Android Development Environment
The Android Platform

A multi-layered software stack for building and running mobile applications
The Android Development Environment

Your workbench for writing Android applications

See:

https://developer.android.com/studio/intro/
Today’s Topics

Downloading Android SDK
Using the Android Studio IDE
Using the Android emulator
Debugging Android applications
Other tools
Prerequisites

Supported Operating Systems:

Microsoft Windows 7/8/10 (32- or 64-bit)
Mac OS X 10.10 (Yosemite) up to 10.12 (Sierra)
GNOME or KDE desktop (tested on Ubuntu 14.04 LTS, Trusty Tahr)
Prerequisites

3 GB RAM min, 4 GB RAM rec
1 GB+ for Android SDK, emulator system images, and caches
1280 x 800 min screen resolution
Getting Started

Download & install Android Studio

See: https://developer.android.com/studio/index.html#downloads
Android Studio

Android platform

Android Studio IDE

Key development tools

System image for emulator
Hello World!
package course.examples.helloworld;

...

public class MainActivity extends Activity {

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

The Android Emulator

Runs virtual devices
## Your Virtual Devices

### Android Virtual Device Manager

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Resolution</th>
<th>API</th>
<th>Target</th>
<th>CPU/ABI</th>
<th>Size on Disk</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaxy Nexus API 18</td>
<td>Galaxy Nexus API 18</td>
<td>720 x 1280: xhdpi</td>
<td>18</td>
<td>Android 4.3 (Google APIs)</td>
<td>x86</td>
<td>4 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Galaxy Nexus API 22</td>
<td>Galaxy Nexus API 22</td>
<td>720 x 1280: xhdpi</td>
<td>22</td>
<td>Android 6.1 (Google APIs)</td>
<td>x86</td>
<td>1 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Nexus 7_API 18</td>
<td>Nexus 7_API 18</td>
<td>800 x 1280: tvdpi</td>
<td>18</td>
<td>Android 4.3</td>
<td>arm</td>
<td>123 MB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Nexus 5 API 22 v1</td>
<td>Nexus 5 API 22 v1</td>
<td>1080 x 1920: xhdpi</td>
<td>22</td>
<td>Android 5.1 (Google APIs)</td>
<td>x86</td>
<td>1 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Nexus 6 API 22 v2</td>
<td>Nexus 6 API 22 v2</td>
<td>1080 x 1920: xhdpi</td>
<td>22</td>
<td>Android 6.1 (Google APIs)</td>
<td>x86</td>
<td>1 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Nexus 5X API 23 New</td>
<td>Nexus 5X API 23 New</td>
<td>1080 x 1920: xhdpi</td>
<td>23</td>
<td>Android 6.0 (Google APIs)</td>
<td>x86</td>
<td>1 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Pixel API 26</td>
<td>Pixel API 26</td>
<td>1080 x 1920: xhdpi</td>
<td>26</td>
<td>Android 8.0 (Google APIs)</td>
<td>x86</td>
<td>2 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
<tr>
<td>Pixel XL API 26 invent</td>
<td>Pixel XL API 26 invent</td>
<td>1440 x 2560: xhdpi</td>
<td>26</td>
<td>Android 8.0 (Google APIs)</td>
<td>x86</td>
<td>1 GB</td>
<td><img src="download_icon" alt="Download" /></td>
</tr>
</tbody>
</table>
The Android Emulator

Pros

- Doesn’t require an actual phone
- Hardware is reconfigurable
- Changes are non-destructive
The Android Emulator

Cons

Slower than an actual device

Some features unavailable
  e.g., no support for Bluetooth or USB connections

Performance / user experience can be misleading
Advanced Features

Can emulate many different device/user characteristics, such as:

- Network speed/latencies
- Battery power
- Location coordinates
Advanced Features

Change network speeds
Advanced Features

Emulate incoming phone calls & SMS messages
Nougat is sweet!
Nougat is sweet!
The Android Emulator

Can interconnect multiple emulators
Advanced Features

Many more options

See:

Debugger

Tool for examining the internal state of a running application
The answer to life, the universe and everything is:

We may never know
public class TheAnswer extends Activity {

    private static final int[] answers = { 42, -10, 0, 100, 1000 };
    private static final int answer = 42;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Required call through to Activity.onCreate()
        super.onCreate(savedInstanceState);

