# ANDROID ON GROOVY

Alexander (Sascha) Klein <alexander.klein@codecentric.de

# ALEXANDER (SASCHA) KLEIN

Principal Consultant
codecentric AG
Stuttgart

Groovy, JavaFX, UI / UX

Griffon committer

alexander.klein@codecentric.de
@saschaklein
http://gplus.to/karfunkel

## ANDROID

2003 Android, Inc.

2005 Google

Open Source (Apache 2.0)

## TOOLS

Android SDK

Gradle

Android Studio

## APP ENTWICKLUNG

- C/C++
- Java
    - Java 6 SE ( Java 7 Sprachfeatures)
    - ohne AWT/Swing
    - ohne java.beans.* und andere
    - Android API

## GROOVY

2003 James Strachan

2007 V 1.0 / G2One Inc.

2008 SpringSource

2013 Pivotal

2015 Apache Incubator

Open Source (Apache 2.0)

## WAS IST GROOVY

Programmiersprache

Skriptsprache

läuft auf der JVM

98% abwärtskompatibel zu Java

Voll mit Java integriert

## WARUM GROOVY ?

- moderneres Java
  - Python, Ruby, Perl, Smalltalk
- objekt-orientiert
- funktional
- optional typisiert
- reduziert Boilerplate-Code
- pragmatisch

# JAVA VS. GROOVY

## Java Code

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    void onClick(View v) {
        startActivity(intent);
    }
});
```

## Groovy Code

```
button.onClickListener = {
    startActivity(intent)
}
```

codecentric

# JAVA VS. GROOVY

## Java Code

```java
public class User {
    private String name;

    String getName() {
        return name;
    }
    void setName(String name) {
        this.name = name;
    }
    String toString() {
        return "User(" + name + ")";
    }
}

User user = new User();
user.setName("Sascha");
System.out.println(user.getName());
```

# JAVA VS. GROOVY

## Groovy Code

```groovy
@ToString
class User {
    String name
}

def user = new User(name: "Hans")
user.name = "Sascha"
println user.name
```

## AST-TRANSFORMATIONEN

### Java Code

```java
public final class ToBeImmutable {
    private final String variable;

    public ToBeImmutable(String variable) {
        this.variable = variable;
    }

    public String getVariable() { return variable; }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result +
               ((variable == null) ? 0 : variable.hashCode());
        return result;
    }
```

# AST-TRANSFORMATIONEN

## Java Code

```java
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        ToBeImmutable other = (ToBeImmutable) obj;
        if (variable == null) {
            if (other.variable != null) return false;
        } else if (!variable.equals(other.variable))
            return false;
        return true;
    }
    @Override
    public String toString() {
        return "ToBeImmutable(" + variable + ")";
    }
}
```

# AST-TRANSFORMATIONEN

## Groovy Code

```groovy
@Immutable final class ToBeImmutable {
    String variable
}
```

# AST-TRANSFORMATIONEN

@Singleton

@EqualsAndHashCode

@TupleConstructor

@Canonical

@InheritConstructors

@AutoClone

@Delegate

@Lazy

@Builder

u.v.m.

# TRAITS

```
@CompileStatic
@SelfType(Context)
trait GoogleApiProvider {
    GoogleApiClient googleApiClient
    void createGoogleApi() {
        googleApiClient = new GoogleApiClient.Builder(this)
                                          .addApi(Wearable.API).build()
    }
}
```

```
class MyService extends Service implements GoogleApiProvider {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState)
        contentView = R.layout.activity_presentation
        createGoogleApi()
    }
    // ...
}
```

```
class MyActivity extends Activity implements GoogleApiProvider {
// ...
}
```

# GROOVY EXTENSIONS

## Java Code

```java
import static android.support.v4.app.NotificationCompat.*;
Intent intent = new Intent(this, WearPresentationActivity.class);
PendingIntent pending;
pending = PendingIntent.getActivity(this, 0, intent,
                                                              FLAG_UPDATE_CURRENT);
NotificationCompat.BigTextStyle bigStyle;
bigStyle = new NotificationCompat.BigTextStyle();
bigStyle.bigText("Time left: " + timeLeft);
Bitmap icon;
icon = BitmapFactory.decodeResource(getResources(),R.drawable.speaker);
```

