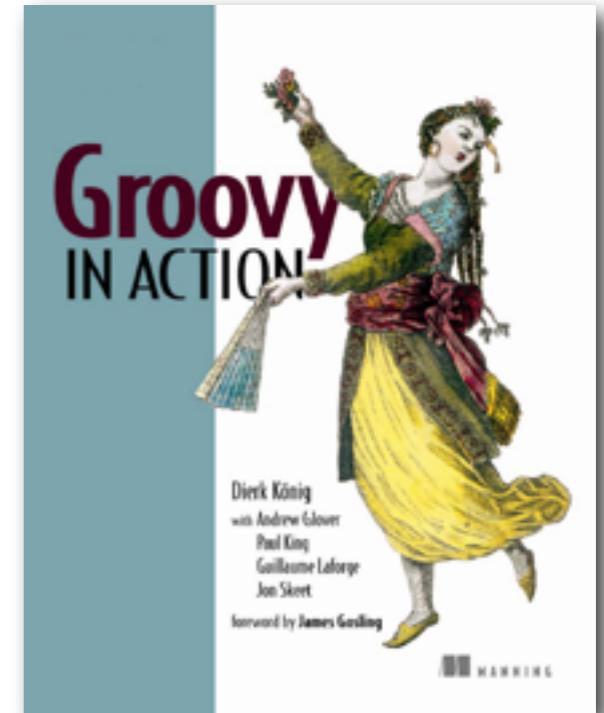


Google App Engine, Groovy and Gaelyk



- **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, SpringOne, JAX, Dynamic Language World, IJTC, and more...



Marc-Antoine Guerrigue & Gaël Lazzari



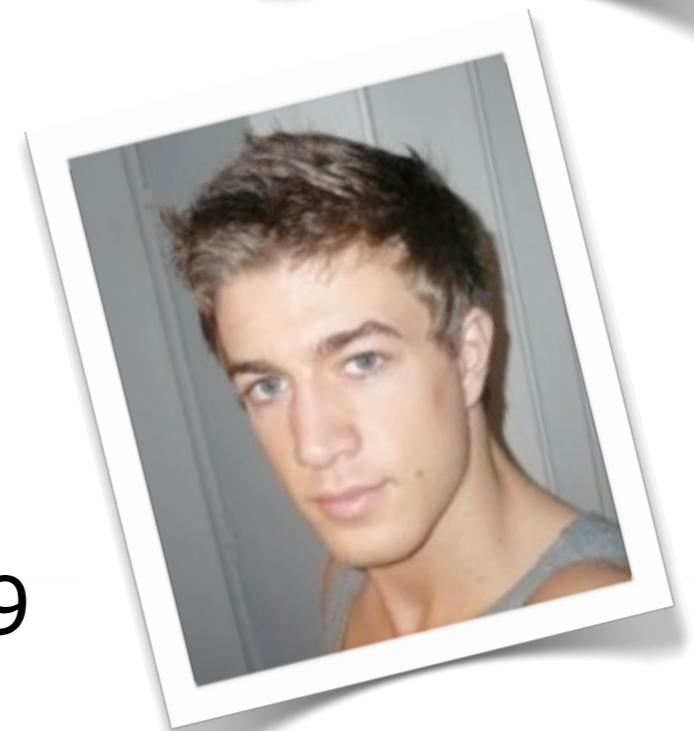
● Marc-Antoine

- Architecte et **Directeur de la R&D chez OCTO Technology** depuis 2001
 - Missions d'architecture d'applications Java EE
 - «Open Sourceur» dans l'âme
 - JCaptcha
 - Membre de l'OSSGTP
 - Développement des sociétés
 - MassiveBrainGames.com
 - XDepend.com



