GR8 FESTivities: Automated UI Testing of Griffon Applications with Groovy, FEST & Jenkins

Nick Bartlow, Ph.D.

Noblis

http://www.noblis.org/

@jenkinsconf

Background / Motivation

- Building a Griffon Application?
- Need to test your UI and like FEST?
- Warning: As Andres Almiray says in Griffon in Action, UI testing is not for the faint of heart…
- Disclaimers aside, here's how you can create a GR8 FESTival in Jenkins!

GRIFFON

"Griffon is an application framework for developing desktop applications in the JVM, with Groovy being the primary language of choice."

FEST Fixtures for Easy Software Testing

"FEST is a collection of libraries, released under the Apache 2.0 license, whose mission is to simplify software testing. It is composed of various modules, which can be used with TestNG or Junit."

Required Tooling

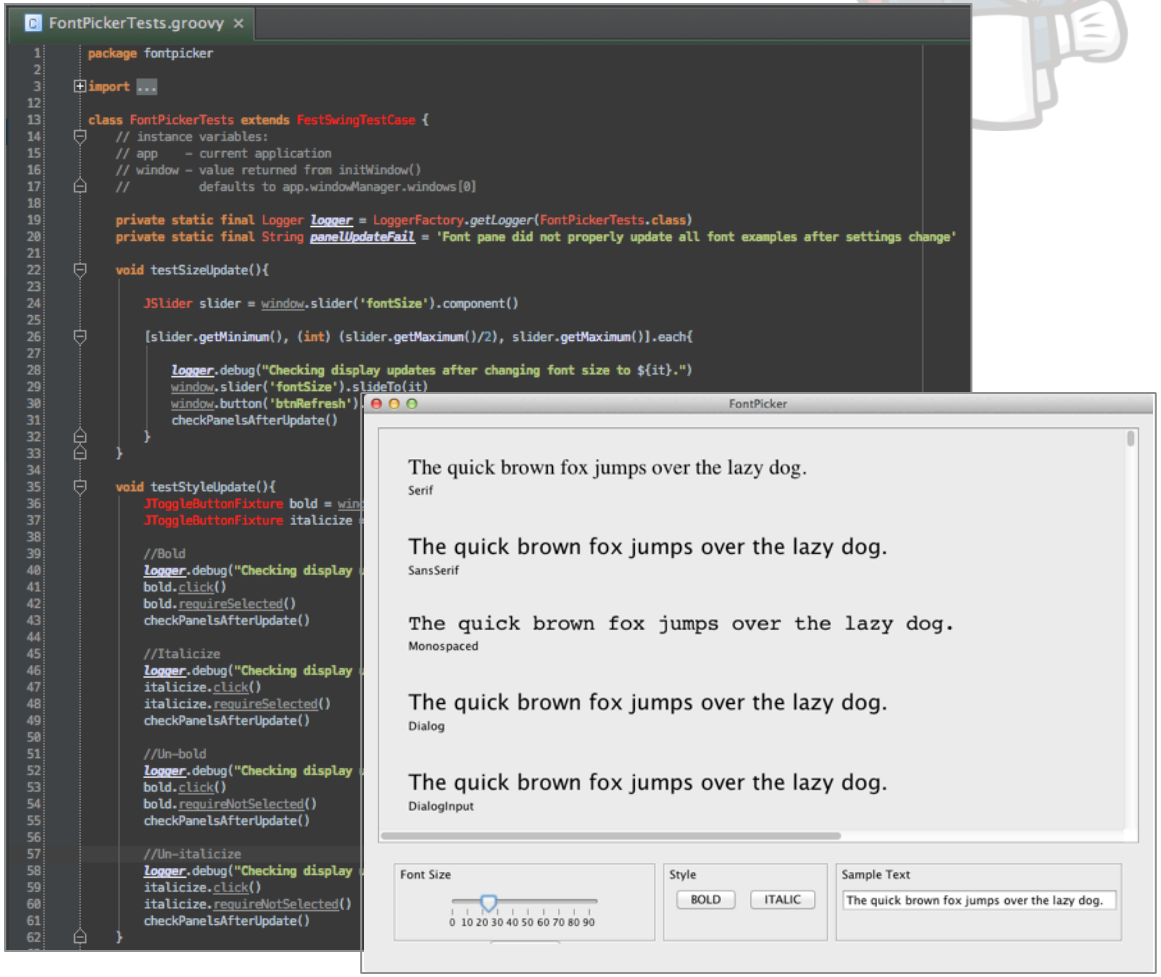
- Provided Scripts
  - GriffonFestRunner.groovy
    - Runs Griffon FEST tests tailored for headless execution
  - VideoEdit.groovy
    - Removes boring parts of recorded videos
  - FESTival.sh
    - Wraps GriffonFestRunner.groovy and VideoEdit.groovy, records test videos and splits video per test removing idle parts
- Groovy enVironment Manager (GVM)
  - Install and manage Groovy and Griffon
- Groovy
  - Used to run provided scripts
- Griffon
  - Runs FEST test for Griffon application in question
- vnc2flv
  - Record test videos in .flv format
- FFmpeg
  - Used by VideoEdit to split videos
- VLC Media Player
  - Playback of test videos
- Ant
  - Used to assemble test reports

