How to connect fragments to REST services

with live demo

About me

Dennis Geurts

@dennisgggg

+Dennis Geurts

dennis.geurts@luminis.eu

luminis

Conversing worlds

Disclaimer

In Android there are numerous ways to reach a certain objective

These views are mine. With it I do not disqualify those views held by others...

Previously on Android...

Activities were simple

```
  purpose ( Intent... )
```

- execution of a single look-up task
- presentation of the result
- take action based on user input
  (usually a new intent)

Previously on Android...

Activities: accessing remote services

asynchronous

- AsyncTask<X,Y,Z> within the Activity
- new Thread() in the Activity
- inside a Handler
- inside a *bound* Service
- inside an *Intent*Service

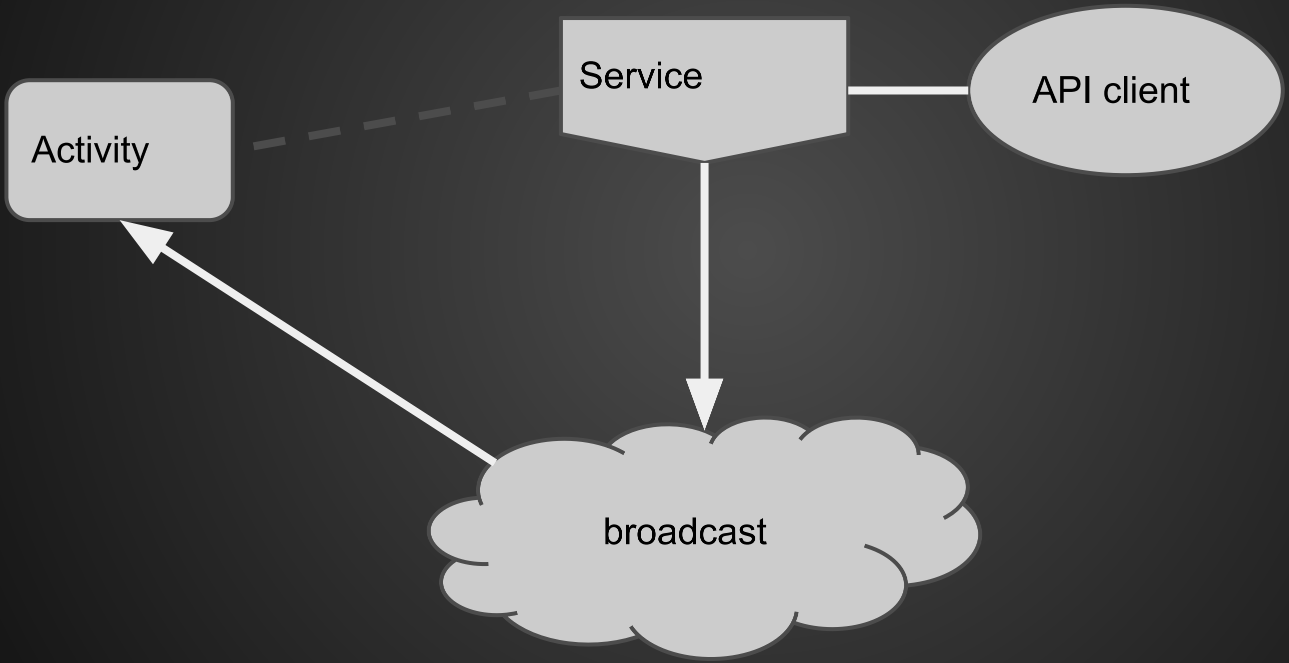
Previously on Android...

Activities: accessing remote services

issues

- Activity life-cycle not guaranteed during asynchronous call
- Activity might not be alive anymore when the result is returned...

Previously on Android...



Now that Tablets have arrived...

Large screens

multi purpose screens

- multiple sources of information
- multiple life-cycles within one activity
- multiple concurrent asynchronous requests

Now that Tablets have arrived...