● Gaël Lazzari

- Consultant OCTO depuis septembre 2009





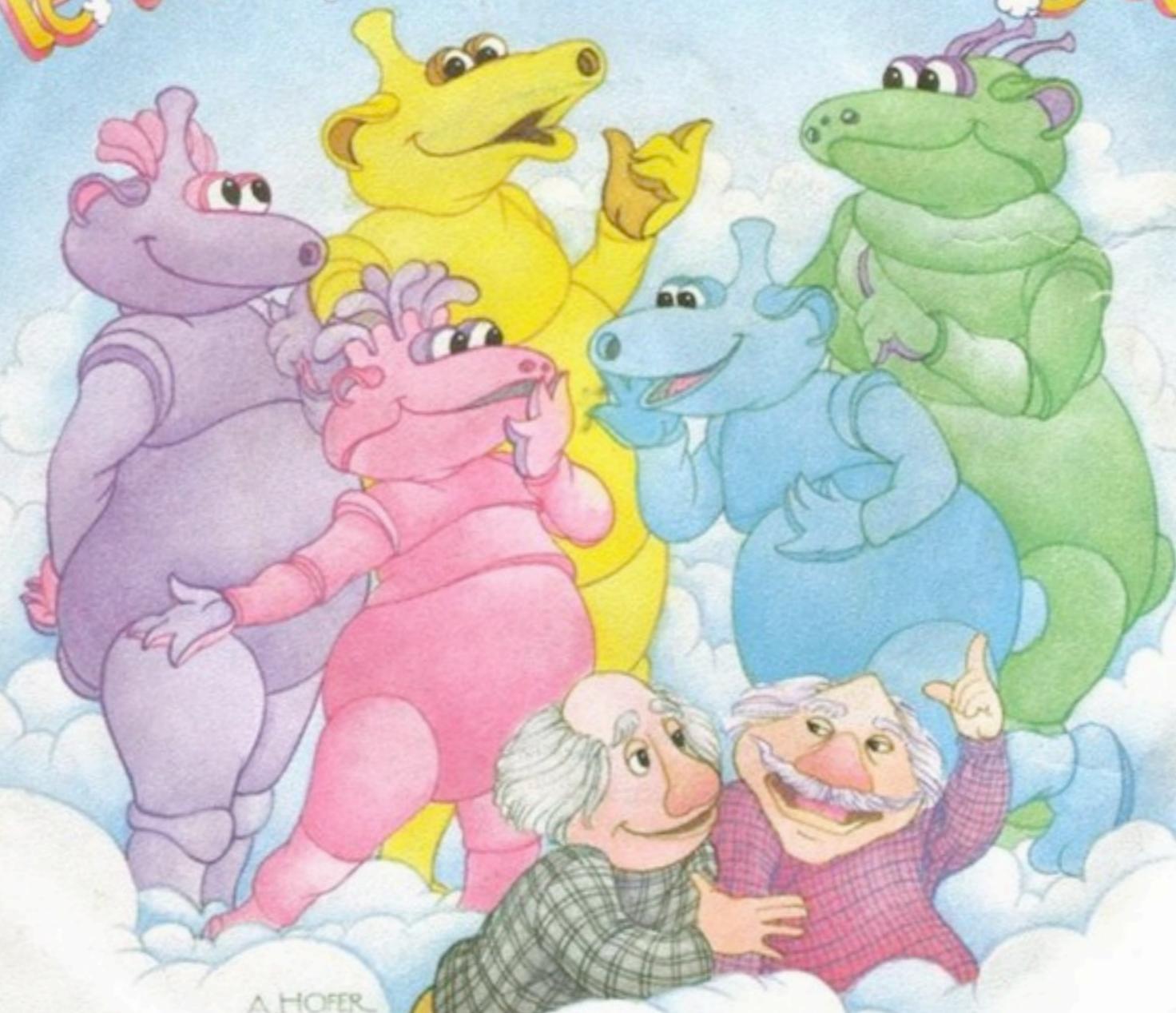
jeudi 12 novembre 2009

Cloud Computing

GÉNÉRIQUE ORIGINAL DE L'ÉMISSION TÉLÉVISÉE DE CHRISTOPHE IZARD 11.000

Le village dans les nuages

tf1



A.HOFER

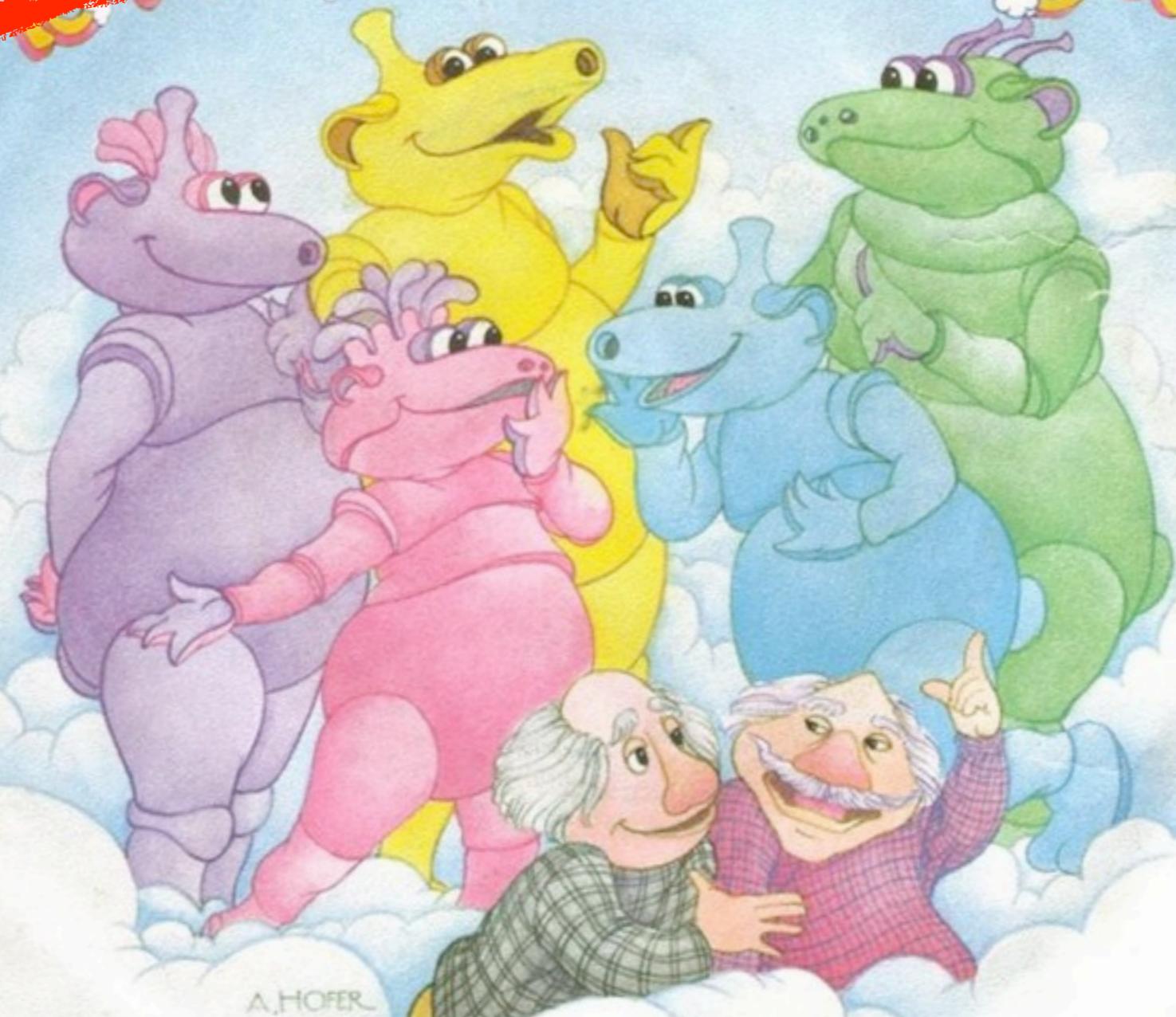
disques
Adès

45^T

© TF1 1982