```java
NotificationCompat.Builder builder;
builder = new NotificationCompat.Builder(this);
builder.setSmallIcon(R.drawable.ic_action_alarms)
  .setLargeIcon(icon)
  .setContentTitle("Time left")
  .setContentText(timeLeft + " (Elapsed: "+rounded+"%)")
  .setContentIntent(pending)
  .setStyle(bigStyle);
NotificationManagerCompat manager=NotificationManagerCompat.from(this);
manager.notify(NOTIFICATION_ID, builder.build());
```

# GROOVY EXTENSIONS

## Groovy Code

```groovy
private @Lazy Bitmap cachedBitmap =
            BitmapFactory.decodeResource(resources, R.drawable.speaker)

notify(NOTIFICATION_ID) {
    smallIcon = R.drawable.ic_action_alarms
    largeIcon = cachedBitmap
    contentTitle = 'Time left'
    contentText = "$timeLeft (Elapsed: ${rounded}%)"
    contentIntent = pendingActivityIntent(0,
                 intent(WearPresentationActivity), FLAG_UPDATE_CURRENT)
    ongoing = true
    style = bigTextStyle {
        bigText "Time left: $timeLeft"
    }
}
```

codecentric

## GROOVY EXTENSIONS

### Extension Module

```
@CompileStatic
class ContextGroovyMethods {
  static void notify(Context self, int notificationId,
                                           Notification notification) {
      getNotifyManager(self).notify(notificationId, notification)
  }
  static void notify(Context self, int notificationId,
                     @DelegatesTo(NotificationCompat.Builder)
                     Closure notifySpec) {
      notify(self, notificationId, notification(self, notifySpec))
  }
```

```
  static NotificationManagerCompat getNotifyManager(Context self) {
      NotificationManagerCompat.from(self)
  }
  static Notification notification(Context self,
               @DelegatesTo(NotificationCompat.Builder) Closure spec) {
      def builder = new NotificationCompat.Builder(self)
      builder.with(spec)
      builder.build()
  }
}
```

## GROOVY EXTENSIONS

```
// src/main/resources/org.codehaus.groovy.runtime.ExtensionModule

moduleName=AndroidExtensions
moduleVersion=1.0
extensionClasses=my.extension.package.ContextGroovyMethods
```

ℹ Für @CompileStatic in einem Gradle-Projekt müssen Extension Modules in einem separaten Projekt liegen

## GROOVY AUF ANDROID

## WARUM ?

Groovy ist kompakter → bessere Wartbarkeit

- schlankerer Code
- höhere Ausdruckskraft

Vorwärts-Kompatibilität mit Java8

Funktionaler Programmierstil

Groovy API

Mehrfachvererbung / Traits

Android API kann vereinfacht werden

## WAS ?

Groovy >= 2.4.0

aktuell: 2.4.3

Java 6 JDK

Android SDK

Android Studio

Gradle

# WIE ?

1. AndroidStudio → Groovy
2. Lazybones → AndroidStudio

# ANDROIDSTUDIO

Erzeuge ein neues Projekt mit Android Studio

Passe build.gradle auf Modulebene an:

```
buildscript {
  repositories {
    jcenter()
  }
  dependencies {
    classpath 'com.android.tools.build:gradle:1.2.3'
    classpath 'org.codehaus.groovy:gradle-groovy-android-plugin:0.3.6'
  }
}

apply plugin: 'groovyx.grooid.groovy-android'

dependencies {
  compile 'org.codehaus.groovy:groovy:2.4.3:grooid'
}
```

## ANDROIDSTUDIO

mkdir app/src/main/groovy

mv app/src/main/java app/src/main/groovy

.java-Dateien in .groovy umbenennen

Java Code 'groovyfizieren'

# JAVA → GROOVY

```
public class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

## JAVA → GROOVY

```
class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
```

```
    @Override
    boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu)
        return true
    }
```

```
    @Override
    boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId()
        if (id == R.id.action_settings) {
            return true
        }
        return super.onOptionsItemSelected(item)
    }
}
```

codecentric

## JAVA → GROOVY

```
class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState)
        contentView = R.layout.activity_main
    }
```

```
    @Override
    boolean onCreateOptionsMenu(Menu menu) {
        menuInflater.inflate(R.menu.menu_main, menu)
        return true
    }
```

```
    @Override
    boolean onOptionsItemSelected(MenuItem item) {
        int id = item.itemId
        if (id == R.id.action_settings) {
            return true
        }
        return super.onOptionsItemSelected(item)
    }
}
```

## LAZYBONES

gvm installieren

```
$ curl -s get.gvmtool.net | bash
```

lazybones installieren

```
$ gvm install lazybones
```

Android Lazybones templates konfigurieren

```
bintrayRepositories = [
    "marioggar/grooid-templates",
    "pledbrook/lazybones-templates"
]
```

## LAZYBONES

Neues Projekt anlegen

```
$ lazybones create grooid-new-project MyApp
```

Android Studio > File > New > Import Project ...

