DEPLOY APPLICATION ON CLOUD SOURCE CODE

Prepared By: DUGGASANI NAGA SUREKHA

SeleniumCloudTest.java

```
   package com.seleniumcloud.test;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class SeleniumCloudTest {

    private WebDriver driver;

     @BeforeClass
      public void setUp() throws
MalformedURLException {

         ChromeOptions chromeOptions = new
ChromeOptions();

 //chromeOptions.setCapability("browserVersion",
"89");
         chromeOptions.setCapability("platformName",
"Linux");
```

```java
         driver = new RemoteWebDriver(new
URL("http://localhost:4444/wd/hub"), chromeOptions);

 driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

      }


     @Test
      public void sampleTest() {
        System.out.println("Selenium Test Starts");
        driver.get("https://www.swiggy.com");
        System.out.println("Selenium Test Success");
    }


     @AfterClass
      public void tearDown() {
        driver.quit();
      }

}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.selenium.cloud</groupId>
  <artifactId>SeleniumCloudProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
  <!--
https://mvnrepository.com/artifact/org.seleniumhq.sel
enium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.testng/testng
-->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>6.14.3</version>
    <scope>test</scope>
```

```
</dependency>
  </dependencies>
  <build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.21.0</version>
      <configuration>
                   <!-- TestNG Suite XML files list
for test execution -->
                   <suiteXmlFiles>

 <suiteXmlFile>testng.xml</suiteXmlFile>

                   </suiteXmlFiles>
               </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Testing.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-
1.0.dtd">
<suite name="Testing Swiggy App">
    <test name="Regression">
         <classes>
```

```
                <class
name="com.seleniumcloud.test.SeleniumCloudTest"/>

            </classes>
     </test>
</suite>
```

MavenWrapperDownloader.java

```
import java.net.*;
import java.io.*;
import java.nio.channels.*;
import java.util.Properties;

public class MavenWrapperDownloader {

    private static final String WRAPPER_VERSION =
"0.5.6";
    /**
     * Default URL to download the maven-wrapper.jar
from, if no 'downloadUrl' is provided.
     */
    private static final String DEFAULT_DOWNLOAD_URL =
"https://repo.maven.apache.org/maven2/io/takari/maven-
wrapper/"
        + WRAPPER_VERSION + "/maven-wrapper-" +
WRAPPER_VERSION + ".jar";

    /**
     * Path to the maven-wrapper.properties file, which
might contain a downloadUrl property to
```

```
     * use instead of the default one.
     */
    private static final String
MAVEN_WRAPPER_PROPERTIES_PATH =
            ".mvn/wrapper/maven-wrapper.properties";

    /**
     * Path where the maven-wrapper.jar will be saved
to.
     */
    private static final String MAVEN_WRAPPER_JAR_PATH =
            ".mvn/wrapper/maven-wrapper.jar";

    /**
     * Name of the property which should be used to
override the default download url for the wrapper.
     */
    private static final String
PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";

    public static void main(String args[]) {
        System.out.println("- Downloader started");
        File baseDirectory = new File(args[0]);
        System.out.println("- Using base directory: " +
baseDirectory.getAbsolutePath());

        // If the maven-wrapper.properties exists, read
it and check if it contains a custom
        // wrapperUrl parameter.
        File mavenWrapperPropertyFile = new
File(baseDirectory, MAVEN_WRAPPER_PROPERTIES_PATH);
```

```
        String url = DEFAULT_DOWNLOAD_URL;
        if(mavenWrapperPropertyFile.exists()) {
            FileInputStream
mavenWrapperPropertyFileInputStream = null;
            try  {
                mavenWrapperPropertyFileInputStream =
new FileInputStream(mavenWrapperPropertyFile);
                Properties mavenWrapperProperties = new
Properties();

mavenWrapperProperties.load(mavenWrapperPropertyFileInpu
tStream);
                url =
mavenWrapperProperties.getProperty(PROPERTY_NAME_WRAPPER
_URL, url);
            } catch (IOException e) {
                System.out.println("- ERROR loading '" +
MAVEN_WRAPPER_PROPERTIES_PATH + "'");
            } finally {
                try {

if(mavenWrapperPropertyFileInputStream != null) {

mavenWrapperPropertyFileInputStream.close();
                    }
                } catch (IOException e) {
                    // Ignore ...
                }
            }
        }
        System.out.println("- Downloading from: " +
url);

        File outputFile = new
File(baseDirectory.getAbsolutePath(),
MAVEN_WRAPPER_JAR_PATH);
        if(!outputFile.getParentFile().exists()) {
            if(!outputFile.getParentFile().mkdirs()) {
                System.out.println(
                        "- ERROR creating output
directory '" +
outputFile.getParentFile().getAbsolutePath() + "'");
            }
        }
        System.out.println("- Downloading to: " +
outputFile.getAbsolutePath());
        try {
            downloadFileFromURL(url, outputFile);
            System.out.println("Done");
            System.exit(0);
        } catch (Throwable e) {
            System.out.println("- Error downloading");
            e.printStackTrace();
            System.exit(1);
        }
    }

    private static void downloadFileFromURL(String
urlString, File destination) throws Exception {
        if (System.getenv("MVNW_USERNAME") != null &&
System.getenv("MVNW_PASSWORD") != null) {
```

```
            String username =
System.getenv("MVNW_USERNAME");
            char[] password =
System.getenv("MVNW_PASSWORD").toCharArray();
            Authenticator.setDefault(new Authenticator()
{
                @Override
                protected PasswordAuthentication
getPasswordAuthentication() {
                    return new
PasswordAuthentication(username, password);
                }
            });
        }
        URL website = new URL(urlString);
        ReadableByteChannel rbc;
        rbc = Channels.newChannel(website.openStream());
        FileOutputStream fos = new
FileOutputStream(destination);
        fos.getChannel().transferFrom(rbc, 0,
Long.MAX_VALUE);
        fos.close();
        rbc.close();
    }

}
```

<u>VersionController.java</u>

```
package com.example.demo.controller;
```

```
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class VersionController {

    @GetMapping("/version")
    public String getVersion(){
        return "1.0";
    }
}
```

<u>ServletInitializer.java</u>

```
package com.example.demo;

import
org.springframework.boot.builder.SpringApplicationBuilde
r;
import
org.springframework.boot.web.servlet.support.SpringBootS
ervletInitializer;

public class ServletInitializer extends
SpringBootServletInitializer {

     @Override
     protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
		return 
application.sources(DemoApplication.class);
	}

}
```

DemoApplicationTest.java

```
package com.example.demo;

import org.junit.jupiter.api.Test;
import 
org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class DemoApplicationTests {

	@Test
	void contextLoads() {
	}

}
```

X=========================X===========================X