GSoC 2015 - CLI Tool for WSO2 AppFactory

Contents

Identify

Project Title	AppFactory Command Line Tool
Project Link	https://docs.wso2.com/display/GSoC/Project+Proposals+for+2015#ProjectProposalsfor2015-Proposal6:AppFactory-CommandLineTool
Project Repo	https://github.com/Dilhasha/AppFacCLI
Assigned Mentors	Anuruddha Premalal (anuruddha@wso2.com) Dimuthu Leelarathne (dimuthul@wso2.com)
Assigned Contributor	M.N.F.Dilhasha

Project Summary

A Command Line Tool(CLI) is a mechanism which provides speed and control to a Software Product. CLI tools make the product traceable, scriptable and adds automation capabilities for developers. There is a large community of developers who are attracted to CLI based tools as opposed to GUI based interfaces due to the simplicity provided by CLI tools.

This project focuses on building such a CLI to interact with WSO2 AppFactory, so that a user can perform functionalities on AppFactory platform via the CLI tool similar to using the Web based GUI interface. The CLI tool will support basic operations on App Factory such as creating, building applications, getting build status, deployment status, checking build history and checking the health of servers.

Required Deliverable

The deliverable for the project is a command line tool which provides operations on App Factory and which is compatible with linux, windows and mac operating systems.

Project implementation plan

1. Write build script for different OS using ‘GO’
2. Design a Command Evaluation engine for the tool
3. Implement General Commands
4. Implement App Specific Commands
5. Implement User Authentication
6. Improve error handling for user inputs
7. Testing for the CLI tool

Project schedule

Duration	Description
May 1st - May 7th	● Use and study existing CLI tools ● Research and decide on the approach to be taken ● Get familiarized with ‘GO’
May 18th - May 25th	● Write build script to develop CLI tool for main operating systems ● Get familiarized with ‘GO’
May 22nd - June 4th	● Decide on the approach for processing commands ● Design a Rule based Command Evaluation engine for the tool
June 5th - June 25th	● Implement the Command Evaluation Engine

	• Implement API Features for a general command to list applications of a user
June 26th -July 16th	• Implement other general commands • Implement App specific commands
July 17th -August 6th	• Implement user authentication • Verify user authentication before specific commands
August 7th - August 13th	• Refine error handling for user inputs
August 14th-August 22th	• Testing of the CLI tool

The comprehensive milestone plan can be found at [4]

Design Decisions

Why Use Golang?

'Go' is a language designed with system programming in mind and is gaining attraction as a language designed to develop highly concurrent applications. It also provides the ability to create cross-compiled binaries with no dependencies. It's capability of compiling to a single, statically linked binary is a high performance improvement as well.[1]

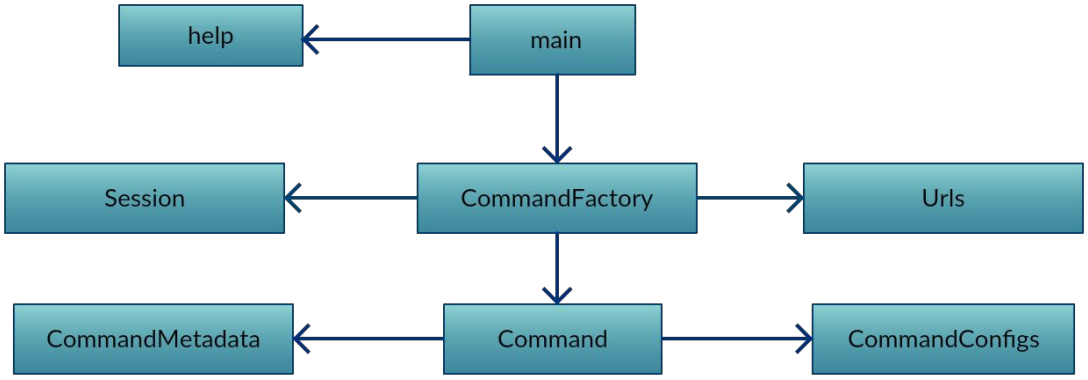
Above features in 'GO' address the core requirements for the CLI tool. There are additional reasons for considering 'GO'.

Recently, many development communities have moved their CLI tools written in other languages to 'GO'. One such example is 'CF CLI' for Cloud Foundry and the community admits in it release that the transformation has resulted in higher performance and cleaner code base [2].

