Groovy 2.3.0-beta-1 is out!

Hi all,

The Groovy team is very pleased to announce the first beta of the Groovy 2.3 release!

Groovy 2.3.0-beta-1 is actually already a feature-complete beta, so the release cycle towards the final version will be very short. Furthermore, Grails 2.4 is looking forward to integrating it rapidly as well for its upcoming major release too! So you can expect a general availability version of Groovy in the coming weeks, as we'll move directly in "release candidate" mode after this beta. But your feedback, as usual, will be very important for trying out this beta in your respective projects, to tell us about any problem you might encounter.

In this email, below, I'll give you a few pointers regarding the scope of the release, but we'll be fleshing out a more detailed release notes document in the coming days and weeks with more information.

Warning: this is a lengthy email, but you'll get to see all the new stuff and improvements the Groovy team and contributors have been working on!

The key highlights of Groovy 2.3 are:

- Official support for running Groovy on JDK 8

  This is the first version of Groovy to be officially compatible with JDK 8.

  JDK 8 and its interface default methods introduced some incompatibilities with a few methods of the Groovy Development Kit, so we had to adapt to the situation, introducing minor breaking changes for the affected methods and their outcome.

  Note that we're not planning to backport the changes to older versions of Groovy, so if you want to run Groovy on JDK 8, you'll have to upgrade to the shiniest version of Groovy!

- Traits

  A major highlight for Groovy 2.3 is the introduction of the concept of traits.

  Traits are reusable components of behavior that you can make your class implement, and are an additional Object-Oriented concept alongside classes and interfaces.

  Groovy traits are stateful (unlike Java 8 interface default methods).

  They allow the composition of behavior without going into the "diamond inheritance" problem allowing you to decide which behavior prevails upon conflict.

  Traits support inheritance, thus a trait can extend another trait or implement an interface.

  Traits are compatible with static type checking and compilation, as well as our usual dynamic behavior. Trait mixed-in methods are actually "real" methods (ie. visible from Java as well) and not just dynamic.

  Traits can also be implemented at runtime with "as" or with "withTraits" if you just want to add behavior of a trait to an object you're instantiating.

  You can find more information on traits in the exhaustive trait documentation.

- New and updated AST transformations

  - @TailRecursive on methods adds tail recursion to methods which are recursive and call themselves at the last operation of the method body, which helps blowing up the stack with the recursive calls (GROOVY-6570).
  - @Sortable on classes implement comparison methods for you, according to the declaration order of your properties (GROOVY-6649).
  - @Delegate supports includes and excludes attributes so as to let you decide if certain methods should be included or excluded from the delegation (GROOVY-6329).

- New NIO module for Java 7+

  On JDK 7 and beyond, you can benefit from the same methods as the ones of File, URLs, the various Stream classes, etc, but for the new NIO2 methods, like Path.

  See GROOVY-6377 and the pull request for some further hints of the new methods.

- Various minor performance improvements across the board, for static compilation, the "invoke dynamic" backend, as well as "normal" dynamic Groovy

- Drastic JSON parsing and serialization performance improvements

  The Rick / Andrey duo spent a fair amount of time optimizing our JSON support, making Groovy 2.3's JSON support usually faster than

all the JSON libraries available in the Java ecosystem.

With JsonSlurper, you'll be able to set different parser types depending on the kind of input you wish to parse, particularly if you know the size of the payload you expect to parse, or whether you want a more tolerant parser which accepts elements like comments which are not normally supported by the JSON specification (GROOVY-6546).

On the output front, with JsonBuilder and StreamingJsonBuilder, which both use JsonOutput for output generation, the serialization has also been tremendously improved (GROOVY-6554).

- Closure type parameter inference

  We closed a gap which forced you to type your closure parameters to get correct type inference with static type checking or static compilation enabled. In situations like the following, you would have to explicitly give the type of the parameter, but it's no longer required:

  ['a','b'].each { it.toUpperCase() }

  In the signature of your methods taking closures as arguments, you'll also be able to annotate the closure parameter with @ClosureParams to give additional hints to the type checker to infer the type of the parameters passed to your closure.

  You can also find more about this in Cédric's blog post on closure parameter type inference.

- New Markup template engine

  Groovy now has an additional template engine, in the form of the Markup template engine, which gives you a very fast template engine, based on the familiar Markup builder approach and notation, but also offering formatting options (indentation, escaping), internationalization, includes, as well as proposing type checked templates and models.

