University of Central Lancashire
UCLan

CO3409 Distributed Enterprise Systems

Lab : Testing

Summary

Develop a web application and test its interface and components.

Purpose

On completion of this you should be able to

- Use NetBeans to create a Maven Web Application
- Test the front end using Selenium
- Unit Test a component in isolation by using Mockito to generate mock objects that mimic the behaviour of objects that haven't yet been written
- Test your tests by using mutation testing
- Use SpotBugs (née Findbugs) to check for potential errors.

Things to Remember

Some things to remember...

- Run Netbeans as "Administrator".
- When running your application, if you are using a database please ensure your database is running first (see previous lab worksheets).
- To deploy an Enterprise Application, I suggest using Clean and Build and then Deploy.

Activities

1. Creating a Web Application

Create a new Maven web application called palindrome to run on Payara, JavaEE 7. Use the **Group ID** com.uclan.

1. Modify the `index.html` to :

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title>Palindrome Checker</title>
</head>
<body>
<h3>Please enter a palindrome</h3>
<form  action="result.jsp" method="post">
<table>
  <tr>
    <th>Potential Palindrome</th>
    <th><input name="palindrome" type="text" /></th>
  </tr>
  <tr>
    <td colspan="2" align="right"><input type="submit" value="Check"/>
    </td>
  </tr>
</table>
</form>
</body>
```

2. Create a new standard JSP page called `result.jsp` and modify it to:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ page import="com.uclan.palindrome.*" %>
<%@ page import="java.util.*" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Result</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%
            System.out.println("Evaluating date now");
            Date date = new Date();
            PalinStore st = new PalinStore();
            Palin p = new Palin(st);
            String param = request.getParameter("palindrome");
            boolean isPalin = p.isPalindrome(param);
        %>
```

```
        Hello!  The time is now <%= date%>
        <%= param %> is palindrome? <%= isPalin %>
        <a href="index.html">Back</a>
    </body>
</html>
```

3. To allow the demonstration of behaviour testing in mocking, we are going to provide a class to store palindromes. Create the following class in Source Packages in the package `com.uclan.palindrome`

```
public class PalinStore {

    List<String> l ;

    public void initialise(){
        l = new ArrayList<String>();
    }

    public void Add(String s) {
        String half = s.substring(0, s.length() / 2);
        if (!l.contains(half)) {
            l.add(half);
        }
    }

    public boolean isIn(String s) {
        String half = s.substring(0, s.length() / 2);
        return l.contains(half);
    }

    public List GetAll() {
        List<String> res = new ArrayList<>();
        for (String s : l) {
            res.add(s + s);
        }
        return res;
    }
}
```

4. It would be better for the class that uses this to access it through an interface. Right-click on the PalinStore class and choose **Refactor**, **Extract Interface**. Call the interface, `IStore`. Create a class to recognise palindromes:

```
public class Palin {
```

```
  IStore p;

  public Palin (IStore p){
    this.p = p;
    p.initialise();
  }

  public boolean isNewPalindrome(String inputString) {
    if (isPalindrome(inputString)) {
      return !p.isIn(inputString);
    } else {
      return false;
    }
  }

  public boolean isPalindrome(String inputString) {
    if (inputString.length() == 0) {
      return true;
    } else {
      char firstChar = inputString.charAt(0);
      char lastChar = inputString.charAt(inputString.length() - 1);
      String mid = inputString.substring(1, inputString.length() - 1);
      return (firstChar == lastChar) && isPalindrome(mid);
    }
  }
}
```

2. Testing the front end with Selenium

Selenium is a tool that can be used for “remote control” of a browser. It can also extract information from the browser – hence it can help automate testing the user interface. There are two components: the web-driver, which allows a java program to control a browser and the IDE, which allows the recording and playback of the interaction with a browser. It will also export the recording to java.

1. Right-click on the project name, New, Selenium Tests, New Selenium Test Case. Call it ITUITest in **Package**, test. NetBeans will initially show that the Selenium packages are missing. It will eventually download the packages and modify pom.xml.

   Update the version of selenium to the latest.

   > You can update versions in a pom.xml by deleting from the end to the first number, then entering a fullstop and waiting for the autocomplete, then delete all version. Type the highest digit that has an autocompletion.

