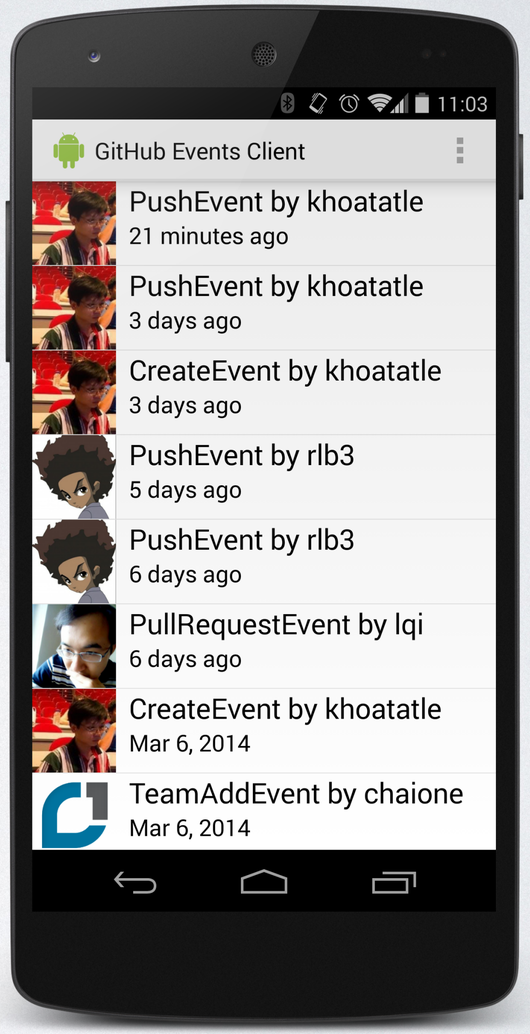
DEVELOPING FOR ANDROID

by Andy Dyer

BUILDING A GITHUB EVENTS CLIENT

We'll cover:

1. Project setup
2. Gradle dependencies
3. API requests & JSON parsing
4. Activities & fragments
5. Lists, adapters, and custom item layouts