GVM
the Groovy enVironment Manager

Test a Griffon Sample App

- FontPicker
  - Nothing fancy, simple to deal with
- We'll ensure that interactions with
  - Font Size
  - Style
  - Sample Text

  Appropriately update the main panel



FontPickerTests.groovy

```
package fontpicker

import ...

class FontPickerTests extends FestSwingTestCase {
    // instance variables:
    // app    - current application
    // window - value returned from initWindow()
    //          defaults to app.windowManager.windows[0]

    private static final Logger logger = LoggerFactory.getLogger(FontPickerTests.class)
    private static final String panelUpdateFail = 'Font pane did not properly update all font examples after settings change'

    void testSizeUpdate(){

        JSlider slider = window.slider('fontSize').component()

        [slider.getMinimum(), (int) (slider.getMaximum()/2), slider.getMaximum()].each{

            logger.debug("Checking display updates after changing font size to ${it}.")
            window.slider('fontSize').slideTo(it)
            window.button('btnRefresh')
            checkPanelsAfterUpdate()
        }
    }

    void testStyleUpdate(){
        JToggleButtonFixture bold = win
        JToggleButtonFixture italicize

        //Bold
        logger.debug("Checking display
        bold.click()
        bold.requireSelected()
        checkPanelsAfterUpdate()

        //Italicize
        logger.debug("Checking display
        italicize.click()
        italicize.requireSelected()
        checkPanelsAfterUpdate()

        //Un-bold
        logger.debug("Checking display
        bold.click()
        bold.requireNotSelected()
        checkPanelsAfterUpdate()

        //Un-italicize
        logger.debug("Checking display
        italicize.click()
        italicize.requireNotSelected()
        checkPanelsAfterUpdate()
    }
```

FontPicker

The quick brown fox jumps over the lazy dog.
Serif

The quick brown fox jumps over the lazy dog.
SansSerif

The quick brown fox jumps over the lazy dog.
Monospaced

The quick brown fox jumps over the lazy dog.
Dialog

The quick brown fox jumps over the lazy dog.
DialogInput

Font Size

Style

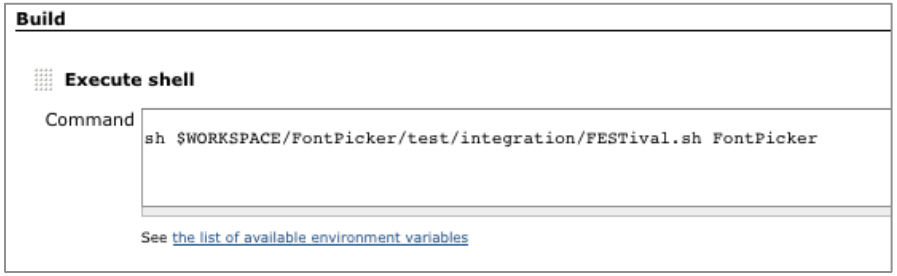
Sample Text: The quick brown fox jumps over the lazy dog.

