GROOVY CASE STUDIES

Web Scraping & Data Extraction with Groovy & Jsoup

Solution

- Used Groovy + Jsoup to extract pricing data.
- Scheduled the script using cron jobs for daily updates.
- Exported data into a CSV file for analysis.

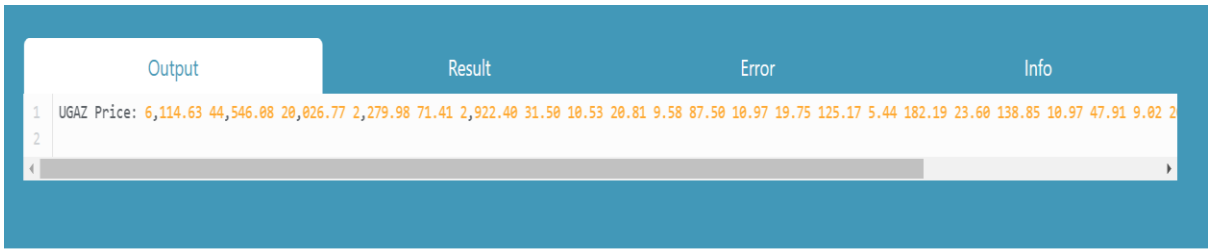

```
@Grab('org.jsoup:jsoup:1.13.1')
import org.jsoup.Jsoup

def url = "https://finance.yahoo.com/quote/UGAZ"
def doc = Jsoup.connect(url).get()

// Inspect the Yahoo page source and find the correct selector
def priceElement = doc.select("fin-streamer[data-field=regularMarketPrice]")

if (priceElement) {
    def price = priceElement.text()
    println "UGAZ Price: $price"
} else {
    println "Price element not found!"
}
```

Output : Scraping prices from website



Groovy DSL for Configuration Management

Solution:

- Developed a Groovy DSL to define infrastructure configurations.
- Allowed users to write settings in Groovy instead of JSON/YAML.
- Integrated with Terraform for cloud infrastructure provisioning.

Groovy Web Console

System Github

```
class Config {
    String environment
    Database database = new Database()
    Server server = new Server()

    void environment(String env) {
        environment = env
    }

    void database(Closure closure) {
        closure.delegate = database
        closure.resolveStrategy = Closure.DELEGATE_FIRST
        closure()
    }

    void server(Closure closure) {
        closure.delegate = server
        closure.resolveStrategy = Closure.DELEGATE_FIRST
        closure()
    }

    void printConfig() {
        println "Environment: $environment"
        println "Database: Host=${database.host}, User=${database.user}, Password=${database.password}"
        println "Server: Memory=${server.memory}, CPU=${server.cpu}"
    }
```

```
}

class Database {
    String host
    String user
    String password

    void host(String h) { host = h }
    void user(String u) { user = u }
    void password(String p) { password = p }
}

class Server {
    String memory
    String cpu

    void memory(String m) { memory = m }
    void cpu(String c) { cpu = c }
}

// Define a method that accepts a closure
def config(Closure closure) {
    def conf = new Config()
    closure.delegate = conf
    closure.resolveStrategy = Closure.DELEGATE_FIRST
    closure()
    conf.printConfig()  // Output the final configuration
}
```

```
// Use the custom DSL
config {
    environment "production"
    database {
        host "db.example.com"
        user "admin"
        password "securePassword"
    }
    server {
        memory "8GB"
        cpu "4"
    }
}
```

Groovy 4.0 | Execute | Share as Link | History | Class Generation | Inspect Ast

Output | Result | Error | Info

```
Environment: production
Database: Host=db.example.com, User=admin, Password=securePassword
Server: Memory=8GB, CPU=4
```