

Mobile Application Development

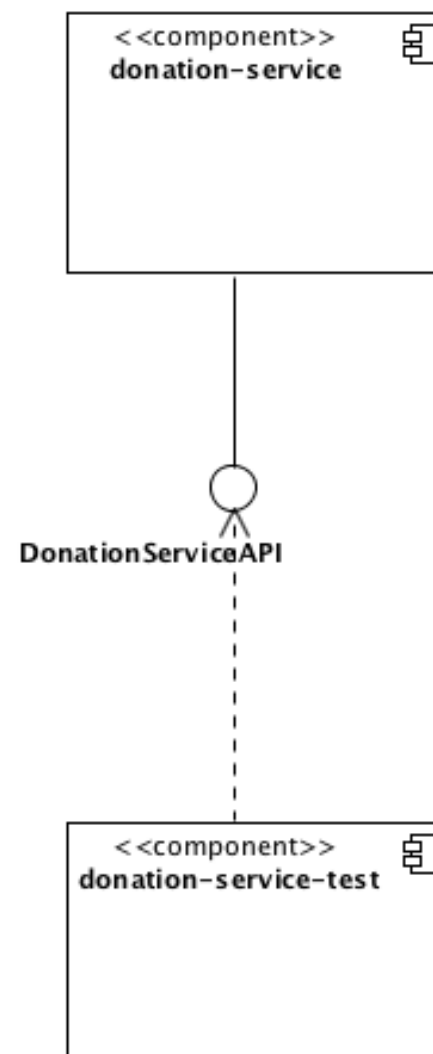
donation-web
api

```
{ method: 'GET', path: '/api/candidates', config: CandidatesApi.find },
{ method: 'GET', path: '/api/candidates/{id}', config: CandidatesApi.findOne },
{ method: 'POST', path: '/api/candidates', config: CandidatesApi.create },
{ method: 'DELETE', path: '/api/candidates/{id}', config: CandidatesApi.deleteOne },
{ method: 'DELETE', path: '/api/candidates', config: CandidatesApi.deleteAll },

{ method: 'GET', path: '/api/users', config: UsersApi.find },
{ method: 'GET', path: '/api/users/{id}', config: UsersApi.findOne },
{ method: 'POST', path: '/api/users', config: UsersApi.create },
{ method: 'DELETE', path: '/api/users/{id}', config: UsersApi.deleteOne },
{ method: 'DELETE', path: '/api/users', config: UsersApi.deleteAll },

{ method: 'GET', path: '/api/donations', config: DonationsApi.findAllDonations },
{ method: 'GET', path: '/api/candidates/{id}/donations', config: DonationsApi.findDonations },
{ method: 'POST', path: '/api/candidates/{id}/donations', config: DonationsApi.makeDonation },
{ method: 'DELETE', path: '/api/candidates/{id}/donations', config: DonationsApi.deleteDonations },
{ method: 'DELETE', path: '/api/donations', config: DonationsApi.deleteAllDonations },
```

