Android Development Tutorial

ANDROID

David Weiser
Marlo Häring

Who are we?

David Weiser

NABU team member

Marlo Häring

Audi team supervisor

GitHub

- Create your account on GitHub:
  - https://github.com/

- Simple Guides
  - https://try.**github**.io/
  - https://rogerdudler.github.io/git-guide/index.html

Outline of the Talk

1. **Introduction**
2. Android Components (Activities, Layouts, Controls)
3. Resource Management (R class, action listener)
4. Meta information and Interactions (Manifest, Intents, Permissions)
5. Testing (Debug, Deploy)

Introduction

- What is Android?
- Why Android?
- Android Architecture
- Basic Android Application Components
- Android distribution
- Integrated Development Tools
- The Android developer lifecycle

What is Android?

- Linux based operating system

- Built for mobile devices like smartphones, tablets, watches, cars

- Developed by Android, Inc. (acquired by Google in 2005)

- Open source – Apache License

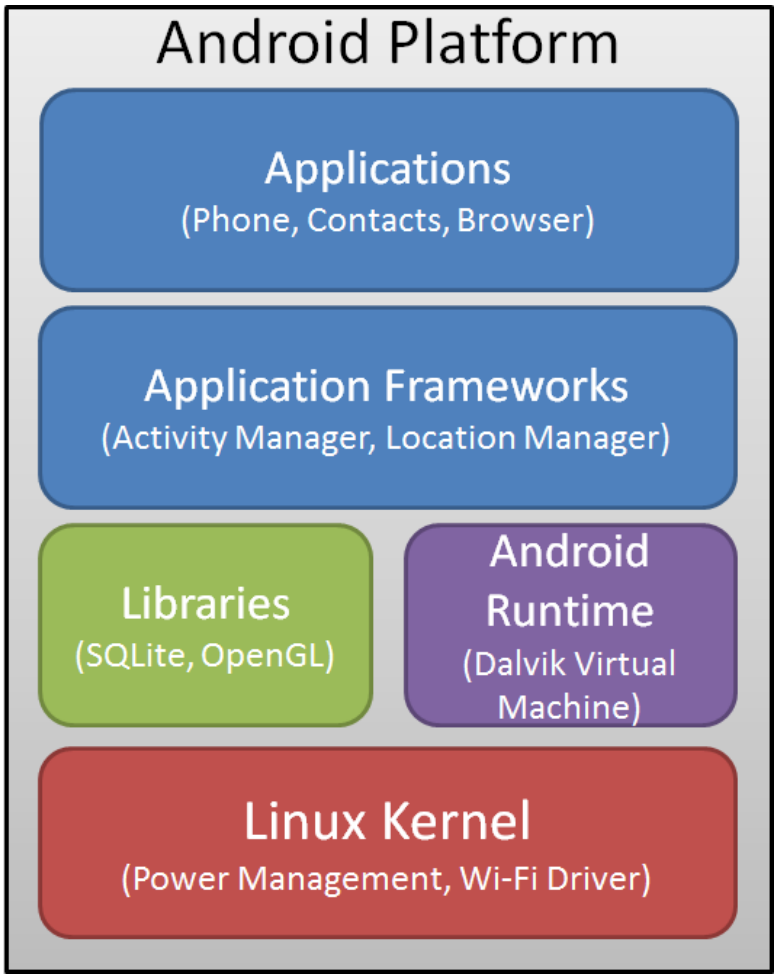
Why Android?

- There are more mobile devices than computers
- Android has the highest market share

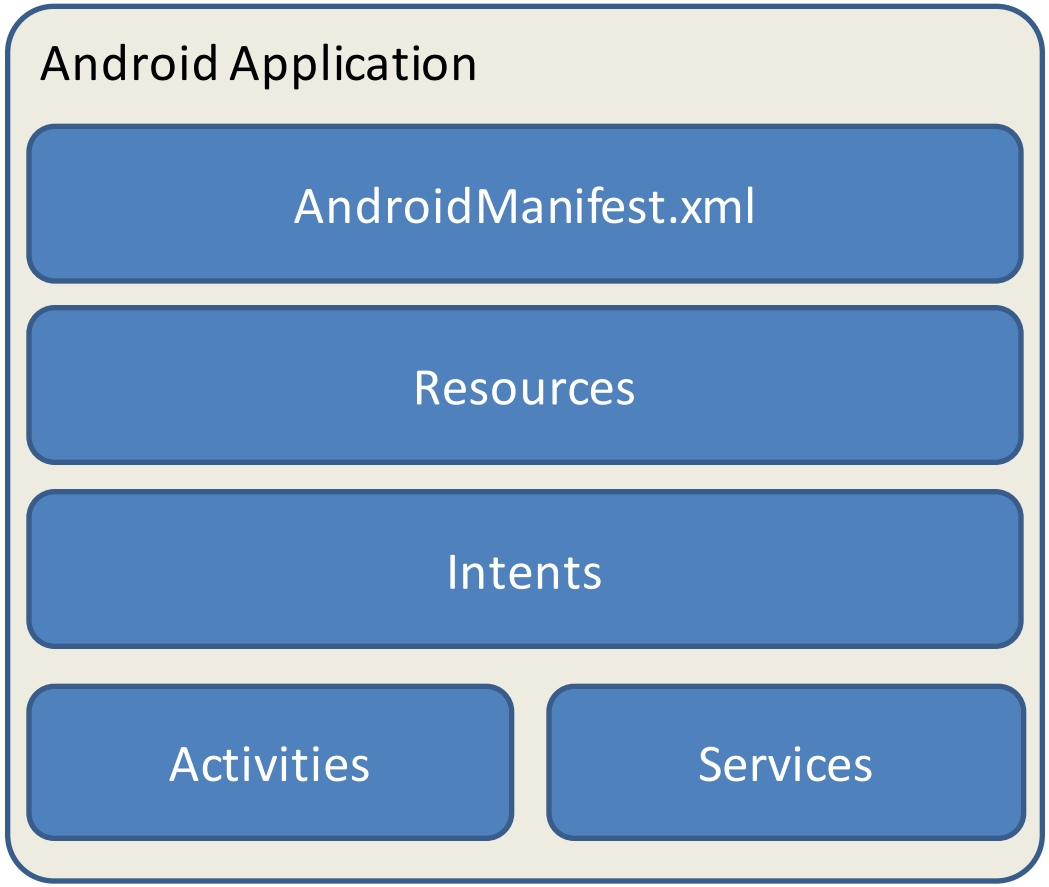
Operating System	2015 Q2 Market Share
Android	82.8%
iOS	13.9%
Windows Phone	2.6%
Blackberry	0,3%
Others	0,4%

