



OW2 Shelbie: Innovative Shell

guillaume.sauthier@bull.net (project leader)
benoit.pelletier@bull.net

www.ow2.org

March 2009



Plan

Once upon a time ...

Shelbie in a nutshell

Existing shells comparisons

Features

EoU, scriptable and remotable

Modular, flexible and dynamic

Architecture & APIs

Technological choices

OSGi™, Groovy, Jline, ...

Some code !

Project's relationships

What's next ?

Once upon a time ... [1]

JOnAS shell

- No evolutions since ...
- Not user friendly
- Limited command set

Heavy JOnAS 5 refactoring

- Modular application server
- Dynamism support

Surf the wave and provides a new shell's generation



```

Terminal
-----
sauthieg@sauthierg:~$ jonas admin
-----
- JOnAS 5 - OSGi on Unix platform -
- JOnAS 5: Java(TM) Open Application Server -
- http://jonas.ow2.org -
- Contact: jonas-team@ow2.org -
-----
You are administering server named jonas
  service:jmx:rmi:///jndi/rmi://localhost:1099/jrmpconnector_jonas
Admin (jonas) > listmodules
ListModules:
//localhost/jonasAdmin
//localhost/juddi
mejb
//localhost/

Admin (jonas) > help
addfile      adds beans/servlets/j2ee app/rars based upon the file extension
custom       dump jonas customization
env          JOnAS properties used by the server
gc           run the garbage collector
help         help
jndinames    lists registered JNDI names
listbeans    lists beans
listmodules  lists modules
listapps     lists applications
name         to identify a current JOnAS server
quit         quit JonasAdmin
removefile   remove beans/servlets/j2ee app/rars (based upon the file extension)
start        stop target servers
stop         stop target servers
halt         halt target servers
sync         synchronize all entities
passivate    passivate all entities
trace        get/set monolog topics
ttimeout     set default transaction timeout
target       set list of servers where command must be applied
Admin (jonas) >
  
```

Once upon a time ... [2]

Existing shells comparison

	Modulaire	Dynamique	Evolutif	Portable	Accès distant	Script	Ergonomie
<i>Bash</i>		++	++	--	++	++	++
<i>Gshell</i>	++	-	+	++	+	+	-
<i>Felix Shell</i>	+	+	+	++	+	--	--
<i>ServiceMix Shell (Gshell based)</i>	++	++	+	++	+	--	+
<i>Groovy Shell</i>	+	-	+	++	--	++	++
<i>JOnAS Shell</i>	-	--	-	++	--	--	--