GETTING STARTED

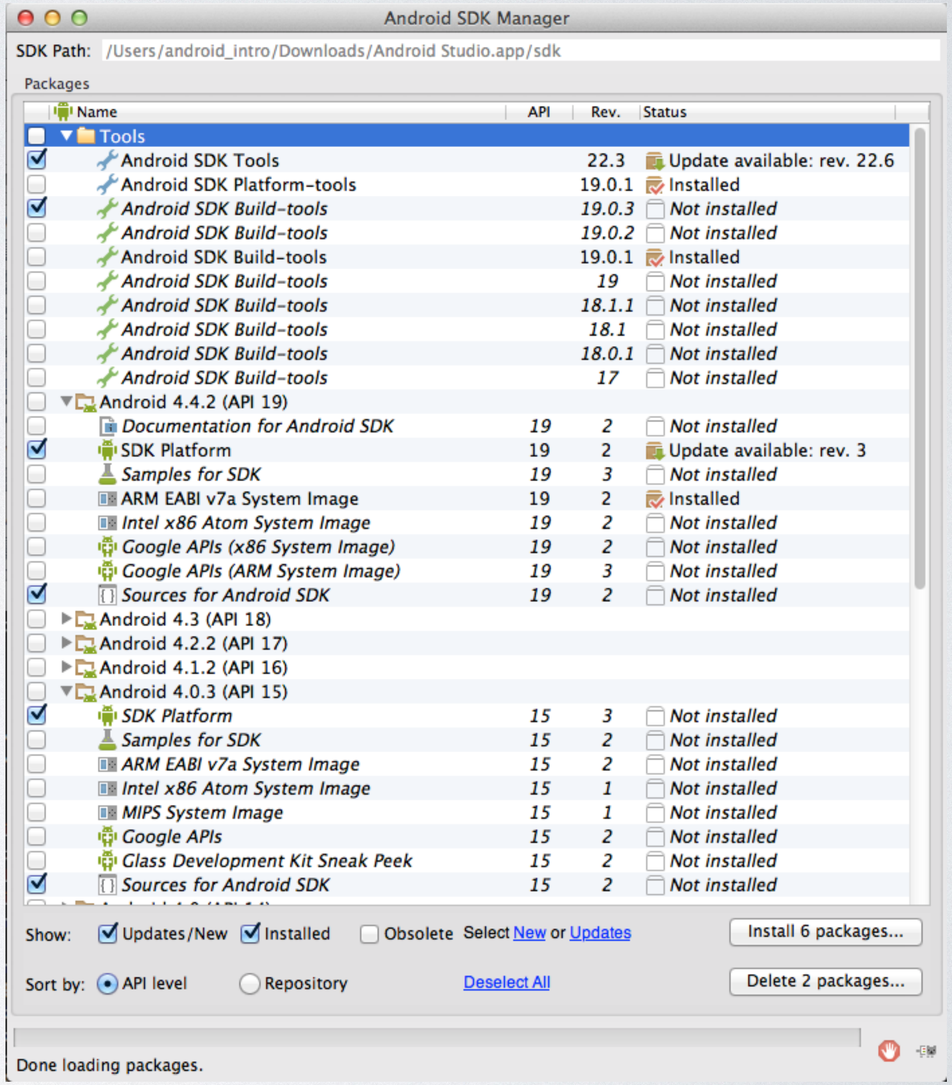
1. Download and install Android Studio

2. Launch the SDK Manager from the command line:

```
/Applications/Android\ Studio.app/sdk/tools/android
```

GETTING STARTED

3. Install the SDK tools and platform/ sources for KitKat & Ice Cream Sandwich



GETTING STARTED

4. Download and install the Genymotion emulator (choose the free version)

5. Clone the repo. Checkout the tags for each phase of the tutorial.

   `https://github.com/abdyer/android-intro`

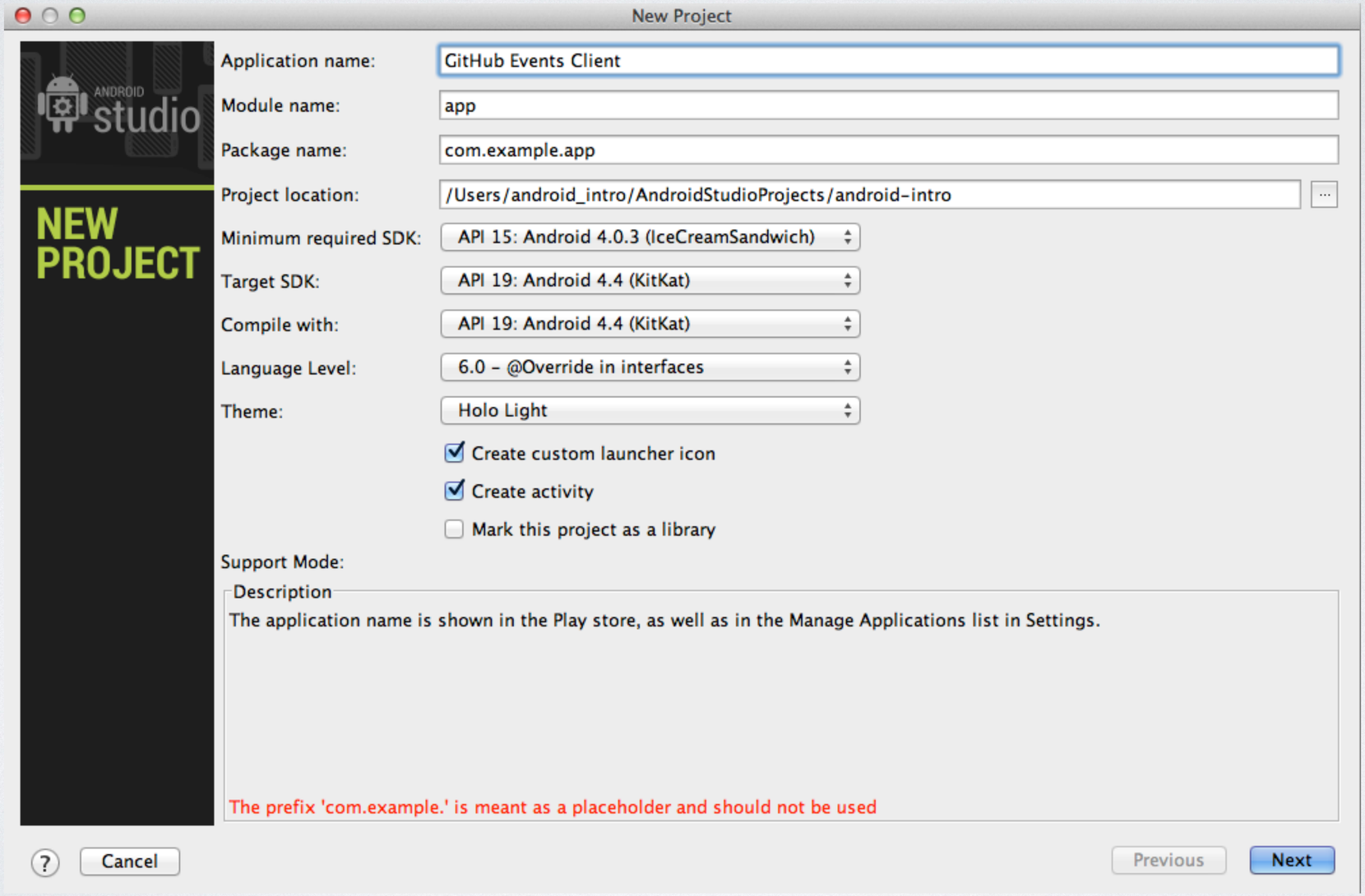
v0.1 ←

TUTORIAL

- Git tags for each section of the tutorial
- Look for the tag name in the upper left of each slide
- Stash changes or reset to move between steps