  More details about the new Markup template engine in the documentation (GROOVY-6596), as well as in Cédric's blog, if you want to learn more about the "behind the scenes" stories!

- Further Groovysh enhancements

  Along with some slightly reduced startup time, Groovysh has seen new improvements in its code-completion capabilities:

  - line comment command (GROOVY-6459)
  - completion for keywords (GROOVY-6399)
  - completion for properties (GROOVY-6395)

- GroovyConsole enhancements

  It is now possible to configure the font used by the console, and also to be able to run a selected snippet of code reusing the imports defined in your script making it easier to just run quick snippets of your script.

- JUnit 4 GroovyAssert class

  The venerable GroovyTestCase (JUnit 3 based approach) has often been used as a base class for your test classes — unless you've been using the Spock testing framework, of course. One of the drawback of this class is that your test classes can't extend your own classes, but must derive from GroovyTestCase to benefit from the additional assertion methods.

  Groovy 2.3 introduces the JUnit 4-friendly GroovyAssert, which is a convenient class offering the usual assertion methods of GroovyTestCase, but in the form of static methods that you can static import in your test class.

  We have not included the myriads of assertEquals method from GroovyTestCase as they have become useless with Groovy's power assert, but it provides the handy shouldFail*() methods (GROOVY-6588).

- ConfigSlurper

  ConfigSlurper just supported a single "environments" non-configurational conditional block, but you couldn't let you define your own. With Groovy 2.3 you can also create your own ones too, for instance if you wanted to support "flavors" like OS variants (GROOVY-6383).

  In addition, the isSet() / hasSet() combo methods (GROOVY-4639) have been added so you can double check if a given node of your configuration has been defined. Before, whether the node wasn't defined or containing null, you couldn't differentiate either case easily.

- @BaseScript class improvements

  @BaseScript is a fairly recent addition in Groovy, and it allowed to annotate a variable in your script to instruct the compiler to use a particular base script class for this script. Now we have another notation which is nicer as you can annotate an import or a package (GROOVY-6592) to indicate that base script class:

  ```
  @BaseScript(MyScript)
  import groovy.transform.BaseScript
  ```

Additionally, base script classes can now use any abstract method for the script body. This means that you can implement the run() method to implement specific behavior like setup and tear down in tests (GROOVY-6585 and GROOVY-6615).

- New style for the GroovyDoc documentation of Groovy and the GDK methods

  GroovyDoc has been updated with a new fresh and modern skin that will be part of the future visual identity of the Groovy website. Those style updates are also available by default for your own usage of GroovyDoc, making your own documentation nicer on the eye.

  You can have a look at the GroovyDoc documentation for Groovy 2.3.0-beta-1.

  We also took the opportunity to apply the same stylesheet to our “DocGenerator” tool which is responsible for the generation of the GDK documentation, showing the methods the Groovy library adds on top of the JDK classes.

  Please also have a look at the new restyled GDK documentation.

- New documentation

  We are still working on the brand new documentation for Groovy (in Asciidoc(tor) format), so you can already have a glimpse at what’s already covered or not.

  We’re looking forward for help for fleshing out the various TBD (To Be Done) sections of the documentation, as it’s a gigantic taks to re-document each and every aspect of the language and its libraries! So please shout if you want to contribute to the new documentation! All help is warmly welcome!

- Dependency upgrades
  - Gradle 1.10 for building Groovy
  - ASM 5.0.1 library for generating our bytecode (also needed for our JDK 8 support)
  - JLine 2.11 and JANSI 1.11 library for Groovysh

Groovy wouldn’t be what it is without your support, your bug reports and feature requests, your contributions and pull requests, so big thanks to all those who contributed to this release in a way or another, including the usual suspects of the core team (Jochen, Paul, Cédric and myself), but also André (documentation, GroovyAssert, Groovy Console), Andres (Config slurper, @Sortable with Paul), Tim, Pascal (ConfigSlurper), Johanes (@TailRecursive), Paolo (new NIO module), Thibault (groovysh), Rick and Andrey (for the JSON support), Jim (base script class), Damien (GroovyDoc / GDK styling), Kenneth and several others I’ve forgotten to mention!

And in honor of this first beta release of Groovy 2.3, Cédric even decided to put his birthday on the same day, so happy birthday Cédric, and great work on this release!

We’re looking forward to your feedback running this beta to its pace!

It’s important to make this 2.3 release rock solid!

Download Groovy 2.3-beta-1 and have a look at the JIRA release notes for additional details on issues and bug fixes closed.

Keep on groovy’ing!