Groovy and **Grails** in **Google App Engine**

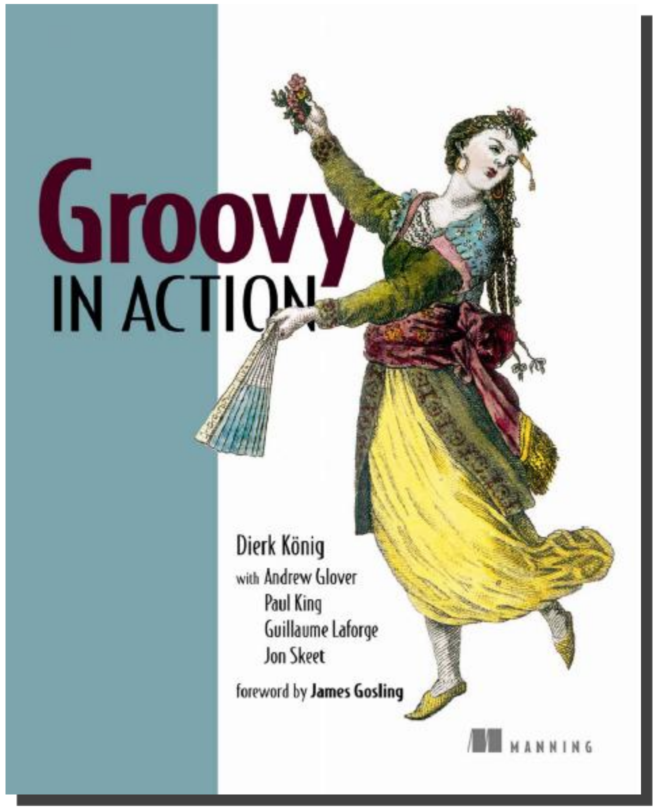
Benefit from a Java-like dynamic language
to be more productive on App Engine

Guillaume Laforge

Head of Groovy Development

Guillaume Laforge

- **Groovy Project Manager**
- JSR-241 Spec Lead
- Head of Groovy Development at **SpringSource**
- Initiator of the **Grails** framework



- Co-author of **Groovy in Action**
- Speaker: JavaOne, QCon, JavaZone, Sun TechDays, Devoxx, The Spring Experience, JAX, Dynamic Language World, IJTC, and more...

Agenda

- **Groovy overview**
  - **Features at a glance**
  - **Groovlets**
  - **Groovy templates**
- Grails overview
  - The Grails stack
  - The App Engine plugin
- Summary
- Q&A

A few words about Groovy

- Groovy is a **dynamic** language for the JVM
  - with a **Meta Object Protocol**
  - compiles directly to bytecode, seamless Java interop
- Open Source **ASL 2** project hosted at Codehaus
- Relaxed grammar **derived from Java 5**
  - Java 5 features out-of-the-box: annotations, generics, static imports, enums, varargs...
  - borrowed good ideas from Ruby, Python, Smalltalk
  - flat learning curve for Java developers

Features at a glance

- Fully Object-Oriented
- **Joint compiler**: seamless Java integration
- **Closures**: reusable/assignable code blocks
- **Properties**: no more getters / setters
- Optional typing: you decide!
- Various shortcut notations: less verbose than Java
- Handy wrapper APIs
  - **Groovlets**, XML, JDBC, JMX, **template engine**, Swing UIs, collections...
- Strong ability for authoring Domain-Specific Languages

A “normal” servlet

- Writing a servlet is no different than a classical standard Java servlet
  - only a bit leaner: properties, semis, public keyword...

-
```
import javax.servlet.http.*

class HelloServlet extends HttpServlet {
    void doGet(HttpServletRequest req,
               HttpServletResponse resp)
        throws IOException {
        resp.contentType = "text/plain"
        resp.writer.println "Hello World!"
    }
}
```

Groovylets: Groovy servlets

- General idea
  - use scripts over classes
  - use Groovy’s markup builder to easily create HTML

-
```
html.html {
     head {
          title "Hello"
     }
     body {
          p "Hello Groovy World!"
     }
 }
```

