and add the following information to the end, before the document closing tag:

Web.xml:

```
<servlet>
    <servlet-name>
        MySqlConnection
    </servlet-name>
    <servlet-class>
        com.yourdomain.projectname.server.MySqlConnection
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>
        MySqlConnection
    </servlet-name>
    <url-pattern>
        /MySqlConnection
    </url-pattern>
</servlet-mapping>
```

The information in the `servlet-name` tags are fairly arbitrary, but must be identical in both, the `servlet` and `servlet-mapping` tags. Essentially, for every Servlet you have running within the application, you will need to set these two tags within the `web.xml` file accordingly. Under `servlet-class`, you will have to enter the fully qualified package name of your Servlet (in this case of the `MySQLConnection` file, as it implements your synchronous and asynchronous interfaces and extends class `RemoteServiceServlet` of GWT. Most importantly, you need to set the correct URL under `url-pattern`. The path there, is the same one that you have specified in your GWT XML configuration file. Remember your `LoginScreen` class? This one implements `EntryPoint` and establishes the MySQL Service Target. It also comes with a `LoginScreen.gwt.xml` file, in which you once set a `servlet` tag. In that tag, you specified the path of the package of the Servlet as well as a path. The path you have to set in `url-pattern` is the same one you set under `path` in the `servlet` tag in `LoginScreen.gwt.xml`. Sounds complicated, but it's easy... I promise.

6.4. Create WAR File.

I am strongly hoping that you are using Cypal Studio or some other GWT plugin for Eclipse ;-) Otherwise, you might have to go about creating this WAR file manually. This can be quite the challenge and trust me – you don't want to do that by hand. Thankfully, Cypal Studio can help there.

First, clean your project by going to `Project` then `Clean...` in Eclipse's menu bar. Select your GWT app project and Eclipse will clean, compile, and build your project. Next, select your project in your Project View of your workspace. Then, in Eclipse's menu bar, click on `File`, then `Export...` In this Dialog, go all the way down to where it says `Web` and click on the plus sign to open whatever is underneath. And this “whatever” should be an item called `WAR file`. Click on it and select `Next`. Follow the Wizard and save the file to where ever you like. Congratulations, your WAR file is created! By the way – a WAR file is essentially just like a JAR file with a special folder structure inside. You can even open it with your favorite Zip compression tool or even `jarutility` that comes with Java.

6.5 Install MySQL into Tomcat.

This step is really simple. Locate the MySQL Connector/J Driver that you have added to your GWT project's classpath in your filesystem. If you can't find it, that's kind of bad. Refer to my first GWT tutorial for a clue where you might have put it or get a new version at [12]. This file needs to be both in your GWT Application's classpath on the Tomcat server, as well as in Tomcat's common libraries folder. The first is done automatically, if you created the WAR file like I said in section 6.4. The latter is done very easily, by simply copying the jar (should be something like `mysql-connector-java-5.1.6-bin.jar`) into the `$TOMCAT_HOME/common/lib` folder of your Tomcat. `$TOMCAT_HOME` refers to the path in which you have installed Tomcat.

6.6. Install WAR file into Tomcat.

This one is just as easy as putting the MySQL Driver into Tomcat. Simply take the WAR file from where ever you told the Wizard in section 6.4 to put your WAR file and copy it into `$TOMCAT_HOME/webapps` folder of your Tomcat. That's it.

In fact deployment is complete at this point, but let's just keep working for just a little bit to verify that stuff worked.

6.7. Launch Tomcat.

This step is going to be simple, too. I could actually leave this one out, but I decided to keep it in here for completeness.

Once the WAR file is deployed, Tomcat will take care of the rest. This even works *while Tomcat is running*, which is called “hot deploying”. Just to be safe, we will shutdown the server and restart it.

To shutdown, simply execute the `shutdown.sh` (on Linux/Unix or Mac OS) or `shutdown.bat` (on Windows) scripts in the `$TOMCAT_HOME/bin` folder.

To start the server again, execute the `startup.sh` or `startup.bat` script in the same folder. Easy as pie.

Tomcat should now deploy that WAR file by unpacking everything therein into a directory with the same name as your WAR file. **Existing files will not be replaced!** So make sure that it is empty or make sure that you know what you are doing. If you want to replace a previous build, just delete the

existing directory. If you want to test two different builds concurrently, just rename the existing folder or the new WAR file to something else. You get the idea, I am sure.

6.8. Verify success of Deployment.

Now that Tomcat is up and running again, your GWT Application is deployed, and hopefully Tomcat didn't complain during deployment, it is time to verify that it works. In your favorite browser, go to the URL of your server, with the port Tomcat is running on and to the folder, you deployed to. This may be according to this pattern:

```
http://servername:port/warfilename/EntryPointFile.html
```

`servername` here stands for the URL you can reach your server under. This may be `localhost` or something like `yourdomain.com`.