2. Because the NetBeans-generated code is out of date, it doesn't add a recently required component. Add the following dependency to `pom.xml`

```
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>2.21</version>
  </dependency>
```

3. Geckodriver is a component that interfaces between the Selenium core and FireFox. Download the appropriate version of geckodriver from https://github.com/mozilla/geckodriver/releases, extract `geckodriver.exe` and put it in an appropriate folder.

4. Just before creating the WebDriver object in ITUITest, add

   `System.setProperty("webdriver.gecko.driver", "<PATHTODRIVER>/geckodriver.exe");`

   Where `<PATHTODRIVER>` is the Windows path, with `\` rather than `.`.

5. Set the contents of `pom.xml` to:

```
<?xml version="1.0" encoding="UTF-8"?>
<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.uclan</groupId>
  <artifactId>palindrome</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>palindrome</name>

  <properties>
    <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <scope>test</scope>
```

```
    <version>3.141.59</version>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>2.35.1</version>
  </dependency>
  <dependency>
    <groupId>com.opera</groupId>
    <artifactId>operadriver</artifactId>
    <scope>test</scope>
    <version>1.5</version>
    <exclusions>
      <exclusion>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-remote-driver</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <scope>test</scope>
    <version>4.11</version>
  </dependency>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>7.0</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.pitest</groupId>
    <artifactId>pitest-parent</artifactId>
    <version>1.4.7</version>
    <type>pom</type>
  </dependency>
  <!-- Dependency for Mockito -->
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-all</artifactId>
    <version>1.10.19</version>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>com.github.spotbugs</groupId>
    <artifactId>spotbugs</artifactId>
    <version>3.1.12</version>
```

```
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.pitest</groupId>
        <artifactId>pitest-maven</artifactId>
        <version>1.4.7</version>
        <configuration>
          <targetClasses>
            <param>com.uclan.palindrome.Palin</param>
          </targetClasses>
          <targetTests>
            <param>test.PalinTest</param>
          </targetTests>
        </configuration>
        <executions>
          <execution>
            <id>pit-report</id>
            <!-- optional, this example attached the goal into mvn test
phase -->
            <phase>test</phase>
            <goals>
              <goal>mutationCoverage</goal>
            </goals>
          </execution>
        </executions>

      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
          <compilerArguments>
            <endorseddirs>${endorsed.dir}</endorseddirs>
          </compilerArguments>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.2</version>
        <configuration>
          <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
```

```
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-dependency-plugin</artifactId>
        <version>3.1.1</version>
        <executions>
          <execution>
            <phase>validate</phase>
            <goals>
              <goal>copy</goal>
            </goals>
            <configuration>
              <outputDirectory>${endorsed.dir}</outputDirectory>
              <silent>true</silent>
              <artifactItems>
                <artifactItem>
                  <groupId>javax</groupId>
                  <artifactId>javaee-endorsed-api</artifactId>
                  <version>7.0</version>
                  <type>jar</type>
                </artifactItem>
              </artifactItems>
            </configuration>
          </execution>
        </executions>
      </plugin>

      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M3</version>
        <configuration>
          <excludes>
            <exclude>**/IT*.java</exclude>
          </excludes>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-site-plugin</artifactId>
        <version>3.7.1</version>

      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>2.9</version>
        <dependencies>
          <dependency>
```

```
            <groupId>org.apache.maven.shared</groupId>
            <artifactId>maven-shared-jar</artifactId>
            <version>1.2</version>
            <exclusions>
              <exclusion>
                <groupId>com.google.code.findbugs</groupId>
                <artifactId>bcel-findbugs</artifactId>
              </exclusion>
            </exclusions>
          </dependency>
          <dependency>
            <groupId>org.apache.bcel</groupId>
            <artifactId>bcel</artifactId>
            <version>6.2</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
  </build>
  <reporting>
    <plugins>

      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>2.9</version>

      </plugin>
      <plugin>
        <groupId>com.github.spotbugs</groupId>
        <artifactId>spotbugs-maven-plugin</artifactId>
        <version>3.1.11</version>
         <configuration>
        <effort>Max</effort>
        <threshold>Low</threshold>
         </configuration>
      </plugin>
    </plugins>
  </reporting>
</project>
```