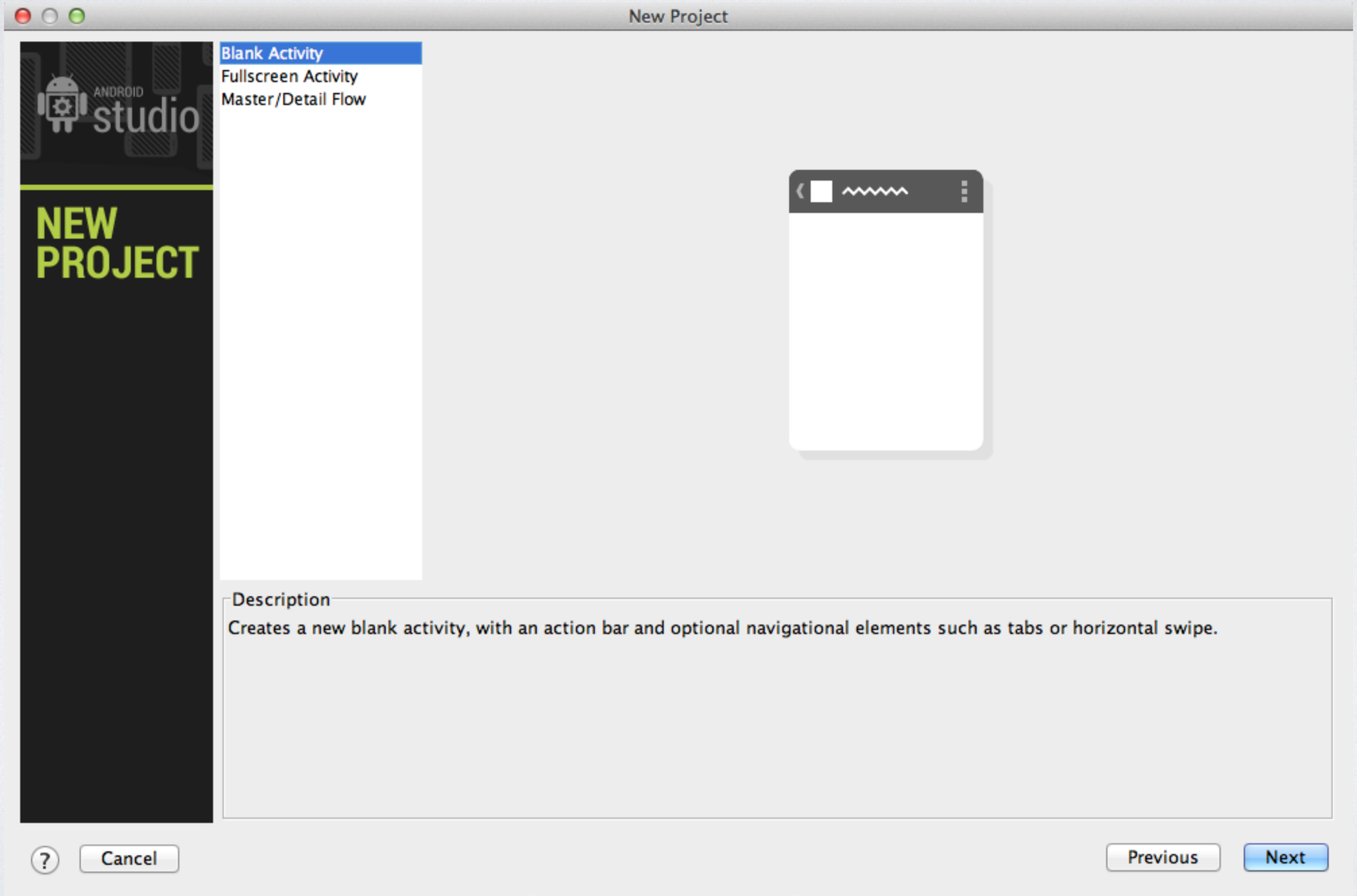
v0.1

FILE > NEW PROJECT…



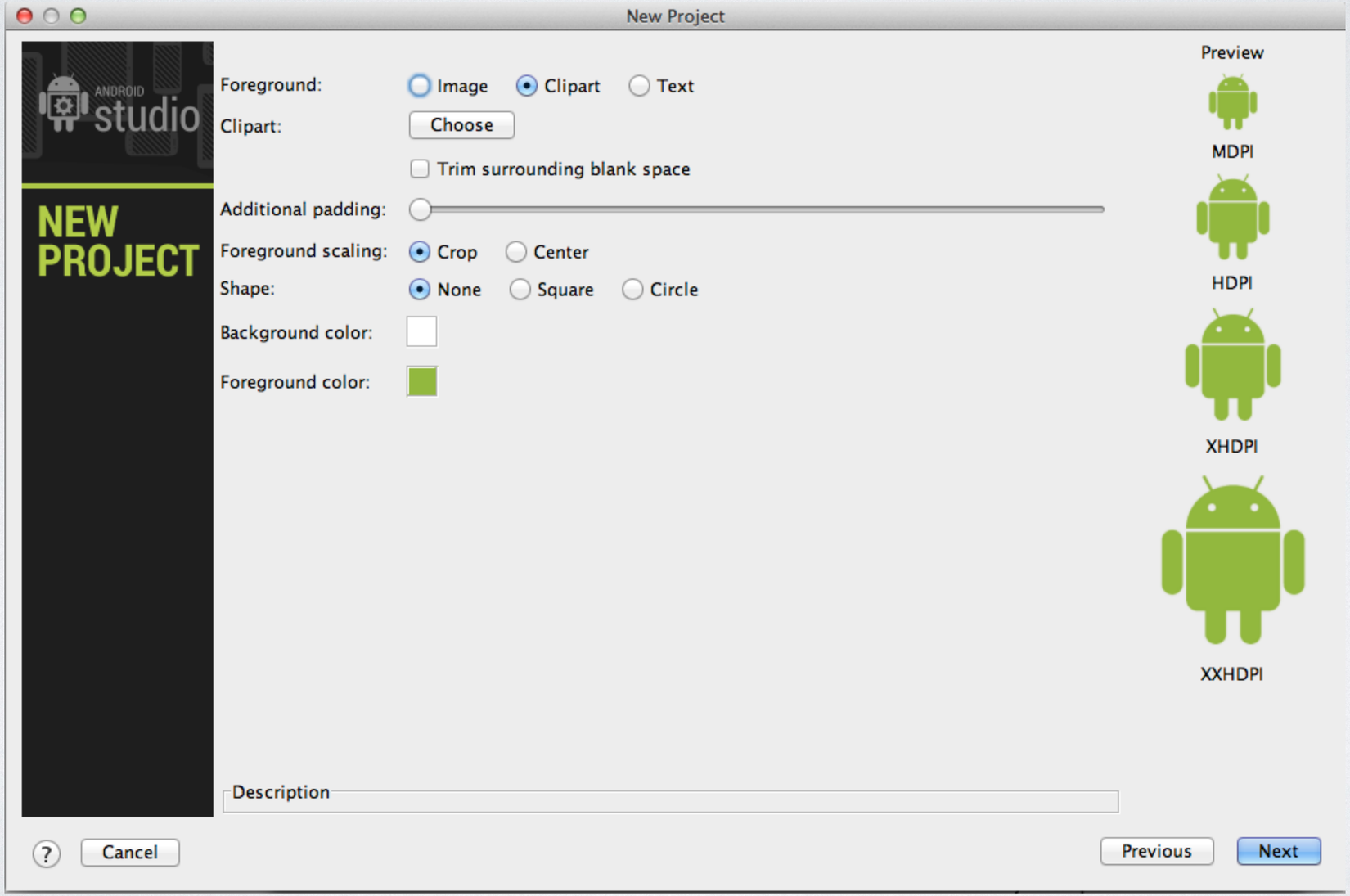
v0.1

FILE > NEW PROJECT…



v0.1

FILE > NEW PROJECT…



v0.1

ANATOMY OF AN ANDROID PROJECT

- Manifest file - `AndroidManifest.xml`
- Resources - `app/src/main/res`
- Java source code - `app/src/main/java`