Set Things Up

- Create Jenkins slave(s) with the tooling in place
  - In the interests of time, see set up steps included with provided scripts
- Make sure your repository is in place
  - FontPicker is available with any griffon installation, (we've made a few edits to accommodate FEST that are available)
- Create the Jenkins job
  - (next slide)

Create the Jenkins Job

- Standard stuff
  - Set up your workspace FontPicker is available in any Griffon installation (we've made a few edits to accommodate FEST that are available)
  - Set up your build trigger(s)
- Environment – More on this later
- The Build Step – Run your FESTival



Build

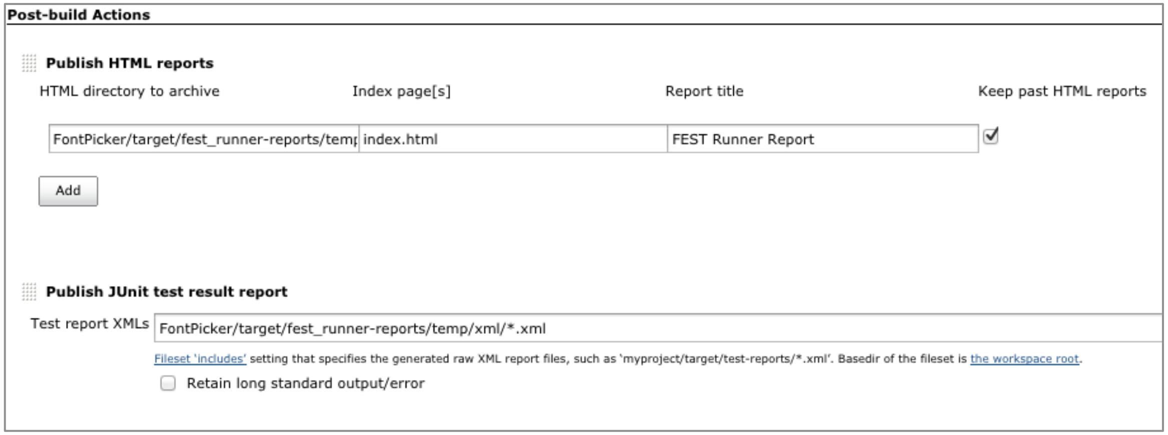
Execute shell

Command

```
sh $WORKSPACE/FontPicker/test/integration/FESTival.sh FontPicker
```

See the list of available environment variables

- Post-build Action – Add Test Reports



Post-build Actions

Publish HTML reports

HTML directory to archive	Index page[s]	Report title	Keep past HTML reports
FontPicker/target/fest_runner-reports/temp	index.html	FEST Runner Report	☑

Publish JUnit test result report

Test report XMLs: FontPicker/target/fest_runner-reports/temp/xml/*.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

☐ Retain long standard output/error

Environments on Jenkins slaves and in the Job

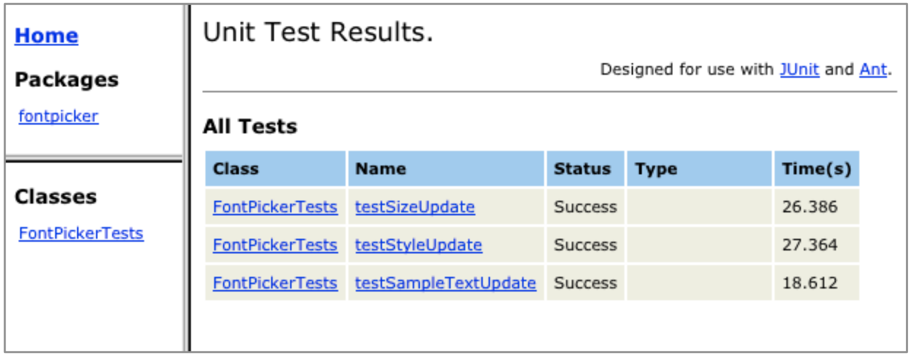
- For the slaves you'll need to ensure
  - Groovy is installed
  - Griffon is installed
    - We like to use GVM for those
      - curl -s get.gvmtool.net | bash
      - gvm install groovy
      - gvm install griffon
  - A vnc server is installed (dependent on OS)
  - vnc2flv is installed
  - FFmpeg is installed
- Again, see set up steps provided