[IDC, 2015]

Android Architecture

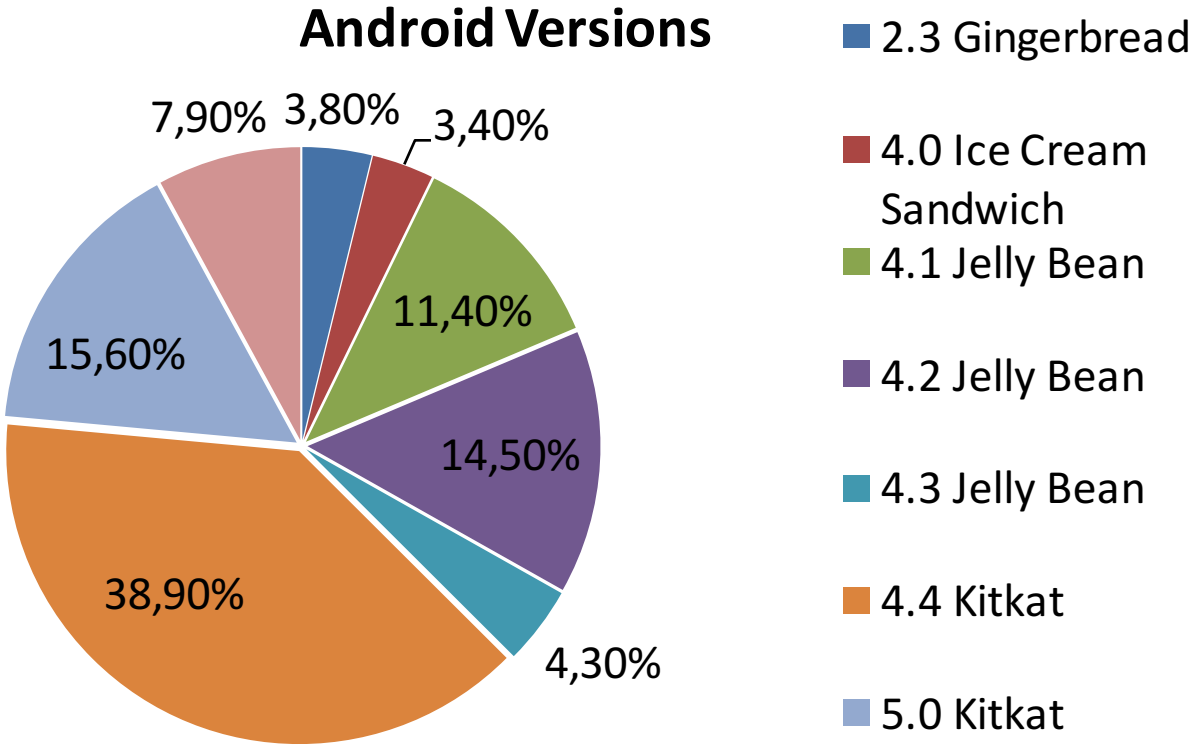


Basic Android Application Components



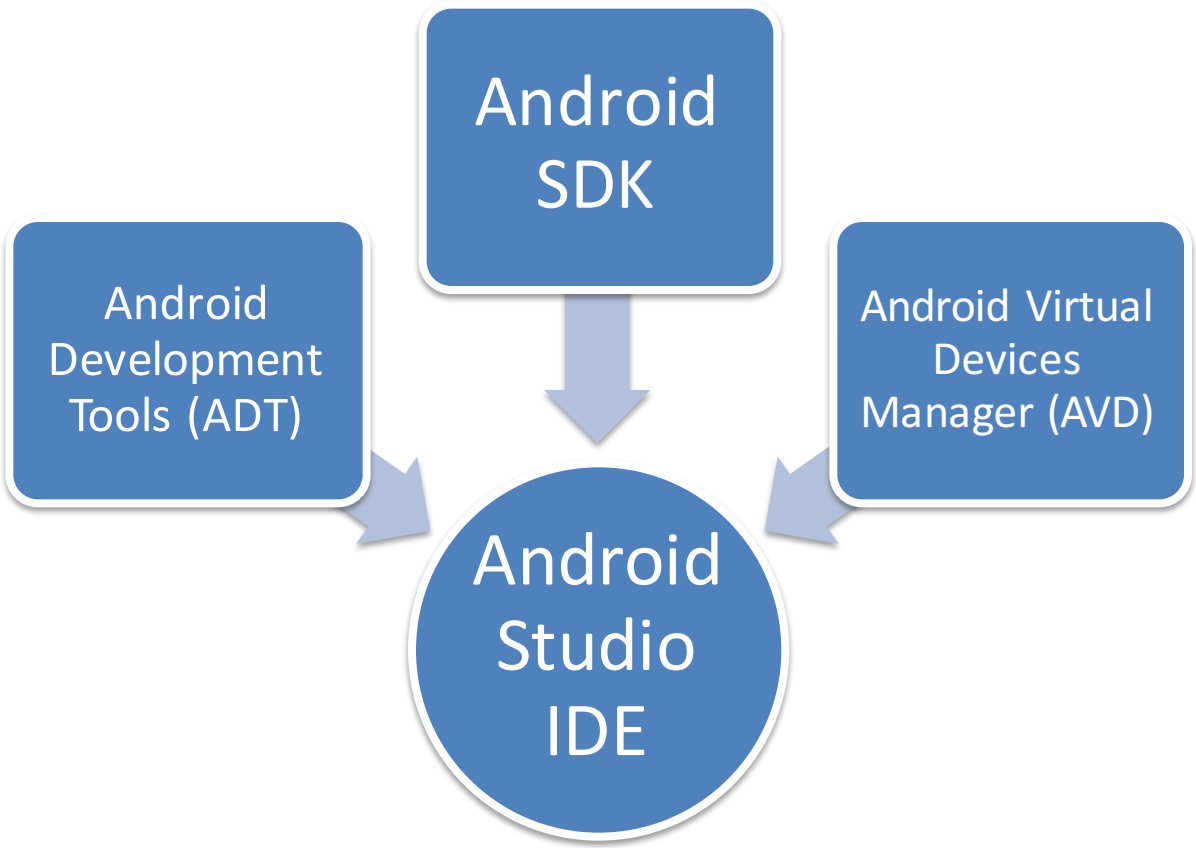
Distribution of Platform Versions

- You cover ~95% of all Android devices if you develop for version 4.0 Ice Cream Sandwich