Bei Bedarf Versionen in MyApp/build.grade anpassen

Bringt noch hilfreiche Bibliotheken mit

## ANDROID API VEREINFACHEN

# ASYNCTASK - JAVA

## AsyncTask

```
final long waitTime = 5000;

button.setOnClickListener(new View.OnClickListener() {
    @Override
    void onClick(View v) {
        new AsyncTask<Void, Long, String>() {

            @Override protected String doInBackground(Void[] params) {
                long time = waitTime + 1000;
                while (time > 1000) {
                    if (isCancelled()) break;
                    time -= 1000;
                    publishProgress(time);
                    Thread.sleep(1000);
                }
                return getResources().getString(R.string.finished);
            }
```

# ASYNCTASK - JAVA

## AsyncTask

```
@Override protected void onPreExecute() {
    String text = getResources().getString(R.string.wait);
    text = String.format(text, (int) (waitTime / 1000));
    textField.setText(text);
}

@Override protected void onPostExecute(String result) {
    textField.setText(result);
}
```

## ASYNCTASK - JAVA

### AsyncTask

```
            @Override protected void onProgressUpdate(Long... data) {
                String text = getResources().getString(R.string.wait);
                text = String.format(text, (int) (waitTime / 1000));
                textField.setText(text);
            }

            @Override protected void onCancelled(String s) {
                String text= getResources().getString(R.string.cancel);
                textField.setText(text);
            }
        }
    }
});
```

codecentric

# ASYNCTASK - GROOVY

```
def waitTime = 5000
button.onClickListener = {
    Fluent.async {
        long time = waitTime + 1000
        while (time > 1000) {
            if (isCancelled()) break
            time -= 1000
            progress(time)
            sleep(1000)
        }
        return resources.getString(R.string.finished)
    }.first {
        def text = resources.getString(R.string.wait)
        textField.text = String.format(,text (int) (waitTime / 1000))
    }.then { String result ->
```

```
        textField.text = result
    }.onProgress { Long[] data ->
        def text = resources.getString(R.string.wait)
        textField.text = String.format(text, (int) (data.first()/1000))
    }.onCancelled { String result ->
        textField.text = resources.getString(R.string.cancel)
    }()
}
```

## ASYNCTASK - ALTERNATIVE

```
button.onClickListener = {
    Fluent.async this.&doInBackground
          .first this.&doFirst
          .then this.&doAfter
          .onProgress this.&onProgress
          .onCancelled this.&onCancelled
          .call()
}

String doInBackground(def params) { ... }
String doFirst() { ... }
String doAfter(String result) { ... }
String onProgress(long[] data) { ... }
String onCancelled(String result) { ... }
```

## ASYNCTASK - FLUENT IMPLEMENTIERUNG

```
https://gist.github.com/karfunkel/6eba3c237890f90c2779
```

## PERFORMANCE

- GR8Conf Agenda
  - Groovy jar → 4.5 MB
  - Application → 2 MB
  - nach ProGuard → 1 MB
  - ~ 8.2 MB RAM (viele Bilder)
    - mit CompileStatic

# PROGUARD

```
-dontobfuscate
-keep class org.codehaus.groovy.vmplugin.**
-keep class org.codehaus.groovy.runtime.dgm*
-keepclassmembers class org.codehaus.groovy.runtime.dgm* {
    *;
}
-keepclassmembers class ** implements
                        org.codehaus.groovy.runtime.GeneratedClosure {
    *;
}
-dontwarn org.codehaus.groovy.**
-dontwarn groovy**
```

codecentric

## COMMUNITY

## SWISSKNIFE

- Annotationsbasiert
  - View injection
  - Multithreading
- Ideen von ButterKnife und AndroidAnnotations
- aber zur Compilezeit über ASTTransformationen
- https://github.com/Arasthel/SwissKnife

codecentric

## SWISSKNIFE

```
class MyActivity extends Activity {
    @ViewById(R.id.myField) TextField mTextField

    @OnClick(R.id.button)
    void onButtonClicked(Button button) {
       Toast.makeText(this, "Button clicked", Toast.LENGTH_SHOT).show()
    }

    @OnBackground
    void doSomeProcessing(URL url) {
       // Contents will be executed on background
       ...
    }
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState)
       contentView = R.layout.activity_main
       // Needed for injection of views and callbacks to take place
       SwissKnife.inject(this)
    }
}
```