        // Set up the application's user interface (content view)
        setContentView(R.layout.answer_layout);

        // Get a reference to a TextView in the content view
        TextView answerView = findViewById(R.id.answer_view);
    }
}
... int val = findAnswer();
String output = (val == answer) ? "42" : "We may never know";

// Set desired text in answerView TextView
answerView.setText(String.format("The answer to life,
   the universe and everything is:\n\n%s", output));

private int findAnswer() {
    for (int val : answers) {
        if (val != answer)
            return val;
    }
    return -1;
}
The answer to life, the universe and everything is:

42
Development Tools

Android Studio provides numerous tools for monitoring application behaviors.
Example Tools

Device File Explorer
Logcat
Traceview
Hierarchyview
Device File Explorer

View, copy, and delete files on your device

Often used to examine and verify file creation and transfer

Added in Android Studio 3.0
private int findAnswer() {
    int val = -1;
    for (int val : answers) {
        if (val == answer)
            return val;
    }
    return val;
}
public class TheAnswer extends Activity {

    private static final int[] answers = {42, -10, answer = 42;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Required call through to Activity.onCreate
        // Restore any saved instance state
        super.onCreate(savedInstanceState);

        // Set up the application's user interface
        setContentView(R.layout.answer_layout);

        // Get a reference to a Textview in the content
        TextView answerView = findViewById(R.id.answer);

        int val = findAnswer();
        String output = (val == answer) ? String.valueOf(val) :

        // Set desired text in answerView Textview
        answerView
            .setText(output);

    }

    private int findAnswer() {
        for (int val : answers) {
            if (val == answer)
                return val;
        } return -1;
    }
}
Logcat

Write and review log messages
Apps use Log class to write messages to log
Developer can search and filter log messages
```java
TheAnswer

private int findAnswer() {
    Log.d(TAG, msg: "Entering findAnswer()";
    for (int val : answers) {
        if (val == answer)
            return val;
    } // Set desired text in answerView
    answerView
        .setText(output);
}
```
setContentView(R.layout.answer_layout);

    // Get a reference to a TextView in the content view
    TextView answerView = findViewById(R.id.answer_view);

    int val = findAnswer();
    String output = (val == answer) ? String.valueOf(answer) : "We may never know";

    // Set desired text in answerView
    answerView
        .setText(output);

private int findAnswer() {
    Log.d(TAG, msg: "Entering findAnswer()");
    for (int val : answers) {
        if (val != answer)
            return val;
    }
    Log.e(TAG, msg: "Unexpected behavior");
    return -1;
}
Traceview

Logs execution sequences and timing taken from a running application

Graphically displays method traces and metrics
public class TheAnswer extends Activity {

    private static final int[] answers = {1, 2, 3, 4};
    private static final int[] answer = {1000};
    private static final String TAG = "TheAnswer";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Required call through to Activity.onCreate()
        super.onCreate(savedInstanceState);
        // Set up the application's user interface (content view)
        setContentView(R.layout.answer_layout);

        // Get a reference to a TextView in the content view
        TextView answerView = findViewById(R.id.answer_view);

        int val = findAnswer();
        String output = (val == answer) ? String.valueOf(answer) : "We may never know";

        // Set desired text in answerView
        answerView.setText(output);
    }

    private int findAnswer() {
        Log.d(TAG, msg: "Entering findAnswer()";
        for (int val : answers) {
            if (val != answer)
                return val;
        }
        return -1;
    }
}
UI Hierarchy Viewer

Shows the runtime organization of the user interface
public class TheAnswer extends AppCompatActivity {

    private static final int[] answers = {1000};
    private static final String TAG = "TheAnswer";

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

        // Get a reference to a TextView in the content view
        TextView answerView = findViewById(R.id.answer_view);
        int val = findAnswer();
        String output = (val == answer) ? String.valueOf(answer) : "We may never know";
        answerView.setText(output);
    }

    private int findAnswer() {
        Log.d(TAG, msg: "Entering findAnswer()");
        for (int val : answers) {
            if (val != answer)
                return val;
        }
        Log.e(TAG, msg: "Unexpected behavior");
        return -1;
    }
}
Next

Application Fundamentals