Setting Up a Mmowgli Project in Eclipse

(The most up-to-date version of this document is in the root of the SourceForge Mmowgli file tree.)

Table of Contents

Introduction 1

Build/Development/Deployment Requirements 2

Import mmowgli application from SourceForge 2

To add the appropriate local server runtime to Eclipse: 6

Adjusting the Java Run-time environment 7

Compile the Theme and Widget-Sets 8

Choose an Encryption Password 10

Clean and Refresh 11

Define the Local Server 12

Run the Application 15

Log into Mmowgli 16

Development Notes 17

# Introduction

Mmowgli is a server-based web application written in Java. It is run within the context of a "servlet container", such as Tomcat or Glassfish. At the time of this writing, no servlet container other than Tomcat v7.X has been tested. To build the rich user interface seen in the browser, the Vaadin Web Framework is used, along with custom artwork, specified through both CSS files and Java code. Mmowgli cannot be run without a database, and although any could be used, Mysql is the only one which has been tested. Mmowgli has been developed using the Eclipse IDE, although periodic efforts are made to provide Netbeans support. There are Netbeans configuration files in the source tree, but beginning developers must almost certainly use Eclipse to get going.

Mmowgli can be developed and debugged on the developers local machine. This is the desirable configuration, since you can insert breakpoints, etc., from within Eclipse to assist development. When running locally like this, Eclipse starts a local version of Tomcat, so Tomcat must be available for the development machine and installed. You may also install MySql on your local development machine, which is another convenient configuration. Eclipse, Tomcat and Mysql running locally makes for a very manageable development environment.

Mmowgli exists in its public repository in a directory tree which, when replicated locally, should be recognizable by Eclipse as an Eclipse project.

This document describes how to proceed to producing Mmowgli's equivalent of "Hello World". The work required falls into the following general areas:

1. Install the build tools.
2. Download the Mmowgli source tree.
3. Create the local mmowgli database.
4. Create and configure the Eclipse project.
5. Adjust the source for first compile.
6. Launch the application (i.e., Tomcat with a deployed Mmowgli) within Eclipse.
7. Bring up the opening page from a local browser.
8. Buy drinks all around.

# Build/Development/Deployment Requirements

1. Eclipse J2EE Luna
2. Vaadin plugin
3. Tomcat 7\* local installation
4. Java Cryptography Extension (JCE) installed in JRE 8
5. MySql database
6. Ivy dependency manager (part of Eclipse IDE)
7. (Optional for cluster deployment) ApacheMQ Message Server
8. (Optional for cluster deployment) Apache GateKeeper

Eclipse and Tomcat can be freely downloaded from https://eclipse.org/ and http://tomcat.apache.org/ . The Vaadin plugin to Eclipse is described here: https://vaadin.com/eclipse . MySql here: https://www.mysql.com/ . The Java Cryptography Extension has recently been found here:

http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html.

The Ivy dependency manager handles retrieving the libraries used by Mmowgli. It is part of eclipse as the “Ivy” plugin. If it is not found, it can be installed through the “Apache Software Foundation” provider at the “Install New Software” menu item/wizard.

Important: Get the J2EE version of Eclipse. Also, Mmowgli has only been tested on the Community Edition of MySQL.

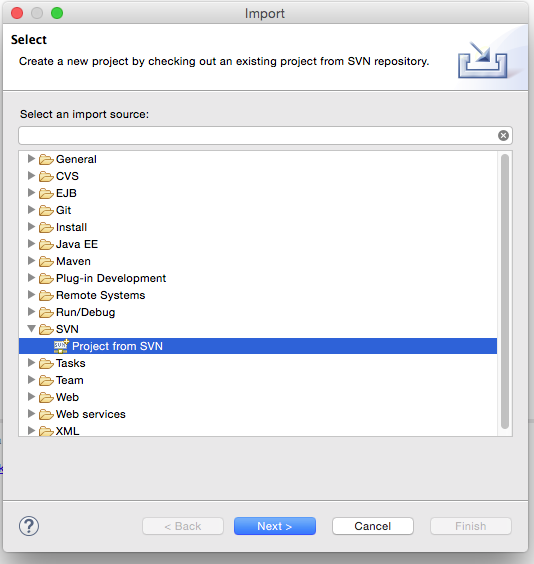
You do not download the Vaadin Framework itself -- only the Eclipse Vaadin plugin. Dependencies within the Mmowgli Eclipse project are handled through the Ivy dependency manager, and this includes the retrieval of the proper Vaadin Framework version. More on that below.

Install the build tools per their instructions.

