# Griffon 0.9.3-beta-2

Error rendering macro 'toc' : null

## Overview

Griffon 0.9.3-beta-2 – "Aquila pennata" - is a maintenance release of Griffon 0.9.

## New Features

### Buildtime

#### IDE Support

It's been a while since IDEA added support for Groovy DSL descriptors. Recently Eclipse gained the same capabilities via DSLD (explained by Andrew Eisenberg, Vladimír Oraný posted a thorough tutorial here). During Gr8conf Copenhagen 2010 a group of Griffon enthusiasts banded together under the Hackergarten space and created a couple of GDSLs specifically tailored for Griffon. This year was no different, a similar group managed to build improved versions of these GDSLs plus the first cur of DSLDs. These scripts should be automatically picked up by either IDE as soon as the Griffon jars are placed in the classpath. So what exactly do you gain now?

- autocompletion of Swing nodes when working in a Griffon View script.
- autocompletion of methods from standard Griffon artifacts.
- autocompletion of methods from the following AST transformations: @Bindable, @Vetoable, @EventPublisher, @MVCAware, @ThreadingAware

#### Configure Application's Manifest

It's now possible to configure the manifest that's placed inside the application's jar. Griffon will automatically create the following entries in the application's manifest

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.8.1
Created-By: ${jvm.version} (${jvm.vendor})
Main-Class: ${griffonApplicationClass} | ${griffonAppletClass}
Built-By: ${user.name}
Build-Date: dd-MM-yyyy HH:mm:ss
Griffon-Version: ${griffonVersion}
Implementation-Title: capitalize(${griffonAppName})
Implementation-Version: ${appVersion}
Implementation-Vendor: capitalize(${griffonAppName})
```

There might be times when you must specify additional attributes or override existing ones. You can do this by adding a new block of configuration to `BuildConfig.groovy`, for example

```
griffon {
    jars {
        manifest = [
            'Foo': 'Bar'
            'Built-By': 'Acme'
        ]
    }
}
```

#### Merge duplicate files when packaging

There's a high chance of some files to have duplicates, e.g. griffon-artifacts.properties if you have installed a plugin that provides MVC groups. It's possible to instruct the build to merge duplicate files by specifying a regular expression and a merging strategy. The following table explains the different merging strategies available

| Strategy | Description |
|---|---|
| Skip | Do not perform any merge. Duplicate is discarded. |
| Replace | Duplicate is preferred and overwrites previous. |
| Append | Duplicate is appended at the end of previous. |
| Merge | Common lines found in duplicate are discarded. New lines found in duplicate are appended at the end. |
| MergeManifest | Duplicate keys override the previous ones. New keys are added to the merged result. |
| MergeProperties | Duplicate keys override the previous ones. New keys are added to the merged result. |
| MergeGriffonArtifacts | Merges artifact definitions per type. |

You can specify merging preferences in @BuildConfig.groovy@ like this

```
griffon {
    jars {
        merge = [
            '.*.xml': org.codehaus.griffon.ant.taskdefs.FileMergeTask.Replace
        ]
    }
}
```

This setting will overwrite any XML file found in the path with the last version encountered as jars are processed.
The griffon build defines a set of default mappings, which are the ones found in the next table

| Regexp | MergeStrategy |
|---|---|
| META-INF/griffon-artifacts.properties | MergeGriffonArtifacts |
| META-INF/MANIFEST.MF | MergeManifest |
| META-INF/services/.* | Merge |
| .*.properties | MergeProperties |

Merging preferences must be defined from the most specific to the least. Your preferences will override any default settings.

# Runtime

## Configurable Platform Customizations

In the past, platform customizations like the handling of the About and Preferences menu in OSX, were handled internally by the Griffon runtime, giving you no chance to override or alter the default behavior. That has been changed now. Starting with this release you should be able to instruct the runtime how you want those customizations to be applied. You only need to implement the `griffon.util.PlatformHandler` inter face and register your implementation. The following configuration in `Config.groovy` specifies a different handler for `macosx`:

```
platform {
    handler = [
        macosx: 'com.acme.MyMacOSXPlatformHandler'
    ]
}
```

Now you only need to create such handler, like this:

```
package com.acme

import griffon.core.GriffonApplication
import griffon.util.PlatformHandler

class MyMacOSXPlatformHandler implements PlatformHandler {
    void handle(GriffonApplication app) {
        System.setProperty('apple.laf.useScreenMenuBar', 'true')
        ...
    }
}
```

The following platform keys are recognized by the application in order to locate a particular handler: `linux`, `macosx`, `solaris` and `windows`.

### New AST Transformations

It's possible for non-artifact classes to participate in the MVC group mechanism (but not he life cycle itself) by implementing the `griffon.core.MVCHandler` interface. This task is easily achieved by annotating the class with `griffon.transform.MVCAware`. The same can be said for classes that would like to gain the capabilities of executing code using the threading facilities exposed by Griffon. The interface is `griffon.core.ThreadingHandler` and the transformation is `griffon.transform.ThreadingAware`.

## Breaking Changes

### Runtime Behavior

All AST xforms have been relocated to package `griffon.transform` to align them with Groovy 1.8.0 where most are now found in `groovy.transform`; this results in the following changes

- `griffon.beans.Listener` -> `griffon.transform.PropertyListener`
- `griffon.util.EventPublisher` -> `griffon.transform.EventPublisher`
- `griffon.util.Threading` -> `griffon.transform.Threading`

The `package griffon.transform` is not auto imported by default. Usage of these AST transformations must be declared explicitly.

Both `griffon.core.GriffonApplication` and `griffon.core.GriffonArtifact` now extend `griffon.core.MVCHandler` and `griffon.core.ThreadingHandler`.

The interface `griffon.util.EventPublisher` is no longer an AST transformations (because it was relocated), it now identifies a class that can publish events using an EventRouter.

`griffon.util.EventRouter` moved to `org.codehaus.griffon.runtime.core.EventRouter`

`griffon.util.UIThreadHelper` moved to `griffon.core.UIThreadManager`

## Sample Applications

Griffon 0.9.3-beta-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.

# 0.9.3-beta-2 Release Notes

## Griffon 0.9.3-beta-2 Resolved Issues (14 issues)

| T | Key | Summary |
|---|---|---|
| | GRIFFON-360 | Single jar package does not merge special files |
| | GRIFFON-285 | Add back missing GDSL files to griffon-cli |
| | GRIFFON-361 | Rework DSL support for IDEA |
| | GRIFFON-362 | Add support for Eclipse's DSLD |
| | GRIFFON-363 | SwingPad has wrong dependency version on gfx-builder |
| | GRIFFON-364 | Thread injection does not work for controller actions that make use of the default parameter |
| | GRIFFON-365 | [SwingPad] Cannot run the application due to an unexpected error when fetching an image |
| | GRIFFON-366 | Switch default imports feature to CompilerCustomizers provided by Groovy 1.8.0 |
| | GRIFFON-368 | 'prefix' for addons ignored |
| | GRIFFON-369 | Variables written to binding in Event NewInstance will be thrown away if it is an instance of Script |
| | GRIFFON-370 | Allow the manifest of the app's jar to be configured with external properties |
| | GRIFFON-372 | Allow platform customizations to be extenarlly configured |
| | GRIFFON-375 | Java based View template does not work in applet mode |
| | GRIFFON-371 | Launching the application in applet mode displays the wrong name in the menu bar (OSX) |

14 issues