# SWISSKNIFE - PERSISTENZ

```
@SaveInstance
private int myInt

// You can also set a custom tag to your variable
@SaveInstance("MYSTRING")
private String myString

@Override
void onCreate(Bundle savedInstanceState){
    // Your previous code
    SwissKnife.restoreState(this, savedInstanceState)
}
```

# SWISSKNIFE - NETZWERK 'SERIALISIERUNG'

## Java Code

```java
public class ParcelableClass implements Parcelable {
    private int id;
    private String name;

    public ParcelableClass(Parcel source) {
        this.id = source.readInt();
        this.name = source.readString();
    }

    public void writeToParcel(Parcel out) {
        out.writeInt(id);
        out.writeString(name);
    }
    public String getId() { return id; }
    // ...
}
```

# SWISSKNIFE - NETZWERK 'SERIALISIERUNG'

Groovy Code

```groovy
@Parcelable
class ParcelableClass {
    int id
    String name
}
```

# GROOID TOOLS

**Ziel**

Builder für Android UI’s ähnlich SwingBuilder

Views ohne xml

Dynamisch erzeugte Views

```
View view = new AndroidBuilder().build(this) {
    relativeLayout(width: MATCH_PARENT, height: MATCH_PARENT,
                                                  padding: [dp(64), dp(16)]) {
        textView(width: MATCH_PARENT, height: dp(20),
                                                   text: R.string.hello_world)
    }
}
```