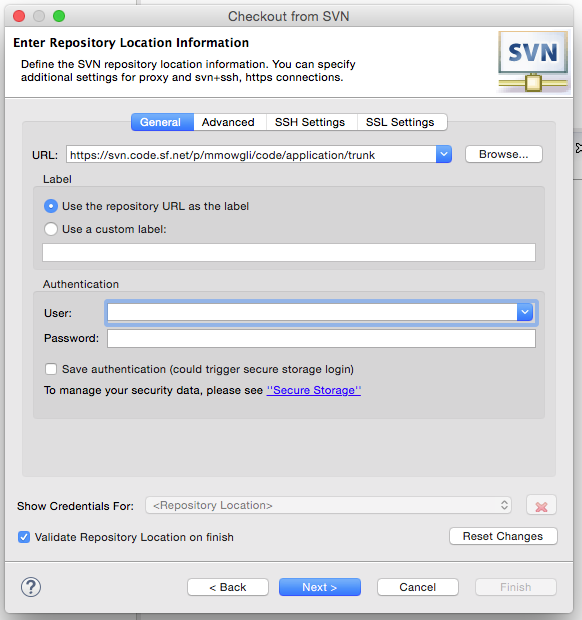
# Import mmowgli application from SourceForge

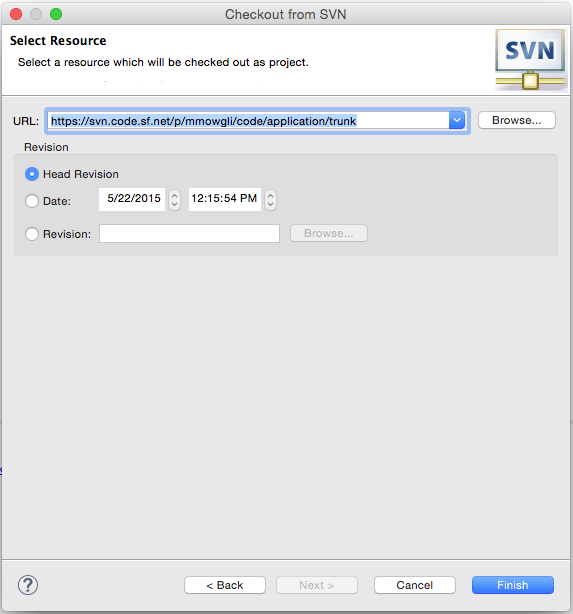
These instructions leave out the information about configuring the initial database. Before trying to launch Mmowgli, read the separate document in the repository on building the database. You will want to direct your attention to one of the files in the distribution, mmowgli\_bootstrap.sql. This is a complete Mmowgli database, consisting of a set of SQL statements which, when run by a mysql database manager, will construct a database on you local system.

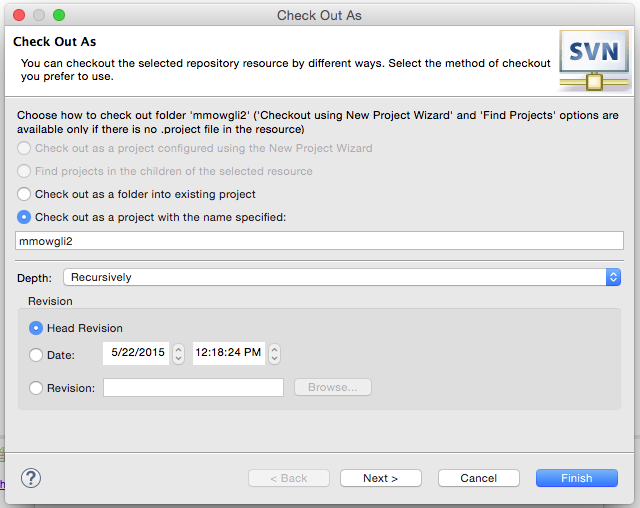
From the Eclipse File->Import… menu, choose to import a project from SVN.