v0.1

ANDROID MANIFEST

- Package information
- Target version - Highest version on which app was tested, aim for latest version
- Minimum version - Minimum supported version. Based on current usage stats, Ice Cream Sandwich (v4.03, API 15)
- Permissions
- Application components - Name, icon, activities, services, receivers

v0.1

APP RESOURCES

- Values - strings, integers, dimensions
- Styles, Themes, and Colors
- Drawables - images, PNG or nine patch
- Layouts & Menus - XML based, similar to HTML
- Animations

v0.1

KNOCK. KNOCK. WHO'S THERE?

...

...

...

...

Java

v0.2

IMPLEMENTING THE API

Add Volley and GSON dependencies to `build.gradle` file:

```
dependencies {
    compile 'com.mcxiaoke.volley:library:1.0.+@aar'
    compile 'com.google.code.gson:gson:2.2.4'
}
```

Pro Tip: http://gradleplease.appspot.com/

v0.2

IMPLEMENTING THE API

Create the application subclass & request queue instance:

```
package com.example.app;

import android.app.Application;

import com.android.volley.RequestQueue;
import com.android.volley.toolbox.Volley;

public class GithubEventsApp extends Application {

    private static RequestQueue requestQueue;

    @Override
    public void onCreate() {
        super.onCreate();
        requestQueue = Volley.newRequestQueue(getApplicationContext());
    }

    public static RequestQueue getRequestQueue() {
        return requestQueue;
    }
}
```

v0.2

IMPLEMENTING THE API

Specify the application subclass in the `AndroidManifest.xml` file:

```
<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:name=".GithubEventsApp">
  <!-- ... -->
</application>
```

v0.2

IMPLEMENTING THE API

Create POJOs for the parsed JSON response:

```
package com.example.app.api;

public class Event {

    int id;
    String type;
    Actor actor;
    Repo repo;
    String created_at;

    public int getId() { return id; }

    public String getType() { return type; }

    public Actor getActor() { return actor; }

    public Repo getRepo() { return repo; }

    public String getCreatedAt() { return created_at; }
}
```

v0.2

IMPLEMENTING THE API

Create POJOs for the parsed JSON response (continued):

```
package com.example.app.api;

import java.util.ArrayList;

public class EventsResponse extends ArrayList<Event> {
}
```

GSON can parse arrays, but without a root element, we need a strongly typed collection object

v0.2

IMPLEMENTING THE API

Create request subclass with JSON parsing:

```
package com.example.app.api;

import com.android.volley.Cache;
import com.android.volley.NetworkResponse;
import com.android.volley.ParseError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.toolbox.HttpHeaderParser;
import com.google.gson.Gson;
import com.google.gson.JsonSyntaxException;

import java.io.UnsupportedEncodingException;

public class GitHubRequest<T> extends Request<T> {

    private static final String API_URL = "https://api.github.com/";

    private Class<T> clazz;
    private final Response.Listener<T> listener;

    public GitHubRequest(int method, String path, Response.ErrorListener errorListener, Response.Listener<T> listener, Class<T> clazz) {
        super(method, API_URL + path, errorListener);
        this.listener = listener;
        this.clazz = clazz;
    }
```

v0.2

IMPLEMENTING THE API

Create request subclass with JSON parsing (continued):

```
// ...

    @Override
    protected Response<T> parseNetworkResponse(NetworkResponse response) {
        try {
            String jsonString = new String(response.data, HttpHeaderParser.parseCharset(response.headers));

            Gson gson = new Gson();
            Cache.Entry entry = HttpHeaderParser.parseCacheHeaders(response);

            //If requested type is Void, requesting class is not expecting a result. Handle separately
            if(clazz == Void.class) { return Response.success(null, entry); }

            return Response.success(gson.fromJson(jsonString, clazz), entry);
        }
        catch (UnsupportedEncodingException e) { return Response.error(new ParseError(e)); }
        catch (JsonSyntaxException e) { return Response.error(new ParseError(e)); }
    }

    @Override
    protected void deliverResponse(T response) {
        if(listener != null) { listener.onResponse(response); }
    }
}
```

