

Griffon 1.1.0

- 1 [Overview](#)
 - 1.1 [Dependencies](#)
- 2 [Features](#)
 - 2.1 [Buildtime](#)
 - 2.1.1 [Developer Specific Configuration](#)
 - 2.2 [Runtime](#)
 - 2.2.1 [I18N Aware Configuration](#)
 - 2.2.2 [I18N API](#)
 - 2.2.3 [Resource Management and Injection](#)
 - 2.2.4 [Toolkit Agnostic Action Manager](#)
- 3 [Fixes](#)
 - 3.1 [Buildtime](#)
 - 3.1.1 [Griffon command execution failure in Linux Mint](#)
 - 3.2 [Runtime](#)
 - 3.2.1 [EventRouter spins up non-daemon thread](#)
 - 3.3 [Compatibility](#)
- 4 [Sample Applications](#)
 - 4.1 [File Viewer](#)
 - 4.2 [GroovyEdit](#)
 - 4.3 [Font Picker](#)
 - 4.4 [Greet](#)
 - 4.5 [SwingPad](#)
 - 4.6 [GroovyFXPad](#)
 - 4.7 [FxBrowser](#)
 - 4.8 [WeatherWidget](#)
- 5 [Release Notes](#)
 - 5.1 [1.1.0](#)

Overview

Griffon 1.1.0 - is the latest release in the 1.x series

Dependencies

The following dependencies have been upgraded

- ant 1.8.4
- ant-junit 1.8.4
- ant-launcher 1.8.4
- commons-beanutils 1.8.3
- commons-io 2.4
- core-renderer R8
- grails-docs 2.1.0
- grails-gdoc-engine 1.0.1
- groovy-all 1.8.8
- jansi 1.9

- jcl-over-slf4j 1.7.1
- jsch -0.1.48
- jul-to-slf4j 1.7.1
- jzlib 1.1.1
- log4j 1.2.17
- slf4j-api 1.7.1
- slf4j-log4j12 1.7.1
- snakeyaml 1.9

Features

Buildtime

Developer Specific Configuration

There are two places where project specific configuration may be placed

```
$USER_HOME/.griffon/settings.groovy  
griffon-app/conf/BuildConfig.groovy
```

The first is distribution wide (affects all projects) while the second is project specific. The first is never checked into SCM but the second may be. Where then, should a developer put specific configuration that only affects his or her environment without propagating those changes to the rest of the team? This is precisely the goal met by this feature, by reusing the global configuration file (settings.groovy) allowing a new configuration block where the keys are the name of the projects themselves, for example

```
projects {  
    sample {  
        griffon.cli.verbose = true  
    }  
    'custom-app' {  
        griffon.'default'.artifact.repository  
= 'my-local'  
    }  
}
```

Runtime

I18N Aware Configuration

The application's runtime configuration is available through the config property of the application instance. This is a ConfigObject whose contents are obtained by merging Application.groovy and Config.groovy. Builder configuration is available through the builderConfig property and reflects the contents of Builder.groovy. Configuration files may also be provided as properties files; settings on the matching script will be overridden by those set in the properties file.

Configuration files are i18n aware which means you can append locale specific strings to a configuration file, for example Config_de_CH.groovy. Locale suffixes are resolved from least to most specific; for a locale with language = 'de', country = 'CH' and variant = 'Basel' the following files are loaded in order

- Config.groovy
- Config.properties
- Config_de.groovy
- Config_de.properties
- Config_de_CH.groovy
- Config_de_CH.properties
- Config_de_CH_Basel.groovy
- Config_de_CH_Basel.properties

By default the current Locale is used to determine values for language, country and variant; you can change this value by tweaking the application's locale.

I18N API

I18N support has been provided by the [i18n](#) plugin so far. Starting with this release the i18n API is now found in core. There's a new chapter in the Griffon Guide describing how messages can be configured and resolved.

Resource Management and Injection

Paired with I18N it's now possible to define resources that can be resolved and injected at boot time. For example

resources.properties

```
sample.SampleModel.griffonLogo =  
/griffon-logo-48x48.png  
logo = /griffon-logo-{0}x{0}.png
```

SampleModel.groovy

```
package sample
import
griffon.core.resources.InjectedResource
import javax.swing.Icon

class SampleModel {
    @InjectedResource Icon griffonLogo
    @InjectedResource(key='logo',
args=['16']) Icon smallGriffonLogo
    @InjectedResource(key='logo',
args=['64']) Icon largeGriffonLogo
}
```

The Griffon Guide includes a new chapter that describes all configuration options.

