Groovy Academy

Fabian Förster

- Seit 1998 Entwickler
  - E-Post (seit ...)
- Schwerpunkt Java & Groovy
- Speaker by: Groovy ...

Sven Ehmann

- Seit 2003 Entwickler
  - Electric Paper
  - Brands4Friends (eBay)
  - E-Post (seit 10.2013)
- Schwerpunkt: Java & Scala & Groovy
- Meetup: „Agile Developer Berlin“

Agenda

- Goals
- „Was wollen wir nicht“
- Tools & Frameworks (Vorstellung & Übung)
  - GVM
  - Gradle
  - Lazybones, Ratpack
- Groovy Koans (Coding Session)
- Battleship (Coding Session)

Goals

- Mit Gradle ein Groovy Projekt aufsetzen
- Grundkenntnisse in Groovy
  - Sprachfeatures: JSON, Collections, ...
- Grundkenntnisse in Ratpack
  - Handler, Modul-Binding, Registery...
- Web-App Battleship
- Testen mit Spock

„Was wollen wir nicht“

- Bashing
  Java8 <> Groovy <> Scala
- Bashing
  Play <> Ratpack
- Groovy: optional typed and dynamic

GVM

GVM is a command line tool for managing parallel Versions of multiple Software Development Kits on most Unix based systems.

Quelle: http://gvmtool.net/

GVM

- GVM install
  curl -s get.gvmtool.net | bash

- Gradle install
  gvm install gradle
  gvm list gradle
  See all possible versions, current installed version and as default set version

  gvm install gradle 2.2
  gvm use gradle 2.2
  switch the candidate version for the current shell only.
  gvm default gradle 2.2
  switch permanent to version
  gvm uninstall gradle 2.2

Quelle: http://gvmtool.net/

GVM

- Other helpfull GVM commands:
  gvm current
  To see what is currently in use for all candidates
  gvm help

- GVM uninstall
  rm -rf ~./gvm
  remove from ~/.bashrc, ~/.profile and ~/.zshrc

```
#THIS MUST BE AT THE END OF THE FILE FOR GVM TO WORK!!!
[[ -s "/home/sven/.gvm/bin/gvm-init.sh" ]] && source "/home/sven/.gvm/bin/gvm-init.sh"
```

Quelle: http://gvmtool.net/

Gradle

- What is Gradle ?
  A very flexible general purpose build tool like … .
  - powerful support for multi-project builds
  - powerful dependency management
  - full support of existing Maven or Ivy repository
  - Groovy build scripts
- Gradle install
  - Download binary only distribution: http://gradle.org/gradle-download/
  - Unpacking
  - export GRADLE_HOME to the path that you unpack binary and add GRADLE_HOME/bin to your PATH variable
  - check install by: ***gradle -v***
  - additional options can be set within GRADLE_OPTS

Quelle: https://docs.gradle.org/current/userguide/introduction.html
https://docs.gradle.org/current/userguide/installation.html

Gradle – first steps

- run ***gradle***
- run ***gradle tasks***
  display the „default“ tasks of every gradle project, e.g. init & wrapper
- create ***build.gradle*** script by ***gradle init*** (Notice: all lines in comment !!!)
  add ***task hugo << { println 'hugo task running' }***
  run ***gradle -q hugo***
- ***-q*** suppress Gradle's log messages
- Build script are (Groovy) code !
- Add ***task boss { println „I'm the boss“ }***
  run ***gradle -q***
  change task boss: ***task boss << { println „I'm the boss“ }***
  run ***gradle -q***
  all tasks without **<<** operator run by default
- define default tasks, executed if no other tasks are specified

  defaultTasks 'hugo', 'boss'

Quelle: https://docs.gradle.org/current/userguide/tutorial_using_tasks.html

Gradle – Build script are (Groovy) code !

