

## Tutorial 5: TWBotExtension Basics

Contributed by D1st0rt  
Saturday, 20 November 2004  
Last Updated Thursday, 25 May 2006

Standalone bots are nice, but some times its more effective to write a module that can be run from a TWBot. TWBots can run multiple modules at once, and you only have to have one bot running to use them, which handles a lot of the basic bot functions for you. The structure of a module is slightly different because it extends TWBotExtension instead of SubspaceBot. Because the BotAction class is not passed in the constructor, if you want to use it, you have to get it statically using BotAction.getBotAction(). Another thing to note is the two abstract methods that TWBotExtension defines: getHelpMessages() and cancel(). The first is what is shown when you say !help <module name>, and cancel is called right before you unload the module. This is used for cancelling any TimerTasks you have running, or anything else that does stuff tied to the bot instead of the module. Here is how I would port mybot to a TWBotExtension.

```
package twcore.bots.twbot;

// Notice how this time we put it in the twbot package instead of mybot

import twcore.core.*;

public class twbotmybot extends TWBotExtension

{
    // Class names for modules must start with "twbot"

    public twbotmybot()
    {

        // nothing to do here since we don't need to use BotAction

        // since the BotAction is not passed in the constructor,

        // we have to get it this way if we need it:

        // BotAction.getBotAction();

    }
}
```

---

```
// What to do when a player enters the arena (copy & pasted from mybot.java)
```

```
public void handleEvent(PlayerEntered event)

{

    // Get the name of player that just entered

    String name = event.getPlayerName();

    // Greet them

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

}
```

```
// I don't like to use CommandInterpreters in modules
```

```
// Here's the alternate way to do commands
```

```
public void handleEvent(Message event)

{

    // We only want private messages from ER's or above

    if(event.getMessageType() == Message.PRIVATE_MESSAGE)

    {

        String name = m_botAction.getPlayerName(event.getPlayerID());

        String message = event.getMessage();

        if(m_opList.isER(name) && message.startsWith("!"))

            // Passed the security check, so we process the command

            delegateCmd(name,message);

    }

}
```

```
}

// This takes the validated message, determines the command,

// and passes it to the proper method

public void delegateCmd(String name, String message)

{
    // This chops off the ! and makes it case-insensitive

    String cmd = message.substring(1).toLowerCase();

    // Pass to proper command method

    if(cmd.equals("welcome"))

        c_welcome(name,message);

    // Other commands would go here as an else-if chain

}

// This one is self-explanatory

public void c_welcome(String name, String message)

{

    m_botAction.sendArenaMessage(name + " welcomes you all!");

}
```

// These are the abstract methods, it won't compile without them:

```
public String[] getHelpMessages()

{

    return new String[] {"[mybot]",

                        "!welcome - welcomes people"};

}

public void cancel()

{

    // Nothing here we need to do

    // If we had any TimerTasks running, this is where they would be stopped

}

}
```

As you can see, this accomplishes more with less coding, because the basic tasks are done for you. Notice how you don't need to request any events, the TWBot does it for you. Notice also how I didn't use a CommandInterpreter in this example. I don't like using CommandInterpreters in modules because they are created by each bot or module that uses them. While it's no problem to have a single bot create one, it means that each module you load that uses one will create a separate instance just for that module, which is wasteful and inefficient. The build command for extensions is "bld twbot", and the way you would use this bot is to spawn a TWBot, !lock it, and then !load mybot.