

# Tupelo Server Webapp Configuration

The Tupelo Server Webapp is a web application that allows remote access to a Tupelo Context. This document describes how to acquire it and configure it.

## Acquiring the Tupelo Server Webapp

**Prebuilt:** The easiest way is to simply download the war from the tupelo website:

[tupelo-server-webapp-2.4.war](#)

**Manual:** You may also acquire and deploy the server manually. After [downloading the Tupelo distribution](#) and [building it using maven](#), copy the generated Web Archive (WAR) file to the webapp directory of your web server installation. For example, if you are deploying under an Apache Tomcat server located at /usr/local/tomcat, you would use the following command from the tupelo build directory:

```
cp tupelo-server-webapp/target/tupelo.war ${TOMCAT_HOME}/webapps
```

Restart your web server, or use its management web page to load and start the tupelo server servlet.

In its initial state, the Tupelo server is backed by a non-persistent MemoryContext. To reconfigure the server to use a context backed by a persistent triple / blob store, follow the instructions below.

## Installation Preliminaries

The default deployment supports a Context backed by PostgreSQL, MySQL, or Sesame, so you need to have one of these installed. Other Contexts may be used and the detailed procedure is [here](#). A functional servlet container such as Tomcat, Jetty or Resin is also required. This has been tested with Tomcat and works with version 5.5 or above. The deployment description that follows is for Tomcat.

By default, authentication is required. The next two sections describe how to set it up or disable authentication.

### Setting up authentication.

In order to set access you will need to edit \$CATALINA\_HOME/conf/tomcat-users.xml and add the tupelo-user role, then either create new users or add this to any existing users.

Example. For a new user, bob with password "changeme" you would add the following:

```
<role rolename="tupelo-user" />
<user username="bob" password="changeme" roles="tupelo-user" />
```

Example. If the user Bob already exists with role of manager, you would add the tupelo-user role as

```
<role rolename="tupelo-user" />
```

and set Bob's entry as

```
<user username="bob" password="changeme" roles="manager,tupelo-user" />
```

### Disabling Authentication.

Note that the first person coming to the service will be able to configure it, so disabling authentication should only be done for a good reason. The easiest way is to deploy the server and afterwards do the following: Go to

```
`${CATALINA_HOME}/webapps/tupelo/web.xml
```

and open it in an editor. Comment out or delete the sections "security-constraint", "login-config" and "security-role". You should stop then restart the service.

## Deploying the server

To deploy the server, go to the main Tomcat page and click on Tomcat Manager. This gives a list of running services. At the bottom is a section entitled "war file to deploy". Browse to the location of the war file, then click deploy.

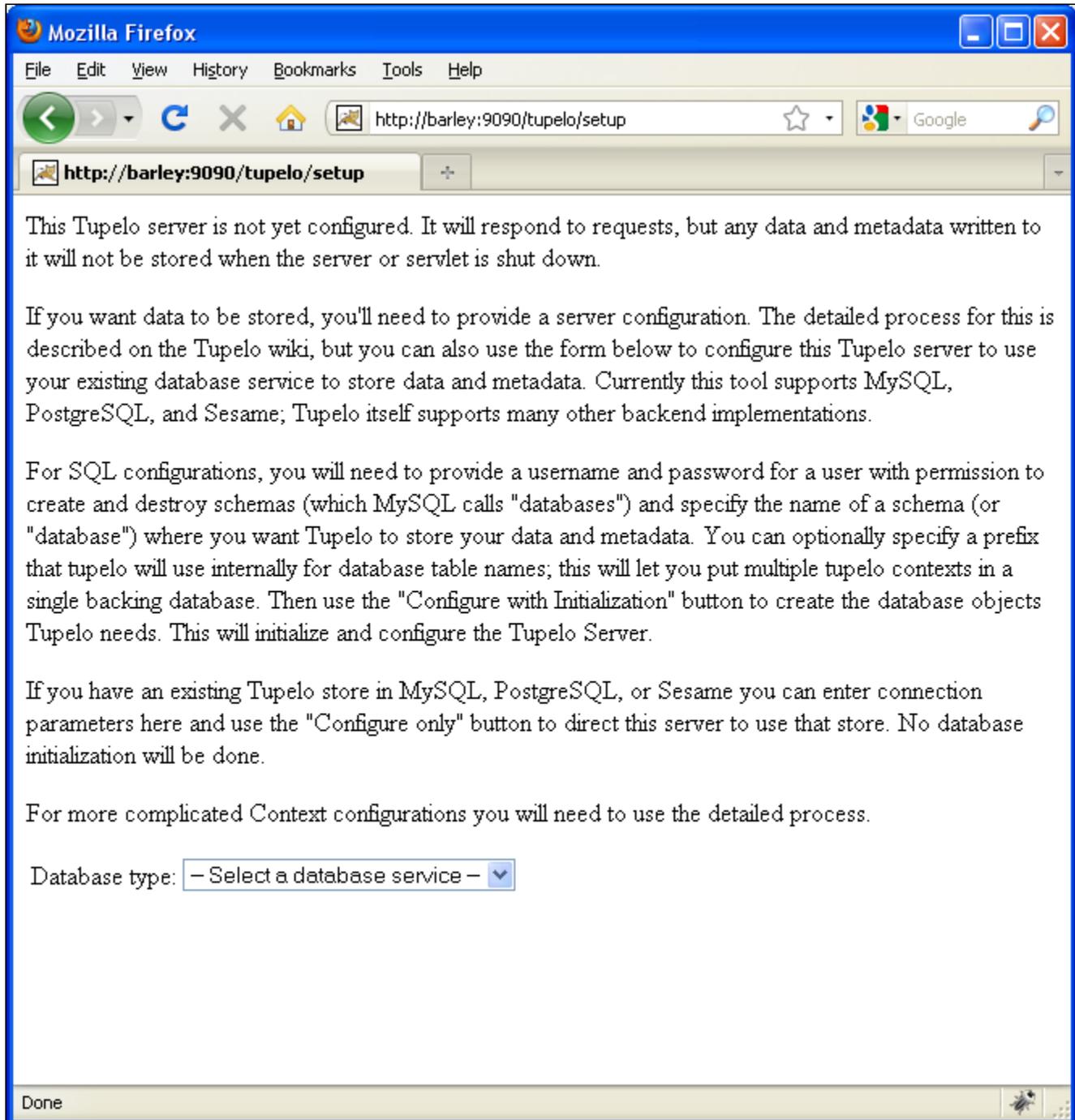
At this point, the tupelo server should show up in the list of running applications.