- Use donation-service for
 - Login
 - Signup
 - Donate
 - List All Donations



donation-android

Donation
Android
Client

Android Project Configuration

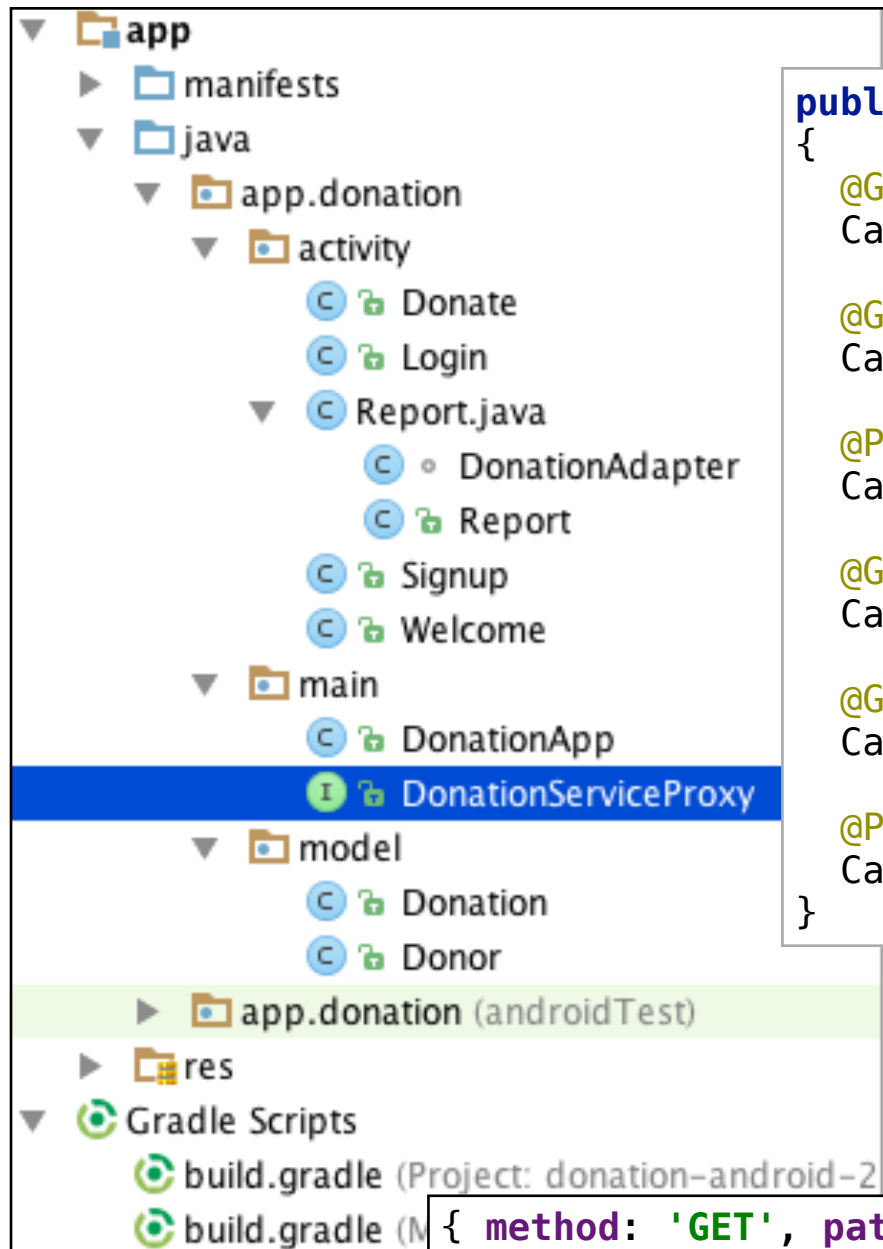
- Gradle - new libraries

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    testCompile 'junit:junit:4.12'  
    compile 'com.android.support:appcompat-v7:23.4.0'  
    compile 'com.squareup.retrofit2:retrofit:2.1.0'  
    compile 'com.google.code.gson:gson:2.7'  
    compile 'com.squareup.retrofit2:converter-gson:2.0.2'  
}
```

- AndroidManifest.xml
 - new 'permission' to access internet

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="app.donation" >  
  
    <uses-permission android:name="android.permission.INTERNET"/>  
  
    <application  
        android:name=".main.DonationApp"  
        android:allowBackup="true"  
        android:icon="@mipmap/ic_launcher"
```

Android Project New class - DonationService



```
public interface DonationService
{
    @GET("/api/users")
    Call<List<User>> getAllUsers();

    @GET("/api/users/{id}")
    Call<User> getUser(@Path("id") String id);

    @POST("/api/users")
    Call<User> createUser(@Body User user);

    @GET("/api/donations")
    Call<List<Donation>> getAllDonations();

    @GET("/api/candidates")
    Call<List<Candidate>> getAllCandidates();

    @POST("/api/candidates/{id}/donations")
    Call<Donation> createDonation(@Path("id") String id, @Body Donation donation);
}
```

```
{ method: 'GET', path: '/api/users', config: UsersApi.find },
{ method: 'GET', path: '/api/users/{id}', config: UsersApi.findOne },
{ method: 'POST', path: '/api/users', config: UsersApi.create },
{ method: 'GET', path: '/api/donations', config: DonationsApi.findAllDonations },
{ method: 'GET', path: '/api/candidates', config: CandidatesApi.find },
{ method: 'POST', path: '/api/candidates/{id}/donations', config: DonationsApi.makeDonation },
```