Toolkit Agnostic Action Manager

The [actions](#) plugin provides a mechanism that can harvest Controller actions and create a matching toolkit action (Swing based) on the group's builder. This behavior has been extended to support UI toolkits other than Swing, starting with JavaFX. The Griffon Guide includes a full description on how to setup this feature, or disable it if the occasion calls for it.

Fixes

Buildtime

Griffon command execution failure in Linux Mint

Default settings found on Linux Mint prevented the Griffon command from working properly as it contained bash specific code for verifying the location of \$JAVA_HOME. Octavian Nita contributed a patch that makes use of standard shell instructions to accomplish the same goal.

Runtime

EventRouter spins up non-daemon thread

EventRouter makes use of an internal thread to dispatch events asynchronously. Unfortunately this thread was not

set as a daemon thread which resulted in JavaFX applications not exiting as expected. The thread is now marked as a daemon one.

Compatibility

The following plugins are now obsolete:

- i18n
- i18n-support
- actions

The following plugins need to be updated to their latest versions

- dialogs
- spring

Sample Applications

Griffon 1.1.0 ships with 8 sample applications of varying levels of complexity demonstrating various parts of the framework. In order of complexity they are:

File Viewer

File Viewer is a simple demonstration of creating new MVCGroups on the fly.

Source: `samples/FileViewer`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

GroovyEdit

GroovyEdit is an improved version of FileViewer that uses custom observable models.

Source: `samples/GroovyEdit`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

Font Picker

Font Picker demonstrates form based data binding to adjust the sample rendering of system fonts.

Source: `samples/FontPicker`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

Greet

Greet, a full featured Griffon Application, is a Twitter client. It shows Joint Java/Groovy compilation, richer MVCGroup interactions, and network service based data delivery.

Source: `samples/Greet`

To run the sample from source, change into the source directory and run `griffon run-webstart` from the command prompt. Because Greet uses JNLP APIs for browser integration using `run-app` will prevent web links from working.

SwingPad

SwingPad, a full featured Griffon Application, is a scripting console for rendering Groovy SwingBuilder views.

Source: `samples/SwingPad`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

GroovyFXPad

GroovyFXPad, a full featured Griffon Application, is a scripting console for rendering [GroovyFX](#) views.

Source: `samples/GroovyFXPad`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

FxBrowser

FxBrowser is a trivial JavaFX powered browser that demonstrates Griffon's integration with JavaFX.

Source: `samples/FxBrowser`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

WeatherWidget

WeatherWidget demonstrates binding, threading and plugin usage.

Source: `samples/WeatherWidget`

To run the sample from source, change into the source directory and run `griffon run-app` from the command prompt.

Release Notes

1.1.0

 The JIRA server does not support trust requests. Issues have been retrieved anonymously. You can set the macro to always use an anonymous request by setting the `anonymous` parameter to `true`

JIRA Issues (6 issues)										
Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due

	GRIFFO N-520	Allow developer specific buildtime configuration to be defined	Andres Almiray	Andres Almiray		 Closed	Fixed	Jun 12, 2012	Jan 11, 2013	
	GRIFFO N-524	Add i18n support to core	Andres Almiray	Andres Almiray		 Closed	Fixed	Jun 18, 2012	Jan 11, 2013	
	GRIFFO N-525	Add resource management API	Andres Almiray	Andres Almiray		 Closed	Fixed	Jun 18, 2012	Jan 11, 2013	
	GRIFFO N-544	UberBuilder does not use resolveFactory	Danno Ferrin	Danno Ferrin		 Closed	Fixed	Jul 31, 2012	Jan 11, 2013	
	GRIFFO N-546	Support toolkit-agnostic actions in ActionManager	Andres Almiray	Andres Almiray		 Closed	Fixed	Sep 03, 2012	Jan 11, 2013	
	GRIFFO N-549	EventRouter spins out a non-daemon thread	Andres Almiray	Andres Almiray		 Closed	Fixed	Sep 11, 2012	Jan 11, 2013	