[developer.android.com, 2015]

Integrated Development Tools



https://developer.android.com/sdk/index.html

The Android developer lifecycle

Design

- Design the user interface
- How will user interact with your Application?
- Keep it simple, make it powerful and intuitive

Develop

- Get the tools
- Get to know the Android API
- Start coding and testing

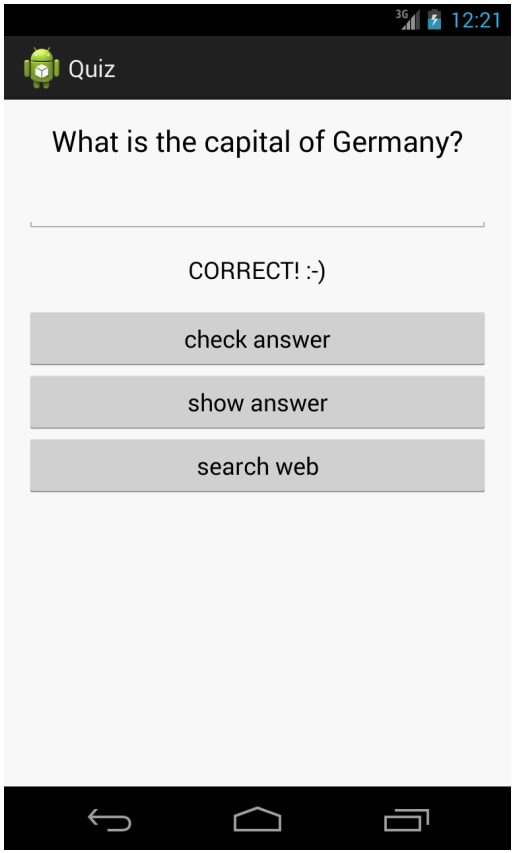
Distribute

- Get to know Google Play and create an account
- Think about visibility, monetizing and distribution
- Check Google's publishing checklist

A Quiz App

David Weiser
Marlo Häring

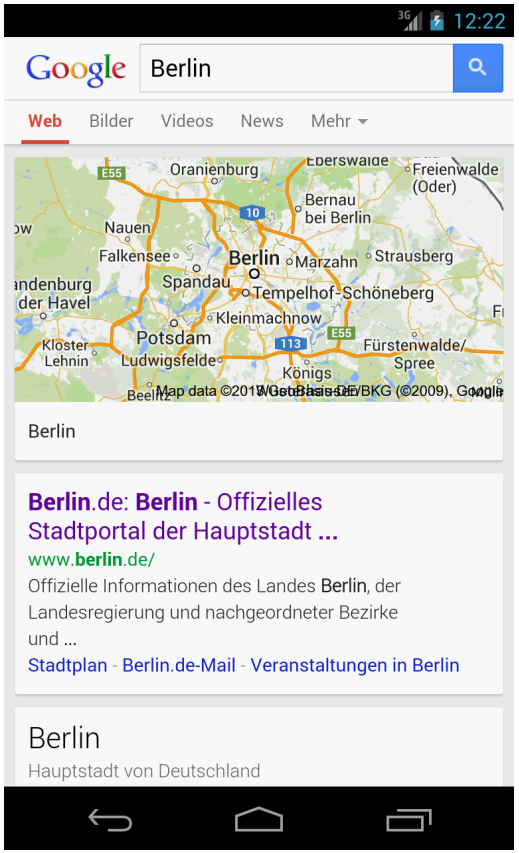
Target for the day



← Quiz

with

websearch→



Exercise

- Create your first project
- Use your own device or create an emulator.
- Browse through the project structure of the Android application

Outline of the Talk

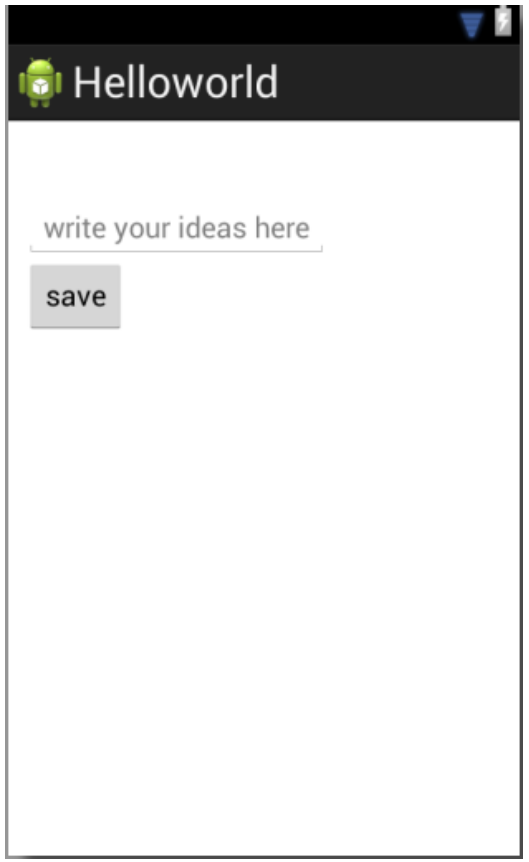
1. Introduction
2. **Android Components (Activities, Layouts, Controls)**
3. Resource Management (R class, action listener)
4. Meta information and Interactions (Manifest, Intents, Permissions)
5. Testing (Debug, Deploy)

Activities, Layouts, Controls

- One simple example
  - What's behind the scene?
- What are Activities?
- Activity lifecycle
- Layouts
  - Other application components
- Controls