Groovylets: Groovy servlets

- Refererence the GroovyServlet in web.xml
- Add a URL mapping

-
  ```
  <servlet>
    <servlet-name>GroovyServlet</servlet-name>
      <servlet-class>
        groovy.servlet.GroovyServlet
      </servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>GroovyServlet</servlet-name>
    <url-pattern>*.groovy</url-pattern>
  </servlet-mapping>
  ```

Groovy templates

- Groovy comes built-in with templating capability

-
  ```
  import groovy.text.SimpleTemplateEngine

  def text = '''Dear $firstname,
  So nice to meet you in <% print city %>.
  See you in ${month},
  <%= signed %>'''

  def binding = [firstname: "Guillaume", month: "June",
                 city: "San Francisco", signed: "Duke"]

  def engine = new SimpleTemplateEngine()

  template = engine.createTemplate(text)
                   .make(binding)
  ```

The Servlet template

- Similar to JSPs

  -
    ```
    <html>
      <body>
        <p>${message}</p>
      </body>
    </html>
    ```

- In your web.xml

  -
    ```
    <servlet>
      <servlet-name>GroovyTemplateServlet</servlet-name>
        <servlet-class>
          groovy.servlet.TemplateServlet
        </servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>GroovyTemplateServlet</servlet-name>
      <url-pattern>*.gtpl</url-pattern>
    </servlet-mapping>
    ```

F GAELYK

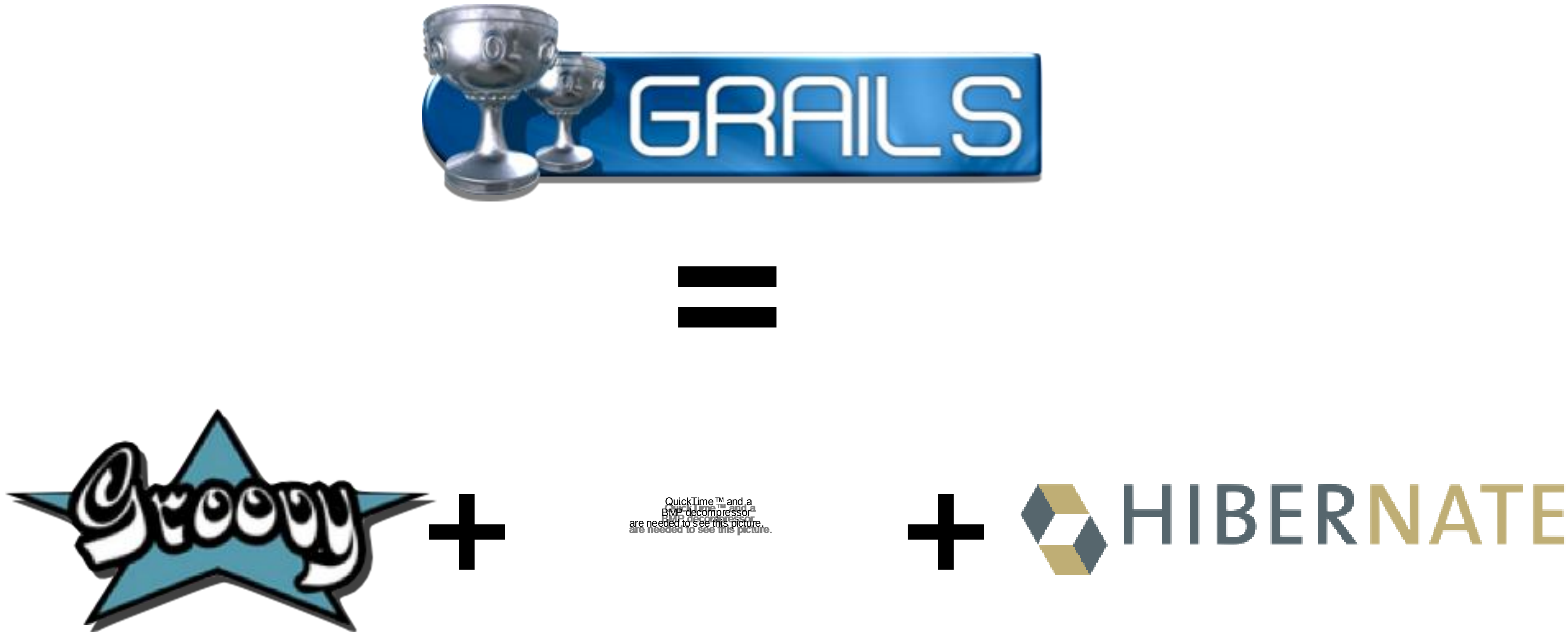
- A thin layer on top of:
  - Groovlets & servlet template
  - Google App Engine Java SDK APIs

