

Tutorial 3: Commands

Contributed by D1st0rt
Friday, 19 November 2004
Last Updated Wednesday, 02 March 2005

Uh oh, we have no control over our bot. Not to fear, that's what commands are for. Commands mainly utilize the CommandInterpreter, though technically you can do it without one. In order to make sure the wrong people don't use certain commands, we will use an OperatorList, which gives access to staff at the same level they are in your zone's moderate, smod and sysop.txt files.

```
//Make the package name the same as
the bot's name so you can spawn it

package twcore.bots.mybot;

//Import all of the TWCore classes so you can use them
import twcore.core.*;

public class mybot extends SubspaceBot
{

    //Requests all of the events from the core
    private EventRequester events;

    //Handles all commands sent to bot
    by players

    private CommandInterpreter cmds;

    //Stores Staff Access Levels
    private OperatorList oplist;

    //Creates a new mybot
    public mybot(BotAction botAction)
    {

        //This instantiates your BotAction
        super(botAction);

        //Instantiate your EventRequester
        events = m_botAction.getEventRequester();
```

```
//Request PlayerEntered events

events.request(EventRequester.PLAYER_ENTERED);

//Request
chat message events

events.request(EventRequester.MESSAGE);

//Instantiate
your CommandInterpreter

cmds = new CommandInterpreter(m_botAction);

//Instantiate
your Operator List

oplist = m_botAction.getOperatorList();

//Set up your
interpreter

addCommands();

}

//What to do when the bot logs on
public void handleEvent(LoggedOn event)
{
    //Get the data from mybot.cfg
    BotSettings config = m_botAction.getBotSettings();

    //Get the initial arena from config and
enter it
    String initial = config.getString("InitialArena");
    m_botAction.joinArena(initial);

    //NOTE: m_botAction is inherited from SubspaceBot
}

//What to do when a player enters the arena
public void handleEvent(PlayerEntered event)
{
    //Get the name of player that just entered
```

```
String name = event.getPlayerName();

//Greet them

m_botAction.sendPrivateMessage(name,"Welcome!");

}

//What to do when somebody says something

public void handleEvent(Message
event)

{

//Pass it to the
interpreter

cmds.handleEvent(event);

}

//Set up commands

public void addCommands()

{

//Allowed message
types for commands

int ok = Message.PRIVATE_MESSAGE |
Message.PUBLIC_MESSAGE;

//Add any commands
as you see fit

cmds.registerCommand("!help",ok,this,"help");

cmds.registerCommand("!die",ok,this,"die");

cmds.registerCommand("!go",ok,this,"go");

}

//Got command: !help

public void help(String name, String msg)

{

//Help Message
```

```
String[] help =
{"!help - this message",
"!die - kills bot",
"!go (arena) - sends bot to arena"};

//Send player the
help message

m_botAction.privateMessageSpam(name,help);
}

//Got command: !die

public void die(String name, String msg)
{
//Make sure player
has clearance

if(oplist.isER(name))

//Destroy bot, allows another to be spawned

m_botAction.die();
}

//Got command: !go

public void go(String name, String msg)
{
//Make sure player
has clearance

if(oplist.isER(name))

//Join arena specified by player

m_botAction.changeArena(msg);
}
}
```

The first parameter in registerCommand is what the player types.

The second is the allowed message types, the third is the object that the method will be called in, and the last is the name of the method to be called. oplist.isER() will return true if the player is at least an <ER>, the staff hierarchy is as follows:

- Owner (is set in owners.cfg in the corecfg folder)

- Sysop

- Smod
- High Mod (set in highmod.cfg)

- Mod

- ER
- Outsider (set in outsider.cfg, used for people to have bot access without being staff)

- ZH