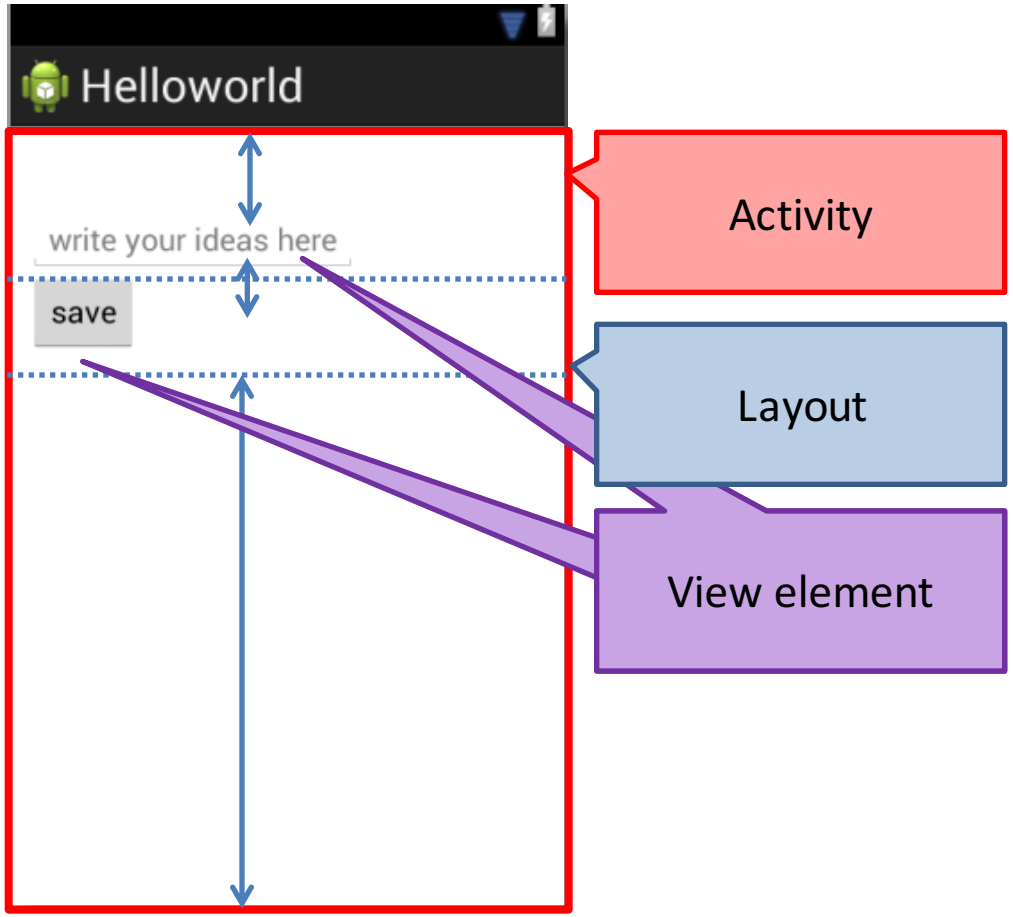
One simple example

- Screenshot from the emulator
- What you can see:
  - Icon and application name on the top
  - A text-input field
  - A button called “save”



Screenshot from Emulator

What's behind the scene?



Activity

An Activity represents one screen

Layout

A Layout arranges the components

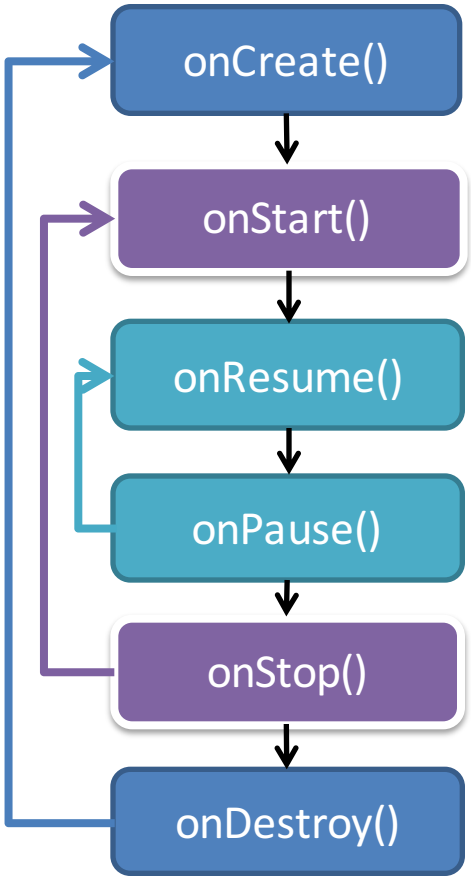
View element

Add View elements for user interactions, to display something, etc.

What are Activities?

- Activities are Java classes
- They represent a single screen with a user interface
- Specific lifecycle

Activity lifecycle



Simplified lifecycle

Called when the activity is first created **(mandatory!)**

Called just before the activity becomes visible

Called just before the activity is interacting with the user

Another activity comes to the foreground

The activity is no longer visible

Destroyed by the system or finished by the user

Other application components

- Service
  - A Service runs in the background and has no user interface
  - E.g.: playing audio in the background

- Content Provider
  - Manages shared data of your application
  - E.g.: applications can access your contacts information on your phone

- Broadcast Receiver
  - Listens and responds to system wide announcements
  - E.g.: low battery, picture captured

Which layouts exists? – some examples

- Linear layout
  - Arrange items horizontal or vertical

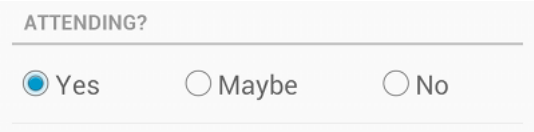
- Relative layout
  - Arranges items relative to each other

- List view
  - Shows one item under another

- Grid view
  - Display a set of columns and rows

Controls – some examples

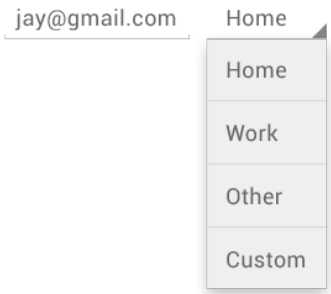
Button, Image button



Radio button

Textfields

Check box



Spinner (drop down menu)

Exercise

- Create a GUI for the Quiz
- Deploy the app on the emulator or on your own device
- Launch the app