Mongoose Models

```
const userSchema = mongoose.Schema({  
  firstName: String,  
  lastName: String,  
  email: String,  
  password: String,  
});
```

```
const candidateSchema = mongoose.Schema({  
  firstName: String,  
  lastName: String,  
  office: String,  
});
```

```
const donationSchema = mongoose.Schema({  
  amount: Number,  
  method: String,  
  donor: {  
    type: mongoose.Schema.Types.ObjectId,  
    ref: 'User',  
  },  
  candidate: {  
    type: mongoose.Schema.Types.ObjectId,  
    ref: 'Candidate',  
  },  
});
```

Mongoose Models

Android Models

```
const userSchema = mongoose.Schema({
  firstName: String,
  lastName: String,
  email: String,
  password: String,
});
```

```
public class User
{
    public String _id;
    public String firstName;
    public String lastName;
    public String email;
    public String password;

    public User(String firstName, String lastName, String email, String password)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
    }
}
```

```
const candidateSchema = mongoose.Schema({
  firstName: String,
  lastName: String,
  office: String,
});
```

```
public class Candidate
{
    public String _id;
    public String firstName;
    public String lastName;
    public String office;

    public Candidate(String firstName, String lastName, String office)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        this.office = office;
    }
}
```

```
const donationSchema = mongoose.Schema({
  amount: Number,
  method: String,
  donor: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
  },
  candidate: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Candidate',
  },
});
```

```
public class Donation
{
    public Long id;
    public int amount;
    public String method;

    public Donation (int amount, String method)
    {
        this.amount = amount;
        this.method = method;
    }
}
```

DonationApp - Attributes

```
public class DonationApp extends Application
{
    public String          service_url    = "http://10.0.2.2:4000"; //Standard Emulator IP
    public DonationService donationService;
    public boolean         donationServiceAvailable = false;
    public User            currentUser;
    public List <Donation>  donations      = new ArrayList<Donation>();
    public List <User>      users          = new ArrayList<User>();
    public List <Candidate> candidates    = new ArrayList<Candidate>();
    ...
}
```

- 'service_url' string for remote service
- donationService member for accessing the service
- donationServiceAvailable flag if service offline
- currentUser - the logged in user
- donations - the list of donations made so far
- users - list of valid users
- candidates - list of available candidates

DonationApp - onCreate

```
@Override
public void onCreate()
{
    super.onCreate();
    Gson gson = new GsonBuilder().create();

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(service_url)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build();
    donationService = retrofit.create(DonationService.class);

    Log.v("Donation", "Donation App Started");
}
```

- Create the proxy service 'donationService', with the appropriate Gson parsers

Signup - standalone version

```
public class Signup extends AppCompatActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);
    }

    public void registerPressed (View view)
    {
        TextView firstName = (TextView) findViewById(R.id.firstName);
        TextView lastName = (TextView) findViewById(R.id.lastName);
        TextView email = (TextView) findViewById(R.id.Email);
        TextView password = (TextView) findViewById(R.id.Password);

        Donor user = new Donor(firstName.getText().toString(), lastName.getText().toString(),
                                email.getText().toString(), password.getText().toString());

        DonationApp app = (DonationApp) getApplication();
        app.newUser(user);

        startActivity (new Intent(this, Welcome.class));
    }
}
```