GÉNÉRIQUE ORIGINAL DE L'ÉMISSION TÉLÉVISÉE DE CHRISTOPHE IZARD 11.000

~~Le village dans les nuages~~ dans les nuages **tf1**



A.HOFER

disques
Adès

45^T

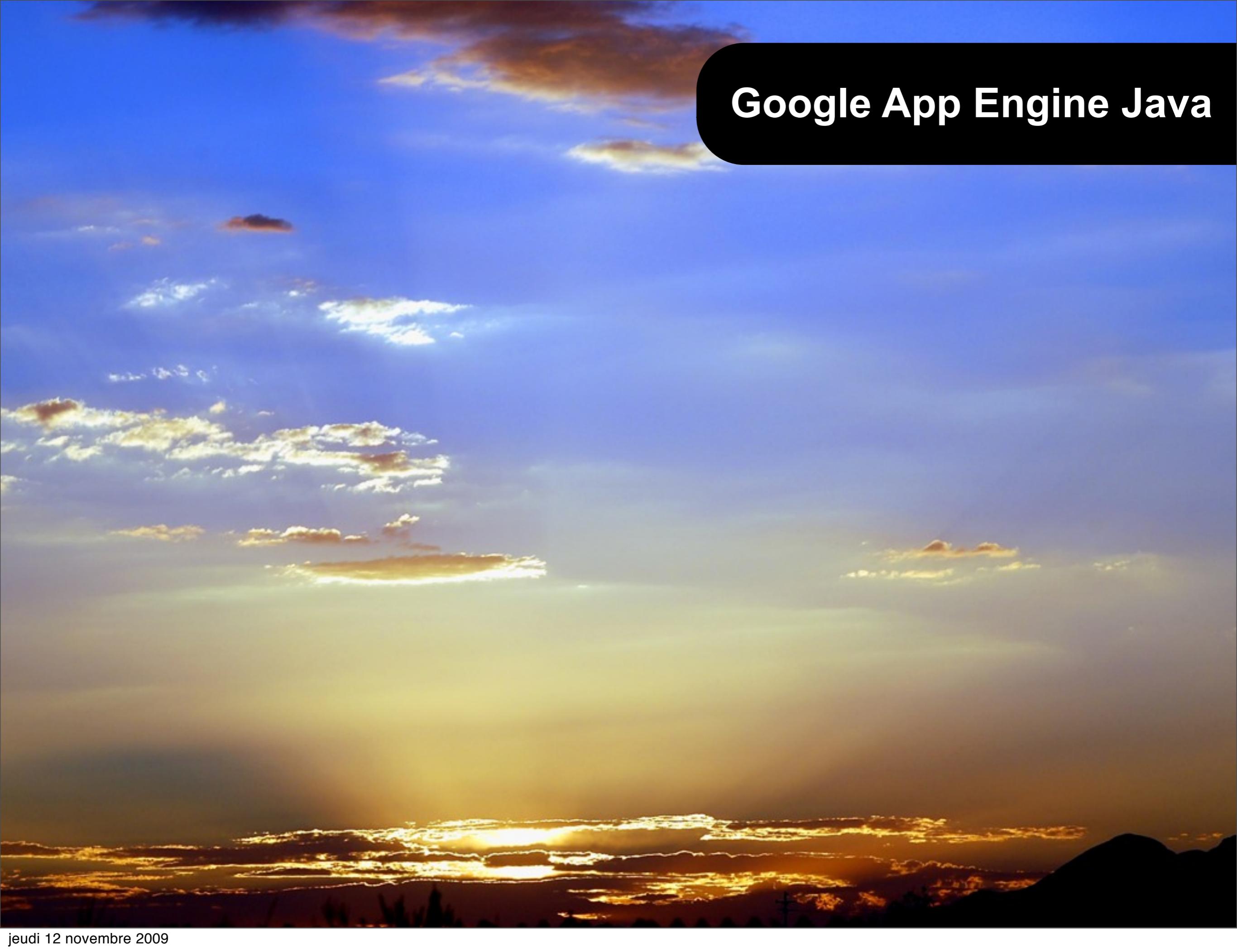
© TF1 1982

l'informatique



- Software as a Service
 - Gmail, SalesForce.com
- Platform as a Service
 - Google App Engine
- Infrastructure as a Service
 - Amazon EC2



A wide-angle photograph of a sunset over a valley. The sky is filled with large, wispy clouds illuminated from below by the setting sun, transitioning from deep orange and yellow at the horizon to a clear blue above. In the foreground, dark silhouettes of mountain peaks are visible against the bright sky.

