Code Coverage

Paolo Aronne

Agenda

introduction to code coverage

code coverage and continuous integration

code complexity

Testing increases your business value

Reliability

Quality

Customer satisfaction

Automated Testing

Keep your code free of regressions

Correct immediately code errors

Adapt continuous build and continuous integration strategies

Use latest Servoy release

How much are you confident for the next release ?

- “Testing is in place and all my tests are successful !!!”
- “Good ! There are no regressions ? Are all functionalities working properly ?”
- “Well it is a pretty big product... I cannot say for sure”
- “Can you measure how much of the product has been tested ?”
- “How can I measure that !?”

Code Coverage

Measures the degree of how much of the source code has been tested

Highlight which part of your code has been tested

Code Coverage

What does code coverage highlights

- Lines
- Statements
- Branches (Condition)
- Functions

How does code coverage works

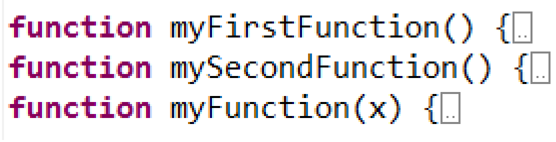
- Instrument source code: inject code in order to collect stats on which part of the code is executed
- Run tests and collect coverage stats
- Output collected stats data and generate analysis report

Code Coverage

A simple Use Case

Source code

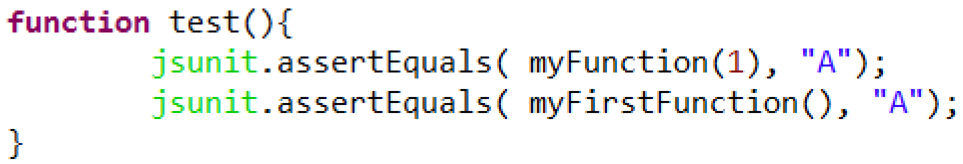
- app.js



```
function myFirstFunction() {□
function mySecondFunction() {□
function myFunction(x) {□
```

Unit test

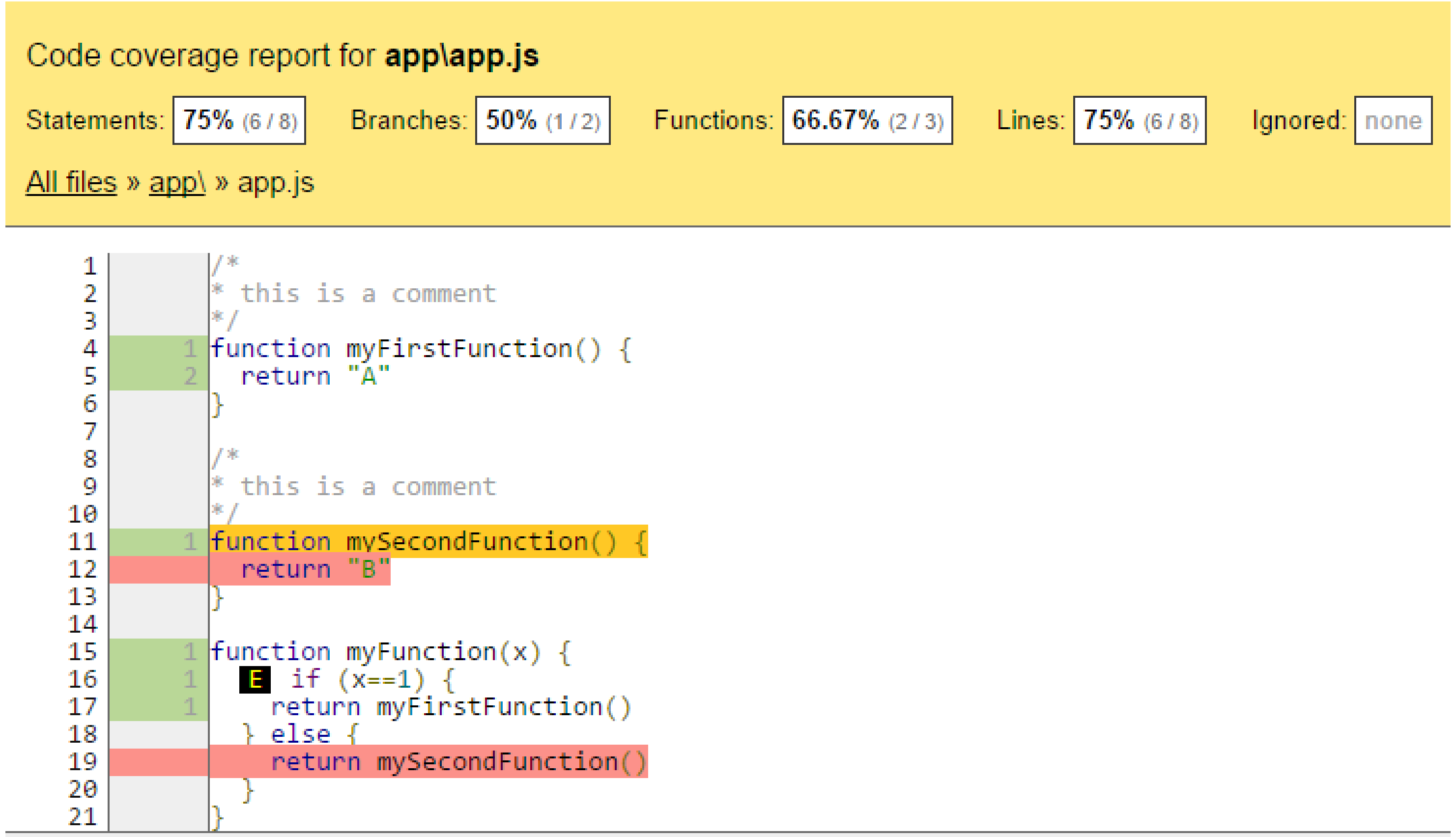
- app_test.js



```
function test(){
        jsunit.assertEquals( myFunction(1), "A");
        jsunit.assertEquals( myFirstFunction(), "A");
}
```