6. Modify the generated test method to visit http://localhost:8080/palindrome/index.html instead of NetBeans.

7. Modify the test to check for the correct title.

8. Install the latest version of FireFox.

9. Run the project, which will deploy it to the server.

10. Run the test by right-clicking on `ITUITest.java` in the **Projects** tab and choosing **Debug Test File**.

11. Install the latest version of Selenium IDE in Firefox.

    > See https://www.seleniumhq.org/selenium-ide/docs/en/introduction/getting-started/

12. Run Firefox and launch the Selenium IDE by clicking on the toolbar button.

13. Select **Record a new test in a new project**. Call the project `SeleniumTest.` Set the `BASE URL` to the URL of the palindrome application (e.g. http://localhost:8080/palindrome/index.html).

14. Click on the text box and enter the string, “abba”, Click on the Check button. When the result page is displayed, click on the Back button.

15. Click again on the toolbar button. When the IDE is displayed, click on the **stop recording** button. Enter the TEST NAME, palin. Click on OK.

16. The recorded sequence of commands is displayed. Select the command that corresponds to clicking on the Check button. The command should be displayed in the Command/Target/Value section. Click on the Target dropdown, and select the xpath identification of the button (xpath=//input[@value='Check']

17. If there is a command to set the window size, select it, click on the vertical dots to the right, and select **Delete**.

18. Save the project in a suitably named file in a suitably named folder.

19. Select the test, palin. Click on the three dots, and choose Export. Give the java file an appropriate name.

20. Open the java file. Use the recorded test to modify `ITUITest.java` so it does the following:
    - Navigate to the application home URL
    - Check that the title contains “Palindrome Checker”

You can already see how to wait for the title to be set: `assert (driver.getTitle().contains("Palindrome Checker"));`

- will check that that the title is correct.
- Clicks on the text-entry box.
- Enters the text “abba”
- Clicks on the Check button.
- Waits for up to 30 seconds for the title to become “Result”
- Check that the result is correct:

```
List<WebElement> list =
driver.findElements(By.xpath("//p[contains(.,'abba is palindrome?
true')]"));
Assert.assertTrue("Incorrect result", list.size() == 1);
```

- Click on the Back link
- Wait for the title to change and check that it has the correct value.
- Fix any imports.

21. Make sure the application is running. Right-click on the `ITUITest.java` and Test File to run the test. Make sure that the test and the application are correct

3. Testing a component with Unit and Mockito

Mockito is a tool that allows the creation of objects that simulate (mock-up) the behaviour of other classes. Mock objects can create an environment to allow the testing of other objects. This allows unit testing of an object isolated from other objects that have not yet been written.

1. Right-click on the Test Packages folder and choose **New**, **Unit Tests**, **Test for Existing Class**. Browse for the class, Palin, check all the options and create the test. Replace the class with the following:

```
@RunWith(MockitoJUnitRunner.class)
public class PalinTest {

  private static IStore p;
  private static String s1;
  private static String s2;

  public PalinTest() {
  }
```

```
@BeforeClass
public static void setUpClass() {
}

@AfterClass
public static void tearDownClass() {
}

@Before
public void setUp() {
  //Create mock object of PalinStore
  p = mock(IStore.class);

  //Create few instances of Book class.
  s1 = "abba";

  s2 = "madamImadam";

  //Stubbing the methods of mocked store with mocked data.
  when(p.GetAll()).thenReturn(Arrays.asList(s1, s2));
  when(p.isIn(s1)).thenReturn(true);
  when(p.isIn("cooc")).thenReturn(false);
  when(p.isIn(s2)).thenReturn(true);
}

@After
public void tearDown() {
}

/**
 * Test of isPalindrome method, of class Palin.
 */
@Test
public void testIsPalindrome() {
  System.out.println("isPalindrome");
  String inputString = "abaraccaraba";
  Palin instance = new Palin(p);
  boolean expResult = true;
  boolean result = instance.isPalindrome(inputString);
  assertEquals(expResult, result);
}

@Test
public void testPalinStore() {
  System.out.println("Palindrome Store Test");
  String inputString = "abba";
  Palin instance = new Palin(p);
  boolean expResult = false;
  boolean result = instance.isNewPalindrome(inputString);
```

```
    assertEquals("abba is not a new result", expResult, result);
    expResult = true;
    result = instance.isNewPalindrome("cooc");
    assertEquals("cooc is a new result", expResult, result);
  }

  @Test
  public void testPalinStoreOrder() {
    System.out.println("Palindrome Order Test");
    //create an inOrder verifier for a single mock
    InOrder inOrder = inOrder(p);
    ArgumentCaptor<String> argument =
ArgumentCaptor.forClass(String.class);

    String inputString = "cooc";
    Palin instance = new Palin(p);
    boolean expResult = false;
    boolean result = instance.isNewPalindrome(inputString);
    assertEquals("abba is not a new result", expResult, result);
    expResult = true;
    result = instance.isNewPalindrome("abba");
    assertEquals("cooc is a new result", expResult, result);
    //following will make sure that initialise is called first then isIn is
called with arguments “abba” and “cooc” in that order.
    inOrder.verify(p).initialise();
    inOrder.verify(p, atLeast(0)).isIn(argument.capture());
    List expected = asList("abba", "cooc");
    assertEquals(expected, argument.getAllValues());
  }
}
```