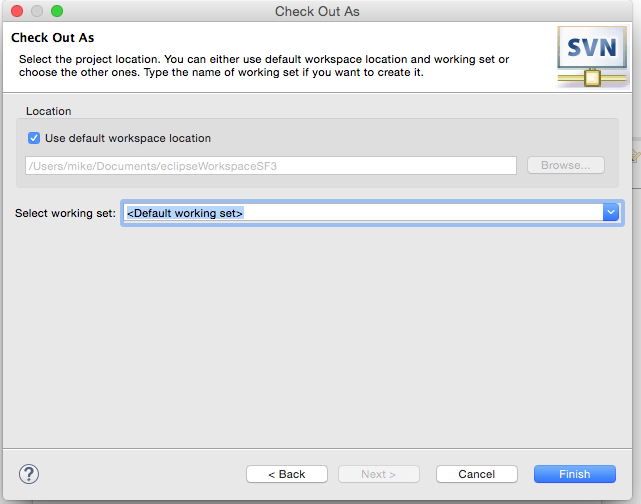


The repository URL is https://svn.code.sf.net/p/mmowgli/code/application/trunk





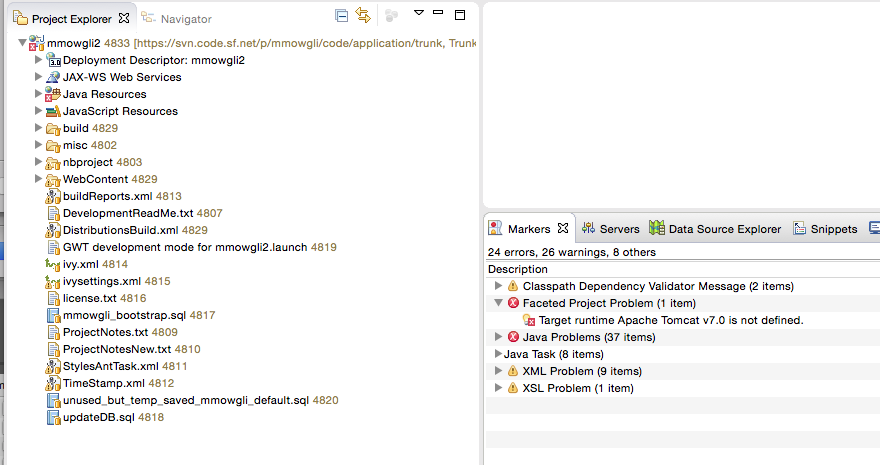




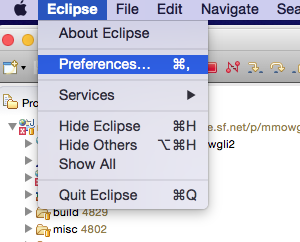
At this point, configure the WebContent/web.xml file with the proper information about your environment and database. Build the database and populate it with the mmowgli\_bootstrap.sql file found at the application project root. These steps are described in detail elsewhere.

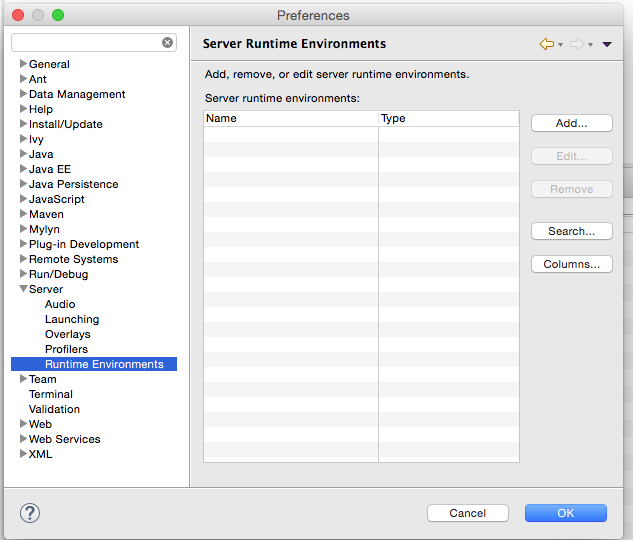
After successfully retrieving the application files, you have a project tree as below. Note the red badge meaning there are errors which must be resolved. Depending on the existing Eclipse configuration, these errors may involve:

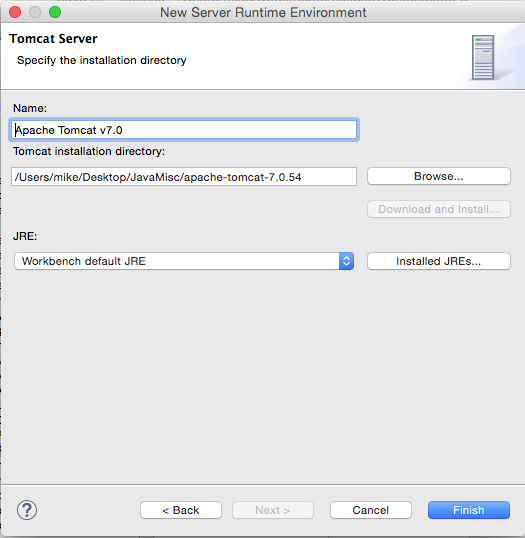
1. A missing Apache Tomcat server.
2. An alternately-named Java runtime environment



