2.Automate the Web Application

Code:

ANNOTATIONS:

Page:

```
package com.demo.seleniumspring.annotation;

import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import java.lang.annotation.*;

@Lazy
@Component
@Scope("prototype") // to create new instances of bean instead of
sharing; helps during parallel runs
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Page {
}
```

Page Fragment:

```
package com.demo.seleniumspring.annotation;

import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import java.lang.annotation.*;

@Lazy
@Component
@Scope("prototype") // to create new instances of bean instead of
sharing; helps during parallel runs
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface PageFragment {
}
```

CONFIG:

Browser Scope:

```
package com.demo.seleniumspring.config;

import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.SessionId;
import org.springframework.beans.factory.ObjectFactory;
import org.springframework.context.support.SimpleThreadScope;

import java.util.Objects;

// custom scope to prevent multiple browsers from launching
// see video: https://bah.udemy.com/course/cucumber-with-spring-
boot/learn/lecture/20184630#overview
public class BrowserScope extends SimpleThreadScope {

    @Override
    public Object get(String name, ObjectFactory<?> objectFactory) {
        return super.get(name, objectFactory);
        // Object o = super.get(name, objectFactory);
        //
        // SessionId sessionId =
((RemoteWebDriver)o).getSessionId();
        // if (Objects.isNull(sessionId)){
        //     super.remove(name);
        //     super.get(name,objectFactory);
        // }
        // return o;
    }

    @Override
    public void registerDestructionCallback(String name, Runnable
callback) {
    }
}
```

Browser Scope Config:

```
package com.demo.seleniumspring.config;

import
org.springframework.beans.factory.config.BeanFactoryPostProcessor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

// this is how spring boot reads custom scope
@Configuration
public class BrowserScopeConfig {

    @Bean
```

```
    public static BeanFactoryPostProcessor
beanFactoryPostProcessor(){
        return new BrowserScopePostProcessor();
    }
}
```

Browser Scope Post Processer:

```
package com.demo.seleniumspring.config;

import org.springframework.beans.BeansException;
import
org.springframework.beans.factory.config.BeanFactoryPostProcessor;
import
org.springframework.beans.factory.config.ConfigurableListableBeanFac
tory;

// registers new scope created: BrowserScope.java
public class BrowserScopePostProcessor implements
BeanFactoryPostProcessor {
    @Override
    public void
postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory)
throws BeansException {
        beanFactory.registerScope("browserscope", new
BrowserScope());

    }
}
```

Remote Web Driver Config:

```
package com.demo.seleniumspring.config;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnMissin
gBean;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnProper
ty;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Profile;

import java.net.URL;

// for selenium grid run; this will be the remote profile
@Lazy
@Configuration
@Profile("remote") // only activate this class if remote profile
public class RemoteWebDriverConfig {

    @Value("${selenium.grid.url}")
    private URL url;


    @Bean
    @ConditionalOnProperty(name = "browser", havingValue =
"firefox")
    public WebDriver remoteFirefoxDriver(){
        return new RemoteWebDriver(this.url,
DesiredCapabilities.firefox());
    }

    @Bean
    @ConditionalOnMissingBean // to catch invalid browser values
    public WebDriver remoteChromeDriver(){
        return new RemoteWebDriver(this.url,
DesiredCapabilities.chrome());
    }

}
```

Web Driver Config:

```
package com.demo.seleniumspring.config;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnMissin
gBean;
import
org.springframework.boot.autoconfigure.condition.ConditionalOnProper
ty;
import org.springframework.context.annotation.*;

@Configuration
@Profile("!remote") // to avoid loading for remote runs
public class WebDriverConfig {
```

```
    @Bean
    @ConditionalOnProperty(name = "browser", havingValue = "edge")
    public WebDriver edgeDriver() {
        // this is the bean class for edge driver

        if (System.getenv("CLOUD_RUN_FLAG") == null) {
            WebDriverManager.edgedriver().setup();
        }
        return new EdgeDriver();
    }

    @Bean
    // @Primary // this will be the default browser
    @ConditionalOnMissingBean // to catch invalid browser values
    @Scope("browserscope") // use custom scope
    public WebDriver chromeDriver() {
        // this is the bean class for chrome driver

        if (System.getenv("CLOUD_RUN_FLAG") == null) {
            WebDriverManager.chromedriver().setup();
           return new ChromeDriver();
        } else {
            WebDriverManager.chromedriver().setup();
            ChromeOptions options = new ChromeOptions();
            options.addArguments("--no-sandbox");
            options.addArguments("--headless");
            return new ChromeDriver(options = options);
        }
    }
}
```