Shelbie Features

User friendly

Completion

History

Detailed help
generation

Scriptable

Administratives tasks

Remotely accessible

Modular

Application as a module
set

Dynamic

Add/remove modules
during runtime

Extensible

Easy addition of new
modules

Portable

Multi plateform/OS

Architecture – Simple View

Command Sets

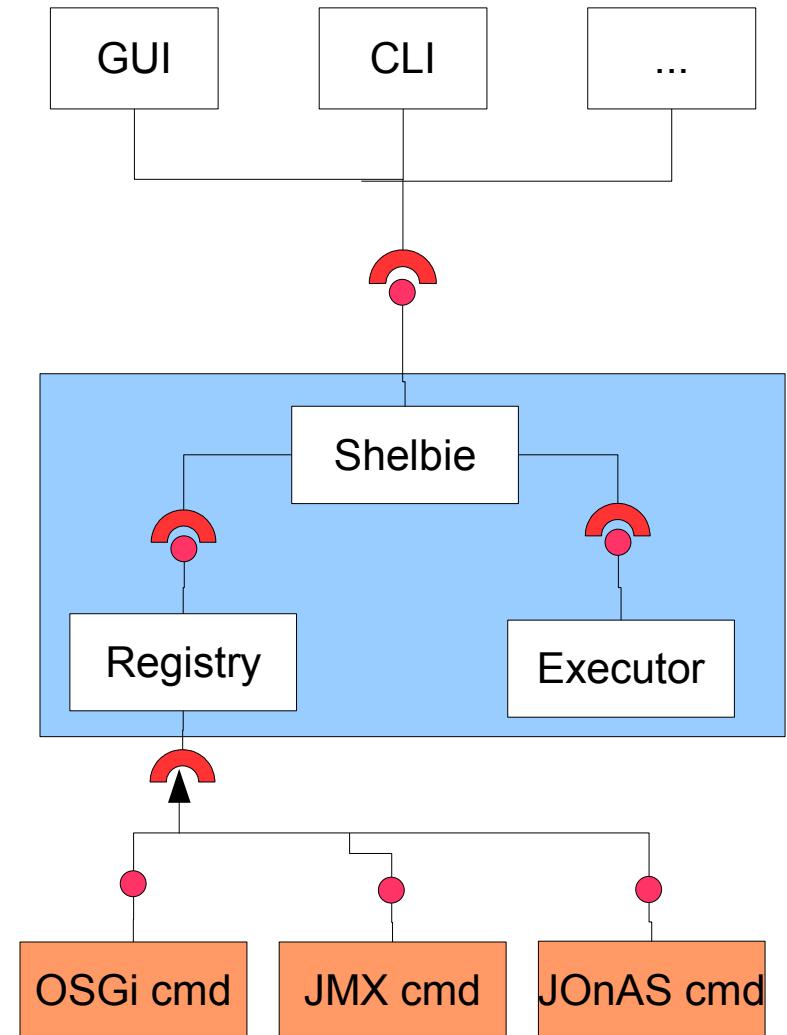
Provides commands
Ex: OSGi, JOnAS, ...

Core

Registry holds the list of available commands
Executor is responsible of command execution
Shelbie represents a shell instance (context)

Clients

Users of the Shelbie API
Ex: GUI, CLI, ...



