GameServer Hosting

Backend Installation Instructions

Table of Contents

Installing MySQL Database

MySQL database will need to be installed on a singular machine. Both the controller and node communicate to this database for various kinds of data. The database does not need to be on the same computer as the controller or node. However, it's recommended that the database is on a computer that can only be accessed on the LAN.

The schema can be anything, as long as the schema is accurate in the property files for the controller and nodes. A recommended schema to use is **gameserver**.

The username can be anything, as long as the username is accurate in the property files for the controller and nodes. A recommended username is **root**.

The password can be anything, as long as the password is accurate in the property files for the controller and nodes. A recommended password is **minecraft**.

Tables

The following is SQL for tables that will need to be created.

gameserver

```
CREATE TABLE `gameserver` (
  `name` varchar(100) NOT NULL,
  `nodeOwner` varchar(45) NOT NULL,
  `specificid` int(11) NOT NULL,
  `servertype` varchar(45) NOT NULL,
  `executableName` varchar(100) NOT NULL,
  PRIMARY KEY (`name`),
  UNIQUE KEY `name_UNIQUE` (`name`)
);
```

minecraftserver

```
CREATE TABLE `minecraftserver` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `maxheapsize` int(11) NOT NULL,
  `restarts` tinyint(1) NOT NULL,
  `arguments` text NOT NULL,
  PRIMARY KEY (`id`)
);
```

node

```
CREATE TABLE `node` (
  `name` varchar(100) NOT NULL,
  `ram` int(11) NOT NULL,
  PRIMARY KEY (`name`),
  UNIQUE KEY `name_UNIQUE` (`name`)
);
```

triggers

```
CREATE TABLE `triggers` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` text NOT NULL,
  `command` text NOT NULL,
  `value` text NOT NULL,
  `serverowner` text NOT NULL,
  `extra` text,
  PRIMARY KEY (`id`)
);
```

Installing Tomcat

The Tomcat version used in development is Tomcat 7. It is recommended to use this version, as I know that it works. The Tomcat server will need to be public to the LAN. It is NOT RECOMMENDED to allow traffic from the Internet to access the Tomcat server. The GameServer Hosting application currently does not require any user logon and all traffic is not encrypted.

Tomcat is needed for both the *controller* and *node* application. Computers that are intended to run servers needs the *node* application. The *node* application is what directly manages server processes and is what the controller contacts to manipulate game servers. The *controller* application is what the browser will contact for both the frontend and will essentially pass information to the corresponding *node* if needed. There are some cases where direct communication between the browser and the *node* application happens. These situations happen during Web Socket communication.

The *controller* can be on the same computer as a *node*. If this is the situation, then both applications will be deployed on the same Tomcat server. Property files for both applications will have addresses be the same.

A library that is needed to connect to the database is required to be placed in the Tomcat's lib directory. This is the mysql-connector-java-8.0.16.jar.

Deploying the Controller

The *controller* is responsible for delivering the frontend and managing multiple *node*s. A WAR file associated with the *controller* can be deployed to the Tomcat server via the Manager screen.

A properties file is required for the *controller*. The path to the properties file will need to be an environmental variable called: `GAME_SERVER_CONTROLLER_PROPERTIES`.

The properties file needs to have at least these properties:

- node_names (Comma separated list of node names)
- node_addresses (Comma separated list of node addresses)
- node_ports (Comma separated list of node ports)
- node_extension (The Tomcat application name, generally "GameServerNode")
- database_url (With MySQL, this is "jdbc:mysql://" + database address + port + schema)
- database_username
- database_password

An example of the *controller* properties file can be found below:

```
node_names=Node1,Node2,Node3

node_addresses=192.168.1.1,192.168.1.2,192.168.1.3

node_ports=8080,8080,8090

database_url=jdbc:mysql://192.168.1.1:3306/gameserver

database_username=root
```

```
database_password=minecraft
```

Deploying the Node

The *node* is responsible for managing the processes and file system for deployed servers. The *node* is also responsible for sending server output data, server running status, and system resource data directly to the client's browser via Web Sockets. Only ONE *node* should be deployed on a computer at a time. A WAR file associated with the *node* can be deployed to the Tomcat server via the Manager screen.

A properties file is required for the *node*. The path to the properties file will need to be an environmental variable called: `GAME_SERVER_NODE_PROPERTIES`.

The properties file needs to have at least these properties:

- name
- deploy_folder (absolute path to folder where servers will be placed)
- database_url (With MySQL, this is "jdbc:mysql://" + database address + port + schema)
- database_username
- database_password
- max_ram (Optional. Max RAM in MB to use for appropriate servers. Servers collectively won't be allowed to have their max ram settings go over this limit.)

An example of the *node* properties file can be found below:

```
node=Node1

deploy_folder=E:\\GameServerNode\\deploy

database_url=jdbc:mysql://127.0.0.1:3306/gameserver

database_username=root

database_password=minecraft
```

Modded Minecraft servers require Java 8 to run properly. Any newer versions will not work. Because of this, all Minecraft servers are ran using Java 8. Computers with the *node* application will need to install Java 8. To not have Java 8 conflict with any version that is needed to run the *node* application, an environmental variable is used to point to the Java 8 VM executable and is executed just for Minecraft servers. This environmental variable needs to be: `JAVA8`.

The first time the *node* application is ran, it will register itself into the database. Because of this, make sure the *node* application is running before running the *controller*.