

UI Test Capture Plugin

Plugin Information

View UI Test Capture [on the plugin site](#) for more information.

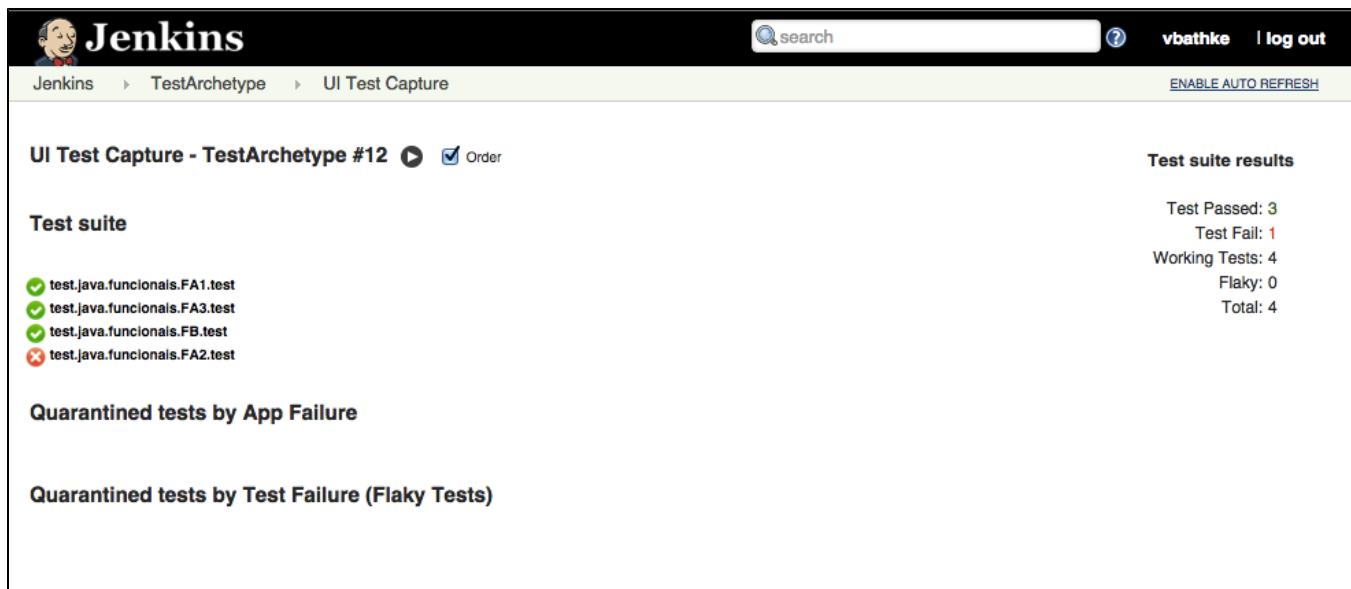
Summary

Stream, persist and classify UI Test Results from a running Build.

The plugin is agnostic about what test engine you use if the required informations was provided.

Features

See the results while they conclude



The screenshot displays the Jenkins web interface for a build named 'UI Test Capture - TestArchetype #12'. The top navigation bar includes the Jenkins logo, a search bar, and user information 'vbathke | log out'. The breadcrumb trail shows 'Jenkins > TestArchetype > UI Test Capture'. A link 'ENABLE AUTO REFRESH' is visible in the top right. The main content area shows the test suite results:

- Test suite**
 - test.java.funcionalis.FA1.test (Passed)
 - test.java.funcionalis.FA3.test (Passed)
 - test.java.funcionalis.FB.test (Passed)
 - test.java.funcionalis.FA2.test (Failed)
- Test suite results**
 - Test Passed: 3
 - Test Fail: 1
 - Working Tests: 4
 - Flaky: 0
 - Total: 4
- Quarantined tests by App Failure**
- Quarantined tests by Test Failure (Flaky Tests)**

If some test fail you can look at its screenshot and Build History

Test suite

- ✔ test.java.funcionais.FA1.test
- ✘ test.java.funcionais.FA2.test

Test suite results

Test Passed: 3
 Test Fail: 1
 Working Tests: 4
 Flaky: 0
 Total: 4

Screenshot:



Stacktrace:

```
-----
Test set: test.java.funcionais.FA2
-----
Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 20.886
sec <<< FAILURE! - in test.java.funcionais.FA2
test(test.java.funcionais.FA2) Time elapsed: 20.886 sec <<< FAILURE!
java.lang.AssertionError: null
    at org.junit.Assert.fail(Assert.java:86)
    at org.junit.Assert.assertTrue(Assert.java:41)
    at org.junit.Assert.assertTrue(Assert.java:52)
    at test.java.funcionais.FA2.test(FA2.java:19)
```