Coverage Report

- executed lines (green)
- not executed line (red)
- function not executed
- branches



Code coverage report for **app\app.js**

Statements: **75%** (6 / 8) Branches: **50%** (1 / 2) Functions: **66.67%** (2 / 3) Lines: **75%** (6 / 8) Ignored: none

All files » app\ » app.js

```
/*
* this is a comment
*/
function myFirstFunction() {
  return "A"
}

/*
* this is a comment
*/
function mySecondFunction() {
  return "B"
}

function myFunction(x) {
  E if (x==1) {
    return myFirstFunction()
  } else {
    return mySecondFunction()
  }
}
```

Code Coverage

- Istanbul
  - Javascript code coverage tool
  - Nodejs module
  - HTML reporting

- Use Istanbul as code coverage tool for Servoy
  - Instrument the solution in workspace
  - Parse the instrumented files
  - Use the onClose Event of the test solution to store the coverage results in a . .json file
  - Generate the HTML coverage report from the .json file

  The whole process can be automated !

Use Jenkins as Continuous Build Software

- Commits are built
- Automatically test builds
- Publish results

Configure Jenkins to integrate Code Coverage

Customized build configuration available on GitHub at:

https://github.com/Servoy/svyJenkinsConfig

Detailed guide available in the wiki at:
https://wiki.servoy.com/display/DOCS/Using+Istanbul+to+integrate+code+coverage+report+in+Jenkins

Code Coverage

Demo

Some thoughts

Servoy Javascript uses Rhino, Istanbul would not be able to istrument:

- Plain XML code
- try {} catch (e if e instanceof ....) {}
- for each loops

Don't try to reach 100% coverage

Trade off between uncovered code and development effort

80% coverage is good !

Code Complexity

Code maintenance

- Correct existing defects
- Improve performance
- Adapt to new requirements

Code maintenance usually takes 40% up to 80% of total project cost

Code complexity

- Reduce code complexity to improve code maintainability and readability
- Use tools for code complexity analysis

Plato

- Tool for javascript code complexity analysis
- Nodejs module
- HTML interactive reporting

Use Plato as code analysis tool for Servoy

- run plato and target the Servoy solution directory in workspace
- integrate Plato with Jenkins using svyJenkinsConfig
- see detailed guide at the wiki page: https://wiki.servoy.com/display/DOCS/Using+Plato+to+integrate+code+analysis+report+in+Jenkins

Code Complexity

Now:

Next Generation Cloud

5:20 pm

Software Licensing Trends in 2014 and what they mean for you