v0.2

IMPLEMENTING THE API

Build the request:

```
package com.example.app.api;

import com.android.volley.Request;
import com.android.volley.Response;

public class GitHubRequestManager {

    public static GitHubRequest<EventsResponse> getEventRequest(Response.ErrorListener errorListener,
                                                                 Response.Listener<EventsResponse> listener,
                                                                 String organization) {
        String path = String.format("orgs/%s/events", organization);
        GitHubRequest<EventsResponse> request = new GitHubRequest<EventsResponse>(Request.Method.GET,
                path, errorListener, listener, EventsResponse.class);
        return request;
    }
}
```

v0.3

LOADING DATA INTO THE LISTVIEW

Add the internet permission in the `AndroidManifest.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.app" >

    <uses-permission android:name="android.permission.INTERNET" />

</manifest>
```

v0.3

LOADING DATA INTO THE LISTVIEW

Create the list fragment:

```
package com.example.app;

// imports...

public class EventListFragment extends ListFragment implements Response.Listener<EventsResponse>, Response.ErrorListener {

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        requestEvents();
    }

    private void requestEvents() {
        getActivity().setProgressBarIndeterminateVisibility(true);
        GitHubRequest<EventsResponse> request = GitHubRequestManager.getEventRequest(this, this, "chaione");
        request.setTag(this);
        GithubEventsApp.getRequestQueue().add(request);
    }

    @Override
    public void onResponse(EventsResponse response) {
        getActivity().setProgressBarIndeterminateVisibility(false);
        setListShown(true);
        ArrayAdapter<Event> adapter = new ArrayAdapter<Event>(getActivity(), android.R.layout.simple_list_item_1, android.R.id.text1, response);
        getListView().setAdapter(adapter);
    }

    @Override
    public void onErrorResponse(VolleyError volleyError) {
        getActivity().setProgressBarIndeterminateVisibility(false);
        Toast.makeText(getActivity(), R.string.unable_to_fech_events, Toast.LENGTH_SHORT).show();
    }
}
```

v0.3

LOADING DATA INTO THE LISTVIEW

Add progress & replace the placeholder fragment in `MainActivity`:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setProgressBarIndeterminate(true);

        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction()
                    .add(R.id.container, new EventListFragment())
                    .commit();
        }
    }
}
```

v0.4

CREATING A CUSTOM LIST ADAPTER

Create the adapter inner class in `EventListFragment`:

```
private class EventsAdapter extends ArrayAdapter<Event> {

    public EventsAdapter(Context context, int resource, List<Event> objects) {
        super(context, resource, objects);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if(convertView == null) {
            convertView = LayoutInflater.from(getContext())
                    .inflate(android.R.layout.simple_list_item_2, parent, false);
        }

        Event event = getItem(position);
        String line1 = String.format("%s by %s", event.getType(), event.getActor().getLogin());
        TextView.class.cast(convertView.findViewById(android.R.id.text1)).setText(line1);
        TextView.class.cast(convertView.findViewById(android.R.id.text2)).setText(event.getCreatedAt());

        return convertView;
    }
}
```

v0.4

CREATING A CUSTOM LIST ADAPTER

Replace the ArrayAdapter with `EventsAdapter`:

```
@Override
public void onResponse(EventsResponse response) {
    getActivity().setProgressBarIndeterminateVisibility(false);
    setListShown(true);
    EventsAdapter adapter = new EventsAdapter(getActivity(),
            android.R.layout.simple_list_item_2, response);
    getListView().setAdapter(adapter);
}
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Create a bitmap cache implementation:

```
package com.example.app.cache;

import android.graphics.Bitmap;
import android.util.LruCache;

import com.android.volley.toolbox.ImageLoader;

public class LruBitmapCache extends LruCache<String, Bitmap> implements ImageLoader.ImageCache {

    public LruBitmapCache(int maxSize) { super(maxSize); }

    @Override
    protected int sizeOf(String key, Bitmap value) {
        return value.getRowBytes() * value.getHeight();
    }