Build history: 12: ✘, 11: ✔, 10: ✔, 8: ✔, 7: ✘, 6: ✘, 5: ✘, 3: ✘, 1: ✘

Move to quarantine

- ✔ test.java.funcionais.FA3.test
- ✔ test.java.funcionais.FB.test



If some tests are Flaky and you want to separate from other tests you can classify them in "App failure" and "Test failure"

Jenkins > TestArchetype > UI Test Capture ENABLE AUTO REFRESH


Test suite

- ✔ test.java.funcionais.FA1.test
- ✔ test.java.funcionais.FA3.test
- ✔ test.java.funcionais.FB.test
- ✘ test.java.funcionais.FA2.test

Test suite results

Test Passed: 3
 Test Fail: 1
 Working Tests: 4
 Flaky: 0
 Total: 4

Screenshot:



Stacktrace:

```
-----
Test set: test.java.funcionais.FA2
-----

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 20.886
sec <<< FAILURE! - in test.java.funcionais.FA2
test(test.java.funcionais.FA2) Time elapsed: 20.886 sec <<< FAILURE!
java.lang.AssertionError: null
    at org.junit.Assert.fail(Assert.java:86)
    at org.junit.Assert.assertTrue(Assert.java:41)
    at org.junit.Assert.assertTrue(Assert.java:52)
    at test.java.funcionais.FA2.test(FA2.java:19)
```

0: ✔, 8: ✔, 7: ✘, 6: ✘, 5: ✘, 3: ✘, 1: ✘


- Move to quarantine
- App Failure
- Test Failure
- Remove from quarantine

Write coments about tests to facilitate maintenance

Jenkins > TestArchetype > UI Test Capture ENABLE AUTO REFRESH

test.java.funcionais.FA2.test

Screenshot:



The screenshot shows a Google search results page for the name 'Victor Augusto Bathe'. It lists various search results including profiles, resumes, and company information. At the bottom of the screenshot area, there is a 'Save' button.

Stacktrace:

```

-----
Test set: test.java.funcionais.FA2
-----
Test Passed: 3
Test Fail: 0
Working Tests: 3
Flaky: 1
Total: 4

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 20.886 sec <<< FAILURE! - in test.java.funcionais.FA2
test(test.java.funcionais.FA2) Time elapsed: 20.886 sec <<< FAILURE!
java.lang.AssertionError: null
    at org.junit.Assert.fail(Assert.java:86)
    at org.junit.Assert.assertTrue(Assert.java:41)
    at org.junit.Assert.assertTrue(Assert.java:52)
    at test.java.funcionais.FA2.test(FA2.java:19)

```

Description:

Test failing because there is a dynamic selector

Configuration

1. Implement the instructions to write the screenshot of the test method before the test close with the following filename convention:

```
target/screenshots/[Test Method].png
```

Example in Selenium with Java:

```
String actualTest = this.getClass().getName()+"."+testname.getMethodName();
File scrFile = ((TakesScreenshot)FabricaWebDriver.getDriver()).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("target/screenshots/"+actualTest+".png"));
```

2. Also before the test finish, append the test result informations in a text file located on "target /teststream.txt" with the following format:

```
{ "metodo": "[Test Method]", "status": "[Test Result]", "classe": "[Test Class]", "descricao": "[Display Test Method]" }
```

Example in Selenium with Java:

```

// Example code for writing to teststream.txt
try {
    FileWriter writer = new FileWriter("target/teststream.txt", true);
    writer.write("{\"metodo\": \"\", \"status\": \"\", \"classe\": \"\", \"descricao\": \"\"}");
    writer.close();
} catch (IOException e) {
    e.printStackTrace();
}

```

```

@Rule
public TestWatcher watcher = new TestWatcher() {    @Override
protected void failed(Throwable e, Description description) {
    status= "falha";
}
@Override
protected void skipped(AssumptionViolatedException e, Description description) {
    status= "skipped";
}
@Override
protected void succeeded(Description description) {
    status= "sucesso";
}
@Override
protected void finished(Description d) {
    try {
        PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("target/teststream.txt",
true)));
        out.println("{\"metodo\": \""+actualTestMethod+"\", \"status\": \""+status+"\", \"
+ \"\"classe\": \""+actualTestClass+"\", \"descricao\": \""+actualTestMethod+"\"}");
        out.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
};

```

3. Jenkins Configuration

1. On the Job configuration active this plugin with the action "UI Test Capture" on Post-build Action
2. Add the action 'Arquive Artifacts' on Post-build Action with the value:

```
target/screenshots/**/*
```

Changelog

Release 1.0.41 (25 October 2015)

- First public release