Web Driver Wait Config:

```
package com.demo.seleniumspring.config;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Lazy;

@Lazy
@Configuration
public class WebDriverWaitConfig {

    @Value("${default.timeout:30}")
    private  int timeout;

    @Bean
    public WebDriverWait webDriverWait(WebDriver driver) {
        return new WebDriverWait(driver, this.timeout);
    }
```

```
    }
```

PAGE:

Google Page:

```
package com.demo.seleniumspring.page.google;

import com.demo.seleniumspring.annotation.Page;
import com.demo.seleniumspring.page.Base;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

// this is the main page class that uses search componet and search
results componet
@Page // using custom annotation created;
src/main/java/com/demo/seleniumspring/annotation/Page.java
public class GooglePage extends Base {

    @Autowired
    private SearchComponent searchComponent;

    @Autowired
    private SearchResult searchResult;

    @Value("${application.url}")
    private String url;

    //launch website
    public void goToGooglePage(){
        this.driver.get(url);
    }

    public SearchComponent getSearchComponent() {
        return searchComponent;
    }

    public SearchResult getSearchResult() {
        return searchResult;
    }

    @Override
    public boolean isAt() {
        return this.searchComponent.isAt();
    }

    public void close(){
        this.driver.quit();
    }
}
```

Search Component:

```
package com.demo.seleniumspring.page.google;

import com.demo.seleniumspring.annotation.PageFragment;
import com.demo.seleniumspring.page.Base;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import java.util.List;

@PageFragment// using custom annotation created;
src/main/java/com/demo/seleniumspring/annotation/PageFragment.java
public class SearchComponent extends Base {

    @FindBy(name = "q")
    private WebElement searchBox;

    @FindBy(name="btnK")
    private List<WebElement> searchBtns;

    public void search(final String keyword) {
        this.searchBox.sendKeys(keyword);
        this.searchBox.sendKeys(Keys.TAB);
        // CLICK first search button
        this.searchBtns
                .stream()
                .filter(e -> e.isDisplayed() && e.isEnabled())
                .findFirst()
                .ifPresent(WebElement::click);
    }

    @Override
    public boolean isAt() {
        return this.wait.until(driver1 ->
this.searchBox.isDisplayed());
    }
}
```

Search Result:

```
package com.demo.seleniumspring.page.google;

import com.demo.seleniumspring.annotation.PageFragment;
import com.demo.seleniumspring.page.Base;
```

```
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

import java.util.List;

@PageFragment// using custom annotation created;
src/main/java/com/demo/seleniumspring/annotation/PageFragment.java
public class SearchResult extends Base {

    @FindBy(css = "div.g")
    private List<WebElement> results;

    public int getCount() {
        return this.results.size();
    }

    @Override
    public boolean isAt() {
        return this.wait.until((d) -> !this.results.isEmpty());
    }
}
```

Base:

```
package com.demo.seleniumspring.page;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.springframework.beans.factory.annotation.Autowired;

import javax.annotation.PostConstruct;

public abstract class Base {

    @Autowired
    protected WebDriver driver;

    @Autowired
    protected WebDriverWait wait;

    @PostConstruct
    private void init(){
        PageFactory.initElements(this.driver, this);
    }

    public abstract boolean isAt();
}
```

UTIL:

ScreenShot Util:

```
package com.demo.seleniumspring.util;
```

```
import org.apache.commons.lang3.ObjectUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Lazy;
import org.springframework.stereotype.Component;
import org.springframework.util.FileCopyUtils;

import java.io.File;
import java.io.IOException;
import java.nio.file.Path;

@Lazy
@Component
public class ScreenShotUtil {

    @Autowired
    private TakesScreenshot driver;

    // location of screenshot file
    @Value("${screenshot.path}")
    private Path path;

    public void takeScreenShot(final String imgName) throws
IOException {
        // takes screenshot as saves to path in app properties file
using given imgName ex. test.png
        if (System.getenv("CLOUD_RUN_FLAG") == null) {
            try {
                File sourceFile =
this.driver.getScreenshotAs(OutputType.FILE);
                FileCopyUtils.copy(sourceFile,
this.path.resolve(imgName).toFile());
                System.out.println("Saving screenshot to " + path);
            } catch (Exception e) {
                System.out.println("Something went wrong with
screenshot capture" + e);
            }
        }

    }
}
```

SELENIUM SPRING APPLICATION:

```
package com.demo.seleniumspring;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
```

```
public class SeleniumSpringApplication {

	public static void main(String[] args) {
		SpringApplication.run(SeleniumSpringApplication.class,
args);
	}

}
```

TEST:

GoogleSearch01 Test:

```
package com.demo.seleniumspring.googletests;

import com.demo.seleniumspring.SpringBaseTestNGTest;
import com.demo.seleniumspring.page.google.GooglePage;
import com.demo.seleniumspring.util.ScreenShotUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.testng.Assert;
import org.testng.annotations.Test;

import java.io.IOException;

public class GoogleSearch1Test extends SpringBaseTestNGTest {

    @Autowired
    private GooglePage googlePage;

    @Lazy // only create the object when needed
    @Autowired
    private ScreenShotUtil screenShotUtil;

    @Test
    public void GoogleTest() throws IOException,
InterruptedException {
        this.googlePage.goToGooglePage();
        Assert.assertTrue(this.googlePage.isAt());

        this.googlePage.getSearchComponent().search("spring boot");
        Assert.assertTrue(this.googlePage.getSearchResult().isAt());

Assert.assertTrue(this.googlePage.getSearchResult().getCount() > 2);
        System.out.println("Number of Results: " +
this.googlePage.getSearchResult().getCount());
        // wait 3 seconds
        // Thread.sleep(3000);
        //take screenshot
        //this.screenShotUtil.takeScreenShot("Test.png");
        //this.googlePage.close();
    }
}
```

GoogleSearch02 Test:

```
package com.demo.seleniumspring.googletests;

import com.demo.seleniumspring.SpringBaseTestNGTest;
import com.demo.seleniumspring.page.google.GooglePage;
import com.demo.seleniumspring.util.ScreenShotUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.testng.Assert;
import org.testng.annotations.Test;

import java.io.IOException;

public class GoogleSearch2Test extends SpringBaseTestNGTest {

    @Autowired
    private GooglePage googlePage;

    @Lazy // only create the object when needed
    @Autowired
    private ScreenShotUtil screenShotUtil;

    @Test
    public void GoogleTest() throws IOException,
InterruptedException {
        this.googlePage.goToGooglePage();
        Assert.assertTrue(this.googlePage.isAt());

        this.googlePage.getSearchComponent().search("Selenium");
        Assert.assertTrue(this.googlePage.getSearchResult().isAt());

Assert.assertTrue(this.googlePage.getSearchResult().getCount() > 2);
        System.out.println("Number of Results: " +
this.googlePage.getSearchResult().getCount());
        // wait 3 seconds
        // Thread.sleep(3000);
        //take screenshot
        //this.screenShotUtil.takeScreenShot("Test.png");
        //this.googlePage.close();
    }
}
```

Selenium Application Test:

```
package com.demo.seleniumspring;

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

@SpringBootTest
class SeleniumSpringApplicationTests {

     @Test
     void contextLoads() {
     }
}
```

Spring Base Test Ng Test:

```
package com.demo.seleniumspring;

import org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.test.context.testng.AbstractTestNGSpringContextT
ests;

@SpringBootTest
public class SpringBaseTestNGTest extends
AbstractTestNGSpringContextTests {
}
```