Если вы видите что-то необычное, просто сообщите мне.

Пример использования Jenkins REST API на Golang

Я собираюсь описать как вызывать Jenkins REST API используя GO. Создадим три Go файла:

1. packages/helpers/helpers.go
2. packages/jenkins/jenkins.go
3. main.go

Исходный код доступен по адресу: https://github.com/mohdnaim/jenkins_rest_api

`helpers.go` содержит функции помощники. В нем будут расположены функции для чистоты и организации кода.

`jenkins.go` содержит функции связанные с Jenkins. Это функции `IsJobExist()` для проверки существования конвеера задачи или билда, `CopyJenkinsJob()` для копирования Jenkins задания и `DownloadConfigXML()` для скачивания конфигурации задачи в формате XML.

`main.go` это главный файл, точка старта нашей программы.

```
//main.go
package main

import (
"fmt"
"log"
"strings"
helpers "./packages/helpers"
jenkins "./packages/jenkins"
)

func main() {
```

```
// Обязательные настройки
jenkinsURL := "АДРЕС СЕРВЕРА"
jenkinsUsername := "ИМЯ ПОЛЬЗОВАТЕЛЯ"
jenkinsAPIToken := "API ТОКЕР"
jenkins.JenkinsDetails = jenkins.Details{jenkinsURL, jenkinsUsername, jenkinsAPIToken}
xmlFolder := "xml"
// 1. Получаем все доступные провекты\задачи
allProjectNames := jenkins.GetAllProjectNames()
// 2. Фильтруем наши проекты которые мы ищем
filteredProjectNames := make([]string, 0)
for _, projectName := range allProjectNames {
  // Что-то делаем
  // Добавляем в другой срез основываясь на условии
  if strings.HasPrefix(projectName, "prefix") {
    filteredProjectNames = append(filteredProjectNames, projectName)
  }
}
// 3. Для каждого проекта получаем его  config.xml
for _, projectName := range filteredProjectNames {
  xmlPath := fmt.Sprintf("%s/%s.xml", xmlFolder, projectName)
  if err := jenkins.DownloadConfigXML(projectName, xmlPath); err != nil {
    log.Println("error download config.xml for project:", projectName)
    continue // пропускаем
  }
}
// 4. Изменяем config.xml
files := helpers.GetFilenamesRecursively(xmlFolder)
for _, xmlFile := range files {
  log.Println(xmlFile)
}
// 4b. Переписываем  config.xml
// 5. http POST запрос обновления config.xml
for _, xmlFile := range files {
  tmpSlice := strings.Split(xmlFile, "/")
  projectName := tmpSlice[len(tmpSlice)-1]
  log.Println(projectName)
  if err := jenkins.PostConfigXML(projectName, xmlFile); err != nil {
    log.Println("error postconfigxml:", projectName)
  }
}
```

```
}
```

Нам нужно настроить детали Jenkins объекта. Укажем корректное значение для следующих переменных:

```
jenkinsURL := "АДРЕС СЕРВЕРА"
jenkinsUsername := "ИМЯ ПОЛЬЗОВАТЕЛЯ"
jenkinsAPIToken := "API ТОКЕР"
```

Строка #18 Создаем структуру содержащую три переменных выше. Структура используется функции в `jenkins.go` поэтому перед вызовом любой функции из `jenkins.go` модуля, нам нужно настроить эти значения.

Строка #19 Укажем папку в которой будем хранить XML файл, который является конфигурацией Jenkins задачи.

Строка #22 Получаем имя всех проектов существующих в Jenkins. Если мы посмотрим на реализацию функции `GetAllProjectNames()` в `jenkins.go`, функция вызывает `DownloadFileToBytes()`. Она делаеет HTTP запрос на `/api/json` используя имя пользователя и пароль в заголовке запроса. Точка доступа вернет имя проекта в JSON формате. Затем преобразуется JSON в `result` словарь. Любое имя в словаре добавляет имя в `allProjectNames` массив.

Строка #24 Отсортируем наши проекты, так как мы хотим. Пройдемся по `allProjectNames` массива и если проект встречает определенный критерий, мы добавляем имя проекта в `filteredProjectNames` массив который мы будем использовать вместо `allProjectNames`.

Строка #35 Для каждого имени проекта в `filteredProjectNames` мы получаем его конфигурационный файл config.xml.

Строка #45 Вызываем фукнкцию `GetFileNamesRecursively()` в `helpers.go` для чтения всех имен файлов в `xmlFolder`.

Строка #50 Мы делаем всё что хотим с `config.xml` файлом, например переименовываем имя проекта, добавляем параметры сборки, изменяем права и т.д.

Я написал скрипт на Python вместо Go для управления `config.xml` файлом, так как Pyehon имеет больше библиотек чем Go, которые могут помочь нам для управления строками и

файлами.