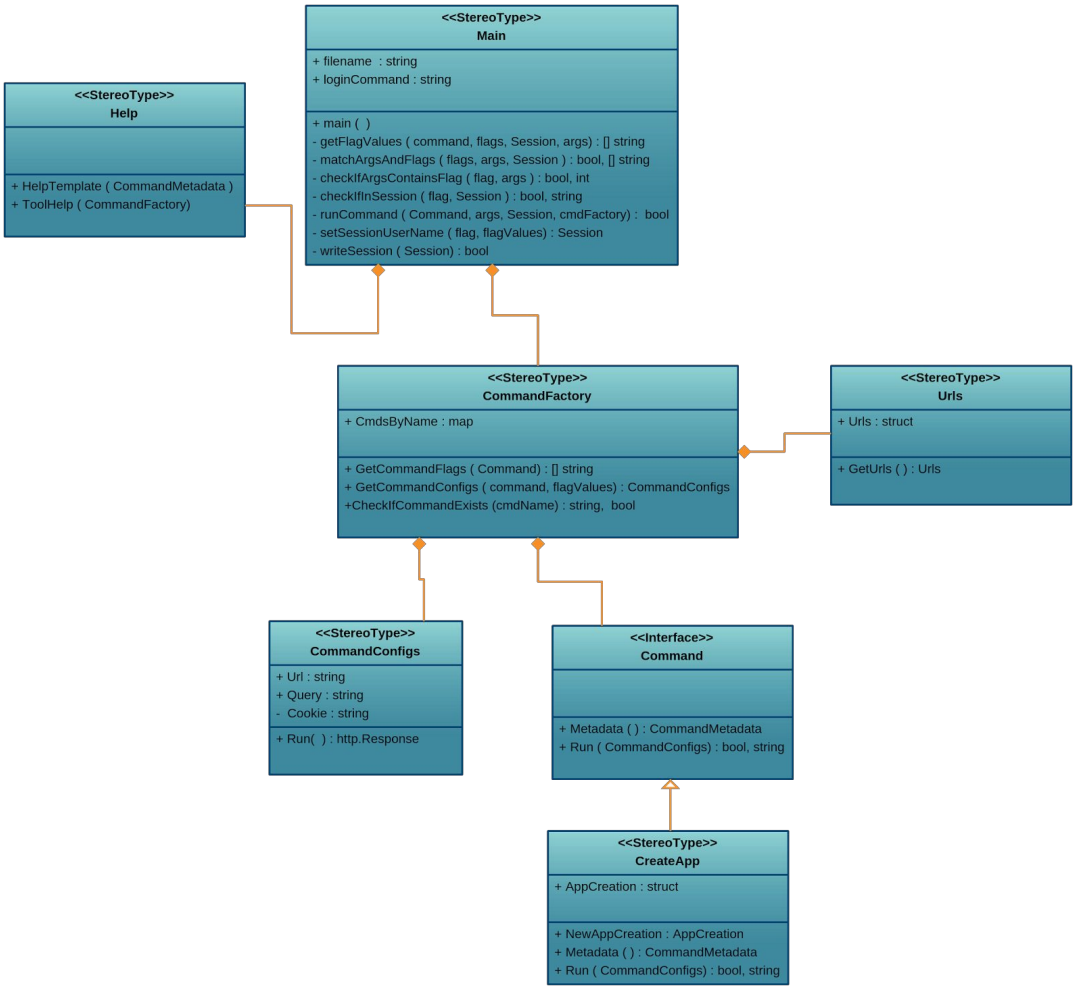
Another example is the Git Hub CLI tool which was initially referred to as 'hub' and was written in Ruby. The developers have pointed out that they were able to overcome the limitation of speed with the use of 'gh', the alternative written in 'GO'. They also found the distribution as a pre-compiled binary, no requirement for pre-installation of any software as very attractive features of 'GO'.[3]

So based on the above mentioned factors, 'GO' is the best scripting language option for developing the CLI tool for WSO2 Appfactory.

Basic Architecture for the Tool



Class Diagram for the rule-based engine



Why use a rule based engine over a switch case based approach?

- reduction in the time taken to evaluate a command over a switch case based approach
- allows for maximum reuse of the code
- easier to extend for any new commands
- have been used by similar CLI tools

Extending the CLI tool

For implementation of a new command, three steps need to be followed.