- Live Coding (use the *-b* option to select another build file)

```
task upper << {
      String someString = 'my_nAMe';
      println "Original: $someString";
      println "Upper case: ${someString.toUpperCase()}"
}
task intro( dependsOn: count ) << { println "I\'m Gradle" }
task count << { 4.times { print " $it " } }
```

- Lazy dependency
  The dependency of task X to task Y is declared before task Y is defined.
  Means, task must not be in ordering.
- Dynamic Tasks

```
4.times { counter ->
   task "dynamicTask$counter" << {
      println "I'm dynamic task number $counter“ }
   };
dynamicTask0.dependsOn dynamicTask2,dynamicTask3
```

Quelle: https://docs.gradle.org/current/userguide/tutorial_using_tasks.html

Gradle – Wrapper, Daemon

The Gradle Wrapper is the preferred way of starting a Gradle build. When you start a Gradle build via the wrapper, Gradle will be automatically downloaded and used to run the build.

- The wrapper should checked into version control.
- Anyone can work, without needing to install Gradle beforehand
- Wrapper guaranteed to use the version of Gradle that the build was designed to work with

Daemon

It is strongly recommended that the Gradle Daemon be enabled on all developer machines.

- - add org.gradle.daemon=true to the GRADLE_HOME/gradle.properties file

Quelle: https://docs.gradle.org/current/userguide/gradle_wrapper.html
https://docs.gradle.org/2.5/userguide/gradle_daemon.html

Gradle – plugins

- IDEA

Generates an IDEA module file. Also generates an IDEA project and workspace file if the project is the root project. One focus of the IDEA plugin is to be open to customization. The plugin provides a standardized set of hooks for adding and removing content from the generated files.

- Other possible plugins:

Java (default :-)), Groovy, Scala, Jetty, Checkstyle, Findbugs, …

Quelle: https://docs.gradle.org/current/userguide/idea_plugin.html
https://plugins.gradle.org/

Lazybones

Lazybones is a command line tool that will bootstrap a project. It allows to create a new project structure for any framework or library for which the tool has a template.

```
gvm install lazybones
Lazybones list
lazybones create ratpack hugo --with-git
```

Quelle: https://github.com/pledbrook/lazybones

Ratpack

Simple, lean & powerful library to build HTTP apps. Built on Java and the Netty event-driven networking engine. The API is optimized for Groovy and Java 8.

Quelle: https://github.com/ratpack/ratpack

Ratpack – continous development

If running the app ***via IDEA***, you must build the project after changing a class to see the change. IDEA does not automatically compile changes when there is an active “run” session.

If running the app via Gradle (i.e. “./gradlew run”) you can open another terminal and recompile via “./gradlew classes”. You do not need to restart the app.

Quelle: https://github.com/ratpack/example-ratpack-gradle-groovy-app

Ratpack – handler

Ratpack is built on top of Netty, a non-blocking HTTP server. Once the Ratpack server is started, it will listen for incoming requests, and for each, will execute all the handlers defined in the handler chain. Conceptually, a handler is just a function that acts on a handling context.

Ratpack is not Servlet API-based, so don’t expect things HttpServletRequest or HttpSession. However, you can think of handlers as a mix of a Servlet and a Filter.

Quelle: http://alvarosanchez.github.io/ratpack-101/

Lib's & Frameworks

Groovy Academy Slides

https://github.com/soehmann/academy

GVM

http://gvmtool.net/

Gradle(Wrapper)

https://docs.gradle.org/current/userguide

https://docs.gradle.org/current/userguide/gradle_wrapper.html

Groovy

http://groovy-lang.org/

http://groovy-lang.org/documentation.html

Lib's & Frameworks

Ratpack

http://ratpack.io

https://github.com/ratpack/hands-on-ratpack

https://github.com/ratpack/example-ratpack-gradle-groovy-app

https://github.com/ratpack

Ratpack with Lazybones

https://github.com/pledbrook/lazybones

http://ratpack.io/manual/current/quick-start.html#using_lazybones_project_templates

Angular

https://angularjs.de/artikel/angularjs-tutorial-deutsch

http://campus.codeschool.com/courses/shaping-up-with-angular-js/contents

Danke