Signup

Sign up for the Donation App

Enter details below

First name

Last Name

Email

Password

REGISTER

Signup - Networked Version

```
public class Signup extends AppCompatActivity implements Callback<Donor>
{
    private DonationApp app;

    //.. as before

    public void registerPressed (View view)
    {
        //.. as before

        DonationApp app = (DonationApp) getApplication();
        Call<Donor> call = (Call<Donor>) app.donationService.createDonor(donor);
        call.enqueue(this);
    }

    @Override
    public void onResponse(Call<User> call, Response<User> response)
    {
        app.users.add(response.body());
        startActivity(new Intent(this, Welcome.class));
    }

    @Override
    public void onFailure(Call<User> call, Throwable t)
    {
        app.donationServiceAvailable = false;
        Toast toast = Toast.makeText(this, "Donation Service Unavailable..",
                                     Toast.LENGTH_LONG);
        toast.show();
        startActivity (new Intent(this, Welcome.class));
    }
}
```

- Callback interface for response from service
- Service Call
- Callback Handlers
 - for success
 - for error

Welcome - standalone version

```
public class Welcome extends AppCompatActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_welcome);
    }

    public void loginPressed (View view)
    {
        startActivity(new Intent(this, Login.class));
    }

    public void signupPressed (View view)
    {
        startActivity (new Intent(this, Signup.class));
    }
}
```

Donation

LOGIN

SIGN UP

Welcome - networked version

```
public class Welcome extends AppCompatActivity implements Callback<List<User>>
{
    private DonationApp app;

    @Override
    public void onResume()
    {
        super.onResume();
        app.currentUser = null;
        Call<List<User>> call = (Call<List<User>>) app.donationService.getAllUsers();
        call.enqueue(this);
    }

    @Override
    public void onResponse(Call<List<User>> call, Response<List<User>> response)
    {
        serviceAvailableMessage();
        app.users = response.body();
        app.donationServiceAvailable = true;
    }

    @Override
    public void onFailure(Call<List<User>> call, Throwable t)
    {
        app.donationServiceAvailable = false;
        serviceUnavailableMessage();
    }

    // ...
}
```

- Callback interface for response from service
- Service Call
- Callback Handlers
- for success
- for error

```
public class Welcome extends AppCompatActivity implements Callback<List<Donor>>
{
    //...

    public void loginPressed (View view)
    {
        if (app.donationServiceAvailable)
        {
            startActivity (new Intent(this, Login.class));
        }
        else
        {
            serviceUnavailableMessage();
        }
    }

    public void signupPressed (View view)
    {
        if (app.donationServiceAvailable)
        {
            startActivity (new Intent(this, Signup.class));
        }
        else
        {
            serviceUnavailableMessage();
        }
    }

    void serviceUnavailableMessage()
    {
        Toast toast = Toast.makeText(this, "Donation Service Unavailable. Try again later", Toast.LENGTH_LONG);
        toast.show();
    }

    void serviceAvailableMessage()
    {
        Toast toast = Toast.makeText(this, "Donation Contacted Successfully", Toast.LENGTH_LONG);
        toast.show();
    }
}
```

