Griffon 0.9.2

Error rendering macro 'toc' : null

Overview

Griffon 0.9.2 – "Aquila pomarina" - is a maintenance release of Griffon 0.9.

Griffon Team

During the development time frame of 0.9.2 the team grew by two new members

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

New Features

Griffon build

The Griffon build has moved from Ant to Gradle with increased boost in build time and reduced setup.

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('eventSetClass
path') ? eventSetClasspath : {cl->}
eventSetClasspath = { cl ->
    eventClosure1(cl)
    if(compilingPlugin('quartz')) return

griffonSettings.dependencyManager.flatDirRes
olver name: 'griffon-quartz-plugin', dirs:
"${quartzPluginDir}/addon"

griffonSettings.dependencyManager.addPluginD
ependency('quartz', [
        conf: 'compile',
        name: 'griffon-quartz-addon',
        group:
'org.codehaus.griffon.plugins',
        version: quartzPluginVersion
    ])

griffonSettings.dependencyManager.addPluginD
ependency('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.

Install plugin updates in batch mode

A couple of releases ago Griffon added a command to list all available plugin updates from currently installed plugins however you were still tasked with updating every single plugin individually. Now you can update all plugins in one step by invoking the `list-plugins-update` command with an additional flag

```
griffon list-plugins-updates -install
```

Override template selection

Every `create-*` command relies on templates in order to generate their target files. Each template matches a naming convention which can be used to your advantage. For example overriding the default template for a view when creating a new MVC groups can be done by calling

```
griffon create-mvc -view=CustomView
```

The command will search for templates in the following locations

- ${basedir}/src/templates/artifacts/
- ${pluginsHome}/*/src/templates/artifacts/
- ${griffonWorkDir}/archetypes/${archetype}/templates/artifacts/
- ${griffonHome}/archetypes/${archetype}/templates/artifacts/
- ${griffonHome}/archetypes/default/templates/artifacts/

The convention is simple, define a flag whose name matches the type of portion you want to override, i.e, -view matches View, -service matches Service and so on. This works with any supported file type (like Groovy and Java).

Configuration flags

All of the command options described in section 4.6 Command Line Options of the Griffon Guide can now be specified in `griffon-app/conf/BuildConfig.groovy` or `$USER_HOME/.griffon/settings.groovy`.

Logging

The logging DSL for execution during runtime is available during buildtime too. You can configure it by editing either `griffon-app/conf/BuildConfig.groovy` or `$USER_HOME/.griffon/settings.groovy` and placing a `log4j` section.

Runtime

Logging DSL

Griffon 0.9.2 has ported over the Log4j DSL found in Grails 1.3.x. You can now configure logging pretty much in the same way. Here's an example of how a typical setup will look like (in `griffon-app/conf/Config.groovy`

```
log4j = {
    error  'org.codehaus.griffon',  //
internals

    warn   'griffon.util',
           'griffon.core'
}
```

The Griffon Guide contains further information on how to use the DSL.

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 still depends on Groovy 1.7.6 (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.

AddonManager

There's a new helper class (`griffon.core.AddonManager`) that will keep track of installed Addons. Each addon now has an associated descriptor of type `griffon.core.GriffonAddonDescriptor` that can helps an application figure out more about addon contributions at runtime.

Services as Application Event Handlers

You might be aware that Griffon automatically manages services instances (with or without installing the Spring plugin). Each service is treated as a singleton, which made it a bit difficult to configure services as application event listeners. Starting with this release services instances will become application event listeners by default, same as controllers. Also, all available service instances can be accessed from the application instance using the `getServices()` method/property. Be warned that services are still instantiated on demand so you won't be able to query all services instances at a given time if not all of them have been instantiated already.

Threading AST Transformation

There's a new AST transformation that can be used to inject proper threading management used to execute code outside/inside the UI thread. Refer to section 9.3 Annotation Based Threading of the Griffon Guide.

Automatic threading management in Controllers

It's been known since the beginning of Swing that executing a long operation inside the EDT is a bad practice and should be avoided. Sadly there is no compile time support for checking for violations of this rule. However starting with this release all controller actions are guaranteed to be executed outside of the EDT. This feature can configured/disabled in several ways

1. Specify `griffon.disable.threading.injection=true` during the compilation process

```
griffon -Dgriffon.disable.threading.injection=true compile
```

2. Alternatively specify `griffon.disable.threading.injection=true` in either `griffon-app/conf/BuildConfig.groovy` or `$USER_HOME/.griffon/settings.groovy`.

3. Follow the instructions found in section 8.1.1 Threads and Actions of the Griffon Guide.

4. Or annotate the action property/method with @Threading(Threading.Policy.SKIP)

This feature complements the @Threading AST transformation, which means it will honor any settings you specify if an action is annotated with @Threading. A controller action is considered to be

- a public method or a closure property (no access modifier)
- the name of the method/property does not match an event handler (it does not begin with 'on')

Publish events in asynchronous mode

There are now 3 ways for publishing an event

- `event(name, args)` - publishes the event in the same thread as the publisher, in other words it is synchronous.
- `eventAsync(name, args)` - publishes the event in a different thread as the publisher, in other words it is asynchronous.
- `eventOutside(name, args)` - publishes the event outside of the UI thread.