Fragments are there to help you cope with added complexity

Again, we have an entity that has a single purpose

- enhanced life-cycle functionality
- application configuration in XML

Fragments - enhanced life-cycle

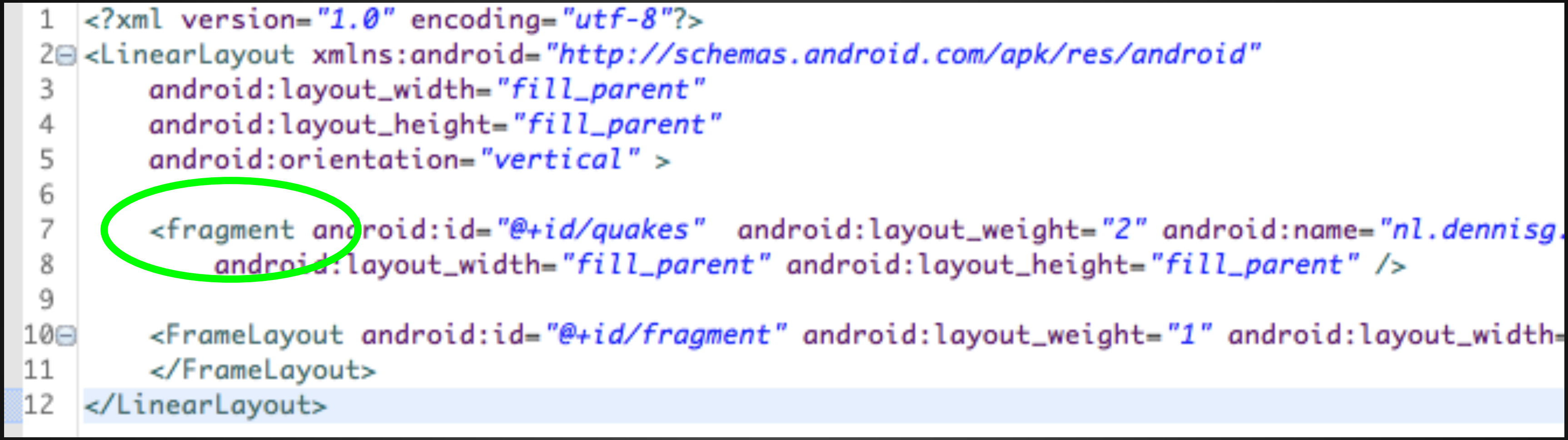
Fragment.setRetainInstance()

retain fragment state across activity re-creation

FragmentTx.addToBackStack()

- adds a whole new meaning to 'backPressed'

Fragments - application config in XML



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <fragment android:id="@+id/quakes"  android:layout_weight="2" android:name="nl.dennisg.
        android:layout_width="fill_parent" android:layout_height="fill_parent" />

    <FrameLayout android:id="@+id/fragment" android:layout_weight="1" android:layout_width=
    </FrameLayout>
</LinearLayout>
```

Here, we add behaviour in XML
not just presentation

Fragments - Remote Service Access

Few simple rules:

- Data for Fragment is loaded within fragment

improve re-usability

Fragments - Remote Service Access

Few simple rules:

- inter-fragment communication through Activity

loose coupling

Fragments - Remote Service Access

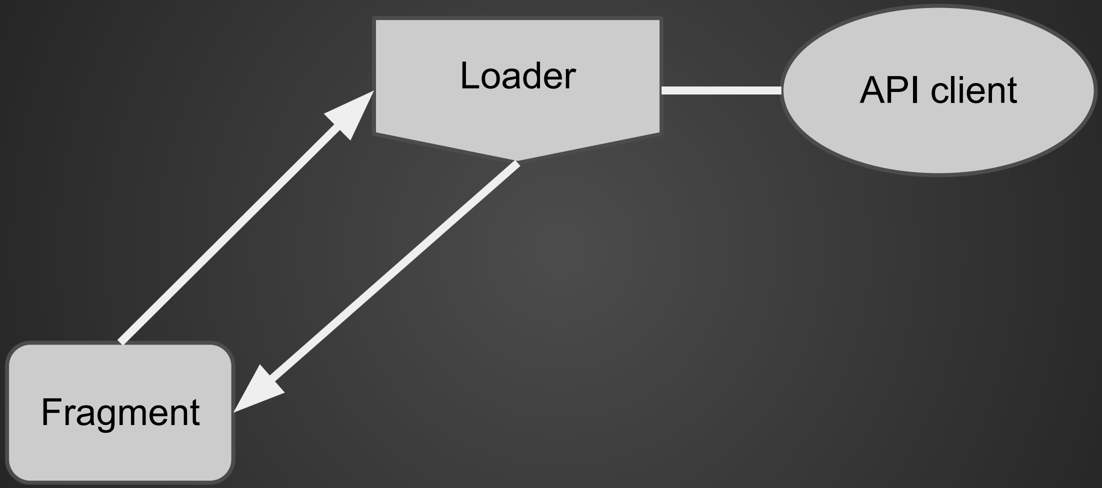
Use LoaderManager