Строка #52 Наконец, после того, как мы создали изменения в строке 50, мы отправляем изменения выполняя POST запрос в Jenkins.

```
//helpers.go
package helpers

import (
"os"
"path/filepath"
)
// StringInSlice ...
func StringInSlice(a string, list []string) bool {
for _, b := range list {
if b == a {
return true
}
}
return false
}
// GetFilenamesRecursively ...
func GetFilenamesRecursively(path string) []string {
var files []string
err := filepath.Walk("xml", func(path string, info os.FileInfo, err error) error {
files = append(files, path)
return nil
})
if err != nil {
panic(err)
}
return files
}
```

```
//jenkins.go
package jenkins
import (
"encoding/json"
"fmt"
"io"
```

```
"io/ioutil"
"log"
"net/http"
"os"
"reflect"
)
// Details ...
type Details struct {
URL      string
Username string
APIToken string
}
// JenkinsDetails ...
var JenkinsDetails = Details{}
// IsJobExist ...
func IsJobExist(projectName string) bool {
fullURL := fmt.Sprintf("%sjob/%s", JenkinsDetails.URL, projectName)
request, err := http.NewRequest("GET", fullURL, nil)
request.SetBasicAuth(JenkinsDetails.Username, JenkinsDetails.APIToken)
client := &http.Client{}
resp, err := client.Do(request)
if err != nil {
log.Println("project doesn't exist:", projectName)
return false
}
if resp.StatusCode == 200 {
// log.Println("project exists:", projectName)
return true
}
log.Println("project doesn't exist:", projectName)
return false
}
// CopyJenkinsJob ...
func CopyJenkinsJob(srcJob string, dstJob string) error {
fullURL := fmt.Sprintf("%s%s?name=%s&mode=copy&from=%s", JenkinsDetails.URL, "createItem",
dstJob, srcJob)
request, err := http.NewRequest("POST", fullURL, nil)
request.SetBasicAuth(JenkinsDetails.Username, JenkinsDetails.APIToken)
client := &http.Client{}
resp, err := client.Do(request)
if err != nil {
```

```
    return err
  }
  if resp.StatusCode == 200 {
    return nil
  }
  return fmt.Errorf("error copying job: %s", srcJob)
}
// DownloadConfigXML ...
func DownloadConfigXML(projectName string, dstFilename string) error {
  fullURL := fmt.Sprintf("%sjob/%s/config.xml", JenkinsDetails.URL, projectName)
  if DownloadFile(fullURL, dstFilename) == nil {
    return nil
  }
  return fmt.Errorf("error downloading file %s", fullURL)
}
// PostConfigXML ...
func PostConfigXML(projectName string, filename string) error {
  // read content of file
  data, err := os.Open(filename)
  if err != nil {
    log.Fatal(err)
    return err
  }
  fullURL := fmt.Sprintf("%sjob/%s/config.xml", JenkinsDetails.URL, projectName)
  request, err := http.NewRequest("POST", fullURL, data)
  request.SetBasicAuth(JenkinsDetails.Username, JenkinsDetails.APIToken)
  client := &http.Client{}
  // perform the request
  resp, err := client.Do(request)
  if err != nil {
    return err
  }
  if resp.StatusCode == 200 {
    // log.Printf("success postConfigXML: %s %s %s", projectName, filename, resp.Status)
    return nil
  }
  return fmt.Errorf("error postConfigXML: %s %s %s", projectName, filename, resp.Status)
}
// DownloadFile ...
func DownloadFile(url string, filepath string) error {
  // Create the file
```

```
	out, err := os.Create(filepath)
	if err != nil {
		return err
	}
	defer out.Close()
	// Get the data
	request, err := http.NewRequest("GET", url, nil)
	request.SetBasicAuth(JenkinsDetails.Username, JenkinsDetails.APIToken)
	client := &http.Client{}
	resp, err := client.Do(request)
	if err != nil {
		return err
	}
	defer resp.Body.Close()
	// Write the body to file
	_, err = io.Copy(out, resp.Body)
	if err != nil {
		return err
	}
	return nil
}
// DownloadFileToBytes ...
func DownloadFileToBytes(url string) ([]byte, error) {
	request, err := http.NewRequest("GET", url, nil)
	request.SetBasicAuth(JenkinsDetails.Username, JenkinsDetails.APIToken)
	client := &http.Client{}
	resp, err := client.Do(request)
	if err != nil {
		return nil, err
	}
	defer resp.Body.Close()
	bodyBytes, err := ioutil.ReadAll(resp.Body)
	if err != nil {
		log.Fatal(err) // writes to standard error
	}
	return bodyBytes, nil
}
// GetAllProjectNames ...
func GetAllProjectNames() []string {
	allProjectNames := make([]string, 0)
	allProjectsURL := fmt.Sprintf("%sapi/json?pretty=true", JenkinsDetails.URL)
```

```
	if respBytes, err := DownloadFileToBytes(allProjectsURL); err == nil {
		// json --> map
		var result map[string]interface{}
		json.Unmarshal([]byte(respBytes), &result)
		// if map has slice 'jobs', iterate over it
		if jobs, keyIsPresent := result["jobs"]; keyIsPresent && reflect.TypeOf(jobs).Kind() ==
reflect.Slice {
			jobs2 := result["jobs"].([]interface{})
			for _, job := range jobs2 {
				// log.Println(job)
				_map, _ := job.(map[string]interface{}) // assert to map
				allProjectNames = append(allProjectNames, _map["name"].(string))
			}
		}
	}
	return allProjectNames
}
```

Revision #3

Created 2022-08-11 20:11:27 UTC by gasick

Updated 2023-04-16 19:30:03 UTC by gasick