

Botz: Internal Bot Library for Openfire

Overview

A common way of creating a new XMPP service is to develop a plugin that will serve the service as a sub domain. That said, if Openfire's domain is **example.com** programmers would develop the new service as an internal or external component and deploy it as **myservice.example.com**.

?

Botz library adds the already rich and extensible Openfire with the ability to create internal user bots. With Botz library, programmers may choose to develop a user bot to run as a service bearing **myservice@example.com** as its JID. To Openfire, the user bot is just like other (human) users.

?

Botz library is strictly internal for Openfire. The notion of a user connection doesn't involve any TCP/IP or socket; hence virtual. There isn't even a C2S implementation.

?

Botz Classes

Botz library contains **BotzConnection** class that allows a user bot to login as a registered or anonymous user. The class optionally automates the creation and registration of the user bot if it has not existed in the database. To make the user bot useful, programmers would implement **BotzPacketReceiver** interface to respond to received packets. **BotzPacketReceiver.processIncomingPacket(Packet)** will be called for every packet received by the user bot. To send packets to other XMPP entities, programmers in turn call **BotzConnection.sendPacket(Packet)**.

?

Botz classes may be used in situations where an internal user bot is needed. Botz most likely proves itself useful in the development of Openfire extensions through plugins.

?

Key Features

- ??? Login anonymously
- ??? Login as an existing Openfire user
- ??? Optionally create a new user as a registered user (bot) if it does not exist. The newly created user account will be stored in the database. Because user creation is done using SQL statements internal to Openfire, this should work for all Openfire-supported databases.
- ??? The above features hide programmers from handling the connection establishment and allow programmers to focus on packet exchanges.
- ??? Change **BotPacketReceiver** on the fly, thus switch behaviors and create multiple personalities of a bot.

?

Using Botz in A Plugin

The following is the code snippet that shows a way to use Botz classes in a plugin. The sample plugin is a parrot bot service that simply echoes `<message/>` packets back to the sender.

?

```
import org.jivesoftware.openfire.botz.BotzConnection;
import org.jivesoftware.openfire.botz.BotzPacketReceiver;
?
public class ParrotBot implements Plugin {
    ??? @Override
    ??? public void destroyPlugin() { }

    ??? @Override
    ??? public void initializePlugin(PluginManager manager, File pluginDirectory) {
    ???????? BotzPacketReceiver packetReceiver = new BotzPacketReceiver() {
    ????????????? BotzConnection bot;

    ?????????????? public void initialize(BotzConnection bot) {
    ?????????????????? & nbsp;? this.bot = bot;
    ?????????????? }

    ?????????????? public void processIncoming(Packet packet) {
    ?????????????????? & nbsp;? if (packet instanceof Message) {
    ?????????????????? & nbsp;? ?????? // Echo back to sender
    ?????????????????? & nbsp;? ?????? packet.setTo(packet.getFrom());
    ?????????????????? & nbsp;? ?????? bot.sendPacket(packet);
    ?????????????????? & nbsp;? }
    ?????????????? }
```

```
????????????? public void processIncomingRaw(String rawText) { };
????????????? public void terminate() { };
????????? };

????????? BotzConnection bot = new BotzConnection(packetReceiver);
????????? try {
????????????? // Create user and login
????????????? bot.login("parrot");
????????????? Presence presence = new Presence();
????????????? presence.setStatus("Online");
????????????? bot.sendPacket(presence);
????????? } catch (Exception e) {
????????????? e.printStackTrace();
????????? }
??? }
}
```

?

?

?

License

This software is published under the terms of the GNU General Public License (GPL).

?

Downloads

Botz Library

Openfire Version	Botz Library	Source Code	Version	License
3.4.*	#	#	1.0.0	GPL
3.3.2-3.3.*	#	#	1.0.0	GPL

?

Sample Bot Plugin

Plugin Name	Openfire Version	Plugin JAR	Source Code	Version	License
ParrotBot	3.4.*	#	#	1.0.0	GPL
ParrotBot	3.3.2-3.3.*	#	#	1.0.0	GPL

?

Installation

The Botz library is not a plugin in itself, and does not contain any plugin-related class. It is meant for use in an application development. To use the library in an Openfire plugin development, copy **botz-openfire-version.jar** file into the **lib** directory of the plugin directory structure. If there are more than one plugins that will use Botz library, the JAR file can be copied to a global class path like **openfire/build/lib/dist** or **{\$OPENFIRE_HOME}/lib**.

?

To install the sample ParrotBot plugin, rename **parrotbot-openfire-version.jar** to **parrotbot.jar** and copy it into **{\$OPENFIRE_HOME}/plugins** directory. To see the ParrotBot in action, send messages to it using your favorite Jabber client.

?

The parrot user bot appears as a real user in Openfire admin console:

?



Figure 1: Parrot Bot As A Registered User

?

When logged in, the bot has a client session:

?



Figure 2: Parrot Bot Logged In

?

?

User-Contributed Bot Plugins

Plugin Name	Plugin JAR	Source Code	Version	Author	License

?

Bot Plugins Wish List

Plugin Name	Description
CAPService	An implementation of XEP-0127:Common Alerting Protocol(CAP) Over XMPP
NameService	MUC Name Service
RoomService	MUC Room Service

?