Überlegungen in SwissKnife zu integrieren

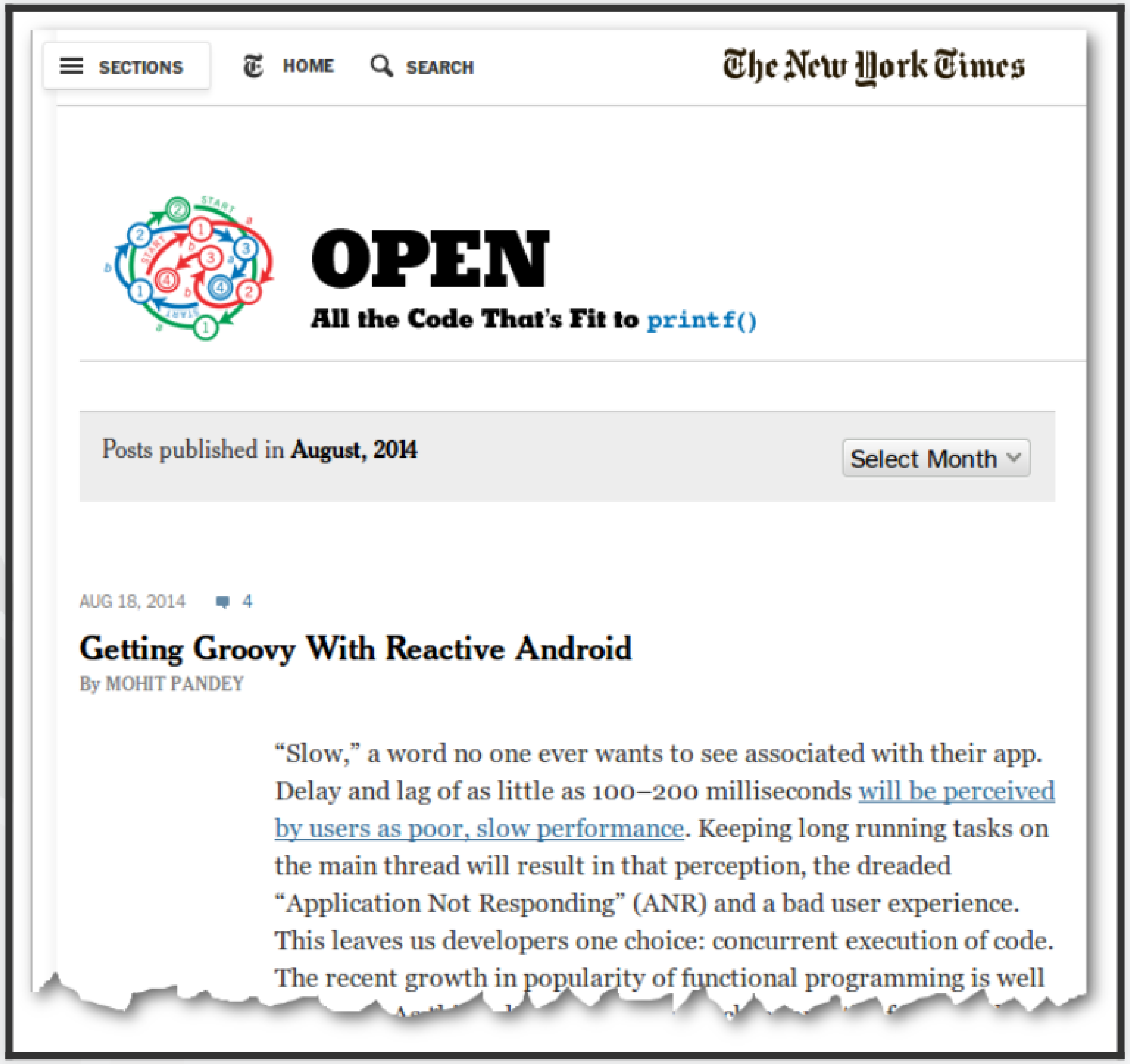
# DYNAMISCHE KOMPILATION

- Kompilation zur Laufzeit ist möglich
  - Android ClassLoader kann nur vom Filesystem lesen
  - Java Bytecode muss in ein JAR abgelegt werden
  - JAR wird in Dex-Format gewandelt
  - Danach wird die Klass geladen
- Dies ist sehr langsam
  - GroovyShell, GroovyClassLoader etc.
  - DSL’s
  - ConfigSlurper

# POTENTIELLE PROBLEME

- Performance auf low-end Geräten
    - @CompileStatic wo möglich
- Berüchtigte 64k Methodengrenze
    - ProGuard verwenden
- Tooling Support
    - AndroidStudio unterstützt Groovy nicht zu 100 %
- Google support
    - Android Gradle Plugin ändert sich sehr häufig