Architecture – Multi-user's View

Remote shell

Concurrent access

Multiple clients

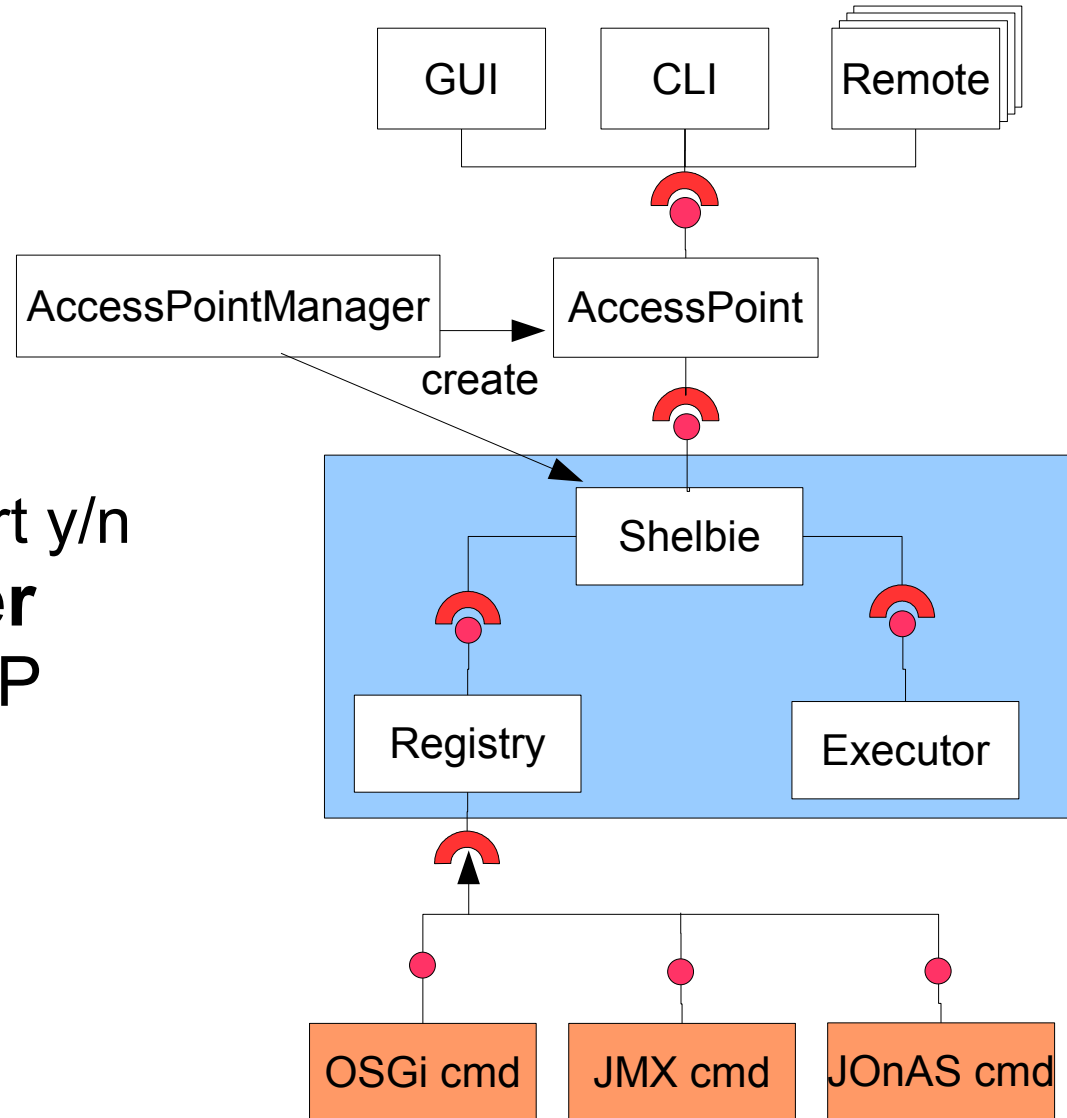
AccessPoint

Handles rich access

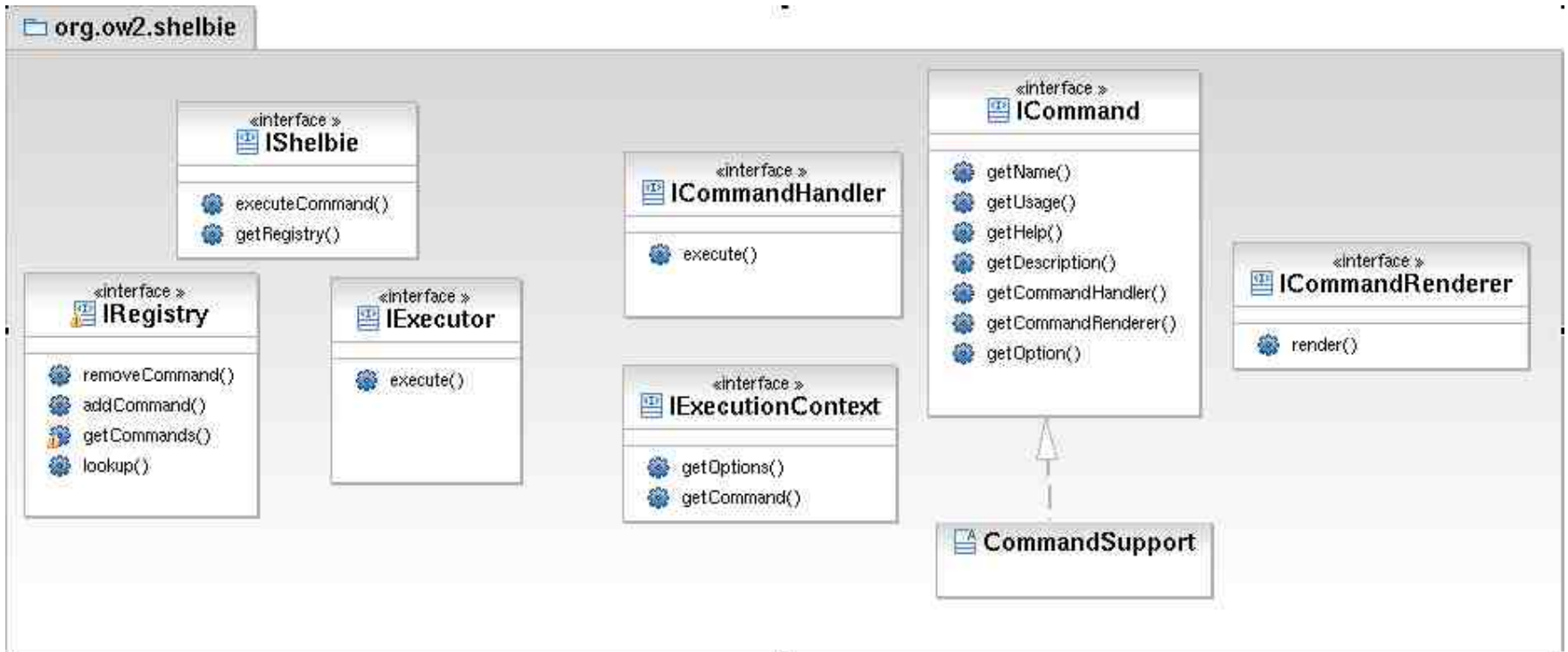
Completion support y/n

AccessPointManager

Handles shell and AP creation for each connected client



Overview



Commands providers only have to take care of *ICommand*, *ICommandHandler* and *ICommandRenderer*

Technological Choices

OSGi (Felix / Equinox)

Modules and dynamism

iPOJO

Services Component Model

Groovy

Scripting capabilities

Args4J

Command line parsing and attributes injection

JLine

History, completion and ANSI support

Jaramiko

SSH daemon and client for remote access

Nice, but ...

What a command looks like ?

```

/**
 * Command used to create {@link Configuration}s.
 * @author Guillaume Sauthier
 */
@Component(name="config-admin:create-configuration")
@Provides
public class CreateConfigurationCommand extends DefaultCommandSupport {

    @Requires
    private ConfigurationAdmin configAdmin;

    /**
     * Return the class that has to be called to execute the command.
     * @return A class that implements the ICommandHandler interface.
     */
    public Class<? extends ICommandHandler> getCommandHandler() {
        return CreateConfigurationHandler.class;
    }

    /**
     * @see org.ow2.shelbie.DefaultCommandSupport#getCommandRenderer()
     */
    @Override
    public Class<? extends ICommandRenderer> getCommandRenderer() {
        return CreateConfigurationRenderer.class;
    }
}

```

Nice, but ...

I've seen a CommandHandler somewhere ...

```

/**
 * The execution part of the echo command. This command take one option (String)