## To add the appropriate local server runtime to Eclipse:

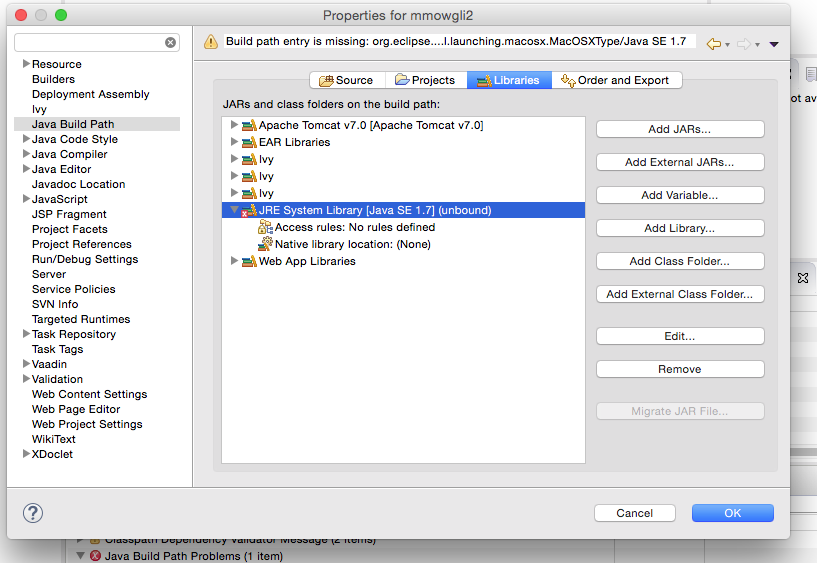


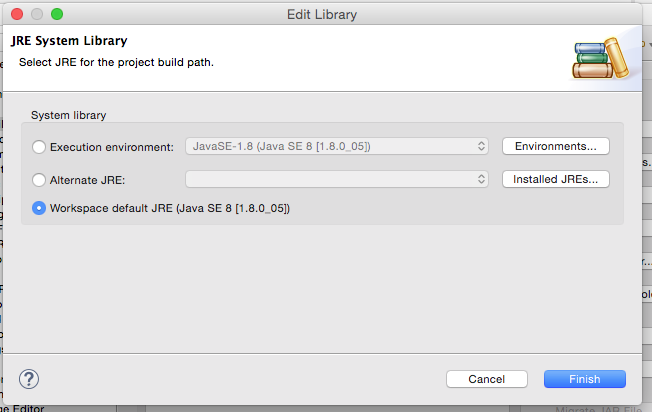




## Adjusting the Java Run-time environment

If you see an error in the project properties like the following, correct the java runtime environment by clicking the “Edit…” button.





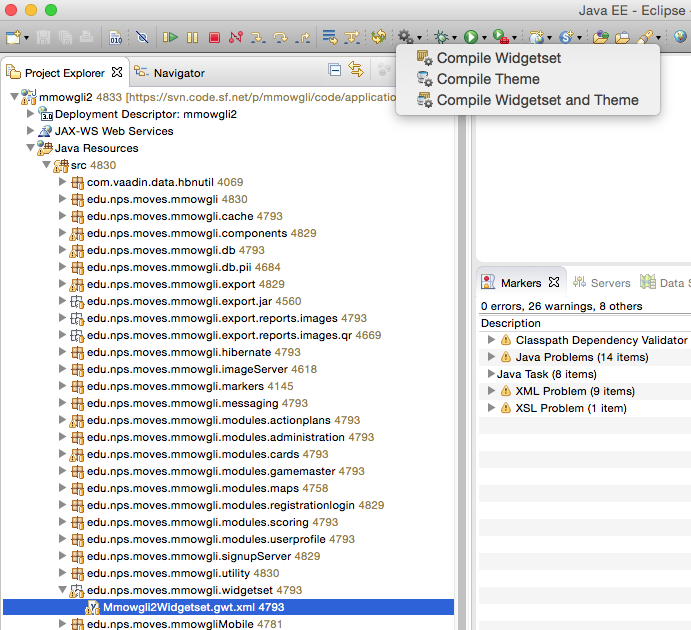
There should be no red badges in the Markers tab at this point.