Exercise – The GUI

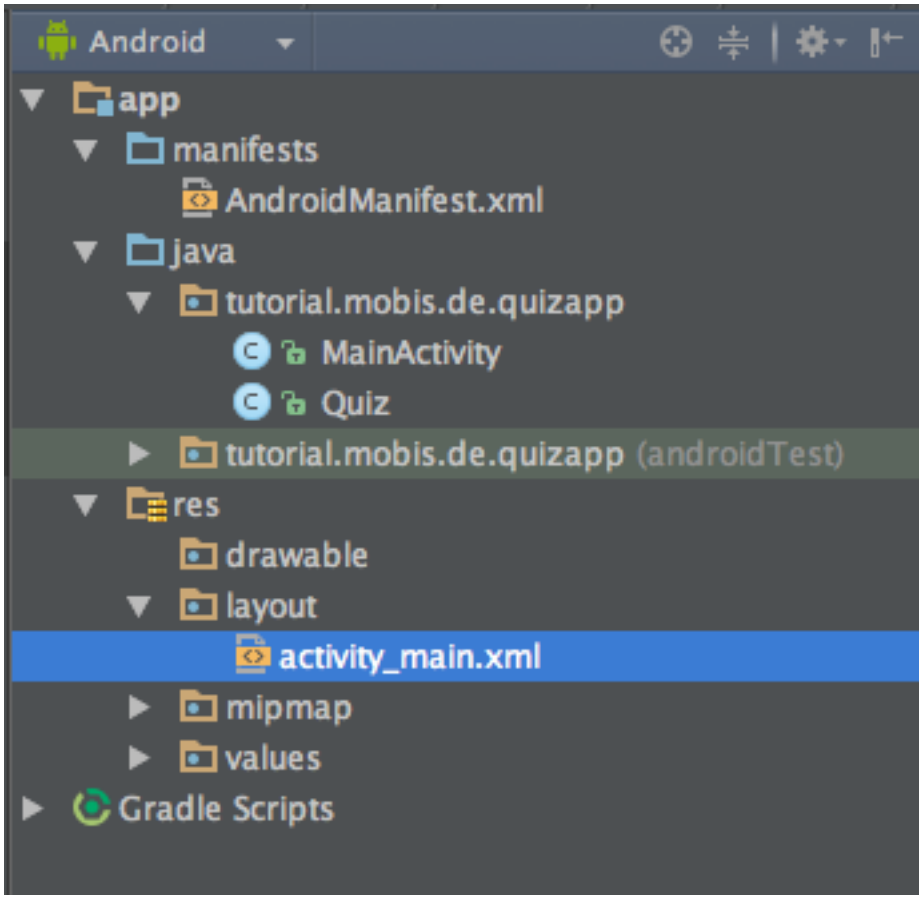
TextView

EditText

TextView

Buttons

Solution – step 1



Open the layout file

Solution – layout file 1/2

```
13      <TextView
14          android:id="@+id/text_question"
15          android:layout_width="wrap_content"
16          android:layout_height="wrap_content"
17          android:layout_gravity="center_horizontal"
18          android:text="@string/text_question"
19          android:textAppearance="?android:attr/textAppearanceLarge" />
20
21      <EditText
22          android:id="@+id/input_answer"
23          android:layout_width="match_parent"
24          android:layout_height="wrap_content"
25          android:layout_marginTop="@dimen/activity_vertical_margin"
26          android:ems="10"
27          android:inputType="text" >
28
29          <requestFocus />
30      </EditText>
31
32      <TextView
33          android:id="@+id/text_solution"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:layout_gravity="center_horizontal"
37          android:layout_marginTop="@dimen/activity_vertical_margin"
38          android:textAppearance="?android:attr/textAppearanceMedium" />
```

Solution – layout file 2/2

```
40      <Button
41          android:id="@+id/button_check_answer"
42          android:layout_width="match_parent"
43          android:layout_height="wrap_content"
44          android:layout_marginTop="16dp"
45          android:onClick="onClick"
46          android:text="@string/button_check_answer" />
47
48      <Button
49          android:id="@+id/button_show_answer"
50          android:layout_width="match_parent"
51          android:layout_height="wrap_content"
52          android:onClick="onClick"
53          android:text="@string/button_show_answer" />
54
55      <Button
56          android:id="@+id/button_search_web"
57          android:layout_width="match_parent"
58          android:layout_height="wrap_content"
59          android:onClick="onClick"
60          android:text="@string/button_search_web" />
```

Outline of the Talk

1. Introduction
2. Android Components (Activities, Layouts, Controls)
3. **Resource Management (R class, action listener)**
4. Meta information and Interactions (Manifest, Intents, Permissions)
5. Testing (Debug, Deploy)

The R class

- R.java provides accessible references to available resources
- The resources that are being referenced are:
  - View elements
  - Layout files
  - Strings
  - Icons

- R.java is dynamically generated at build-time

The R class

```
10 public final class R {
11     public static final class attr {
12     }
13     public static final class dimen {
14         /**  Default screen margins, per the Android Design guidelines.
15
16          Customize dimensions originally defined in res/values/dimens.xml (such as
17          screen margins) for sw720dp devices (e.g. 10" tablets) in landscape here.
18
19          */
20         public static final int activity_horizontal_margin=0x7f040000;
21         public static final int activity_vertical_margin=0x7f040001;
22     }
23     public static final class drawable {
24         public static final int ic_launcher=0x7f020000;
25     }
26     public static final class id {
27         public static final int LinearLayout1=0x7f080000;
28         public static final int action_settings=0x7f080007;
29         public static final int button_check_answer=0x7f080004;
30         public static final int button_search_web=0x7f080006;
31         public static final int button_show_answer=0x7f080005;
32         public static final int input_answer=0x7f080002;
33         public static final int text_question=0x7f080001;
34         public static final int text_solution=0x7f080003;
35     }
36     public static final class layout {
37         public static final int activity_main=0x7f030000;
38     }
```

R class: exemplary usage

- Assign a Layouts to an Activity
  - setContentView(R.layout.activity_main);

- Get references to view elements
  - TextView questionTextView =
    (TextView) findViewById(R.id.text_question);

- Get String resources
  - String helloWorld = getString(R.string.text_hello_world);

Exercise – handle button clicks

- Create an onClick-Method
- Create a pop-up window that will be displayed after a button click
- Test your App in the Emulator

Solution

Pop-up toast on a button click

Exercise – source code

```
public void onClick(View view) {
    switch (view.getId()) {
    case R.id.button_check_answer:

        Toast.makeText(
                this,
                "check answer clicked",
                Toast.LENGTH_LONG
        ).show();

        break;
}
```