 * and print this option in the shell
 * @author Benjamin strappazon
 */
public class EchoCommandHandler implements ICommandHandler {

    /**
     * Execution of the command. Display all parameters.
     * @param executionContext Context which stores all command options and a
     *     reference on {@link EchoCommand}.
     * @return The command execution result.
     */
    public Object execute(final IExecutionContext executionContext) {

        StringBuilder st = new StringBuilder();

        // get the option object
        EchoCommand.EchoOption option = (EchoCommand.EchoOption) executionContext.getOptions();

        if (option.name != null && option.name.length() > 0) {
            st.append(option.name + " says : ");
        }

        for (Object o : option.words) {
            st.append(o + " ");
        }
        return st.toString();
    }
}

```

Nice, but ...

... and also a CommandRenderer ...

```

/**
 * @author Benjamin Strappazon
 */
public class EchoCommandRenderer implements ICommandRenderer {

    /**
     * Display exactly what the handler returns without formatting.
     * @see org.ow2.shelbie.ICommandRenderer#displayResult(java.io.OutputStream,
     *      java.lang.Object)
     * @param out The outputstream to write result.
     * @param commandExecutionResult The execution result of echo command.
     * @param executionContext The context that stores command options.
     */
    public void displayResult(final IConsoleRenderer out,
                             final Object commandExecutionResult,
                             final IExecutionContext executionContext) {

        out.println(commandExecutionResult.toString());

    }
}

```

Nice, but ...