# PRODUKTIVEINSATZ



SECTIONS HOME SEARCH The New York Times

OPEN

All the Code That's Fit to `printf()`

Posts published in **August, 2014** Select Month

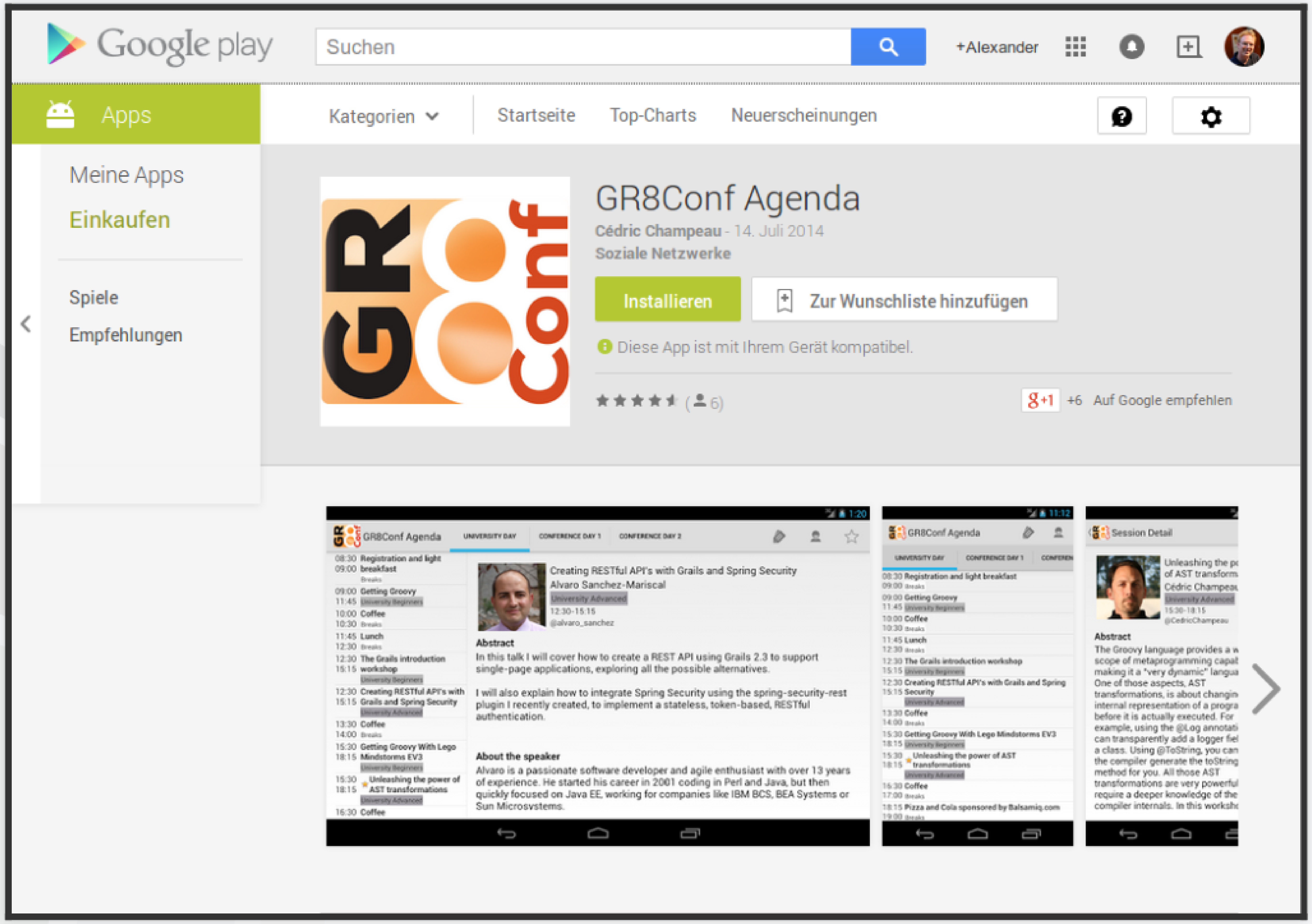
AUG 18, 2014 4

**Getting Groovy With Reactive Android**

By MOHIT PANDEY

"Slow," a word no one ever wants to see associated with their app. Delay and lag of as little as 100–200 milliseconds will be perceived by users as poor, slow performance. Keeping long running tasks on the main thread will result in that perception, the dreaded "Application Not Responding" (ANR) and a bad user experience. This leaves us developers one choice: concurrent execution of code. The recent growth in popularity of functional programming is well

# PRODUKTIVEINSATZ



GR8Conf Agenda

Cédric Champeau - 14. Juli 2014

Soziale Netzwerke

Diese App ist mit Ihrem Gerät kompatibel.

★★★★½ (6)

+1 +6 Auf Google empfehlen

GR8Conf Agenda — UNIVERSITY DAY | CONFERENCE DAY 1 | CONFERENCE DAY 2

08:30–09:00 Registration and light breakfast — Breaks
09:00–11:45 Getting Groovy — University Beginners
10:00–10:30 Coffee — Breaks
11:45–12:30 Lunch — Breaks
12:30–15:15 The Grails introduction workshop — University Beginners
12:30–15:15 Creating RESTful API's with Grails and Spring Security — University Advanced
13:30–14:00 Coffee — Breaks
15:30–18:15 Getting Groovy With Lego Mindstorms EV3 — University Beginners
15:30–18:15 Unleashing the power of AST transformations — University Advanced
16:30–17:00 Coffee — Breaks
18:15–19:00 Pizza and Cola sponsored by Balsamiq.com — Breaks

Creating RESTful API's with Grails and Spring Security
Alvaro Sanchez-Mariscal
University Advanced
12:30-15:15
@alvaro_sanchez

**Abstract**
In this talk I will cover how to create a REST API using Grails 2.3 to support single-page applications, exploring all the possible alternatives.

I will also explain how to integrate Spring Security using the spring-security-rest plugin I recently created, to implement a stateless, token-based, RESTful authentication.

**About the speaker**
Alvaro is a passionate software developer and agile enthusiast with over 13 years of experience. He started his career in 2001 coding in Perl and Java, but then quickly focused on Java EE, working for companies like IBM BCS, BEA Systems or Sun Microsystems.

Session Detail

Unleashing the p
of AST transform
Cédric Champeau
University Advanced
15:30-18:15
@CedricChampeau

**Abstract**
The Groovy language provides a w
scope of metaprogramming capab
making it a "very dynamic" langua
One of those aspects, AST
transformations, is about changin
internal representation of a progra
before it is actually executed. For
example, using the @Log annotati
can transparently add a logger fie
a class. Using @ToString, you can
the compiler generate the toString
method for you. All those AST
transformations are very powerfu
require a deeper knowledge of the
compiler internals. In this worksh

# FRAGEN ?

Alexander (Sascha) Klein

codecentric AG
Curiestr. 2
70563 Stuttgart

tel +49.711.67400-328
mobile +49.172.5294020
alexander.klein@codecentric.de
@saschaklein

http://www.codecentric.de
http://blog.codecentric.de