```
getLoaderManager().initLoader(0, null, this);
```

the Fragment implements LoaderCallbacks<D>

<D> is the type of data you want to receive

Fragments - Remote Service Access



Today's example of remote data

Earthquake feed:

```
http://earthquake.usgs.gov
              /earthquakes/catalogs/eqs7day-M2.5.txt
```

- csv
- updated every 5 min
- quakes over 2.5 on Richter scale
- all quakes of the last 7 days

Today's example of remote data

Earthquake feed:

Header:

```
Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,
Region
```

Data:

```
ak,10425740,1,"Monday, March  5, 2012 17:59:48
     UTC",59.7103,-153.4235,2.8,115.80,59,"Southern
Alaska"
```

Example of Remote data

proposed REST interface:

```
GET /quakes
        - list all quakes (minimal info)
GET /quakes/<id>
        - list detailed information
```

Example of Remote data

Tips:

- version your REST interface !
- add version meta information

allow for backward compatibility

You simply cannot force your customers to update your apps

Example of Remote data

proposed REST interface:

```
GET /
      - version information
GET /v1/quakes
      - list all quakes (minimal info)
GET /v1/quakes/<id>
      - list detailed information
```

Example of Remote data

Version info - meta data

```
GET /
```

```
{ "quakes" : "Welcome", "version" : "v1" }
```

API name

API version

Example of Remote data

List quakes

```
GET /v1/quakes
[
   {"src":"ak","eqid":"10425830",
           "region":"Alaska Peninsula"},
   {"src":"ak","eqid":"10425740",
           "region":"Southern Alaska"},
   ...
]
```

Example of Remote data

Unique identifier

"<src>_<eqid>" is always unique

use it to identify a particular quake

Example of Remote data

List quakes

```
GET /v1/quakes/ak_10425830
{
"src":"ak","eqid":"10425830","version":"1","
datetime":"Monday, March  5, 2012 19:37:00
UTC","lat":"58.5578","lon":"-155.6727","
magnitude":"2.5","depth":"185.70","nst":"
6","region":"Alaska Peninsula"
}
```

The Loaders

```
Loader<List<Quake>>

Loader<Quake>
```

The LoaderCallbacks

```
LoaderCallbacks<List<Quake>> {
    ...
    onLoadFinished(Loader<List<Quake>>   loader,
                         List<Quake> quakes) {

        setListAdapter(new QuakesAdapter(quakes));

    }
}
```

The LoaderCallbacks

```
LoaderCallbacks<Quake> {
    ...
    onLoadFinished(Loader<Quake>   loader, Quake quake)
    {

        setListAdapter(new QuakeAdapter(quake));

    }
}
```

The REST client

Develop and test client outside Android Application

(at least do not use any Android specific classes)

The REST client

```
public String getVersion();
public List<Quake> getQuakes();
public Quake getQuake(String id);
```

The REST client

Technologies used:

- HttpClient
- Gson

leads to clean code...

The REST client

HttpClient

```
  HttpGet req = new HttpGet('/');
  HttpResponse res =
        client.execute(req);
```

use ResponseHandlers !

The REST client

Gson

```
Version ver =
        vrh.handleResponse(res);


Version {
   private String quake;
   private String version;
}
```

The REST client

Gson

```
    List<Quake> list =
                 qlrh.handleResponse(res);

    Quake quake =
                 qrh.handleResponse(res);
```

The REST client

Gson