- What it does
  - Inject all the services in servlets and templates
  - Adds syntax sugar and dynamic methods to the SDK API

Agenda

- Groovy overview
  - Features at a glance
  - Groovlets
  - Groovy templates
- **Grails overview**
  - **The Grails stack**
  - **The App Engine plugin**
- Summary
- Q&A

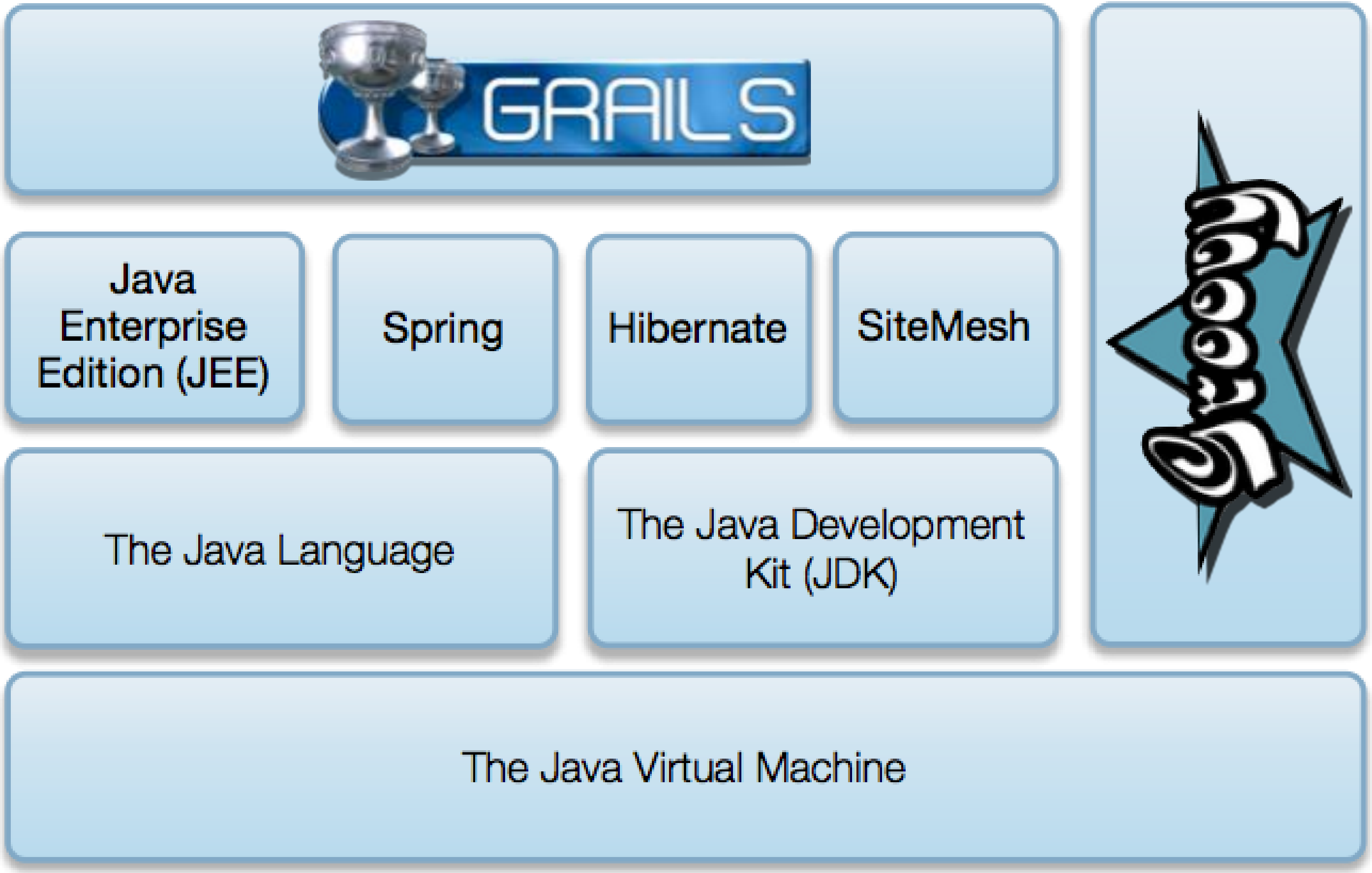
From 10,000 feet…



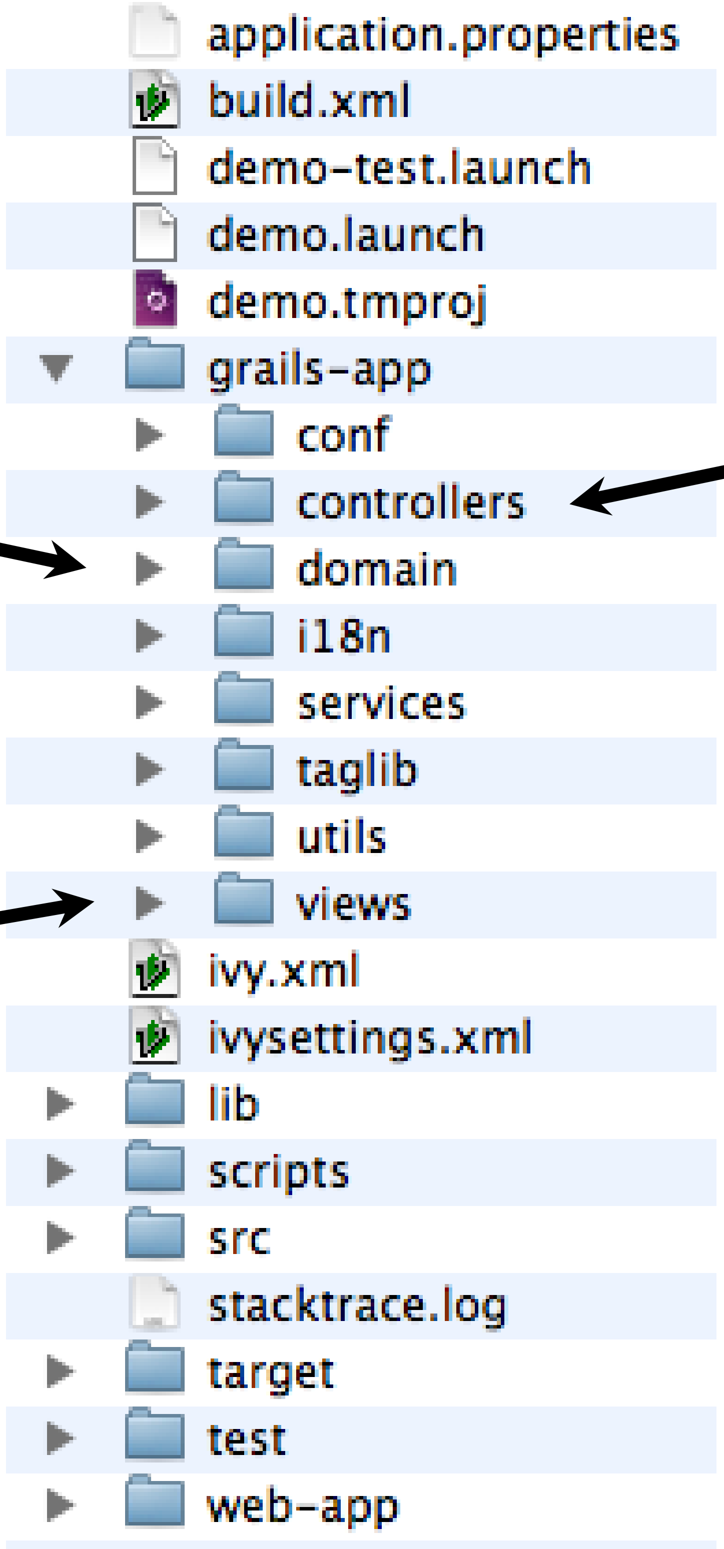
In a few bullet points…

- Grails is a **full stack web app development env.**
  - built on **rock-solid** and **proven OSS frameworks**
    - **Spring**, Spring MVC, Spring WebFlow
    - **Hibernate**, SiteMesh, Quartz
    - Jetty, HSQLDB
  - provides its ORM layer on top of Hibernate
  - has got its own integrated build system and CLI
- Follows good and lean principles
  - **Convention over Configuration**, MVC, DRY…
- Provides powerful time-saving capabilities
  - Scaffolding, an extensible plugin system, taglibs, templates, and more

The big picture



The standard project structure



```
application.properties
build.xml
demo-test.launch
demo.launch
demo.tmproj
grails-app
    conf
    controllers
    domain
    i18n
    services
    taglib
    utils
    views
ivy.xml
ivysettings.xml
lib
