Android Debug Framework
(Logcat, DDMS, Debug, Exception Handling, Third Party APIs)
LogCat

• The LogCat view shows you the log messages of your Android device and helps you to analyze problems. For example Java exceptions in your program would be shown here.

To open this view, select (Alt-6 or Click Android Monitor)
You can see LogCat on Eclipse Bottom.

LogCat

LogCat window in Eclipse showing log messages.
DDMS
(DALVIK DEBUG MONITOR SERVER)
What is DDMS

Android provides a debugging tool called the Dalvik Debug Monitor Server (DDMS). We will cover:

1. What is the advantages of DDMS?
2. How DDMS works?

I) Advantages of DDMS:
   i) Screen capture on the device.
   ii) Thread and heap information on the device.
   iii) View Process
   iv) Radio state information
   v) Incoming call and SMS spoofing, location data spoofing, and more.
   vi) DDMS provides port-forwarding services.

II) How to open DDMS: To use DDMS open the perspective DDMS: go to Window->Open Perspective->Other, DDMS
Open DDMS
Now take a look into the elements of DDMS:
• **The Elements of DDMS:**

  • **Devices:** In the top left corner we can see a tab “Devices”. There we can get the list of devices and emulator which are on line (for more understanding see the below picture). Below each instance of the Emulator or device, we have all processes that are running in this instance.

  • From left to right we will see some icons:
    
    I) **Bug:** This shows you the debugger state. If it is green that means we are connected to the debugger and if it is red we are disconnected from the debugger.
    
    II) **Multiple–rows icon:** It is an “Update thread” button if we will click on this we will get the information about running thread in the “Thread” tab.
    
    III) **Stop icon:** To stop the selected process.
    
    IV) **Screen Capture icon:** By clicking on this we will get a picture of our device/emulator screen.

  • **Threads:** We can view the thread information for the selected process. To check thread information, go to Devices tab; select the process that you want to examine the threads for. Click the Update Threads button. And you will get thread information in Thread tab.

  • **Heap:** this will allows you to view how much heap memory a process is using. This information is useful in tracking heap usage at a certain point of time during the execution of your application.

  • **Go to Devices tab; Click on the update heap -&gt; Click on the Cause GC**
• You will get Heap information like this:
• **Allocation Tracker:** This allows you to track, in real time, where objects are being allocated when you perform certain actions in your application. This information is valuable for assessing memory usage that can affect application performance.

• **Go to Devices tab; select the process -> click the Start Tracking button-> it will begin allocation tracking ->click on the get allocation**
Screen capture From AVD

- Select Device and click screen capture button For capture screen shoot.
**File Explorer**

**File Explorer**: This will allow you to view, copy, and delete files on the device. This feature is useful in examining files that are created by your application or if you want to transfer files to and from the device.

In the Devices tab, select the emulator that you want to view the file system for. **Click the Pull file button**: To copy a file from the device, locate the file in the File Explorer.

**Click the Push file button**: To copy a file to the device.
File Explorer

You can copy and delete file.