```
Quake {
   private String src, eqid, version,
      datetime, nst, region;

   private double lat, lon, depth,
       magnitude;
}
```

The REST service

start simple

- in-memory list of quakes

comfortable with a working API ?
  proceed with more complex version

The REST service

- list of quakes in

```
map
function (doc) {
  emit(null, {
       src : doc.src,
      eqid : doc.eqid,
    region : doc.region });
}
```

Live demo

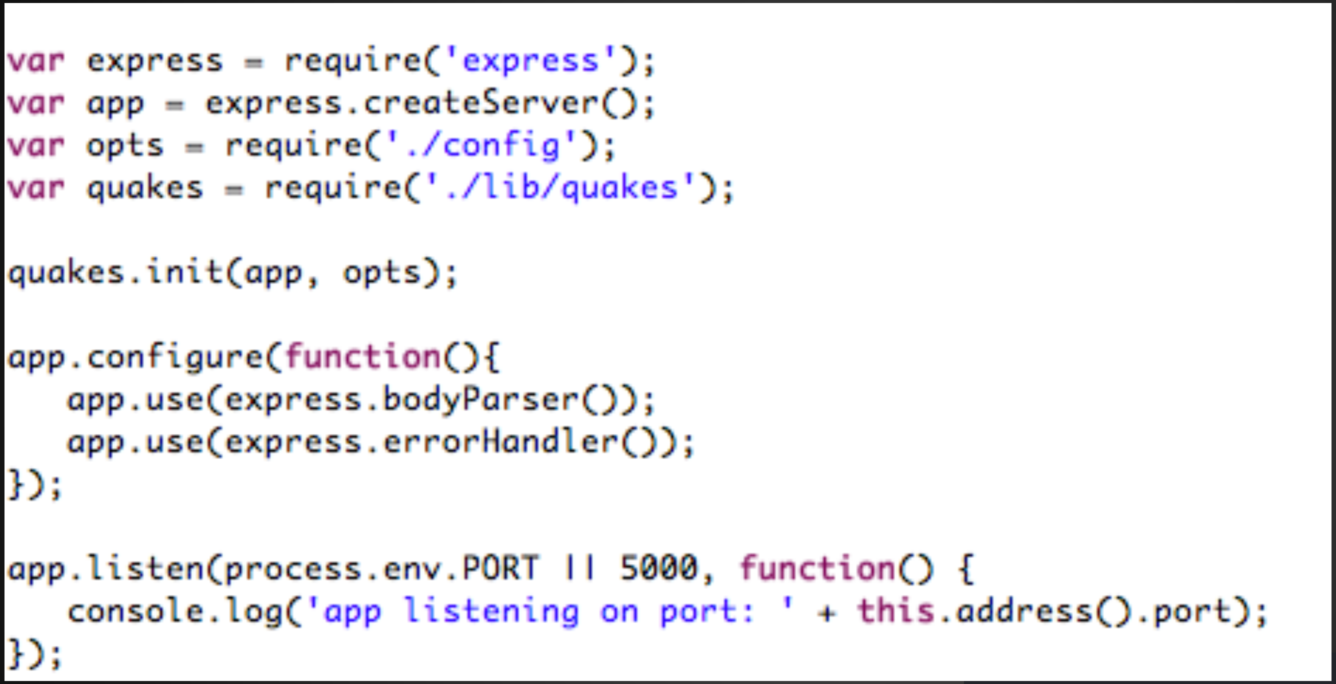
The REST service

Deploy on Heroku!

- heroku create
- git commit
- git push heroku master

added bonus:
signed-cert HTTPS

The REST service



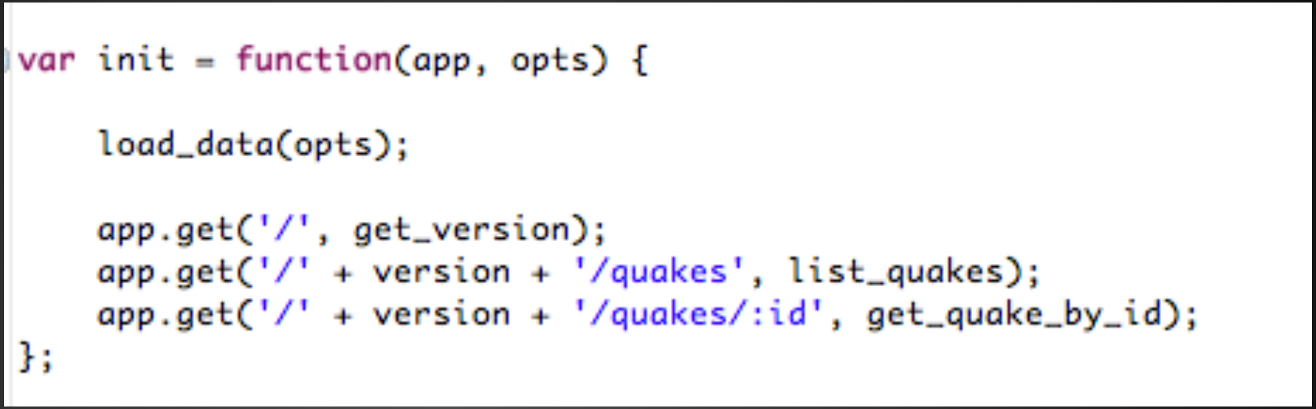
```
var express = require('express');
var app = express.createServer();
var opts = require('./config');
var quakes = require('./lib/quakes');