# Compile the Theme and Widget-Sets

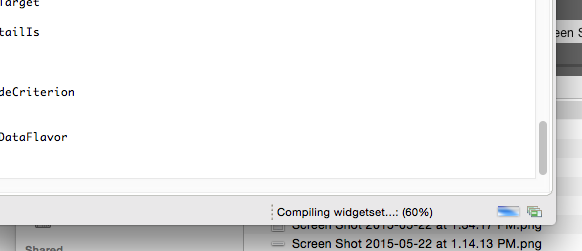
Developers are used to having to "compile" Java source files, converting them to executable (to a JVM) class files. A Vaadin application has 2 other compile steps. Vaadin apps make use of a Google Web Toolkit (GWT) “widgetset” behind the scenes. This is the interface between the Java Vaadin GUI library and Javascript/DOM code and data structures within the browser. It needs to be rebuilt under certain circumstances, such as at first build time (as here), upgrade of Vaadin version, and other more esoteric cases. Also, Vaadin css themes are written in the Sass stylesheet language and need to be compiled before use.

Mmowgli has 2 widget sets: web application and mobile application. They are compiled from the tool bar “gear” icon. Compile the theme and app widgetset with one menu hit of “Compile Widgetset and Theme” after selecting the gwt.xml file, then “Compile Widgetset” after selecting the

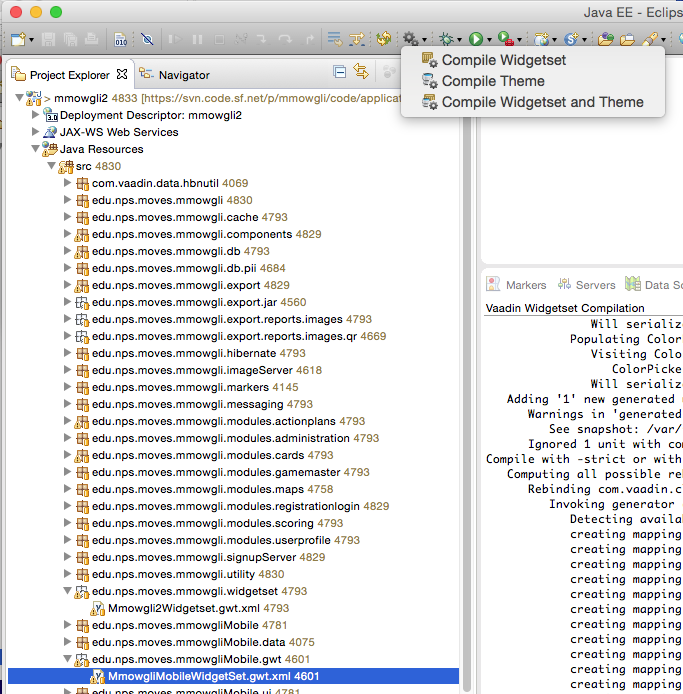
similar file in the mobile branch.



Watch the footer for completion of the task

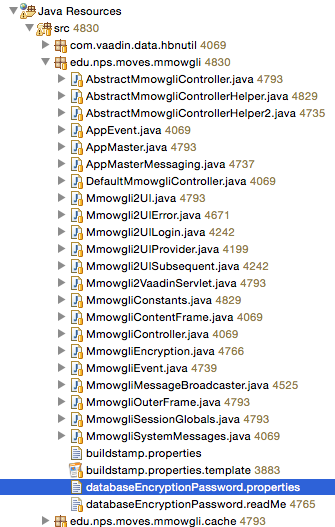


And the mobile widget set:



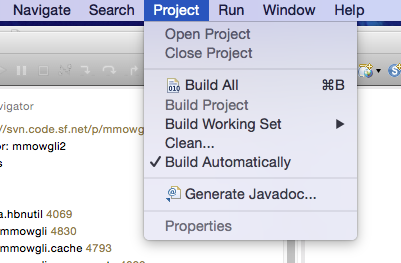
## Choose an Encryption Password

Mmowgli encrypts email addresses and real names before inserting in the database. The encryption requires a "key", which is compiled into the Mmowgli application. Read the file src/edu/nps/moves/mmowgli/databaseEncryptionPassword.readme and make a databaseEncryptionPassword.properties file right next to it, specifying a unique and private key.