How do I use my commands from Groovy scripts ?

```
// Invoke the function using the -m option and store the result in a variable
def listBundle = osgi_ps("-m")

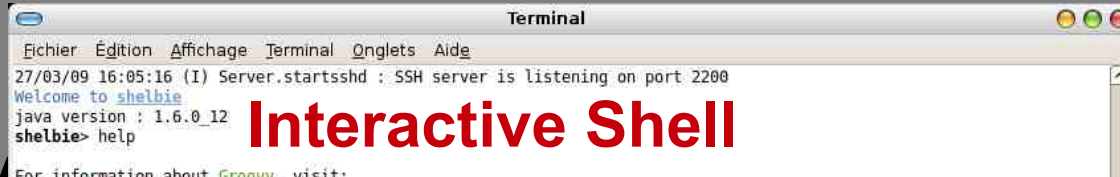
// Iterate over the list (contains real bundle objects)
listBundle.each { bundle->
    // Retrieve the bundle symbolic name and display it
    println(bundle.getSymbolicName());
}
```

```
shelbie> load ./listBundles.groovy
System Bundle
org.ow2.bundles.ow2-util-log
org.ow2.bundles.ow2-util-il8n
org.apache.felix.org.apache.felix.ipoyo
org.apache.felix.configadmin
org.ow2.shelbie.shelbie-api
org.ow2.shelbie.shelbie-core
org.ow2.shelbie.commands.shelbie-echo-command
org.ow2.shelbie.commands.shelbie-osgi-command
org.ow2.shelbie.commands.shelbie-jonas-command
org.ow2.shelbie.commands.shelbie-ca-command
org.ow2.shelbie.shelbie-cli
org.ow2.shelbie.shelbie-groovy
org.ow2.shelbie.shelbie-accesspoint
org.ow2.shelbie.shelbie-sshd
```

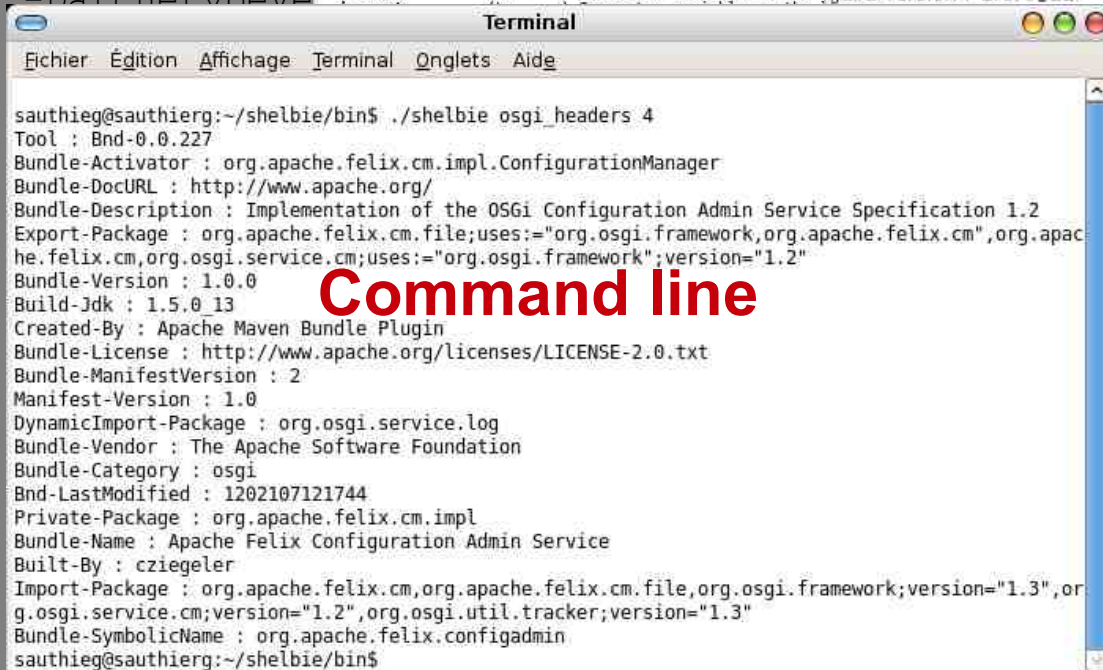
Nice, but ...