quakes.init(app, opts);

app.configure(function(){
   app.use(express.bodyParser());
   app.use(express.errorHandler());
});

app.listen(process.env.PORT || 5000, function() {
   console.log('app listening on port: ' + this.address().port);
});
```

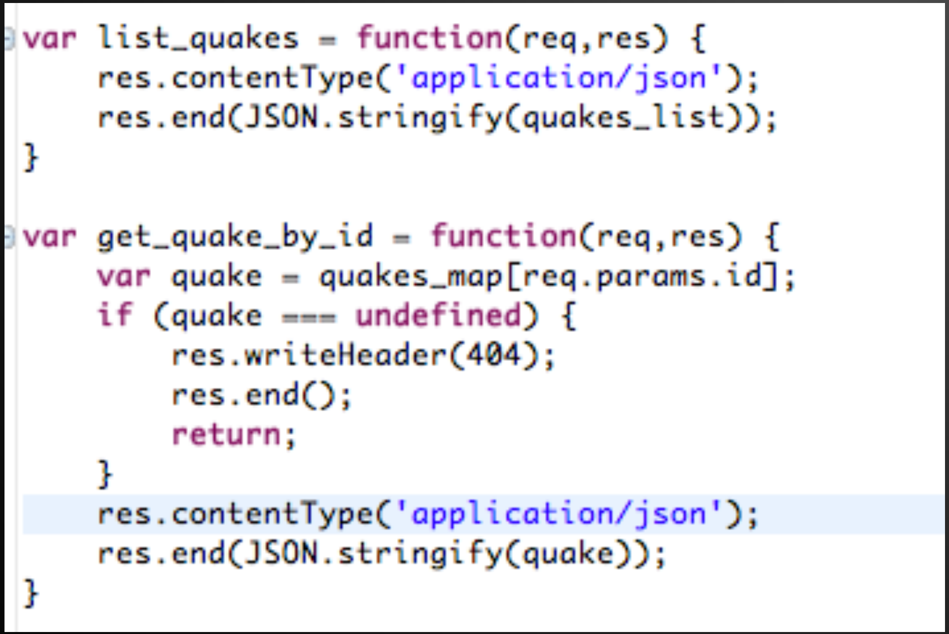
The REST service



```
var init = function(app, opts) {

    load_data(opts);

    app.get('/', get_version);
    app.get('/' + version + '/quakes', list_quakes);
    app.get('/' + version + '/quakes/:id', get_quake_by_id);
};
```



```
var list_quakes = function(req,res) {
    res.contentType('application/json');
    res.end(JSON.stringify(quakes_list));
}

var get_quake_by_id = function(req,res) {
    var quake = quakes_map[req.params.id];
    if (quake === undefined) {
        res.writeHeader(404);
        res.end();
        return;
    }
    res.contentType('application/json');
    res.end(JSON.stringify(quake));
}
```

!! Time for some action !!

Questions