Twister plugin guide

Twister framework is designed to load user created plugins and display them in the main interface. In order for the framework to communicate with the plugin, the plugin must implement the TwisterPluginInterface interface found under com.twister.plugin.twisterinterface in the Twister.jar library. When a new plugin is downloaded from the server it is automatically initialized by JVM, so, the initialization can't be controlled by the framework, this is the reason why the plugin should have an empty constructor. Instead, the initialization should be made in the init method.

- init(ArrayList<Item> suite, ArrayList<Item> suitetest, Hashtable<String, String> variables)

Init method accepts 3 parameters, references to variables found in Twister framework.

- ArrayList<Item> suite : reference to an ArrayList of Items defined by the user, also found under the Suites tab
- ArrayList<Item> suitetest : reference to an ArrayList of Items generated by the user, also found under the Monitoring tab
- Hashtable<String, String> variables: a Hashtable of String that points to diferent paths defined by the user.

The keys of the Hashtable are:

- user : framework user
- password : user password
- temp : temporary folder created by the framework
- Inifile : configuration file
- remoteuserhome : user home folder found on server
- remotconfigdir : config directory found on server
- localplugindir : local directory to store plugins
- httpserverport : server port used by EP to connect to a centralengine
- centralengineport : centralengine port
- resourceallocatorport : resource allocator port
- remotedatabaseparth : directory that contains database config file
- remotedatabasefile : database config file
- remoteemailpath : path to email configuration directory
- remoteemailfile : email configuration file
- configdir : local config directory
- usersdir : local directory to store suites configuration
- masterxmldir : local directory to store generated suite
- testsuitepath: remote directory that contains tc's for suite definition
- logspath: directory to store logs
- masterxmlremotedir : remote directory to store generated suite
- remotehwconfdir : remote directory to store hardware config file
- remoteepdir : remote directory to store EP file
- remoteusersdir : remote directory to store suites configuration

The framework calls terminate() method when the user wants to discard the plugin .The plugin should override the method terminate() to handle the release of all the resources. If some resources are not managed corectly, ex. threads will continue to execute after the call terminate(), these resources will continue to run in background.

The plugin should offer the framework a Component with the content that will be displayed on getContent() method. The framework will take that Component and put it under a new tab with the name provided by the plugin on the method getName(). The plugin should initialize the Component in the init method and should hold a reference to it so that it will serve the framework the same component every time getComponent() method is called.

The getFileName() method should return the name of the file that contains the plugin.

The plugin should be packed in a jar archive and uploaded in the Plugins folder found on server. The jar archive must contain a configuration file found in META-INF/services named com.twister.plugin.twisterinterface.TwisterPluginInterface.
This file contains a single line listing the concrete class name of the implementation, the plugin class name. (More on http://java.sun.com/developer/technicalArticles/javase/extensible/index.html)

For better understanding a brief tutorial for creating a small plugin will be presented.

We will create a plugin to display the username found in the Hashmap and put it in a Jlabel.
First include the Twister.jar library to your favourite ide. In this library we will find, besides the interface to implement, a base plugin class that eases the initialization process.

Create the class UserName that looks like this:

```
import com.twister.Item;
import com.twister.plugin.baseplugin.BasePlugin;
import com.twister.plugin.twisterinterface.TwisterPluginInterface;

public class UserName extends BasePlugin implements TwisterPluginInterface {

}
```

The BasePlugin holds the Hashtap in a variable named variables, the suite array in a variable named suite and the generated suite in a variable named suitetest. So, in order to get the username provided by the framework we will use variables Hastable, and put it in a Jlabel initialized in init.

```
import com.twister.Item;
import com.twister.plugin.baseplugin.BasePlugin;
import com.twister.plugin.twisterinterface.TwisterPluginInterface;

public class UserName extends BasePlugin implements TwisterPluginInterface {
private JPanel p;
private JLabel label;

      public void init(ArrayList<Item> suite, ArrayList<Item> suitetest,
                        Hashtable<String, String> variables) {
            super.init(suite, suitetest, variables);
            p = new Jpanel();
            label = new JLabel(variables.get("user"));
      }
}
```

Observe how we are holding a reference to the JPanel p because this is the component that we will serve to the framework.

```
import java.awt.Component;
import java.util.ArrayList;
import java.util.Hashtable;
import javax.swing.JLabel;
import javax.swing.JPanel;
import com.twister.Item;
import com.twister.plugin.baseplugin.BasePlugin;
import com.twister.plugin.twisterinterface.TwisterPluginInterface;

public class UserName extends BasePlugin implements TwisterPluginInterface {
	private JPanel p;
	private JLabel label;

	@Override
	public void init(ArrayList<Item> suite, ArrayList<Item> suitetest,
			Hashtable<String, String> variables) {
		super.init(suite, suitetest, variables);
		p = new JPanel();
		label = new JLabel(variables.get("user"));
	}

	@Override
	public Component getContent() {
		return p;
	}
}
```

Let us provide a description, filename that contains this plugin, and the title of the plugin tab.

```
@Override
public String getDescription() {
	String description = "Plugin to display user name";
	return description;
}

@Override
public String getFileName() {
	String filename = "UserName.jar";
	return filename;
}

@Override
public String getName() {
	String name = "UserName";
	return name;
}
```

Also for consistency we should release the references on the terminate() method. In case we would have Threads running, we should terminate them here.

```
@Override
public void terminate() {
	super.terminate();
	p = null;
	label = null;
}
```

The final class should look like this:

```
import java.awt.Component;
import java.util.ArrayList;
import java.util.Hashtable;
import javax.swing.JLabel;
import javax.swing.JPanel;
import com.twister.Item;
import com.twister.plugin.baseplugin.BasePlugin;
import com.twister.plugin.twisterinterface.TwisterPluginInterface;

public class UserName extends BasePlugin implements TwisterPluginInterface {
	private static final long serialVersionUID = 1L;
	private JPanel p;
	private JLabel label;

	@Override
	public void init(ArrayList<Item> suite, ArrayList<Item> suitetest,
			Hashtable<String, String> variables) {
		super.init(suite, suitetest, variables);
		p = new JPanel();
		label = new JLabel(variables.get("user"));
		p.add(label);
	}

	@Override
	public Component getContent() {
		return p;
	}

	@Override
	public String getDescription() {
		String description = "Plugin to display user name";
		return description;
	}

	 @Override
	public String getFileName() {
		String filename = "UserName.jar";
		return filename;
	}
```

```
	@Override
	public void terminate() {
		super.terminate();
		p = null;
		label = null;
	}
}
```

We will pack this in an archive named UserName.jar. This archive must contain the folowing directory META-INF/services. In the services directory we must put a file named com.twister.plugin.twisterinterface.TwisterPluginInterface and, in this file we put the name of our plugin class UserName. After we upload the UserName.jar file to the server Plugins directory, we should be able to download the plugin from Twister framework and activate it in the Plugins section.