    @Override
    public Bitmap getBitmap(String url) { return get(url); }

    @Override
    public void putBitmap(String url, Bitmap bitmap) { put(url, bitmap); }
}
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Add bitmap cache & image loader instances to `GithubEventsApp`:

```
// ...

import com.android.volley.toolbox.ImageLoader;
import com.example.app.cache.LruBitmapCache;

// ...

public class GithubEventsApp extends Application {

    // ...

    private static LruBitmapCache bitmapCache;
    private static ImageLoader imageLoader;

    @Override
    public void onCreate() {
        super.onCreate();
        requestQueue = Volley.newRequestQueue(getApplicationContext());
        bitmapCache = new LruBitmapCache(10 * 1024);
        imageLoader = new ImageLoader(requestQueue, bitmapCache);
    }

    // ...

    public static ImageLoader getImageLoader() { return imageLoader; }
}
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Create the custom list item layout in `event_list_item.xml`:

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

    <com.android.volley.toolbox.NetworkImageView
        android:id="@+id/event_list_item_avatar"
        android:layout_width="@dimen/list_image_width_height"
        android:layout_height="@dimen/list_image_width_height"
        android:layout_alignParentLeft="true"
        android:layout_marginRight="@dimen/default_margin"/>

    <TextView
        android:id="@+id/event_list_item_text_line1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/event_list_item_avatar"
        android:textAppearance="?android:attr/textAppearanceLarge"/>

    <TextView
        android:id="@+id/event_list_item_text_line2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/event_list_item_avatar"
        android:layout_below="@+id/event_list_item_text_line1"
        android:textAppearance="?android:attr/textAppearanceMedium"/>

</RelativeLayout>
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Add the Butter Knife library in `build.gradle`:

```
dependencies {
    compile 'com.mcxiaoke.volley:library:1.0.+@aar'
    compile 'com.google.code.gson:gson:2.2.4'
    compile 'com.jakewharton:butterknife:4.0.+'
}
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Create a view holder and prepare views with Butter Knife:

```
class ViewHolder {

    @InjectView(R.id.event_list_item_avatar) NetworkImageView avatar;
    @InjectView(R.id.event_list_item_text_line1) TextView textLine1;
    @InjectView(R.id.event_list_item_text_line2) TextView textLine2;


    public ViewHolder(View view) {
        ButterKnife.inject(this, view);
    }
}
```

Since the list adapter recycles its views, we can improve performance by only doing the view lookups once and "holding" onto them.

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Use the view holder in the adapter's `getView()` method:

```
private class EventsAdapter extends ArrayAdapter<Event> {
    // ...
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;
        if(convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.event_list_item, parent, false);
            holder = new ViewHolder(convertView);
            convertView.setTag(holder);
        }
        else {
            holder = ViewHolder.class.cast(convertView.getTag());
        }

        Event event = getItem(position);
        String line1 = String.format("%s by %s", event.getType(), event.getActor().getLogin());
        holder.textLine1.setText(line1);
        holder.textLine2.setText(event.getCreatedAt());
        holder.avatar.setImageUrl(event.getActor().getAvatarUrl(), GithubEventsApp.getImageLoader());

        return convertView;
    }
    // ...
}
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Update the adapter constructor call:

```
@Override
public void onResponse(EventsResponse response) {

    // ...

    EventsAdapter adapter = new EventsAdapter(getActivity(), response);

    // ...
}
```

v0.5

CREATING A CUSTOM LIST ITEM LAYOUT

Implement a relative time format for the event date:

```
private class EventsAdapter extends ArrayAdapter<Event> {
    // ...
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // ...
        holder.textLine2.setText(getRelativeTime(event.getCreatedAt()));
        // ...
    }

    private String getRelativeTime(String isoDateTime) {
        try {
            TimeZone utc = TimeZone.getTimeZone("UTC");
            SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
            format.setTimeZone(utc);
            GregorianCalendar calendar = new GregorianCalendar(utc);
            calendar.setTime(format.parse(isoDateTime));
            return DateUtils.getRelativeTimeSpanString(calendar.getTimeInMillis(), System.currentTimeMillis(), DateUtils.SECOND_IN_MILLIS).toString();
        } catch (ParseException e) {
            return isoDateTime;
        }
    }
}
```

v0.6

CREATING A WEBVIEW ACTIVITY

Create the layout in `activity_webview.xml`:

```
<?xml version="1.0" encoding="utf-8"?>

<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

v0.6

CREATING A WEBVIEW ACTIVITY

Create the activity class:

```
public class WebViewActivity extends Activity {

    public static final String EXTRA_URL = "url";
    public static final String EXTRA_TITLE = "title";

    @InjectView(R.id.webview) WebView webview;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_webview);
        setProgressBarIndeterminate(true);
        ButterKnife.inject(this);
        getActionBar().setTitle(getIntent().getStringExtra(EXTRA_TITLE));
        getActionBar().setDisplayHomeAsUpEnabled(true);
        loadUrl();
    }
```

v0.6

CREATING A WEBVIEW ACTIVITY

Create the activity class (continued):

```
    // ...

    private void loadUrl() {
        String url = getIntent().getStringExtra(EXTRA_URL);
        webview.setWebViewClient(new WebViewClient() {
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                setProgressBarIndeterminateVisibility(false);
                return false;
            }
        });
        setProgressBarIndeterminateVisibility(true);
        webview.loadUrl(url);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

v0.6

CREATING A WEBVIEW ACTIVITY

Add the activity to `AndroidManifest.xml`:

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme"
    android:name=".GithubEventsApp">
    <!-- ... -->

    <activity android:name=".WebViewActivity" />

   <!-- ... -->
</application>
```

v0.6

CREATING A WEBVIEW ACTIVITY

Add a method in `EventListFragment` to launch the new activity:

```
private void launchWebViewActivity(String path) {
    String url = String.format("http://github.com/%s", path);
    Intent intent = new Intent(getActivity(), WebViewActivity.class);
    intent.putExtra(WebViewActivity.EXTRA_TITLE, path);
    intent.putExtra(WebViewActivity.EXTRA_URL, url);
    startActivity(intent);
}
```

v0.6

CREATING A WEBVIEW ACTIVITY

Launch the activity when a list row is tapped:

```
@Override
public void onResponse(EventsResponse response) {
    // ...
    final EventsAdapter adapter = new EventsAdapter(getActivity(), response);
    // ...
    getListView().setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Event event = adapter.getItem(position);
            launchWebViewActivity(event.getRepo().getName());
        }
    });
}
```

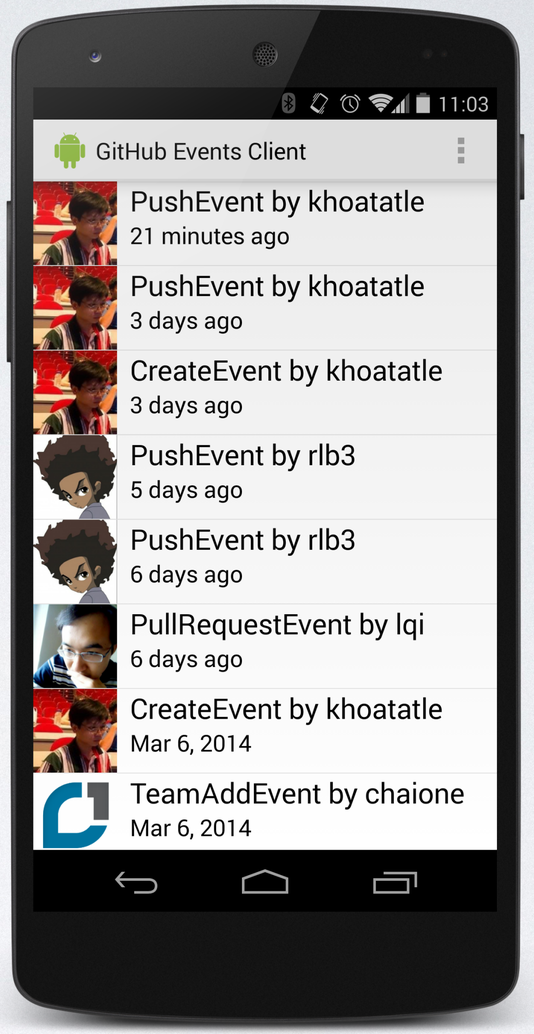
v0.6

CREATING A WEBVIEW ACTIVITY

Launch the activity when an avatar is tapped:

```
private class EventsAdapter extends ArrayAdapter<Event> {

    // ...

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // ...

        final Event event = getItem(position);

        // ...

        holder.avatar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                launchWebViewActivity(event.getActor().getLogin());
            }
        });

        // ...
    }
}
```

QUESTIONS?


11:03
GitHub Events Client
PushEvent by khoatatle
21 minutes ago
PushEvent by khoatatle
3 days ago
CreateEvent by khoatatle
3 days ago
PushEvent by rlb3
5 days ago
PushEvent by rlb3
6 days ago
PullRequestEvent by lqi
6 days ago
CreateEvent by khoatatle
Mar 6, 2014
TeamAddEvent by chaione
Mar 6, 2014