Welcome - Retrieve list of Candidates

```
@Override
public void onResume()
{
    //...

    Call<List<Candidate>> call2 = (Call<List<Candidate>>) app.donationService.getAllCandidates();
    call2.enqueue(new Callback<List<Candidate>>() {
        @Override
        public void onResponse(Call<List<Candidate>> call, Response<List<Candidate>> response) {
            app.candidates = response.body();
        }

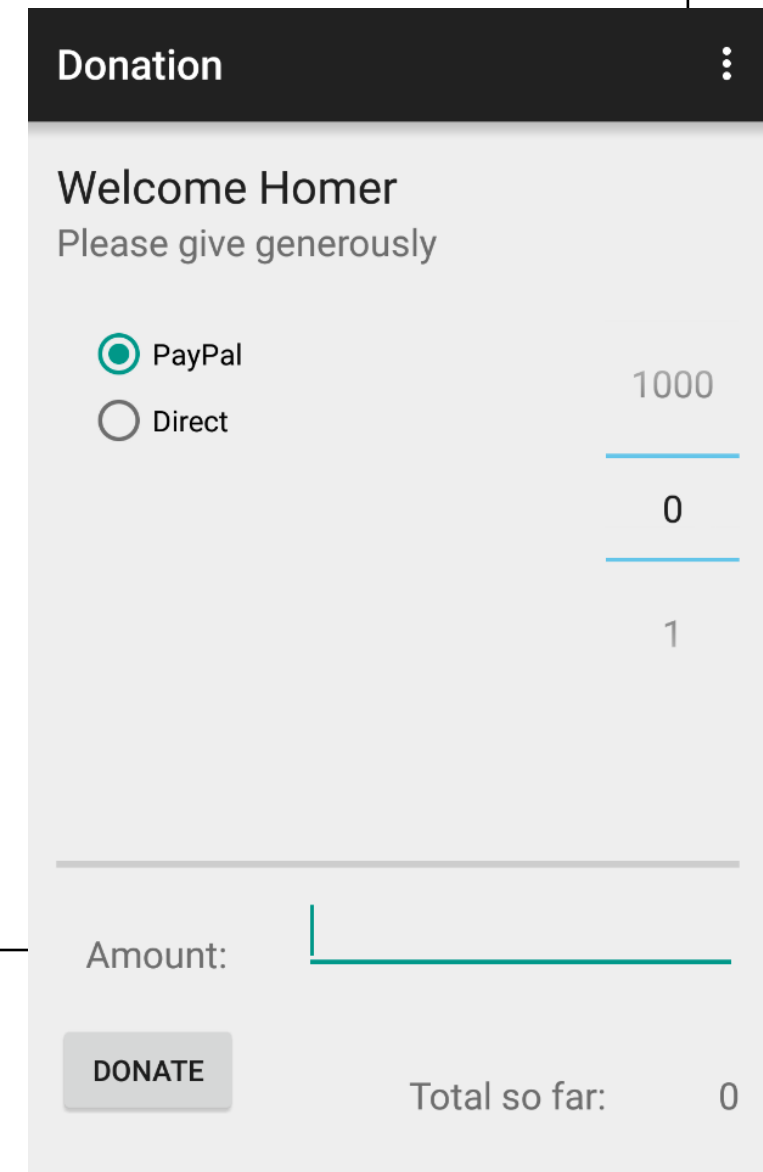
        @Override
        public void onFailure(Call<List<Candidate>> call, Throwable t) {
            app.donationServiceAvailable = false;
            serviceUnavailableMessage();
        }
    });
}
```

- Anonymous inner class version of remote call

Donate - Standalone Version

```
public class Donate extends AppCompatActivity
{

    public void donateButtonPressed (View view)
    {
        String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
        int donatedAmount = amountPicker.getValue();
        if (donatedAmount == 0)
        {
            String text = amountText.getText().toString();
            if (!text.equals(""))
                donatedAmount = Integer.parseInt(text);
        }
        if (donatedAmount > 0)
        {
            app.newDonation(new Donation(donatedAmount, method));
            progressBar.setProgress(app.totalDonated);
            String totalDonatedStr = "$" + app.totalDonated;
            amountTotal.setText(totalDonatedStr);
        }
        amountText.setText("");
        amountPicker.setValue(0);
    }
}
```



The screenshot shows the 'Donate' app interface. At the top is a dark header with the title 'Donation' and a menu icon. Below the header, the text 'Welcome Homer' and 'Please give generously' is displayed. There are two radio buttons: 'PayPal' (selected) and 'Direct'. To the right of these buttons is a vertical stack of numbers: '1000', '0', and '1', each with a horizontal line underneath. At the bottom, there is a text input field labeled 'Amount:' and a 'DONATE' button. To the right of the button, it says 'Total so far: 0'.

Donate - With Candidate Support

- Valid candidates, retrieved in Welcome, displayed in a spinner here

The screenshot shows a mobile application interface for donations. At the top, a blue header bar contains the word "Donation". Below this, a white area displays "Welcome Homer" and "Please give generously". There are two radio buttons: "PayPal" (selected) and "Direct". To the right of these is a numeric input field showing "0". Below the radio buttons is a spinner menu currently displaying "lisa". An arrow from the text "Valid candidates, retrieved in Welcome, displayed in a spinner here" points to this spinner. Below the spinner is a label "Amount:" followed by a red vertical line. At the bottom left is a grey button labeled "DONATE". At the bottom right is the text "Total so far: 0". The bottom of the screen features a black Android navigation bar with back, home, and recent apps icons.

