# Volley: Fast and Easy Networking for Android?

## Präsentation zum GDG-Bremen-Stammtisch

Sven Nobis

7. Oktober 2013


# Back in June, 2013...

# ...wollte ich ein paar Daten in meiner App laden

- JSON und Bilder
- Einfach
  - *Viele Probleme*
- Asynchron
- Memory und Disk Caching
- und ...

# Volley kurz vorgestellt

- **Woher bekommen ich es?**
  - `git clone https://android.googlesource.com/platform/frameworks/volley/`
- **Und wie benutze ich es?**
  - ...
- *Präsentation auf der Google IO:*
  - `https://developers.google.com/events/io/sessions/325304728`

# Wie benutze ich Volley?

## Initialisierung

```
// Somewhere common; app startup or adapter constructor
mRequestQueue = Volley.newRequestQueue(context);
mImageLoader = new ImageLoader(mRequestQueue, new BitmapLruCache
```

# Wie benutze ich Volley?

## Ein Abfrage

```
mRequestQueue.add(new JsonObjectRequest(Method.GET, url, null,
  new Listener<JSONObject>() {
    public void onResponse(JSONObject jsonRoot) {
      mNextPageToken = jsonGet(jsonRoot, "next", null);
      List<Items> items = parseJson(jsonRoot);
      appendItemsToList(item);
      notifyDataSetChanged();
    }
  }
}
```

# Wie benutze ich Volley?

## Bilder

```
- <ImageView
+ <com.android.volley.NetworkImageView
```

```
mImageView.setImageUrl(BASE_URL + item.image_url,mImageLoader);
```

# Wie benutze ich Volley?

*onStop*

```
@Override
public void onStop() {
  mRequestQueue.cancelAll(this);
  [...]
```

# Wie benutze ich Volley?

## Custom requests - z.B. JSON mit Jackson parsen (1)

```
public class JacksonRequest<TResponse> extends JsonRequest<TResponse> {

    private final ObjectMapper mMapper;
    private final Class<TResponse> mResponseClass;

    public JacksonRequest(int method, String url, String jsonRequest,
            Listener<TResponse> listener, ErrorListener errorListener,
            ObjectMapper mapper, Class<TResponse> responseClass) {
        super(method, url, jsonRequest, listener, errorListener);
        mMapper = mapper;
        mResponseClass = responseClass;
    }

    /*
     * Some more constructors  ...
     */
```

# Wie benutze ich Volley?

## Custom requests - z.B. JSON mit Jackson parsen (2)

```
    @Override
    protected Response<TResponse> parseNetworkResponse(NetworkResponse response) {
        try {
            String jsonString = new String(response.data, "UTF-8"); // TODO:
            HttpHeaderParser.parseCharset(response.headers));
            return Response.success(mMapper.readValue(jsonString, mResponseClass),
                                    HttpHeaderParser.parseCacheHeaders(response));
        } catch (UnsupportedEncodingException e) {
            return Response.error(new ParseError(e));
        } catch (JsonParseException e) {
            return Response.error(new ParseError(e));
        } catch (JsonMappingException e) {
            return Response.error(new ParseError(e));
        } catch (IOException e) {
            return Response.error(new ParseError(e));
        }
    }
}
```