Select file and click button for Copy file from AVD

Delete file from AVD
### File Explorer

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
<th>Permissions</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>acct</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>drwxr-xr-x</td>
<td></td>
</tr>
<tr>
<td>cache</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>drwxrwx-</td>
<td></td>
</tr>
<tr>
<td>config</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>dr-x-------</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>lrwxrwxrwx</td>
<td>-&gt; /sys/ker...</td>
</tr>
<tr>
<td>data</td>
<td></td>
<td>2013-05-31</td>
<td>15:47</td>
<td>drwxrwxrwx</td>
<td>-&gt; /system...</td>
</tr>
<tr>
<td>default.pro</td>
<td>116</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rw-r--r--</td>
<td></td>
</tr>
<tr>
<td>dev</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>drwxr-xr-x</td>
<td></td>
</tr>
<tr>
<td>etc</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>lrwxrwxrwx</td>
<td>-&gt; /system...</td>
</tr>
<tr>
<td>init</td>
<td>105204</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rwxr-</td>
<td></td>
</tr>
<tr>
<td>init.goldfis</td>
<td>2344</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rwxr-</td>
<td></td>
</tr>
<tr>
<td>init.rc</td>
<td>17048</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rwxr-</td>
<td></td>
</tr>
<tr>
<td>init.trace.rc</td>
<td>1637</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rwxr-</td>
<td></td>
</tr>
<tr>
<td>init.usb.rc</td>
<td>3915</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rwxr-</td>
<td></td>
</tr>
<tr>
<td>mnt</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>drwxrwrxr-x</td>
<td></td>
</tr>
<tr>
<td>proc</td>
<td></td>
<td>1970-01-01</td>
<td>05:30</td>
<td>dr-xr-xr-x</td>
<td></td>
</tr>
<tr>
<td>root</td>
<td></td>
<td>2011-12-09</td>
<td>04:36</td>
<td>drwx--------</td>
<td></td>
</tr>
<tr>
<td>sbin</td>
<td></td>
<td>1970-01-01</td>
<td>05:30</td>
<td>drwxr-x-</td>
<td></td>
</tr>
<tr>
<td>sdcard</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>lrwxrwxrwrx</td>
<td>-&gt; /mnt/s...</td>
</tr>
<tr>
<td>sys</td>
<td></td>
<td>1970-01-01</td>
<td>05:30</td>
<td>drwxr-xr-x</td>
<td></td>
</tr>
<tr>
<td>system</td>
<td></td>
<td>2012-07-14</td>
<td>02:30</td>
<td>drwxr-xr-x</td>
<td></td>
</tr>
<tr>
<td>uевentd.gc</td>
<td>272</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rw-r--r--</td>
<td></td>
</tr>
<tr>
<td>ueventd.rc</td>
<td>3879</td>
<td>1970-01-01</td>
<td>05:30</td>
<td>-rw-r--r--</td>
<td></td>
</tr>
<tr>
<td>vendor</td>
<td></td>
<td>2013-05-31</td>
<td>15:45</td>
<td>lrwxrwxrwrx</td>
<td>-&gt; /system...</td>
</tr>
</tbody>
</table>
Debugging with Android Studio
Android Studio enables you to debug apps running on the emulator or on an Android device. With Android Studio, you can:

1. Select a device to debug your app on.
2. View the system log.
3. Set breakpoints in your code.
4. Examine variables and evaluate expressions at run time.
5. Run the debugging tools from the Android SDK.
6. Capture screenshots and videos of your app.
Work with Breakpoints

• Breakpoints enable you to pause the execution of your app at a particular line of code, examine variables, evaluate expressions, and continue the execution line by line. Use breakpoints to determine the causes of run-time errors that you can't fix by looking at your code only.

To debug your app using breakpoints:

1. Open the source file in which you want to set a breakpoint.
2. Locate the line where you want to set a breakpoint and click on it.
3. Click on the yellow portion of the side bar to the left of this line, as shown in figure 5.
4. Start your app as described in Next Slide.
Set Breakpoints

```java
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener((view) -> {
    Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
});

EditText et1 = (EditText)findViewById(R.id.editText);
EditText et2 = (EditText)findViewById(R.id.editText2);
TextView tv1 = (TextView)findViewById(R.id.textView1);
TextView tv2 = (TextView)findViewById(R.id.textView2);
Button btn = (Button)findViewById(R.id.btn);

btn.setOnClickListener((view) -> {
    String first = et1.getText().toString();
    a = Integer.parseInt(first);

    String second = et2.getText().toString();
    b = Integer.parseInt(second);

    c = a + b;

    String answer = Integer.toString(c);
    Toast.makeText(getApplicationContext(), answer, Toast.LENGTH_LONG).show();
});
```
Run your App in Debug Mode

1. To debug your app in Android Studio:
2. Open your project in Android Studio.
3. Click **Debug** in the toolbar.
4. On the *Choose Device* window, select a hardware device from the list or choose a virtual device.
5. Click **OK**. Your app starts on the selected device.
Select Device to Debug your App
Debug Window

When the app run as Debug Mode then Debug Window will open as below:
View and Configure breakpoints

To view all the breakpoints and configure breakpoint settings, click View Breakpoints on the left side of the Debug tool window.
In this window you can see all your Breakpoints. You can also add or delete the Breakpoints.
Debug your app with breakpoints

• The *Debug* tool window lets you examine variables and control the execution step by step:

• To examine the object tree for a variable, expand it in the *Variables* view. If the *Variables* view is not visible, click *Restore Variables View*.