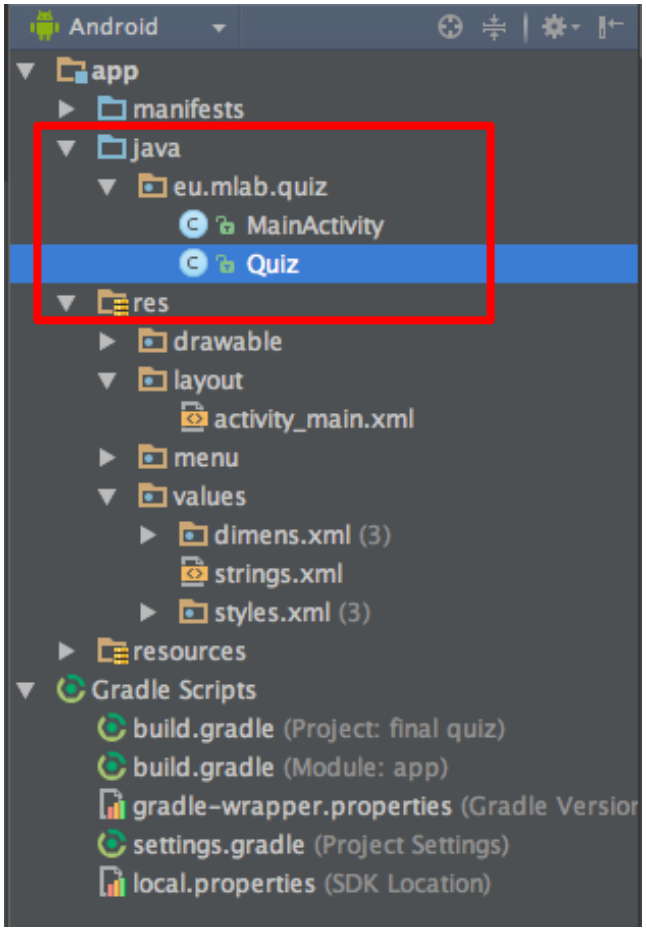
Method from the layout file

The IDs are also written in the layout file

- Create a pop-up window for each button in the layout

The Quiz.java class

- All Java classes are located in the“src“ folder
- Place Quiz.java in the package of your MainActivity



activity_main.xml | MainActivity.java | Quiz.java | Quiz Manifest

```java
package eu.mlab.quiz;

import java.util.ArrayList;

public class Quiz {
    private List<String[]> quizQuestionList;
    private String[] currentQuestion;

    public Quiz() {
        quizQuestionList = new ArrayList<String[]>();
        fillQuizList();
    }

    private void fillQuizList() {
        String[] question1 = {"What is the capital of Germany?", "Berlin"};
        quizQuestionList.add(question1);

        String[] question2 = {"What is the capital of Australia", "Canberra"};
        quizQuestionList.add(question2);

        String[] question3 = {"What is the capital of Denmark", "Kopenhagen"};
        quizQuestionList.add(question3);
    }

    public String getRandomQuestion() {
        int pos = new Random().nextInt(quizQuestionList.size());
        currentQuestion = quizQuestionList.get(pos);

        return currentQuestion[0];
    }

    public boolean checkAnswer(String answer) {
        if(currentQuestion[1].equalsIgnoreCase(answer)) {
            return true;
        }
        return false;
    }

    public String getSolution() {
        return currentQuestion[1];
    }
}
```

Exercise – MainActivity.java

- Use Quiz.java in the MainActivity
- Initialize Quiz.java in MainActivity.java
- Check whether user's input is correct
- Show the answers if the corresponding button is clicked
- Test your implementation on your device or with the emulator

Initialize MainActivity

```
11 public class MainActivity extends Activity {
12     private TextView questionTextView;
13     private TextView solutionTextView;
14     private EditText answerEditText;
15
16     private Quiz quiz;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         init();
24     }
25
26     private void init() {
27         questionTextView = (TextView) findViewById(R.id.text_question);
```

Initialize solutionTextView and answerEditText

```
31         quiz = new Quiz();
32         showNextQuestion();
33     }
```

Implement Click Listener

- checkAnswer():
  - Get the content of the EditText and compare it with the solution in Quiz.java
  - Show the next question if the answer is correct
- showAnwser():
  - Show the solution in the solutionTextView
- searchWeb(): will be implemented later

```
35      public void onClick(View view) {
36          switch (view.getId()) {
37          case R.id.button_check_answer:
38              checkAnswer();
39              answerEditText.setText("");
40              break;
41
42          case R.id.button_show_answer:
43              showAnswer();
44              break;
45
46          case R.id.button_search_web:
47              searchWeb();
48              break;
49          }
50      }
```

Solution

```
52    private void checkAnswer() {
53        String answer = answerEditText.getText().toString();
54
55        if(quiz.checkAnswer(answer)) {
56            solutionTextView.setText("CORRECT! :-)");
57            showNextQuestion();
58        }
59        else {
60            solutionTextView.setText("FALSE! :-(");
61        }
62    }
```

```
64    private void showAnswer() {
65        solutionTextView.setText(quiz.getSolution());
66    }
67
68    private void showNextQuestion() {
69        questionTextView.setText(quiz.getRandomQuestion());
70    }
```

Outline of the Talk

1. Introduction
2. Android Components (Activities, Layouts, Controls)
3. Resource Management (R class, action listener)
4. **Meta information and Interactions (Manifest, Intents, Permissions)**
5. Testing (Debug, Deploy)

Manifest, Intents, Permissions

- One activity isn't enough!
- Two types of Intents
  - Explicit intents
  - Implicit intents

- Data storage options

- Android manifest

- Security and permissions

One activity isn't enough!