Google App Engine Java

Google App Engine Java



- Google's PaaS solution
- Run your app on Google's infrastructure
- Initially just Python supported
- Since this year, **Java** supported  pythonTM
 - Sandboxed **JVM**
 - Jetty **servlet container**
- Several JVM-compatible language supported
- A few services available
 - Email, XMPP, URL Fetch, Image, User, Task queues...



Key aspects

- You can use most of your usual web frameworks for developping apps on App Engine Java
 - **A WAR file, basically!**
- No OS image, or software to install
 - Unlike with Amazon EC2
- All the scaling aspects are handled for you
 - Database / session replication, load balancing
- There are quotas, but you need a high traffic application to start being charged
 - **Free to get started**

Available services

● Memcache

- JCache implementation
- Save on CPU and DB

● URL Fetch

- Access remote resources
- HttpURLConnection

● Mail

- Support both incoming and outgoing emails

● Images

- Resize, crop, rotate...

● XMPP

- Send / receive Jabber messages (GTalk)

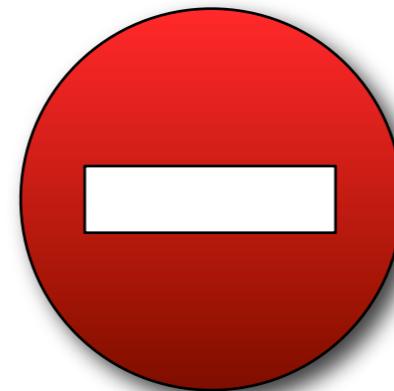
● User

- Use Google's user/authentication system

● Cron & Task queues

- Schedule tasks at regular intervals
- Queue units of work

- **Not our usual relational database**
 - key / value datastore
- **30 seconds request duration limit**
- Forbidden to
 - write on the file system
 - create threads
 - use raw sockets
 - issue system calls
 - use IO / Swing / etc. directly
 - There's a whitelist of classes allowed
- Number of files and their size are limited



Quotas



Quotas (1/2)

● Bandwidth

- 1,3M requests/day
- 1GB/day in/out
- 6.5 CPU hours/day

● Datastore

- 10M calls
- 1GB/day
- 12GB in / 115GB out
- 60 CPU hours/day

● Mail

- 7K calls/day
- 2K recipients/day
- 5K emails/day
- 2K attachments
- 100MB of attachments

● URL Fetch

- 657K calls/day
- 4GB in/out /day

Quotas (2/2)

- **XMPP**

- 657K calls/day
- 4GB data sent/day
- 657K recipients/day
- 1MB invitations/day

- **Image manipulation**

- 864 calls/day
- 1GB in / 5GB out
- 2.5M transforms

- **Memcache**

- 8.6M calls/day
- 10GB in
- 50GB out

- **Task queues**

- 100K calls

Dashboard – Groovy Console

https://appengine.google.com/dashboard?app_id=groovyconsole&version_id=10.3365

Google app engine

groovyconsole 10

Main

- [Dashboard](#)
- [Quota Details](#)
- [Logs](#)
- [Cron Jobs](#)
- [Task Queues](#)

Datastore

- [Indexes](#)
- [Data Viewer](#)
- [Statistics](#)

Administration

- [Application Settings](#)
- [Developers](#)
- [Versions](#)
- [Admin Logs](#)

Billing

- [Billing Settings](#)
- [Billing History](#)

Resources

- [Documentation](#)
- [FAQ](#)
- [Developer Forum](#)
- [Downloads](#)
- [System Status](#)

Charts

Requests/Second

all 24 hr 12 hr 6 hr

Quotas reset every 24 hours. Next reset: 9 hrs

Billing Status: Free - [Settings](#)

Resource	Usage	
CPU Time	6%	0.40 of 6.50 CPU hours
Outgoing Bandwidth	3%	0.03 of 1.00 GBytes
Incoming Bandwidth	0%	0.00 of 1.00 GBytes
Stored Data	0%	0.00 of 1.00 GBytes
Recipients Emailed	0%	0 of 2000

Current Load

URI	Requests last 15 hrs	Avg CPU (API) last hr	% CPU last 15 hrs
/recentscripts.gtpl	39	3650 (1966) ⚠	68%
/executor.groovy	23	500 (0)	6%
/view.groovy	18	1371 (134) ⚠	13%
/	14	1315 (125) ⚠	9%
/atom.groovy	12	661 (124)	4%
/robots.txt	3	0 (0)	0%

Errors

URI	Count	% Errors last 15 hrs
/robots.txt	3	100%

Nice dashboard

The Datastore



The datastore...