`port` is the port that you have set Tomcat to run on. You can manipulate that in Tomcat's `server.xml` file. Pardon me for not detailing this procedure, but this would be beyond the scope of this tutorial, as Tomcat configuration can be quite a pain. So much be said, though: typical port numbers would be something like 8080, which is the dynamic web server port default, or 8100 (which is the JBoss default – and JBoss is yet another Tomcat flavor). I believe the Jakarta/Catalina default (Catalina is the code name of version 5.5 release of Apache Tomcat) is 8010, so you may assume that. Consult your `server.xml` in the `$TOMCAT_HOME/config` folder for the exact port.

`warfilename` is straight forward, right? It is the name of the WAR file that you entered into the Wizard in section 6.4. Unless of course, you changed it during following the steps in section 6.7. Remember that when hot deploying, Tomcat will extract the contents of the WAR file to a folder with the same name.

`EntryPointFile.html` is a little bit more special. As you may recall, your GWT application has a class that implements the `EntryPoint` interface – you know, the one with the `onModuleLoad()` method. This class name depicts also the GWT XML configuration file. During build, the GWT compiler has created an HTML file by that name and this is precisely the one you want to call in your browser. If you followed the example from the other tutorial, this would be `LoginScreen.html`. Hence, also according to that tutorial, your URL would be

```
http://yourdomain.com:8010/projectname/LoginScreen.html
```

You may leave out the HTML file from the URL, if you like. Then, Tomcat will show you a file listing of all files in the folder `projectname`. There is a lot of stuff in there that may sound like gibberish to you, but yet again, you want to look for a HTML file named like your `EntryPoint` class.

7. Summary.

OK, that's it! If everything worked out, you should see your GWT application in your browser, similar to what you were able to see in the hosted mode using GWT's own internal IDE server.

Let's recapitulate for a minute. In this tutorial, you have seen how to modify your GWT application to tell the dynamic web server about the Servlet, how to deploy the necessary resources and the GWT application itself, and how to verify that it works. Essentially, we enabled Tomcat to be able to pull data from some MySQL server, if some deployed GWT application that specifies a MySQL connection Servlet requests so.

But wait a minute... didn't we forget something? How about telling Tomcat were to find the MySQL server? How does Tomcat know where to pull the data from? Well, that is actually part of your Servlet. If you do it right, it won't be necessary to configure Datasources or JDNI's anymore (whatever that is) – this will be added to the “*context*” of the deployed GWT application during hot deployment. Tomcat, ergo, does it by itself. You will specify this in your GWT project, when you write the Servlet. Details can be found in section 6.8 of the other tutorial.

Well, folks, that's it. I hope you like this little tutorial and find it helpful in your next GWT programming venture. Let me know of any successes, failures, and whatnot.

8. Acknowledgements.

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. Also thanks to Doug for the kind support in all my work – your help will never be forgotten. Special thanks to Wayne for assisting me when frustration peeked and to Gilian for keeping me sane.

9. 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's GWT MySQL Example Project, accessed October, 24th, 2008
http://code.google.com/p/gwt-examples/wiki/project_MySQLConn
- [4] Google Search Engine, accessed October, 24th, 2008
<http://www.google.com/>
- [5] GWT – Google Web Toolkit, accessed October, 24th, 2008
<http://code.google.com/webtoolkit/>
- [6] GWT Download Section, accessed October, 24th, 2008
<http://code.google.com/webtoolkit/download.html>
- [7] The Eclipse Project, accessed October, 24th, 2008
<http://www.eclipse.org/>
- [8] 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>
- [9] Cypal Studio, GWT Plug-in for Eclipse Ganymede, accessed October, 24th, 2008
<http://www.cypal.in/studio>
- [10] Cypal Studio Documentation, accessed October, 24th, 2008
<http://www.cypal.in/studiodocs>
- [11] WAMP Server – Apache, MySQL, PHP on Windows, accessed October, 24th, 2008
<http://www.wampserver.com/en/>
- [12] MySQL Connector/J, accessed October, 24th, 2008
<http://www.mysql.com/products/connector/j/>

[13] GWT Serializable Types, accessed October, 24th, 2008

<http://www.gwtapps.com/doc/html/com.google.gwt.doc.DeveloperGuide.RemoteProcedureCalls.SerializableTypes.html>

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

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

McGraw-Hill Higher Education.

[16] Apache Tomcat version 5.5, accessed November, 28th, 2008

<http://tomcat.apache.org/download-55.cgi>