Griffon 0.9.2-beta-3

Error rendering macro 'toc' : null

Overview

Griffon 0.9.2-beta-3 – "Aquila adalberti" - is a maintenance release of Griffon 0.9.

Griffon Team

We would like to welcome two new team members to the core development team:

- René Gröschke (@breskeby)
- Alexander Klein (@saschaklein)

New Features

Buildtime

IDE Integration

The integration files for Eclipse and IDEA have been updated to conform to their latest conventions.

Dependencies

Plugin dependencies declared using the Dependency DSL should be fully honored now.
The DependencyReport script will now skip configurations that may not available (i.e, such as `provided`)

Plugin CLI sources

Plugins can now build and package sources that should be excluded form runtime. Just place the sources under `src/cli` and the build will do the rest. Adding the package classes to the build via the dependency DSL is as easy as pasting the following snippet in the plugin's `_Events.groovy` script

```
def eventClosure1 = binding.variables.containsKey('eventSetClasspath') ?
eventSetClasspath : {cl->}
eventSetClasspath = { cl ->
    eventClosure1(cl)
    if(compilingPlugin('quartz')) return
    griffonSettings.dependencyManager.flatDirResolver name: 'griffon-quartz-plugin',
dirs: "${quartzPluginDir}/addon"
    griffonSettings.dependencyManager.addPluginDependency('quartz', [
        conf: 'compile',
        name: 'griffon-quartz-addon',
        group: 'org.codehaus.griffon.plugins',
        version: quartzPluginVersion
    ])
    griffonSettings.dependencyManager.addPluginDependency('quartz', [
        conf: 'build',
        name: 'griffon-quartz-cli',
        group: 'org.codehaus.griffon.plugins',
        version: quartzPluginVersion
    ])
}
```

Substitute 'quartz' for your plugin name. This feature is available to plugins that also package an addon descriptor.

Runtime

WindowManager DSL

Starting with Griffon 0.9.2 there's a new DSL for configuring show/hide behavior per window. This configuration can be set in `griffon-app/conf/Config.groovy`, and here is how it looks

```
swing {
    windowManager {
        myWindowName = [
            show: {window, app -> ... },
            hide: {window, app -> ... }
        ]
        myOtherWindowName = [
            show: {window, app -> ... }
        ]
    }
}
```

The name of each entry must match the value of the Window's name: property. Each entry may have the following options

- **show** - used to show the window to the screen. It must be a closure that takes two parameters: the window to display and the current application.
- **hide** - used to hide the window from the screen. It must be a closure that takes two parameters: the window to hide and the current application.
- **handler** - a custom WindowDisplayHandler.

The first two options have priority over the third one. If one is missing then the WindowManager will invoke the default behavior. There is one last option that can be used to override the default behavior provided to all windows

```
swing {
    windowManager {
        defaultHandler = new MyCustomWindowDisplayHandler()
    }
}
```

Previous to Griffon 0.9.2 the first window to be displayed during the Ready phase was determined by a simple algorithm: picking the first available window from the managed windows list. With 0.9.2 however, it's now possible to configure this behavior by means of the WindowManager DSL. Simply specify a value for swing.windowManager.startingWindow, like this

```
swing {
    windowManager {
        startingWindow = 'primary'
    }
}
```

This configuration flag accepts two types of values:

- a String that defines the name of the Window. You must make sure the Window has a matching name property.
- a Number that defines the index of the Window in the list of managed windows.

If no match is found then the default behavior will be executed.

Conditional logging

The latest Groovy beta release (1.8b3) includes a new AST transformation (@Log) that can inject a Logger instance (if not present already) and

transforms all logging calls to be conditionally guarded. This means that a simple logging statement as

```
log.info "Elmer Fudd hunts $wabbits"
```

is transformed into

```
if(log.infoEnabled) log.info "Elmer Fudd hunts $wabbits"
```

While Griffon 0.9.2-beta-3 still depends on Groovy 1.7.5 (which doesn't provide access to @Log) it however, does inject conditional logging on all logger instances belonging to Griffon artifacts, that is: controllers, models, views, services, and any additional artifacts added by plugins.

Breaking Changes

Moved Classes

- griffon.core.BaseGriffonApplication -> org.codehaus.griffon.runtime.core.BaseGriffonApplication

Changed Classes

- griffon.core.ArtifactManager turned into an interface
- griffon.core.ArtifactManager.getClassesOfType() returns List<GriffonClass> rather than GriffonClass[]
- griffon.core.ArtifactManager.getAllClasses() returns List<GriffonClass> rather than GriffonClass[]
- ArtifactManager is no longer a singleton. You can get a hold to the current ArtifactManager instance by querying the application directly. ApplicationHolder can be used for all other classes that do not have a direct reference to the running application. The rationale for this change is to have as few magic singleton classes as possible.

New classes

- org.codehaus.griffon.runtime.core.AbstractArtifactManager
- org.codehaus.griffon.runtime.core.DefaultArtifactManager

Dependency Management

Plugins built with Griffon 0.9.2 and upwards should declare all of its dependencies using the Dependency DSL found in `griffon-app/conf/BuildConfig.groovy`, including those that are located in the `lib` directory. For the latter case, you must declare a resolver that is local to the plugin, like this

```
griffon.project.dependency.resolution = {
    inherits "global"
    log "warn"
    repositories {
        flatDir name: 'slickPluginLib', dirs: 'lib'
    }
    dependencies {
        compile 'org.newdawn:slick:274',
                'com.jcraft:jorbis:0.0.17'
    }
}
```

Sample Applications

Griffon 0.9.2-beta-3 ships with 5 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.

0.9.2-beta-3 Release Notes

Griffon 0.9.2-beta-3 Resolved Issues (27 issues)

T	Key	Summary
	GRIFFON-296	Run-App and Installing plugins broke after upgrading to 0.9.2-beta-3
	GRIFFON-161	SecurityException: invalid SHA1 signature file digest' when running the command 'griffon package zip
	GRIFFON-261	Dependency report throws FileNotFoundException when run on a plugin project
	GRIFFON-262	Can't resolve test dependencies declared on a plugin
	GRIFFON-273	Application does not run in applet mode if the UI contains buttons
	GRIFFON-283	IntegrateWith -eclipse does not populate library entries properly
	GRIFFON-284	IntegrateWith -intellij adds outdated IDEA project files, should use the new dir layout
	GRIFFON-287	"GlazedlistsGriffonAddon"'s signer information does not match signer information of other classes
	GRIFFON-288	Exception when using ShutdownHandler using a Map as ShutdownHandler

GRIFFON-289	When using a ShutdownHandler, the window will always close before the canShutdown's are invoked. Should be configurable.
GRIFFON-291	Can't build binary package withour codehaus username/password
GRIFFON-293	executing "./gradlew build" fails due to missing checkstyle configuration
GRIFFON-294	Remove signatures from jars before signing them again
GRIFFON-295	Ability to specify which Window should be displayed first
GRIFFON-297	Dependency problem with a plugin including an addon if it is named camelcased (or contains hyphen ?)
GRIFFON-298	Add conditional logging on artifacts
GRIFFON-299	Use conditional logging on complex expressions
GRIFFON-300	Can't generate guide in PDF
GRIFFON-301	griffon.core.ArtifactManager should be an interface
GRIFFON-302	Let addon jars participate with the DependencyManager using the dependency DSL

Showing 20 out of 27 issues