- **It's not your father's relational database!**
 - You're not even using SQL!
- **Distributed key / value store**
 - Based on Google's «BigTable»
- Supporting
 - Transactions and partitioning
 - Hierarchies through entity groups
- **Schema-less** approach
- Data access APIs
 - JPA and JDO
 - Direct low-level APIs

...and its «limitations»

- You're not using SQL
 - **No joins**
 - No database constraints
 - No aggregation functions (count, avg...)
- You can **only retrieve 1000 records** per query
- In a query, you can **only filter on one column** for inequality
- Transactions only available in entity groups
- You can only update an entity once in a transaction



jeudi 12 novembre 2009



GAEILYK





gaelyk

<http://gaelyk.appspot.com>





GAElyk

<http://gaelyk.appspot.com>

The screenshot shows a web browser window displaying the Gaelyk tutorial page at <http://gaelyk.appspot.com/tutorial/>. The browser's address bar shows the URL. The page has a header with the Gaelyk logo and navigation links for Home, Tutorial, Download, and Community. A Groovy logo is also present. The main content area is titled "Tutorial" and contains text explaining the goal of the tutorial, which is to get started with using Gaelyk to write and deploy Groovy applications on Google App Engine. It assumes the user has already downloaded and installed the Google App Engine SDK. The easiest way to get setup rapidly is to download the template project from the "download section". The page lists several files included in the template project: web.xml, appengine-web.xml, a sample Groovlet and template, and the needed JARs (Groovy, Gaelyk and Google App Engine SDK). It also mentions that you can browse the JavaDoc of the classes composing Gaelyk. Below this, there are sections for "Setting up your project" and "Directory layout", with a note about following the directory layout proposed by the Gaelyk template project.

Gaelyk - a lightweight Groovy toolkit for Google App Engine Java

http://gaelyk.appspot.com/tutorial/

Home Tutorial Download Community

Tutorial

The goal of this tutorial is to quickly get you started with using **Gaelyk** to let you write and deploy your Groovy applications on Google App Engine. We'll assume you have already downloaded and installed the Google App Engine SDK of your machine. If you haven't, please do so by reading the [instructions](#) from Google.

The easiest way to get setup rapidly is to download the template project from the [download section](#). It provides a ready-to-go project with the right configuration files pre-filled and an appropriate directory layout:

- `web.xml` preconfigured with the **Gaelyk** servlets
- `appengine-web.xml` with the right settings predefined (static file directive)
- a sample Groovlet and template
- the needed JARs (Groovy, Gaelyk and Google App Engine SDK)

You can [browse the JavaDoc](#) of the classes composing **Gaelyk**.

Setting up your project

Directory layout

We'll follow the directory layout proposed by the **Gaelyk** template project:

Why Groovy?



- **Groovy** is a **dynamic language** for the JVM
 - very flexible, malleable, expressive and **concise** syntax
 - easy to learn for Java developers
 - deriving from the Java 5 grammar
 - provides powerful APIs to simplify the life of developers
 - possibility to **dynamically enrich existing APIs**
 - support for **Groovlets**
 - features its own **template engine**
- We worked with the Google App Engine Java team before the official launch of the platform, to ensure Groovy would run well on this new environment



- Gaelyk is a **lightweight Groovy toolkit** on top of the Google App Engine Java SDK
- Gaelyk builds on Groovy's servlet support
 - **Groovlets**: Groovy scripts instead of raw servlets!
 - **Groovy templates**: JSP-like template engine
 - Both allow for a clean separation of views and logic
- Gaelyk provides several **enhancements** around the GAE Java SDK to make life easier, thanks to Groovy's dynamic nature



First steps...

