

# **CRaSH**

***CRaSH cookbook***

**Julien Viet**

**eXo Platform**

Copyright © 2011 eXo Platform SAS

## Table of Contents

Preface

### 1. FAQ

1.1. General

1.2. Running CRaSH

1.3. Basic

1.4. Commons problems

### 2. Create your first command

2.1. Run CRaSH

2.2. Add new command

2.3. refresh console

### 3. How to print an array

3.1. Exemple

3.2. Table Elements

3.3. table

3.4. attributes

3.5. styles

## Preface

Cookbook.

# 1

## FAQ

### 1.1. General

#### 1.1.1. What is CRaSH ?

CRaSH is a shell that extends JVM. With CRaSH, you will connect with a shell directly on a JVM. Moreover, you could add your command (Java/Groovy) and that's why CRaSH is really interesting.

#### 1.1.2. What can I do with CRaSH ?

- Monitoring JVM and make your own dashboard command.
- Make command for your application (add data in a cache, add user, monitor jobs ).
- Make your JMX command.

#### 1.1.3. What is the differences between CRaSH and JMX ? Is the same ?

JMX provides only bean and methods. That's all. CRaSH permit to access to JMX and to make command with it. CRaSH also permit to make script with thread, jdbc, entity ...

### 1.2. Running CRaSH

#### 1.2.1. How can I run CRaSH ?

See documentation : <http://www.crashub.org/doc.html#running>

#### 1.2.2. How can I connect Crash to a JVM ?

See documentation Connection in Shell Usage chapter <http://www.crashub.org/doc.html#d5e219>

### 1.3. Basic

#### 1.3.1. What is the best way to create a command ?

The best way to create a command is to use CRaSH utilities. See command as a class : <http://www.crashub.org/doc.html#d5e531>

### 1.3.2. What is the best way to start with CRaSH ?

- Launch CRaSH and play with commands
- Create commands (script command and class command)
- See cookbook and documentation.

## 1.4. Commons problems

### 1.4.1. "Command not found"

In most cases, when you created a command, it's a syntax error in a command. Check your import and syntax with your ide.

### 1.4.2. Can't find crash.properties file

You have to launch CRaSH in standalone mode once. Then, it will appears in `$CRASH_HOME/conf/`

### 1.4.3. "Remoting issue"

It could happen when you have an error in your command. For example :

```
% jdbc select se  
Remoting issue
```

### 1.4.4. Where are base commands ?

They will be in `$CRASH_HOME/cmd/base` directory. You have to launch CRaSH once in standalone mode.

# 2

## Create your first command

In this cookbook, you will learn how to create a simple script command. You will see that you can create it dynamically without restarting CRaSH.

A better solution to create a command is to use CRaSH class. It provides tools to simplify command creation.

### 2.1. Run CRaSH

```
cd $CRASH_HOME/bin  
./crash.sh  
Type help at prompt
```

You will see something like :

Try one of these commands with the -h or --help switch:

NAME	DESCRIPTION
clock	
dashboard	
date	show the current time
env	display the term env
filter	
hello	
help	provides basic help
java	various java language commands
jdbc	JDBC connection
jmx	Java Management Extensions
jndi	Java Naming and Directory Interface
jpa	Java persistance API
jvm	JVM informations
log	logging commands
man	format and display the on-line manual pages
shell	shell related command
sleep	sleep for some time
sort	Sort a map
system	vm system properties commands
thread	JVM thread commands

## 2.2. Add new command

To add a command to CRaSH. You have to add a groovy file in the cmd directory :

```
cd $CRASH_HOME/cmd
vi test.groovy
```

Put the following in test.groovy :

```
for (int i = 0;i < 10;i++) {
    System.out.println("CRaSH is cool !");
}
```

In this example, we create a command by using Java syntax. It's because Groovy understand Java Syntax. So you could begin to develop your command in Java and when you want try cool Groovy stuff.

## 2.3. refresh console

Type help again at prompt and you will see test command.

```
% help
```

Try one of these commands with the -h or --help switch:

NAME	DESCRIPTION
clock	
dashboard	
date	show the current time
env	display the term env
filter	
hello	
help	provides basic help
java	various java language commands
jdbc	JDBC connection
jmx	Java Management Extensions
jndi	Java Naming and Directory Interface
jpa	Java persistence API
jvm	JVM informations
log	logging commands
man	format and display the on-line manual pages
shell	shell related command
sleep	sleep for some time
sort	Sort a map
system	vm system properties commands
test	
thread	JVM thread commands

## How to print an array

### 3.1. Exemple

Here is an example for printing an array :

```
import org.crsh.text.ui.UIBuilder

UIBuilder ui = new UIBuilder();

ui.table(separator: dashed) {
    header(decoration: bold, foreground: black, background: white) {
        label("ATTRIBUTE NAME"); label("ACCESS"); label("TYPE"); label("DESCRIPTION")
    }

    for(Attr tmpAttr : lst) {
        if (null != tmpAttr) {
            row() {
                label(tmpAttr.name, foreground: red);
                label(tmpAttr.access);
                label(tmpAttr.type);
                label(tmpAttr.desc);
                label(tmpAttr.attrs.toString());
            }
        }
    }
}

out << ui;
```

### 3.2. Table Elements

To define an array, you will use elements like header, label ... If you want to see an example, edit dashboard.groovy in \$CRASH\_HOME/cmd/base/

### 3.3. table

- Define table.

### 3.3.1. label

- Print a label.

### 3.3.2. columns

- Define columns.
- e.g : columns: [1]

### 3.3.3. rows

- Define rows.
- rows: [1,1]

### 3.3.4. header

- Define header.
- header element

### 3.3.5. eval

- Execute an other CRaSH command.

```
eval {  
  execute("jvm heap")  
}
```

## 3.4. attributes

Attribute can be add to table element.

### 3.4.1. border

- Define a border.
- e.g: border: dashed

### 3.4.2. row

### 3.4.3. separator

- Define separator style.
- e.g : dashed,star

#### **3.4.4. overflow**

- overflow ?

#### **3.4.5. leftCellPadding**

- Left align in cell.

#### **3.4.6. rightCellPadding**

- Right align in cell
- e.g : rightCellPadding: 1

### **3.5. styles**

Class `org.crsh.text.Style` contains style that we will use when making elements (e.g : header)

#### **3.5.1. bold**

- Type : boolean
- e.g : bold: true

#### **3.5.2. underline**

- Type : boolean

#### **3.5.3. blink**

- Type : boolean

#### **3.5.4. fg, foreground**

- Type : Color
- fg: black

#### **3.5.5. bg, background**

- Type : Color
- bg: white