- There are several options for the job environment variables ranging from various environment prepping plugins to explicit setting in the shell step
  - Go with what you like best

GVM
the Groovy enVironment Manager

Kick off your Pipeline and Let the Debugging Begin!

- The Post-Build Reports you set up give you
  - Standard graphical trend tracking in the job home page
  - ALL FEST job output as if you ran it outside of jenkins
    - Including Properties, System.out and System.err



Unit Test Results.

Designed for use with JUnit and Ant.

All Tests

Class	Name	Status	Type	Time(s)
FontPickerTests	testSizeUpdate	Success		26.386
FontPickerTests	testStyleUpdate	Success		27.364
FontPickerTests	testSampleTextUpdate	Success		18.612

- Can't figure out your bug with the reports (<u>this will happen</u>)? **Roll the Tape!**
  - You decide where to copy videos to in FESTival.sh
  - Cut to some example FontPickerTests videos…

Battle Scars

- **Why do we need GriffonTestRunner.groovy can't we just use griffon test-app integration?**
  - We're running things headless in Jenkins and need to manage the $DISPLAY
  - *Sometimes* FEST component hierarchies hang around across test cases within a test class, this cleans it up
- **Be careful using the Xvnc plugin to manage your sessions in builds.**
  - We've found there are important timing differences between when an X session actually becomes available for use.
  - You might run into Socket error: [Errno 111] Connection refused errors with recording the videos using vnc2flv
- **We find that manually restarting a vncserver on the slave as part of job environment preparation (e.g. service vncserver restart) works well for us**
  - It seems to allow us to avoid Errno 111 above
  - We used to just leave a session running continually on the slaves, we found that occasionally (read: insidious intermittent problem) the session would become unstable this would rear it's head by hanging the griffon process of copying files to the staging dir, for example

```
Starting integration test phase ...
    ...
    [copy] Copying 1 file to /…/FontPicker/staging
```

- **Watch (again videos…) out what's going on in the Desktop Environments of your slaves**
  - Screensavers: turn them off
  - Automatic update popups can get in the way of your application windows and cause tests to fail
  - Keyboard repeat rates can cause strange automated input behavior
- **Videos too big?**
  - Set your Desktop background to all black. Turns out the pretty blue default Gnome CentOS background with all the fancy blue gradient doesn't compress nearly as well as all black
  - Change vnc2flv compression settings or consider additional post-processing with FFMpeg

Step Up Your Game with Concurrency

- We had success running a lot of UI tests pretty quickly with two approaches:
  - Concurrent builds with the Jenkins [Multijob Plugin]
    - Set up each FEST test class as a subjob
    - Allows Jenkins to track history per job nicely
    - Requires more per job maintenance
  - Create a FEST-job template with parameterized concurrent execution
    - No per job maintenance
    - More pieces to put together to easily track success / failures trends

//TODOs / Questions?

- Write a Griffon Plugin for Jenkins
  - Grails one exists, should be a simple extension
  - Maybe by next JUC!
- Lightning Questions?
- Supplemental material available at
  - https://bitbucket.org/noblis/juc2013/wiki/Home

Thank You To Our Sponsors

Platinum

CloudBees

Gold

ZEROTURNAROUND

appvance

black diamond SOFTWARE

New Relic.

JenkinsMobi Mobile ALM

XebiaLabs the deployment automation company

BrightRoll

JFrog

Silver

AppDynamics

Confreaks

LIFERAY.

SOASTA Test Faster. Release Sooner.