scripts
src
stacktrace.log
target
test
web-app
```

M

C

V

The App Engine plugin

- Grails has got a very extensive and rich plugin system and ecosystem
  - 100+ existing and useful plugins available
    - GWT, Flex, YUI, iUI, JMS, search, security, testing...
- A Google App Engine plugin was developed
  - following the Grails development flow
  - abstracts the app engine commands for deployments
  - leverages the local engine

Setting up your project

- Create your Grails application as usual
  - `grails create-app myapp`

- Specify the App Engine app id
  - `google.appengine.application = "myapp"`

- Uninstall hibernate and install the GAE plugin

  -
    ```
    grails uninstall-plugin hibernate
    grails install-plugin app-engine
    ```

- Run the application locally
  - `grails app-engine run`

Deploying applications

- Set the version number to 1, as Grails uses a different numbering scheme (x.y)
  - `grails set-version 1`
- Get your application ready
  - `grails app-engine package`
- On first deployment
  - `$APPENGINE_HOME/appcfg.sh update ./target/war`
- For subsequent deployments
  - `grails app-engine deploy`

DEMO

Agenda

- Groovy overview
  - Features at a glance
  - Groovlets
  - Groovy templates
- Grails overview
  - The Grails stack
  - The App Engine plugin
- **Summary**
- Q&A

Summary

- **Groovy**
  - Groovy is a first-class citizen on Google App Engine
  - Lots of shorcut notations can streamline your code
  - Groovy servlets and templates are great for crafting small applications
  - Gaelyk further simplifies this pure Groovy approach

- **Grails**
  - Grails is the next-generation web development stack
  - Developing Grails application on the Google cloud is almost as lean and simple as for the usual crop of application servers

Further information

- Groovy in Google App Engine
  - http://blog.springsource.com/2009/04/07/write-your-google-app-engine-applications-in-groovy/

- Grails in Google App Engine
  - the Grails 1.1.1 release and App Engine screencast
    - http://blog.springsource.com/2009/05/14/grails-111-released-with-google-appengine-support/
  - the Google App Engine Grails plugin
    - http://grails.org/plugin/app-engine

Agenda

- Groovy overview
  - Features at a glance
  - Groovlets
  - Groovy templates
- Grails overview
  - The Grails stack
  - The App Engine plugin
- Summary
- **Q&A**

Q&A

- Questions & perhaps...

  even answers :-)