Candidate Adapter

- Populate the spinner with the list of candidate names

```
private class CandidateAdapter extends BaseAdapter implements SpinnerAdapter {
    private final List<Candidate> data;

    public CandidateAdapter(List<Candidate> data) {
        this.data = data;
    }

    @Override
    public int getCount() {
        return data.size();
    }

    @Override
    public Object getItem(int position) {
        return data.get(position);
    }


    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int position, View recycle, ViewGroup parent) {
        TextView text;
        if (recycle != null) {
            text = (TextView) recycle;
        } else {
            text = (TextView) getLayoutInflater().inflate(
                android.R.layout.simple_dropdown_item_1line, parent, false
            );
        }
        text.setTextColor(Color.BLACK);
        text.setText(data.get(position).firstName);
        return text;
    }
}
```

Donate - Networked Version

```
public class Donate extends AppCompatActivity implements Callback<Donation>
{
    ...

    public void donateButtonPressed (View view)
    {
        String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
        int donatedAmount = amountPicker.getValue();
        if (donatedAmount == 0)
        {
            String text = amountText.getText().toString();
            if (!text.equals(""))
                donatedAmount = Integer.parseInt(text);
        }
        if (donatedAmount > 0)
        {
            Donation donation = new Donation(donatedAmount, method);
            Candidate candidate = (Candidate) candidateSelection.getSelectedItem();
            Call<Donation> call = (Call<Donation>) app.donationService.createDonation(candidate._id, donation);
            call.enqueue(this);
        }
        amountText.setText("");
        amountPicker.setValue(0);
    }
}
```



• Service Call

Donate - Networked Version

```
public class Donate extends AppCompatActivity implements Callback<Donation>
{
    ...

    @Override
    public void onResponse(Call<Donation> call, Response<Donation> response)
    {
        Toast toast = Toast.makeText(this, "Donation Accepted", Toast.LENGTH_SHORT);
        toast.show();
        app.newDonation(response.body());
        progressBar.setProgress(app.totalDonated);
        String totalDonatedStr = "$" + app.totalDonated;
        amountTotal.setText(totalDonatedStr);
        amountText.setText("");
        amountPicker.setValue(0);
    }

    @Override
    public void onFailure(Call<Donation> call, Throwable t)
    {
        Toast toast = Toast.makeText(this, "Error making donation", Toast.LENGTH_LONG);
        toast.show();
    }

    ...
}
```

- Service Response

Report - Standalone Version

```
public class Report extends AppCompatActivity
{
    private ListView    listView;
    private DonationApp app;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        app = (DonationApp) getApplication();

        listView = (ListView) findViewById(R.id.reportList);
        DonationAdapter adapter = new DonationAdapter (this, app.donationList,
        listView.setAdapter(adapter);
    }
    ...
}
```

Donation



Report

210	paypal
20	cash
330	cash
10	paypal
999	PayPal

Report - Networked Version

```
public class Report extends AppCompatActivity implements Callback<List<Donation>>
{
    private ListView        listView;
    private DonationApp     app;
    private DonationAdapter adapter;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        app = (DonationApp) getApplication();

        listView = (ListView) findViewById(R.id.reportList);
        adapter = new DonationAdapter (this, app.donations);
        listView.setAdapter(adapter);

        Call<List<Donation>> call = (Call<List<Donation>>) app.donationService.getAllDonations();
        call.enqueue(this);
    }

    @Override
    public void onResponse(Call<List<Donation>> call, Response<List<Donation>> response)
    {
        adapter.donations = response.body();
        adapter.notifyDataSetChanged();
    }

    @Override
    public void onFailure(Call<List<Donation>> call, Throwable t)
    {
        Toast toast = Toast.makeText(this, "Error retrieving donations", Toast.LENGTH_LONG);
        toast.show();
    }

    //...
}
```