- Go to <http://gaelyk.appspot.com>
- Download the template project
- Put your first Groovlet in /WEB-INF/groovy
- And your templates at the root
- And you're ready to go!
- Launch dev_appserver.sh
- Go to <http://localhost:8080/>

The web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
  <servlet>
    <servlet-name>GroovletServlet</servlet-name>
    <servlet-class>groovyx.gaelyk.GaelykServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TemplateServlet</servlet-name>
    <servlet-class>groovyx.gaelyk.GaelykTemplateServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>GroovletServlet</servlet-name>
    <url-pattern>*.groovy</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TemplateServlet</servlet-name>
    <url-pattern>*.gtpl</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.gtpl</welcome-file>
  </welcome-file-list>
</web-app>
```

● Variables available

- request / response
- context / applicaiton
- session
- params
- header
- out / sout / html

● Methods available

- include / forward
- print / println

● Google services

- datastoreService
- memcacheService
- urlFetchService
- mailService
- userService
- user
- defaultQueue
- queues
- xmppService



Groovy sugar!

```
mailService.send to: 'admin@example.com',  
from: 'user@example.com',  
subject: 'Hello World',  
htmlBody: '<bold>Hello</bold>'
```

...compared to Java



```
Properties props = new Properties();
Session session = Session.getDefaultInstance(props, null);

String msgBody = "...";

try {
    Message msg = new MimeMessage(session);
    msg.setFrom(new InternetAddress("admin@example.com", "Admin"));
    msg.addRecipient(Message.RecipientType.TO,
                     new InternetAddress("user@example.com", "Mr. User"));
    msg.setSubject("Your Example.com account has been activated");
    msg.setText(msgBody);
    Transport.send(msg);
} catch (AddressException e) {}
} catch (MessagingException e) {}
```

A groovlet

- Instead of writing full-blown servlets, just write Groovy scripts (aka Groovlets)

```
def numbers = [1, 2, 3, 4]
def now = new Date()

html.html {
    body {
        numbers.each { number -> p number }
        p now
    }
}
```

A template

```
<html>
  <body>
    <p><%
        def message = "Hello World!"
        print message %>
    </p>
    <p><%= message %></p>
    <p>${message}</p>
    <ul>
      <% 3.times { %>
        <li>${message}</li>
      <% } %>
    </ul>
  </body>
</html>
```

Accessing the datastore

- Direct interaction with the low-level datastore APIs

```
import com.google.appengine.api.datastore.Entity  
  
Entity entity = new Entity("person")  
  
// subscript notation, like when accessing a map  
entity['name'] = "Guillaume Laforge"  
  
// normal property access notation  
entity.age = 32  
  
entity.save()  
entity.delete()  
  
datastoreService.withTransaction {  
    // do stuff with your entities  
    // within the transaction  
}
```

Querying to be improved...



```
import com.google.appengine.api.datastore.*  
import static com.google.appengine.api.datastore.FetchOptions.Builder.*  
  
// query the scripts stored in the datastore  
def query = new Query("savedscript")  
  
// sort results by descending order of the creation date  
query.addSort("dateCreated", Query.SortDirection.DESCENDING)  
  
// filters the entities so as to return only scripts by a certain author  
query.addFilter("author", Query.FilterOperator.EQUAL, params.author)  
  
PreparedQuery preparedQuery = datastoreService.prepare(query)  
  
// return only the first 10 results  
def entities = preparedQuery.asList( withLimit(10) )
```

...into something groovier?



```
def entities = Query.create {  
    select from: savedscript  
    sort DESC, on: dateCreated  
    where { author == params.author }  
    limit 10  
} as List
```

...into something groovier?