Uncaught exception management

The Griffon runtime will now handle all uncaught exceptions and trigger application events following a naming convention. This way any listener can react to an exception being thrown. Consult section 5.5.10 of the Guide to know more about this feature. In short, if an IllegalArgumentException is thrown and not caught then a listener can handle by registering the following handler

```
def onUncaughtIllegalArgumentException = { iae -> ... }
```

There's also a generic catch all event

```
def onUncaghtExceptionThrown = { e -> ... }
```

Xml externalized Views

Building a Java View might not be so fun as building its Groovy counterpart (thanks to the builder DSL). However you can use an XML definition that closely resembles the builder format but uses XML instead of Groovy. this option is meant for developers that do not want to use Groovy source code in any way. Consult section 13.3 of the guide, specifically subsection 13.3.3 to know more about this feature.

Creating bindings in Java

Continuing with Java specific features, bindings are another missing aspect of the Swing library. Fear not as the new `griffon.util.BindUtils` exposes SwingBuilder's bindings throw a fluent interface design. Here's an example of it

```
BindUtils.binding()
         .withSource(model)
         .withSourceProperty("value")

.withTarget(builder.getVariable("output"))
         .withTargetProperty("text")
         .make(builder);
```

You can use this feature from Groovy code too, as long as you provide an instance of a builder that can handle the `bind()` node.

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'
    }
}
```

Buildtime Dependencies

The jar `org.springframework.test` is no longer provided for the test configuration.

Runtime Behavior

Automatic threading management for controller actions is a very desirable feature but it breaks compatibility with previous releases. You can disable this feature in many ways if it proves to be problematic during transition. Please review all methods and closure properties found in your controllers. Any candidate that follows the rules for actions will be automatically transformed.

Changes in the event publishing mechanism means that calling `eventAsync()` will not work as before. Every call to `eventAsync()` must be substituted with `eventOutside()`

Previous releases in this series

Griffon 0.9.2-beta-1
Griffon 0.9.2-beta-2
Griffon 0.9.2-beta-3
Griffon 0.9.2-rc1

Sample Applications

Griffon 0.9.2 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 Release Notes

> 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`

Griffon 0.9.2 Resolved Issues (7 issues)		

Type	Key	Summary
	GRIFFON-314	DefaultArtifactManager may attempt instantiation of an abstract class
	GRIFFON-324	Ability to install plugin updates in batch mode
	GRIFFON-331	Publish events in asynchronous mode
	GRIFFON-335	Cannot setup an HTTP proxy by calling griffon set-proxy
	GRIFFON-337	Dist directory is deleted after Prepackage event is fired
	GRIFFON-328	UncaughtExceptionHandler support in Griffon core
	GRIFFON-332	slight typo in user guide

0.9.2-rc1 Release Notes

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`

Griffon 0.9.2-rc1 Resolved Issues (10 issues)		
Type	**Key**	**Summary**
	GRIFFON-154	GfxBuilder: Line node cannot be used by itself
	GRIFFON-308	Automatically execute controller actions off the UI thread
	GRIFFON-309	DefaultGriffonControllerClass incorrectly lists some methods as actions
	GRIFFON-311	Ability to specify compile flags in BuildConfig.groovy
	GRIFFON-313	POM file for griffon-scripts is missing dependency on griffon-resources
	GRIFFON-317	Services could be auto registered as app event listeners
	GRIFFON-318	Binary zip has weird file permissions
	GRIFFON-319	Tar gz has weird file permissions
	GRIFFON-326	A collection of multiple bugs in gfxbuilder
	GRIFFON-322	Debian installer does not set executable flag on griffon

0.9.2-beta-3 Release Notes

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`

Griffon 0.9.2-beta-3 Resolved Issues (27 issues)		
Type	**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 without 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
	GRIFFON-303	Allow a plugin to package multiple addon jars
	GRIFFON-304	GriffonApplication.getPhase() should be synchronized
	GRIFFON-305	Allow plugin installs on compatible platforms
	GRIFFON-306	Update Spring to 3.0.5.RELEASE
	GRIFFON-307	Ability to build and package plugin cli sources
	GRIFFON-312	Griffon run-webstart and run-applet do not work on windows
	GRIFFON-254	griffon run-app asks to choose between RunApp and RunApplet

0.9.2-beta-2 Release Notes

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`

Griffon 0.9.2-beta-2 Resolved Issues (11 issues)		
Type	**Key**	**Summary**
	GRIFFON-281	View scripts fail to execute
	GRIFFON-269	Migrate build from Ant to Gradle
	GRIFFON-277	Add logging support
	GRIFFON-282	StackOverflowError when queryin a controller for its metaclass
	GRIFFON-265	CreateApp fails on Windows when fileType parameter is specified
	GRIFFON-270	Full arguments are not sent to application when using RunApp script
	GRIFFON-271	Launch script does not pass arguments to running application
	GRIFFON-272	Can't launch application in webstart mode

	GRIFFON-276	Jar signature fails verification when launch in webstart mode
	GRIFFON-278	Switch application event handling to synchronous mode
	GRIFFON-279	package-plugin command line prompt for svn password prints plaintext password to console

0.9.2-beta-1 Release Notes

> 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`

Griffon 0.9.2-beta-1 Resolved Issues (2 issues)		
Type	**Key**	**Summary**
	GRIFFON-263	Can't install FEST plugin using Griffon 0.9.1a
	GRIFFON-264	Can't run artifact tests in unit mode