1. Add URL for command to the URL storage
2. Implement the command line interface
3. Add the command to Command Factory

Following is an example on how each step can be carried out for implementing the command to delete an application.

1. **Add URL for command to the URL storage**

    - Add the following attribute line to the the 'Urls' struct.
      DeleteApp string

    - Add the 'DeleteApp' value to the 'Urls' object returned in 'GetUrls' function as follows. This depends on your Appfactory set up. (Change the base_url accordingly)

DeleteApp:"https://203.94.95.207:9443/appmgt/site/blocks/application/delete/ajax/delete.jag",

2. **Implement the command line interface**

- Creating and adding basic functions for 'AppDeletion'

    - Create a new go file named 'deleteApp.go'

    - First create a struct with the name 'AppDeletion', it should have an attribute named Url.

      type AppDeletion struct {
      Url string
      }

- Then create a function to return an AppDeletion object

```
func NewAppDeletion(url string) (cmd AppDeletion) {
  return AppDeletion{
    Url : url,
  }
}
```

- Implementing the functions in ‘Command’ interface

  - Implement ‘Metadata()’ function’

```
func (appDeletion AppDeletion)Metadata() CommandMetadata{
        return CommandMetadata{
                Name : "deleteApplication",
                Description : "delete an application of user",
                ShortName : "da",
                Usage : "delete app",
                Url : appDeletion.Url,
                SkipFlagParsing : false,
                Flags : []cli.Flag{
                  cli.StringFlag{Name: "-a", Usage: "appKey"},
                  cli.StringFlag{Name: "-u", Usage: "userName"},
                  cli.StringFlag{Name: "-c", Usage: "cookie"},
                },
        }
}
```

  - Implement ‘Run()’ function

```
func(appDeletion AppDeletion) Run(configs CommandConfigs)(bool , string){
        //Send http request and get response
        resp := configs.Run()
        //if request did not fail
        if(resp != nil){
          defer resp.Body.Close()
        }else{
          //exit the cli
          return true, ""
        }
        body, _ := ioutil.ReadAll(resp.Body)

        if (resp.Status == "200 OK") {
```

```
            bodyString := string(body)
            var errorFormat formats.ErrorFormat
            err := json.Unmarshal([]byte(bodyString), &errorFormat)
            if (err == nil) {
              if (errorFormat.ErrorCode == http.StatusUnauthorized) {
                fmt.Println(errorFormat.ErrorMessage)
                fmt.Println("Your session has expired.Please login and try again!")
                return false , configs.Cookie
              }
            }
            var successMessage formats.SuccessFormat
            err = json.Unmarshal([]byte(bodyString), &successMessage)

            if(err == nil){
              fmt.Println(successMessage.Message)
            }
          }
          return true,configs.Cookie
}
```

3. **Add the command to Command Factory**

Once the command is successfully implemented, the user needs to be aware that ‘deleteApp’ functionality is available. Add the following line to ‘commandFactory.go’ file’s ‘NewFactory()’ function.

```
factory.CmdsByName["deleteApplication"] = NewAppDeletion(urls.DeleteApp)
```

Once these three steps are followed correctly, the command ‘deleteApplication’ is available to the user.

Project Demonstration

A demonstration of the project can be found at [5]
Presentation slides for the demonstration can be found at [6]

Project Executables

Executables of the project can be found at [7]

Related Blog Posts

Blogs written related to the project can be found at [8]

References

[1] http://www.tutorialspoint.com/go/go_overview.htm

[2] http://blog.cloudfoundry.org/2013/11/09/announcing-cloud-foundry-cf-v6/

[3] https://github.com/github/hub/issues/475

[4]https://docs.google.com/spreadsheets/d/1VW_RtTdfUShhClSvfkPSr0b8hDuWM61mX6qjSFE05NY/edit?usp=sharing

[5] https://drive.google.com/file/d/0B5jf9n7hxy8YV1VQMTJGZ3ZKeU0/view

[6]https://docs.google.com/presentation/d/1yNojFbikh09V57tMtcoaacyc_17Ev1xQrraQCBxjezY/edit#slide=id.p

[7]https://drive.google.com/open?id=0B5jf9n7hxy8YfjJ4NnFWY0pXYklIMkFfNnNHSFF5bXB3UjE2eUZjdVJacGM1TDc3Vk4wY1E

[8] http://dilhashatechspace.blogspot.com/2015/07/gsoc-2015-project-for-wso2-appfactory.html