GSoC 2014

This year, the Groovy Project is applying to the Google Summer of Code as an organization. In the previous years, we had a few students from the GSoC programme work on Groovy through the Codehaus organization, but in 2013 we'd like to expand our involvement in the programme by also offering opportunities to other projects of the wider Groovy Ecosystem to participate in this effort.

Below, you will find various project ideas for Google Summer of Code 2014, and we'll update this page as we refine our application and as other project proposals arrive and by including feedback both from the Groovy ecosystem and the students who would like to work on a Groovy based project.

Students, if you wish to have further details about the below projects, don't hesitate to reach out to us through comments on this page, or by discussing those projects ideas on our mailing-lists. We'll be happy to expand the descriptions of those projects, and give more details or guidelines as to what our expectations are.

Project ideas

Groovy on Android

Currently, Groovy is not able to run properly on Google's Android mobile platform out of the box. A couple years ago, a first GSoC project (nicknamed DiscoBot), started porting Groovy to Android, using Groovy 1.7, but performance wasn't there (20s to startup a simple Hello World). The goal of this GSoC project is to work with the Groovy core team and the past contributors of the DiscoBot project, towards the goal of making any Groovy program to run on the Android platform well, so that apps for such mobile phone can be written fully in Groovy.

It will be interesting to investigate what modifications can be brought to Groovy to make it support Android in a more straightforward manner, how we can leverage static compilation capabilities, and also see how Groovy builders and other features can help further streamline the development of Android applications using Groovy.

Project: Groovy

Possible mentors: Cédric Champeau

An Antlr v4 grammar for Groovy

As of today, Groovy 2 still uses Antlr v2 for its grammar. The original grammar was based off of the Java grammar itself. But we would like to create a dedicated grammar for Groovy with the latest version of Antlr, ie. with version 4. Antlr v4 has evolved nicely and makes it easier to evolve grammars, without the painful work of rule disambiguation. So the idea is to develop a clean room implementation of the Groovy grammar for the upcoming versions of Groovy, that would be able to also cover new syntax elements, like the support of Java 8 lambda syntax, or the type annotation JSR, and we'd also take the opportunity to tackle things that we haven't covered so far, like JavaDoc comments in the resulting AST.

Project: Groovy

Possible mentor: Jochen Theodorou

Groovy and Java joint compiler without stubs

Groovy already has a joint compiler, but it works by producing stubs for groovy files. Not only does the resulting disc IO increase the time the compiler needs to compile something a lot, it also has certain limitations. Examples are unapplied xforms and certain situations where direct calls to the super class are required, but the joint compiler cannot produce them, because it has no idea how the class looks like. This proposal is to create a bridge between javac and groovyc internal data structures similar to what was been done for the groovy-eclipse compiler.

Project: Groovy

Possible mentor: Jochen Theodorou

Lazy data structures and comprehensions

The idea would be to have Groovy support generators, lazy evaluations, ... See for example: http://blog.bloidonia.com/post/22117894718/groovy-stream-a-lazy-generator-class-for-groovy

It is also important to take a look at what APIs are being written in JDK 8 because most of the APIs written for the new lambdas are specifically lazy.

Project: Groovy

Possible mentor: Paul King

Groovy DSL for Cascading

Create a Domain-Specific Language in Groovy for the Cascading framework, to simplify authoring of data analytics and data management applications for Apache Hadoop and other datastores.
Possible mentor: Vladimir Vivien

Benchmark suite

Benchmarking a language is a complex task and Groovy misses such a benchmark suite. The idea would be to write a benchmarking module that tests various algorithms using different flavours: untyped, typed, statically compiled, with various code styles (Java-ports, idiomatic Groovy, ...). Of course, such a benchmark would be the base to compare various Groovy versions between them, such as comparing the "classic" call site caching with the new "invokedynamic" implementation, as well as comparing new releases to ensure that no performance regression is accidentally introduced. Such a benchmark suite would be of great help for the developers.

Project: Groovy

Possible mentors: Cédric Champeau, Guillaume Laforge or Jochen Theodorou

Remoting and distribution for GPars

The distinction between single-VM concurrency and concurrency among distributed nodes is becoming blurred. Actors can be distributed across multiple nodes in a network, dataflow variables and channels can transport values across the single-machine boundary as well as the state preserved by agents can reside on a particular place in the network and may need to be accessed from other places. As part of this assignment the participant would design and implement remoting for actors, agents and dataflow channels. Ideally remoting should not have impact on the existing API, should be pluggable to leverage different communication protocols and should be implemented as an optional add-on to limit performance and memory overhead when single-machine concurrency is used.

Project: GPars

Possible mentor: Václav Pech or Russel Winder

Software Transactional Memory implementation for Groovy

STM is one of the promising concepts that aim to enable developers to write safe concurrent code. Several promising open source JVM-based implementations have appeared recently - (http://multiverse.codehaus.org/overview.html,http://code.google.com/p/deuce/, http://akkasource.org/). As an initial part of the assignment the existing options should be investigated in order to build a suitable STM strategy for the GPars project. A second part of the assignment would be to implement a Groovy wrapper around the chosen solution, integrating smoothly STM into Groovy and the GPars library as well as providing an intuitive API or a set of DSLs.
Project: GPars

Possible mentor: Václav Pech or Russel Winder

GPars Monitoring

GPars provides several concurrency and parallel solutions, like actors, dataflow, etc. And in enterprise contexts, it's interesting to know how those solutions perform, how to get feedback from them, in the form of various metrics. For example, how much actor mailboxes may be full or not, which queues are currently used, etc. The idea of this GSoC project is to introduce a way of monitoring those various concurrency constructs so as to be able to gather useful feedback on the usage in a concrete usage context, so as to be able to further fine tune the configuration of those constructs (increase the thread pool, etc). This project likely involves wiring in some JMX beans for the monitoring.

Project: GPars

Possible mentor: Václav Pech or Russel Winder

Document the GPars concurrency concepts

With the gradually increasing adoption of GPars and concurrent programming in general among mainstream developers, we've been constantly hitting the knowledge and terminology barrier, which prevents developers from fully leveraging the powerful concurrency concepts and leaves them with only partial awareness of the available options and their best use. The GPars project would greatly benefit from clear and understandable documentation, including a set of topical tutorials, how-to's as well as an improved User Guide and code samples. Writing these documents as well as coming up with good ways to communicate the principles and their practical use to the GPars users is the main focus of this assignment.

Project: GPars

Possible mentor: Václav Pech or Russel Winder

Groovy to JavaScript transpiler

Groovy runs on the JVM, but we would also like to see a Javascript backend. Of course, a prototype would have limited capabilities compared to what the JVM one offers, mostly because the Groovy language is not only a language but also a rich API relying internally on JDK APIs. A project

called GrooScript is already working, covering various areas of the Groovy language and semantics: http://www.grooscript.org/

The idea of this GSoC project is to further support Groovy language and semantics, improve existing support for various constructs, and optimize the generated code to get good performance out of the generated JavaScript code.

Project: Groovy / GrooScript

Possible mentor: Jorge Franco