

Criação de um projeto Maven no Netbeans com utilização de plugin de cobertura de testes

Luiz Faria

19 de Fevereiro de 2018

1. Criação do projeto Maven

Vamos começar por criar um projeto Maven no Netbeans.

- File -> New Project ...
- Em Categories escolher Maven e em Projects, escolher Java Application
- Introduzir o nome do projeto

Vamos agora declarar uma classe que será usada como exemplo (Palindroma.java):

```
package org.dei.palindroma;  
  
public class Palindroma {  
  
    public boolean isPalindroma(String inputString) {  
        if (inputString.length() <= 1) {  
            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) && isPalindroma(mid);  
        }  
    }  
}
```

2. Preparação do plugin de cobertura de testes JaCoCo

Para utilizar o plugin de cobertura de testes JaCoCo (<http://www.eclemma.org/jacoco/trunk/doc/maven.html>), é necessário acrescentar no ficheiro pom.xml (existente na pasta Project Files) o seguinte elemento <build>:

```
<build>  
    <plugins>  
        <plugin>
```

```

        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.7.7.201606060606</version>
        <executions>
            <execution>
                <goals>
                    <goal>prepare-agent</goal>
                </goals>
            </execution>
            <execution>
                <id>report</id>
                <phase>prepare-package</phase>
                <goals>
                    <goal>report</goal>
                </goals>
            </execution>
        </executions>
    </plugin>
</plugins>
</build>

```

3. Criação de um teste

Vamos criar uma classe de teste para a classe Palindroma.java de forma automática:

- No explorador de projectos seleccionar a classe Palindroma.java com o botão direito do rato e seleccionar Tools -> Create/Update Tests
- Na pasta Test Packages foi criada a classe PalindromaTest.java que inclui já o seguinte método para teste do método isPalindroma():

```

/**
 * Test of isPalindroma method, of class Palindroma.
 */
@Test
public void testIsPalindroma() {
    System.out.println("isPalindroma");
    String inputString = "";
    Palindroma instance = new Palindroma();
    boolean expResult = false;
    boolean result = instance.isPalindroma(inputString);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to
    fail("The test case is a prototype.");
}

```

- Vamos alterar este método de teste de forma a testar o método isPalindroma() para apenas palindromas vazios:

```

/**

```

```

        * Test of isPalindroma method, of class Palindroma.
        */
@Test
public void testIsPalindroma() {
    System.out.println("isPalindroma_vazio");
    String inputString = "";
    Palindroma instance = new Palindroma();
    boolean expResult = true;
    boolean result = instance.isPalindroma(inputString);
    assertEquals(expResult, result);
}

```

4. Execução do teste e visualização do teste de cobertura

Vamos agora proceder à execução do teste.

- No Projects explorer, seleccionar o projeto com o botão direito do rato e seleccionar Code Coverage -> Show Report...
- Pressionar o botão Run All Tests no fundo da janela
- É apresentado o resultado de cobertura de teste. Como seria de esperar a cobertura não é total. Na mesma janela podemos seleccionar o ficheiro Palindroma, sendo apresentado o código da classe Palindroma. Nesta listagem, as linhas de código não cobertas pelo teste aparecem a vermelho:

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package org.dei.palindroma2;

/**
 *
 * @author luizfaria
 */
public class Palindroma {
    public boolean isPalindroma(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) && isPalindroma(mid);
        }
    }
}

```

5. Adição de um novo teste

Vamos agora adicionar um novo teste de modo a testar o comportamento do método isPalindroma() com outros palindromas que não sejam vazios.


Na classe de teste vamos adicionar um novo método de teste na classe PalindromaTest.java:

```
@Test
public void testIsPalindromaNotEmpty() {
    System.out.println("isPalindroma_nao_vazio");
    String inputString = "noon";
    Palindroma instance = new Palindroma();
    boolean expResult = true;
    boolean result = instance.isPalindroma(inputString);
    assertEquals(expResult, result);
}
```

6. Execução dos testes

Por fim vamos voltar a executar os testes e consultar o novo relatório de cobertura. Agora todo o código está coberto pelos testes, como ilustra a figura:

Total Coverage: 100.00 %

Filename	Coverage	Total	Not Executed
 org.dei.palindroma2.Palindroma	100.00 %	7	0
Total	100.00 %	7	0

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package org.dei.palindroma2;

/**
 *
 * @author luizfaria
 */
public class Palindroma {
    public boolean isPalindroma(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) && isPalindroma(mid);
        }
    }
}
```