## Configuration

Once installed, the server will only be backed by a [MemoryContext](#) until configured. In that case, reads and writes from a client will only persist until Tomcat is restarted. The configuration page will be shown by going to the page

<http://host/tupelo>

where "host" is the name of the machine. If authentication is enabled, you will need to log in. Once in the system you should see the initial configuration screen:



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://barley:9090/tupelo/setup`. The page content includes the following text:

This Tupelo server is not yet configured. It will respond to requests, but any data and metadata written to it will not be stored when the server or servlet is shut down.

If you want data to be stored, you'll need to provide a server configuration. The detailed process for this is described on the Tupelo wiki, but you can also use the form below to configure this Tupelo server to use your existing database service to store data and metadata. Currently this tool supports MySQL, PostgreSQL, and Sesame; Tupelo itself supports many other backend implementations.

For SQL configurations, you will need to provide a username and password for a user with permission to create and destroy schemas (which MySQL calls "databases") and specify the name of a schema (or "database") where you want Tupelo to store your data and metadata. You can optionally specify a prefix that tupelo will use internally for database table names; this will let you put multiple tupelo contexts in a single backing database. Then use the "Configure with Initialization" button to create the database objects Tupelo needs. This will initialize and configure the Tupelo Server.

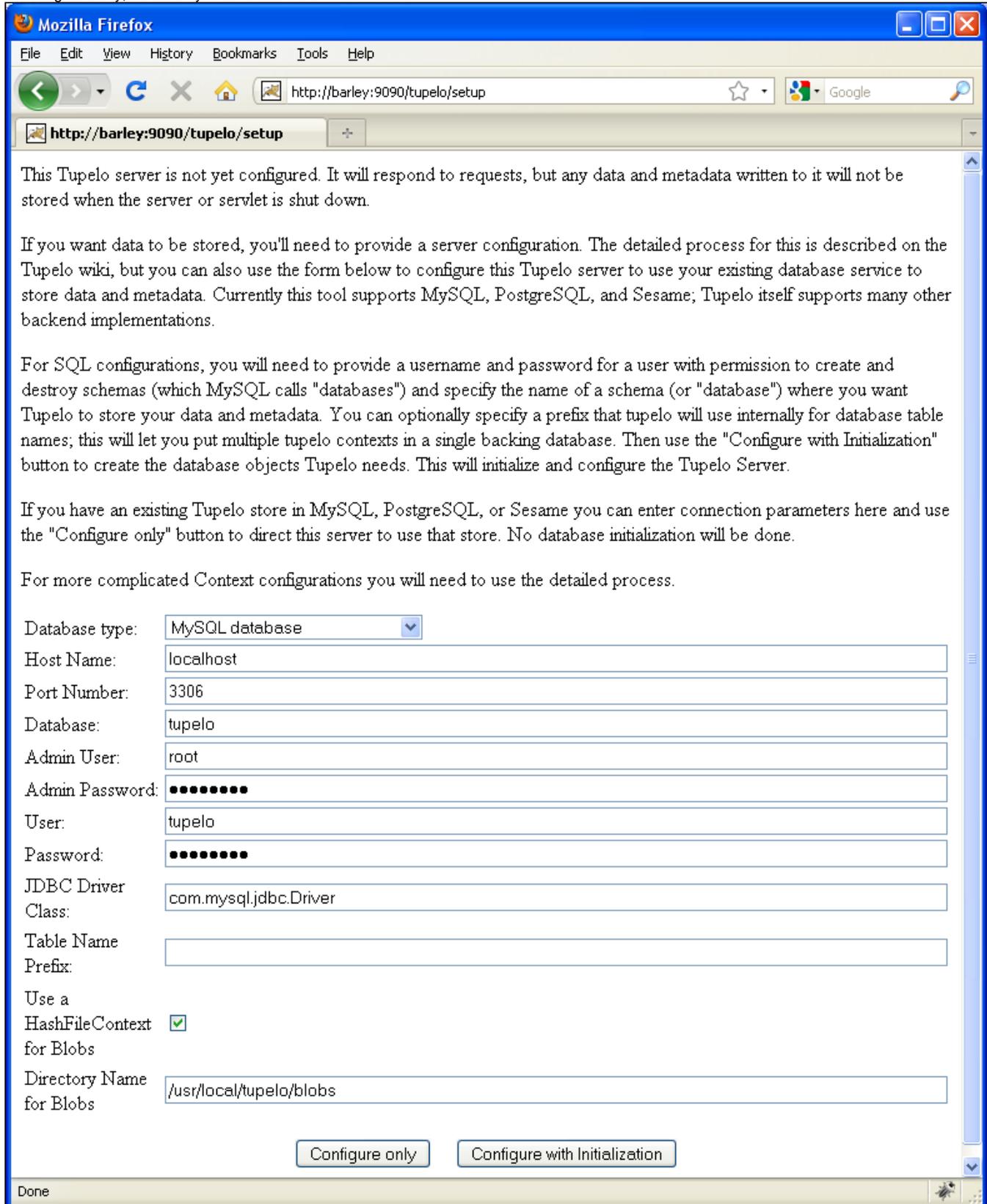
If you have an existing Tupelo store in MySQL, PostgreSQL, or Sesame you can enter connection parameters here and use the "Configure only" button to direct this server to use that store. No database initialization will be done.

For more complicated Context configurations you will need to use the detailed process.

Database type:

The browser status bar at the bottom shows "Done".

Use the *Database Type* menu field to select the type of database you will be using, then fill in the parameter fields as appropriate. The various fields are the same as those described in the corresponding Context implementation documentation. For example, if you choose to use a context backed by MySQL, see the [MySQL context](#) document. In addition, you may elect to have blobs stored in a HashFileContext (a directory on disk) rather than in the same location as for triples. Do this by checking the *Use HashFileContext for Blobs* checkbox, and enter the full file specification, including directory, for where you would like blobs to be stored:



This Tupelo server is not yet configured. It will respond to requests, but any data and metadata written to it will not be stored when the server or servlet is shut down.

If you want data to be stored, you'll need to provide a server configuration. The detailed process for this is described on the Tupelo wiki, but you can also use the form below to configure this Tupelo server to use your existing database service to store data and metadata. Currently this tool supports MySQL, PostgreSQL, and Sesame; Tupelo itself supports many other backend implementations.

For SQL configurations, you will need to provide a username and password for a user with permission to create and destroy schemas (which MySQL calls "databases") and specify the name of a schema (or "database") where you want Tupelo to store your data and metadata. You can optionally specify a prefix that tupelo will use internally for database table names; this will let you put multiple tupelo contexts in a single backing database. Then use the "Configure with Initialization" button to create the database objects Tupelo needs. This will initialize and configure the Tupelo Server.

If you have an existing Tupelo store in MySQL, PostgreSQL, or Sesame you can enter connection parameters here and use the "Configure only" button to direct this server to use that store. No database initialization will be done.

For more complicated Context configurations you will need to use the detailed process.

Database type:

Host Name:

Port Number:

Database:

Admin User:

Admin Password:

User:

Password:

JDBC Driver Class:

Table Name Prefix:

Use a HashFileContext for Blobs

Directory Name for Blobs:

Done

Note that clicking *Configure only* will simply store whatever values you have set where *Configure with Initialization* will attempt to initialize the Context as well as configure the server with that context. If the latter fails you will see an error message. Simply hit the back button on your browser and correct any issues.

If the configuration succeeds, you should see the Welcome screen.

Note that for Tupelo version 2.4.2 and earlier, the server must be restarted for the context configuration to take effect. This is not necessary starting with version 2.4.3.