• To evaluate an expression at the current execution point, click *Evaluate Expression*.

• To advance to the next line in the code (without entering a method), click *Step Over*.

• To advance to the first line inside a method call, click *Step Into*.

• To advance to the next line outside the current method, click *Step Out*.

• To continue running the app normally, click *Resume Program*.
Variables

To examine the object tree for a variable, expand it in the Variables view. If the Variables view is not visible, click Restore Variables View.
Step Into.

To advance to the first line inside a method call, click **Step Into**.
Step Over

To advance to the next line in the code (without entering a method), click Step Over.
Step Out

To advance to the next line outside the current method, click Step Out.
Track Object Allocation

Android Studio lets you track objects that are being allocated on the Java heap and see which classes and threads are allocating these objects. This allows you to see the list of objects allocated during a period of interest.

This information is valuable for assessing memory usage that can affect application performance. To track memory allocation of objects:

Start your app as described in Run Your App in Debug Mode.

1. Click **Android** to open the Android DDMS tool window.
2. On the Android DDMS tool window, select the **Devices | logcat tab**.
3. Select your device from the dropdown list.
4. Select your app by its package name from the list of running apps.
5. Click **Start Allocation Tracking**
6. Interact with your app on the device.
7. Click **Stop Allocation Tracking**
LogCat Window

<table>
<thead>
<tr>
<th>Time</th>
<th>PID</th>
<th>TID</th>
<th>Application</th>
<th>Tag</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 150 high_thd 195</td>
</tr>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 151 high_thd 195</td>
</tr>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 149 high_thd 195</td>
</tr>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 150 high_thd 195</td>
</tr>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 150 high_thd 195</td>
</tr>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 150 high_thd 195</td>
</tr>
<tr>
<td>E</td>
<td>355</td>
<td>355</td>
<td></td>
<td>PSD</td>
<td>set_ps_threshold: low_thd 150 high_thd 195</td>
</tr>
<tr>
<td>D</td>
<td>8132</td>
<td>8132</td>
<td></td>
<td>PowerSaveS...</td>
<td>PowerSaveReceiver EXTRA_STATUS: 0</td>
</tr>
<tr>
<td>V</td>
<td>2750</td>
<td>2750</td>
<td></td>
<td>ExReceiver</td>
<td>receive action=android.intent.action.VIBRATE</td>
</tr>
<tr>
<td>D</td>
<td>2750</td>
<td>2750</td>
<td></td>
<td>ExReceiver</td>
<td>doOnReceive action=android.intent.action.VIBRATE</td>
</tr>
</tbody>
</table>
Capture Screenshots

To take a screenshot of your app:

1. Start your app as described in Run your App in Debug Mode.
2. Click Android to open the Android DDMS tool window.
3. Click Screen Capture on the left side of the Android DDMS tool window.
4. Optional: To add a device frame around your screenshot, enable the Frame screenshot option.
5. Click Save.
Exception(java.lang.Exception)

• When exceptions are thrown in the application, it must be caught by the application code
• To be able to catch any exception that isn’t part of a try-catch-clause, you can use the `Thread.setDefaultUncaughtExceptionHandler(Thread.UncaughtExceptionHandler)` method
• In the handler you can then perform whatever action you want. Don’t forget to call “`System.exit()`” though, because otherwise this won’t work!
public class SomeActivity extends Activity {
    private static final String LOG_TAG = "SomeActivity";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler()
        {
            @Override
            public void uncaughtException(Thread paramThread, Throwable paramThrowable)
            {
                Log.e(LOG_TAG, "OMG! Uncaught Exception!");
                // Without calling System.exit() this will not work.
                System.exit(2);
            }
        });
    }
}
Crash Reporting Tools

• There are many crash reporting tools for mobile development. A list:
  http://www.folio3.com/blog/crash-reporting-tools-ios-android/

1) crashlytics:
   https://try.crashlytics.com/

2) Crittercism
   https://app.crittercism.com/developers/login

3) New relic