- Every activity represents one screen
- Most applications have complex UIs
  - Connect multiple activities using intents
- Intents…
  - …connect activities
  - …are requests for actions:
    - Start an activity, service, broadcast intent
  - …can carry data

Explicit and implicit intents

- Explicit intents
  - Handle an intent by a specific component
    - ```
      Intent intent = new Intent(this, Activity.class);
      startActivity(intent);
      ```

- Implicit intents
  - The Android OS looks for components that can handle an intent
    - ```
      Intent intent = new Intent(Intent.ACTION_CALL,
      Uri.parse("tel:(+49)12345789"));
      startActivity(intent);
      ```

Implicit intents

- Some examples of possible actions:

Constant	Target component	Action
ACTION_CALL	Activity	Initiate a phone call
ACTION_MAIN	Activity	Start up as initial activity
ACTION_SYNC	Activity	Syncronize server and mobile phone data
ACTION_BATTERY_LOW	Broadcast receiver	Warn that the battery is low
ACTION_SCREEN_ON	Broadcast receiver	Screen is turned on

Data storage options

- Shared Preferences
  - Stores primitive data as key-value pairs
- Internal Storage
  - Stores private data on the device memory
- External Storage
  - Stores public data on an external storage
- SQLite Databases
  - Stores structured data in a database
- Network Connection
  - Stores data on your own server

Android Manifest

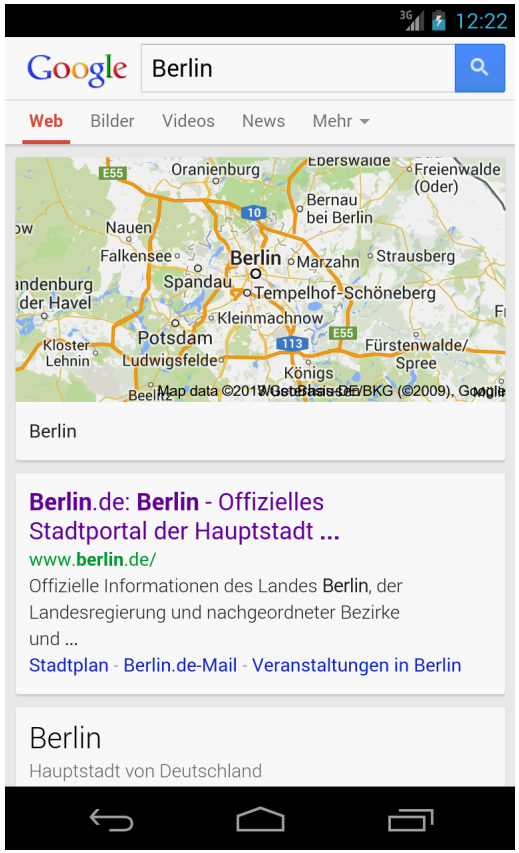
- Every Applications must have an AndroidManifest.xml file
- The manifest is located in the root directory of an Application
- Some of the information it contains:
  - SDK version compatibility
  - Components registration (e.g. activities, services, etc.)
  - The permissions that the Application requires

Permissions

- Permissions define Application's access to a part of the code or data
- Some permissions are:
  - ACCESS_FINE_LOCATION
  - INTERNET
  - WRITE_EXTERNAL_STORAGE
  - WRITE_SMS

Exercise

- Objective: search for the solution with Google in a browser automatically
- Use an implicit intent
- Test the application

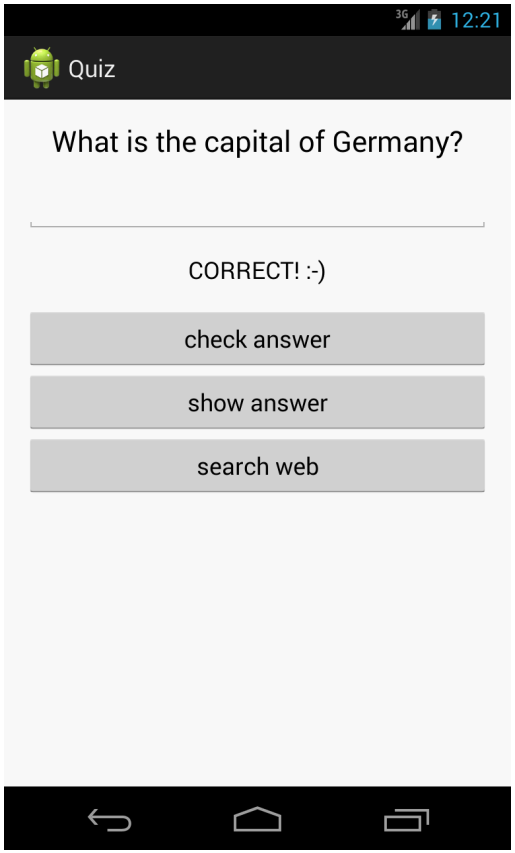


Solution – source code

- The Android system takes care of implicit intents, you just have to:
  - Declare an Action so that the system understands what you want to do
  - Forward data for the action
  - Start the Activity

```
72    private void searchWeb() {
73        Intent searchInternetIntent =
74                new Intent(
75                        Intent.ACTION_VIEW,
76                        Uri.parse("http://www.google.de/search?q=" + quiz.getSolution())
77                );
78        startActivity(searchInternetIntent);
79    }
```