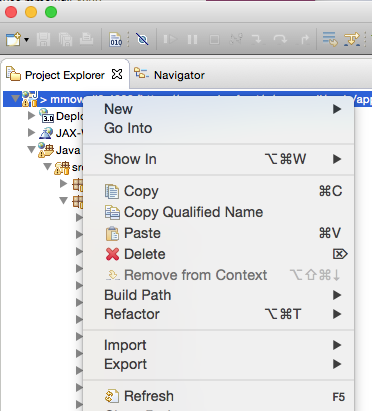


## Clean and Refresh

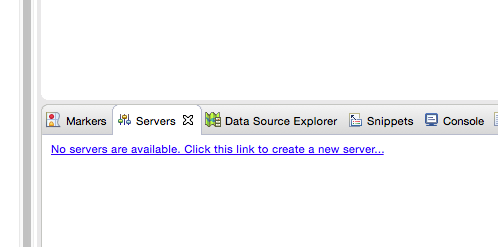
Clean (and rebuild) the application. This adds a build-stamp which is used to mark the time of each build. It will remain in place from this point on, so futher cleans are not required unless otherwise desired.

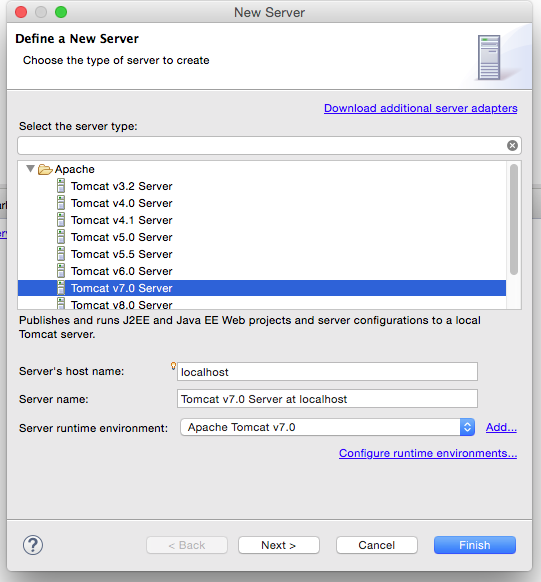


Refresh the project tree, so Eclipse knows about the new file.