```
def entities = Query.create {  
    select from: savedscript  
    sort DESC, on: dateCreated  
    where { author == params.author }  
    limit 10  
} as List
```

Not Yet Implemented!

Task queue API

```
// access a configured queue using the subscript notation
queues['dailyEmailQueue']

// or using the property access notation
queues.dailyEmailQueue

// you can also access the default queue with:
queues.default
defaultQueue

// add a task to the queue
queue << [
    countdownMillis: 1000, url: "/task/dailyEmail",
    taskName: "Send daily email newsletter",
    method: 'PUT', params: [date: '20090914'],
    payload: content
]
```

- Sending instant messages

```
String recipient = "someone@gmail.com"

// check if the user is online
if (xmppService.getPresence(recipient).isAvailable()) {
    // send the message
    def status = xmppService.send(to: recipient,
                                    body: "Hello, how are you?")

    // checks the message was successfully
    // delivered to all the recipients
    assert status.isSuccessful()
}
```

- Sending instant messages with an XML payload

```
String recipient = "service@gmail.com"  
  
// check if the service is online  
if (xmppService.getPresence(recipient).isAvailable()) {  
    // send the message  
    def status = xmppService.send to: recipient, xml: {  
        customers {  
            customer(id: 1) {  
                name 'Google'  
            }  
        }  
    }  
  
    // checks the message was successfully delivered to the service  
    assert status.isSuccessful()  
}
```

- Sending instant messages with an XML payload

```
String recipient = "service@gmail.com"  
  
// check if the service is online  
if (xmppService.getPresence(recipient).isAvailable()) {  
    // send the message  
    def status = xmppService.send to: recipient, xml: {  
        customers {  
            customer(id: 1) {  
                name 'Google'  
            }  
        }  
    }  
  
    // checks the message was successfully delivered to the service  
    assert status.isSuccessful()  
}
```

```
<customers>  
    <customer id='1'>  
        <name>Google</name>  
    </customer>  
</customers>
```

- Receiving incoming instant messages
 - Configure the XmppServlet in web.xml
 - Add the inbound message service in appengine-web.xml

```
// get the body of the message
message.body

// get the sender Jabber ID
message.from

// get the list of recipients Jabber IDs
message.recipients

// if the message is an XML document instead of a raw string message
if (message.isXml()) {
    // get the raw XML
    message.stanza

    // get a document parsed with XmlSlurper
    message.xml
}
```

What's coming next?

- Add more sugar around...
 - The Memcache service
 - The incoming email support in GAE SDK 1.2.6
 - The Datastore query system
 - SQL-like DSL
 - dynamic finders
- More generally...
 - Anything that'll come up in upcoming GAE SDK versions

Summary

- Easy access to a cloud solution
 - Deploying Java apps, as easily as you would with PHP
- Familiar to Java folks
 - Your good old Servlet centric webapps style
- Pretty cheap
 - You need a high-trafficed website to reach the quotas
- Gaelyk provides a simplified approach to creating Servlet centric webapps in a productive manner
 - Leveraging Groovy's servlet / template support and dynamic capabilities



Guillaume Laforge
Head of Groovy Development
glaforge@gmail.com

Q&A

References:

- <http://code.google.com/appengine/>
- <http://gaelyk.appspot.com/>
- <http://groovy.codehaus.org/>
- <http://grails.org/>

Images utilisées dans cette présentation



- Nuages
 - <http://www.morguefile.com/archive/display/627059>
 - <http://www.morguefile.com/archive/display/625552>
 - <http://www.morguefile.com/archive/display/629785>
- Le village dans les nuages
 - [http://www.collectoy.com/photoBDD/Disque/Le%20village%20dans%20les%20nuages%20\(%20face%20\).jpg](http://www.collectoy.com/photoBDD/Disque/Le%20village%20dans%20les%20nuages%20(%20face%20).jpg)
- Duke ok GAE
 - <http://code.google.com/images/duke-on-gae.jpg>
 - http://weblogs.java.net/blog/felipegaudio/archive/ae_gwt_java.png
- Python logo : <http://python.org/images/python-logo.gif>
- Gaelyc cross with clouds : <http://www.morguefile.com/archive/display/37889>
- Speed limit : <http://www.morguefile.com/archive/display/18492>
- Warehouse : <http://www.morguefile.com/archive/display/85628>
- Snow foot steps : <http://www.flickr.com/photos/robinvanmourik/2875929243/>
- Sugar : <http://www.flickr.com/photos/ayelie/441101223/sizes/l/>