2. Run the test by right-clicking on the palindrome project in the Projects tab and choosing Test. Fix the program so it passes the tests as shown in the **Test Results** tab.

4. Reviewing your code with SpotBugs

Spotbugs analyses the bytecode to seek a variety of problems such as poor programming style or potential use of uninitialized variables. This is static analysis.

> For more information, see
>
> https://spotbugs.github.io/
>
> https://spotbugs.github.io/spotbugs-maven-plugin/usage.html

1. Right-click on the palindrome project in the Projects tab and choose **Run Maven**. Enter the **Goal**, site, and run Maven goal.

2. Click on the link shown in the resultant output file to **View Generated Project Site** on the application. Click on **Project Reports** and **SpotBugs**. Fix the problems identified in the report.

5. Checking your tests with PIT

PIT checks the amount (coverage) of the source code that the unit tests have exercised. It goes further by making changes to the source code and running the unit tests against these mutants. Given that mutated programs would be expected to have bugs, if they pass the unit tests, it is likely that the unit tests aren't rigorous enough.

Use Windows Explorer to open your project folder (`<Path to project folder>\palindrome`, open the folder target\pit-reports. Open the latest folder, and `index.html`. Look at the coverage reported by PIT. The existing tests did not kill all the mutants, which probably means the tests are not thorough enough.

Modify the tests to ensure the mutants are killed – or that they are actually the same as the correct program. When you strengthen the tests, you may find that your `isPalindrome()` or `isNewPalindrome()` functions don't pass the tests, indicating that there are bugs that you should fix.

6. Advanced Challenge

See how good you are... write some unit tests for `public boolean isPrime(int n);`

7. Questions: Have You Understood?

(Note: I wouldn't be surprised if you had to do some research to answer questions marked `*`

1. What does `<%@ page import="java.util.*" %>` do?
2. What does `String param = request.getParameter("palindrome");` do?
3. What feature of the way that PalinStore is used would you test by behaviour testing?
4. Can you spot any features of PalinStore that is not in the normal Java style?
5. Why is it better for classes using PalinStore to access it through a variable of an interface type rather than a variable of PalinStore type?
6. Can you spot any bugs by looking at the code for Palin?
7. What is a POM?
8. Why is it useful to automate testing?
9. What is the alternative to recording a user interface test with Selenium IDE

10. Does Selenium test the quality of the user interface?
11. Why do you need a “wait” feature in Selenium?
12. Selenium interacts with the DOM of a page. What is the DOM?
13. What is a mock object?
14. Why use Mockito rather than implementing the mock objects yourself?
15. What is a test runner?
16. What test runner is used for these tests? How do you know?
17. Why is the mock object defined in setUp()?
18. `*` What is a Maven goal?
19. Explain what is meant by the statement “100% line coverage is a necessary, but not sufficient to guarantee thorough testing”.
20. What is the difference between 100% line coverage and 100% path coverage?
21. Explain the theory behind mutation testing.