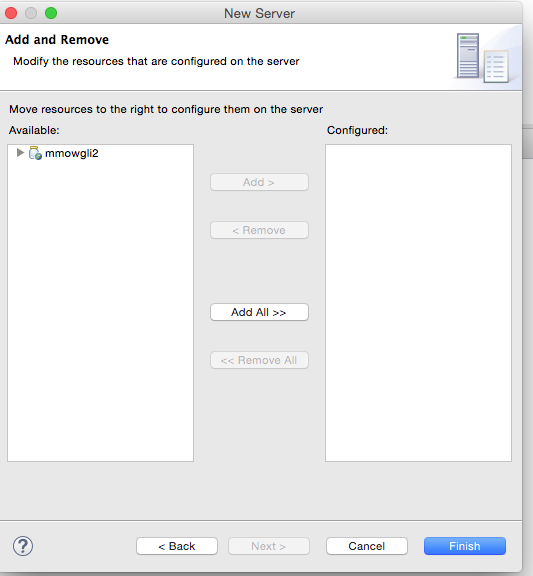


## Define the Local Server



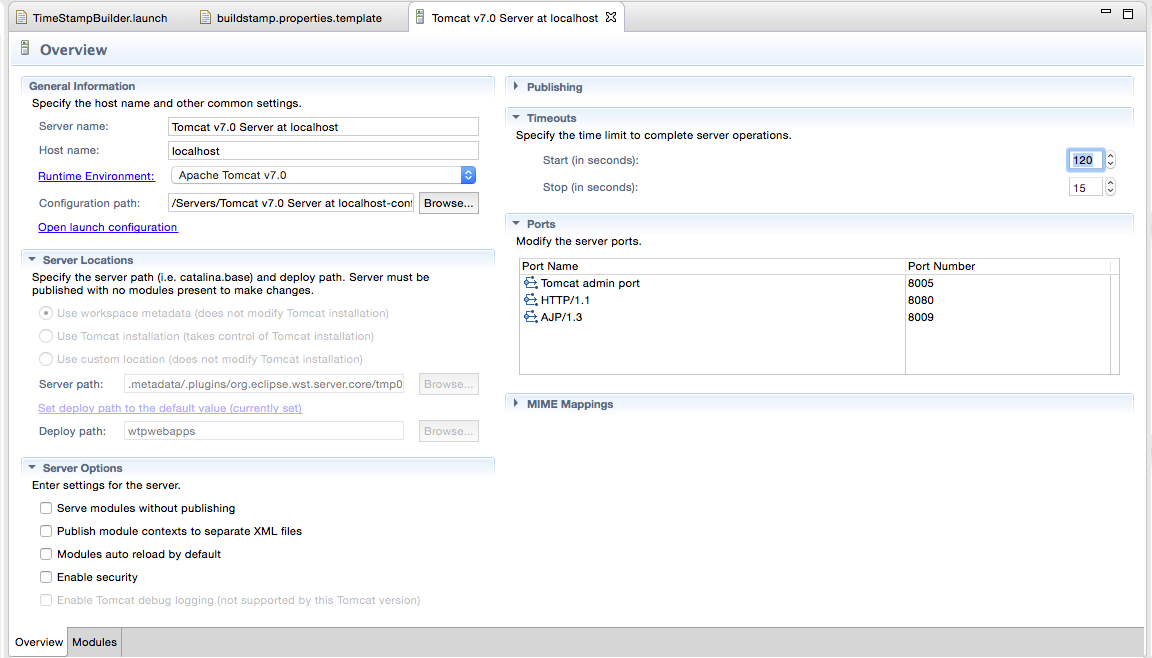


The mmowgli application needs to be installed on the server.



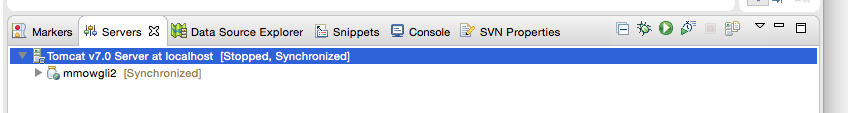
Two server setting may need to be adjusted, depending on the developer’s preference. Startup timeout may need to be bumped. The first Mmowgli developers use 120 seconds.

Server options to automatically reload and “serve without publishing” are turned off.



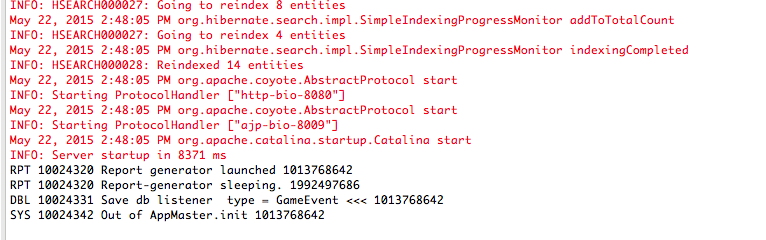
## Run the Application

Make sure all changes are published by clicking the “Publish to the Server” button at the top right.



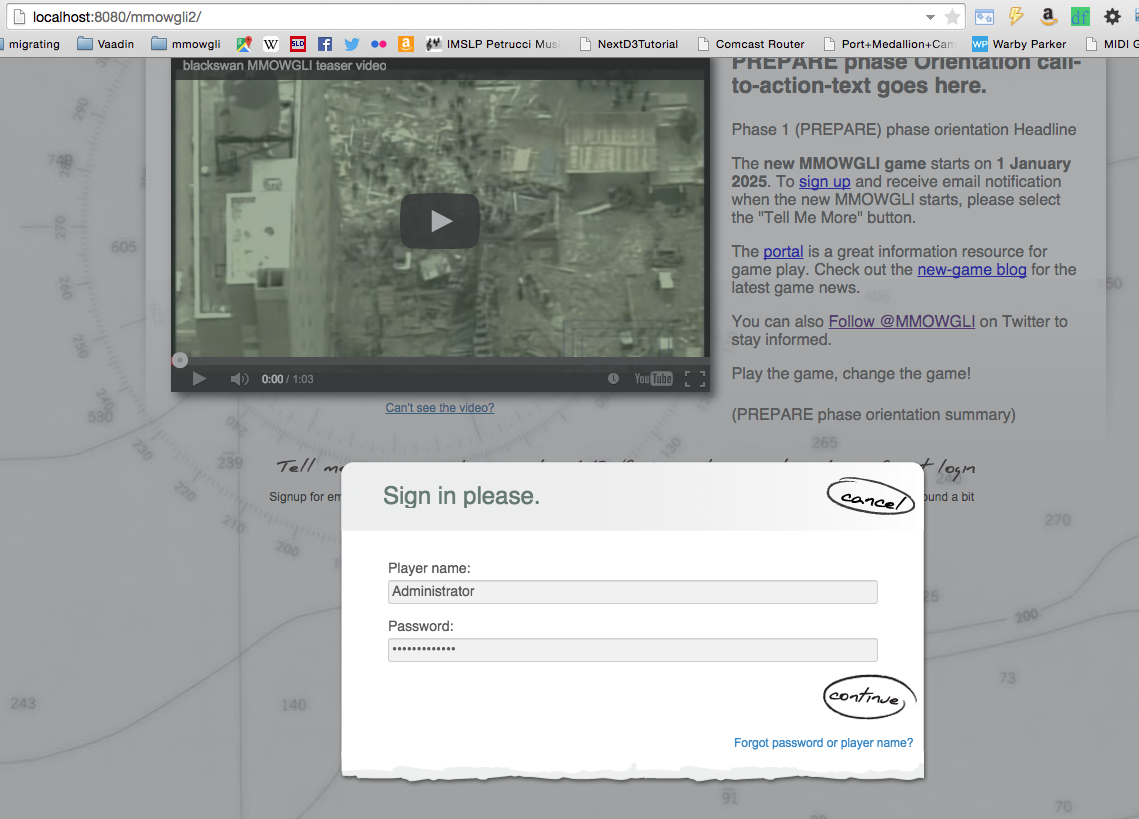
Launch the server, and thus the application, by clicking the green arrow.