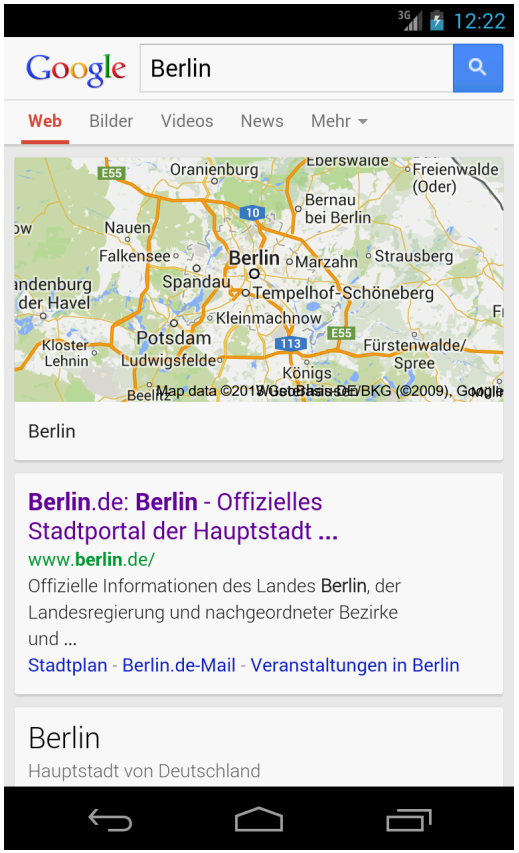
Solution



← Quiz

with

websearch→



Outline of the Talk

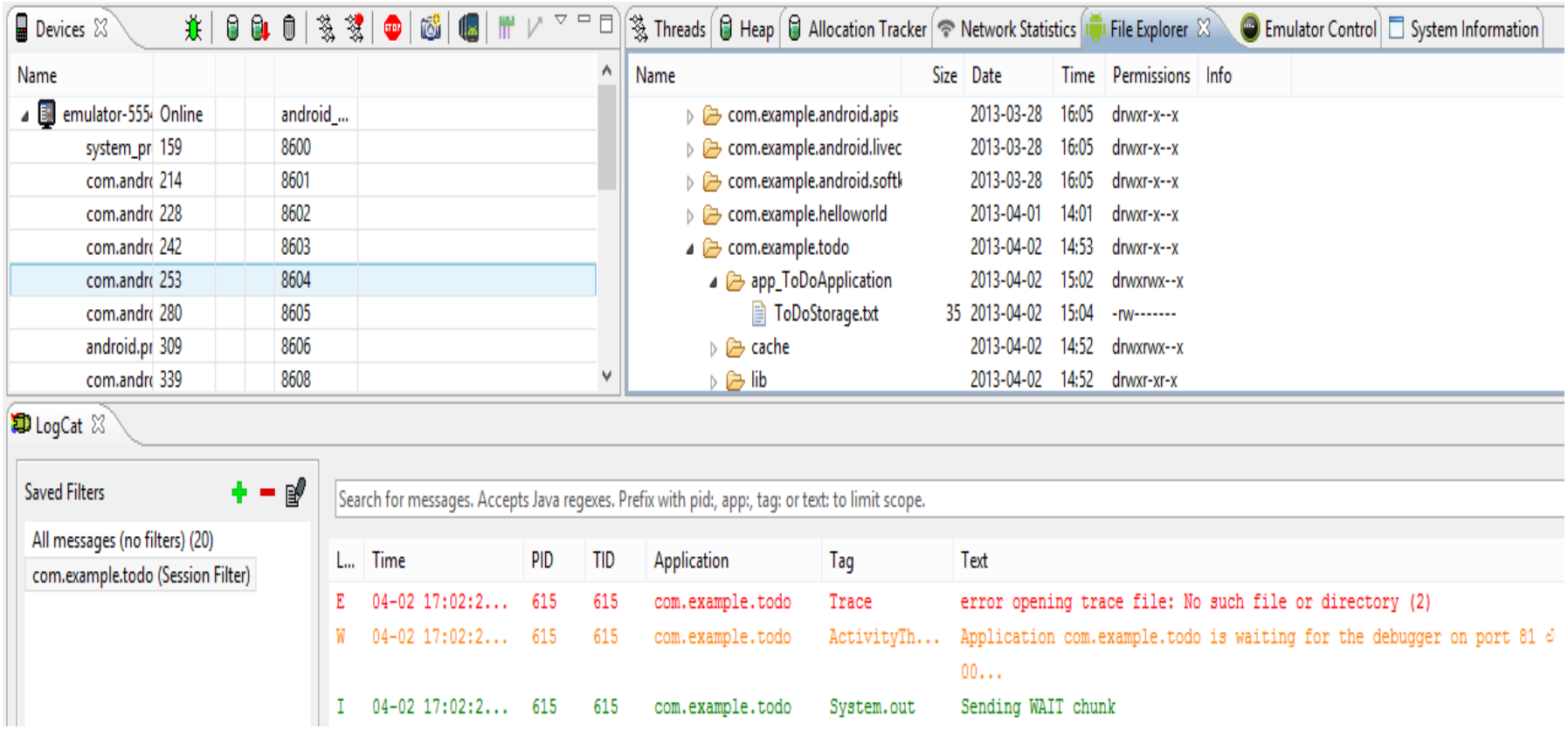
1. Introduction
2. Android Components (Activities, Layouts, Controls)
3. Resource Management (R class, action listener)
4. Meta information and Interactions (Manifest, Intents, Permissions)
5. **Testing (Debug, Deploy)**

Debug, Deploy

- Dalvik Debug Monitor Server (DDMS)
  - Main features

- LogCat

Android Device Monitor (ADM)



ADM - Main Features

- Take screenshots
- View logs in LogCat
- File explorer of the emulator or device
- View heap memory usage of processes
- Mock your location
- Fake incoming sms or calls
- Watch your network traffic

LogCat

- Having trouble finding the source of an exception?
  - Write logs to narrow the location of your exception

- Log levels:
  - VERBOSE
  - DEBUG
  - INFO
  - WARN
  - ERROR
  - ASSERT

Exercise

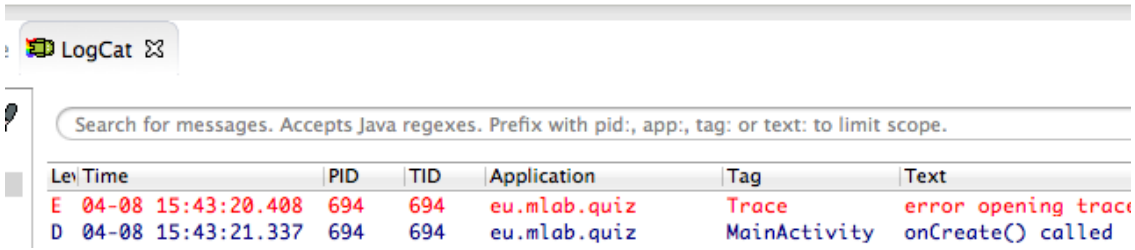
- Explore the Android Device Monitor
- Write your own debug code
- Test your app

Solution

- Code

```
ublic class MainActivity extends Activity {
    public static final String TAG = MainActivity.class.getSimpleName();

    private TextView questionTextView;
    private TextView solutionTextView;
    private EditText answerEditText;

    private Quiz quiz;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Log.d(TAG, "onCreate() called");
        init();
    }
```

- LogCat snippet



Android Development Tutorial

Thank You!

Online Tutorials

- Coursera Online Lecture -

  https://www.coursera.org/course/android

- Vogella Android Tutorial -

  http://www.vogella.com/tutorials/Android/article.html

- Official Android Training -

  http://developer.android.com/training/index.html