# Volley im Vergleich...

## ...mit OkHttp

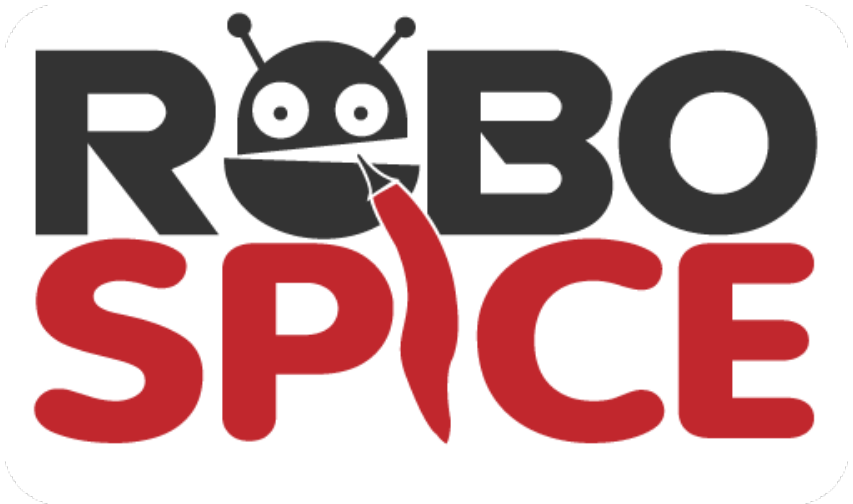
```
OkHttpClient client = new OkHttpClient();

String get(URL url) throws IOException {
  HttpURLConnection connection = client.open(url);
  InputStream in = null;
  try {
    // Read the response.
    in = connection.getInputStream();
    byte[] response = readFully(in);
    return new String(response, "UTF-8");
  } finally {
    if (in != null) in.close();
  }
}
```

(http://square.github.io/okhttp/)

# Volley im Vergleich...

## ...RoboSpice




https://github.com/octo-online/robospice

# Nochmal zurück zu...

*...ich wollte ein paar Daten in meiner App laden*

- Die Daten
  - gering ($< 3$ MB)
  - nicht personalisiert
  - ändern sich gelegentlich/unregelmäßig
  - Offline Support
- Sowas wie HTML5 Application Cache?

→ Android Application Cache

# Nochmal zurück zu...

*...ich wollte ein paar Daten in meiner App laden*

- Die Daten
  - gering ($< 3$ MB)
  - nicht personalisiert
  - ändern sich gelegentlich/unregelmäßig
  - Offline Support
- Sowas wie HTML5 Application Cache?

$\rightarrow$ Android Application Cache

# Nochmal zurück zu...

*...ich wollte ein paar Daten in meiner App laden*

- Die Daten
  - gering ($< 3$ MB)
  - nicht personalisiert
  - ändern sich gelegentlich/unregelmäßig
  - Offline Support
- Sowas wie HTML5 Application Cache?

$\rightarrow$ Android Application Cache

# Android Application Cache

## Application Cache Manifest

```
CACHE MANIFEST

CACHE:
# Cache the session data, some common data,
# information about the location and some pictures:
common.json
sessions.json
location.json
img/location_a.jpg
NETWORK:
# Use network connection for these files:
comments.json
http://api.twitter.com
# Or just an asterik for alle other requests:
*
FALLBACK:
# We don't want do syncronize so many comments
# So we provide a special file with a few comments
# per session when offline:
comments.json comments_light.json
```

# Android Application Cache

## Initialisierung

```
public init(Context ctx, ErrorListener errorListener,
    ApplicationCacheEventListener appCacheEventListener) {
    mApplicationCache = new ApplicationCache();
    mApplicationCache.setEventListener(appCacheEventListener);
    try {
        String url = /* ... */;
        mApplicationCache.init(url, ctx,
          new AppCacheDiskBasedCache(/* ... */));
    } catch(URISyntaxException ex) {
        throw new RuntimeException(ex);
    }
    mRequestQueue = mApplicationCache.getQueue();
    mErrorListener = errorListener;
}
```

# Android Application Cache

## EventListener

```
public interface ApplicationCacheEventListener {
      public void onChecking(ApplicationCache source);
      public void onError(ApplicationCache source,
        boolean wasManifest, VolleyError error);
      public void onNoUpdate(ApplicationCache source);
      public void onDownloading(ApplicationCache source,
        int total);
      public void onProgress(ApplicationCache source,
        String loadedUrl, int loaded, int total);
      public void onUpdateReady(ApplicationCache source);
      public void onObsolete(ApplicationCache source);
}
```

# Fazit

## zu Volley

+ Einfach und Schnell zu Nutzen
− Dokumentation
+ Caching nach dem HTTP-Standard
+ Quellcode

# Vielen Dank für Ihre Aufmerksamkeit!

https://github.com/SvenTo/