I'm convinced

>\$ mvn archetype-
-DarchetypeGroupId=org.apache.maven.archetypes:maven-archetype-quickstart:1.0
-DarchetypeArtifactId=maven-archetype-quickstart



Interactive Shell



Command line

pom.xml

Shelbie Projects Relationships

From within the consortium

OW2 JOnAS

With it's modular OSGi architecture, JOnAS will host the Shelbie server and a set of commands, shell will be accesible from the command line or SSH.

OW2 JASMINe (JaDOrT)

OW2 Utils/Bundles

From external communities

Apache Felix, Apache iPOJO, ...

The component model in use within Shelbie is hosted at Apache.

Codehaus Groovy

Independants: JLine & Jaramiko

What's next ?

Early adopters

- OW2 JaDOrT (Migration orchestration tool)

- OW2 JOnAS (thread dump)

Near future

- Transfert the codebase

- Perform a 1.0 release

- Create a web site

- Start building a community

- Finalize integration within JOnAS (5.2 timeline)

Background Tasks

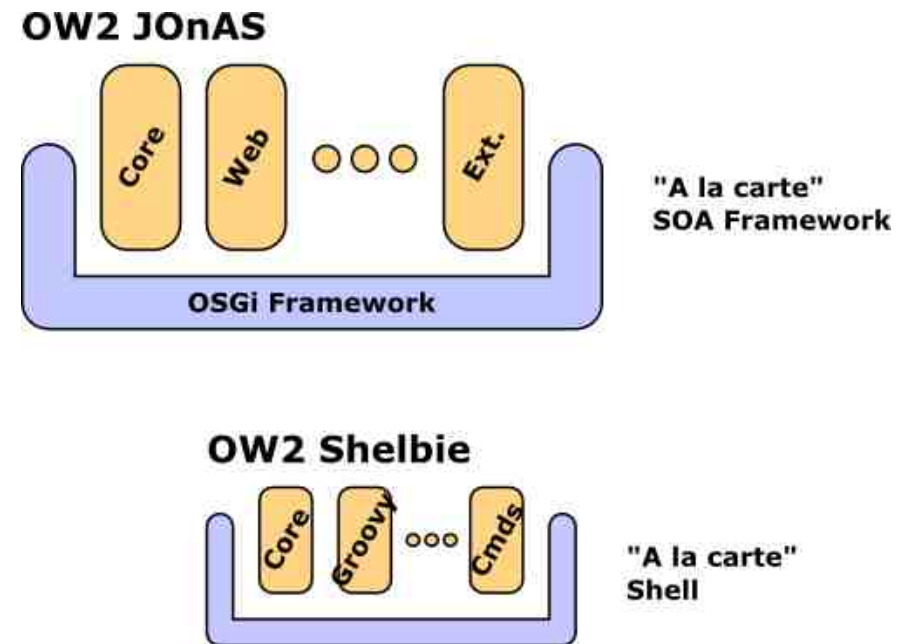
- Design Shelbie v2

 - Piped processes, easier command dev, injection, ...

- Attract users

Wrap up

License: LGPL 2.1
Small and modular
OSGi based
Dynamism
« A la carte » Shell
for right sized deployment
JOnAS independence



shelbie.ow2.org

*Ok, that's quite empty for now, but sources
will be transfered soon, I promise :-)*

GS

For more informations
Please contact
Benoit Pelletier & Guillaume Sauthier
benoit pelletier @ bull net
guillaume sauthier @ bull net