Watch the Output tab for errors. If the game is successfully launched, you should see something like the following at the tail end of the console output.



## Log into Mmowgli

The default server port will be 8080 and the default application context is “mmowgli2”. Enter <http://localhost:8080/mmowgli2/> in your browser. A successfully install should show you something like the following.



Congratulations!

The first log on in a deployed game would be to Administrator/Administrator. There are 3 other initial users: GameMaster/GameMaster, GameBuilder/GameBuilder, and SeedCard/SeedCard. The Administrator would typically change the 4 passwords, logout, then create a regular account. Then logout of the regular account, and back in as Administrator/newpw. Go to the Game Administrator -> Player administration and give the just-made regular accounts the desired permissions.

# Development Notes

Mmowgli performs an indexing task on some database tables during startup. This happens while Tomcat is initializing the application. The initial, nearly empty database takes little or no time to index, but as the database accumulates Cards, Users and ActionPlans, the length of this task can exceed Tomcat's startup timeout. You will need to bump that startup timeout (right-click the Tomcat server in the Servers tab and find "Timeouts") as needed.

Mmowgli depends on many other Java libraries to run. These libraries are specified through the ivysettings.xml and ivy.xml files in the project root directory. "Resolving" the dependencies is a specific step in which the require libraries are retrieved from remote sites on the net. Whenever ivy.xml is changed (to include another dependency, perhaps), the resolve step takes place automatically. At first, however, it must be done manually. Right click on the top project node in the Project Explorer and choose ivy->resolve. This step takes several minutes.

It is very convenient to have Eclipse build your project automatically when you make changes. If this is not your preference, uncheck “Build Automatically” from the Project menu, and be sure to select “Build Project” after you make changes and before launching. This process is separate from putting, or “publishing” your changes to the Tomcat installation of mmowgli. That process is done through

the “publish to the server”.

Some notes on mmowgli art resources: From a programmer/app-packaging/deployment point of view, including all static application images in the created WAR file is most desireable: all files are contained in one war (jar/zip) file along with application class and css files. The image files are served to the client by the Tomcat application server. This method works well, but is not performance-optimal. Most images in the application will be countlessly reloaded by the user when switching pages, etc. It is best if this data were cached on the client end. A web browser can do this, but the web browser is never given a chance to do this if the images are kept "within" the app and loaded "internally" by Tomcat.

When an art resource is used in mmowgli code, it is a simple matter in vaadin to specify how the image is to be referenced. It is marked as a FileResource, ClassResource or ExternalResource. The first is loaded through the (server) filesystem. The latter two are the type discussed in the previous paragraph.

Mmowgli images are strored external to the application code. There is a configuration parameter, “gameImagesUrl”. It contains the root path on the application server where the statically-served media sits. This value is used by internal mmowgli code, but it must also be specified in WebContent/VAADIN/themes/mmowgli2/mmowgliScssParams.scss to be accessible through CSS.

For some images used in the game, and NOT specified by css, the database will have media entries, and those entries will include a path. The standard for specifying media image paths in through the source enum:

1. GAME\_IMAGES\_REPOSITORY, // file name only or relative path plus file name
2. USER\_UPLOADS\_REPOSITORY, // file name only or relative path plus file name
3. FILESYSTEM\_PATH, // full path on server machine
4. WEB\_FULL\_URL, // full url
5. DATABASE // Images table in db, served by ImageServer servlet

There is a small text file in the source tree called edu/nps/moves/mmowgli/buildstamp.properties . Its contents are read by game code to display the date and time of the build. It does not exist in the source repository, but is created and updated by a custom Eclipse build task described in TimeStamp.xml in the root of the source tree. This task gets run during a “Clean” and is set up through Project->Properties->Builders.