CET325

Advanced Mobile Development

Lecture 4A

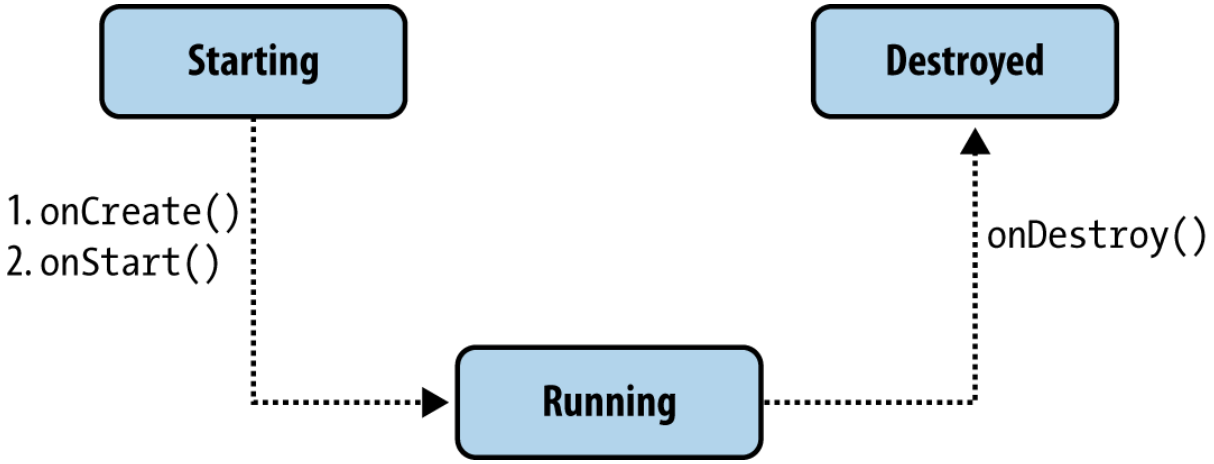
Agenda

- Application components
- Intents
- Event Listeners

Application Components

- Types of core component:
  - **Activities** (represents a single screen in an app)
  - **Services** (performs operations in the background without UI)
  - **Content Providers** (manages a shared set of app data)
  - **Broadcast Receivers** (responds to system-wide broadcast announcements)
- Also of interest:
  - **Intents** (Messages sent among the building blocks)
- Components are usually declared in the application manifest.

Service



- A service runs in the background to perform long term operations
  - Do not provide a UI
  - Is not bound to a particular activity
- Example:
  - Background music
  - Fetch data over a network
  - File I/O
- Created as a subclass of the android Service class

Service

- Provides two features
  - Tells the system that we want to do something in the background
    startService (Intent service)
    - Even if the app closes!
  - The ability to expose functionality to other apps (*bindService())*
- Service is NOT a separate process or thread
- Service is a simple class, you must implement separate threads by yourself.

Service

- Service requests added to ‘manifest file’ at design / development.
  - Device responds to requests at run-time, either granting or denying permission.

- You will remember that the manifest file is the default location for declaration of components and permissions.

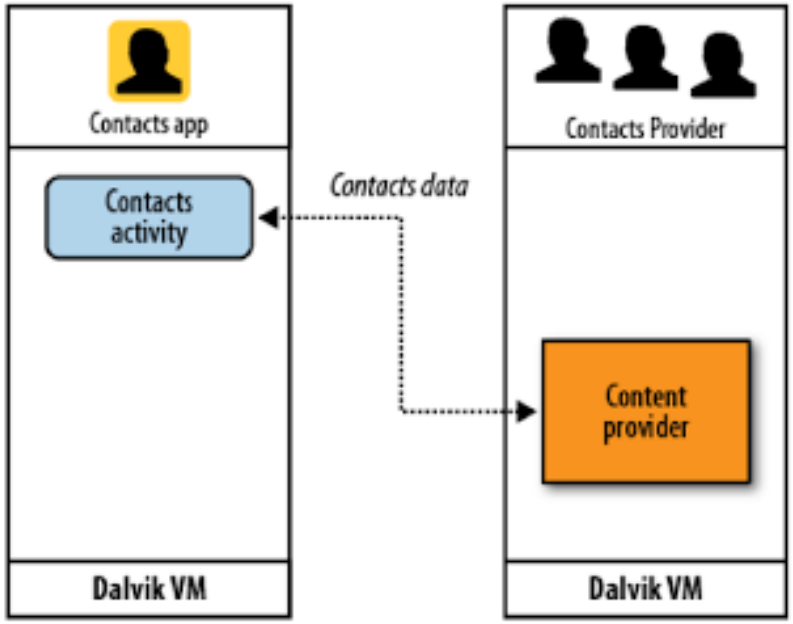
Content Provider

- A content provider makes a specific set of the application's data available to other apps.
  - Share data to other apps
  - Manage persistent data, or external sources of data.
  - Any app with appropriate permission can read / write the data.
- Adheres to CRUD principle

Operation	Method
Create	insert()
Read	query()
Update	update()
Delete	delete()

Content Provider

- Many native databases are available via the content providers, for example Contact Manager
- On-board SQLite database management system to provide organised persistent data storage.
- Common interface for querying the data



Content Provider

- The result of the query: simple table in Query object

- Content Provider exposes a public URI that uniquely identifies its dataset
  - URIs begin with content://
  - Android provides constants for native content providers, for example
    - ContactsContract.Contacts.CONTENT_URI

Broadcast Receiver

- A broadcast receiver is a component that does nothing but receive and react to broadcast announcements
  - Allow you to respond to system conditions, for example low battery or the screen being turned off.
- You can
  - Receive and react to system services
  - Receive and react to other apps broadcast announcements
  - Initiate broadcasts to other apps.
- App is given fixed time to react to a broadcast (10 seconds)
- Implemented as a subclass of *BroadcastReceiver*.

Intents

- Messages sent among the building blocks.
  - Trigger another activity
  - Tell a service to start up
  - Initiate a broadcast
- They allow interaction between any Android component.
- Intents are asynchronous
  - The code that started them doesn't have to wait for them to finish.
- *Explicit (specifies the component on the receiving end)*
- *Implicit (specifies the type of receiver)*

Intents and Intent Filters

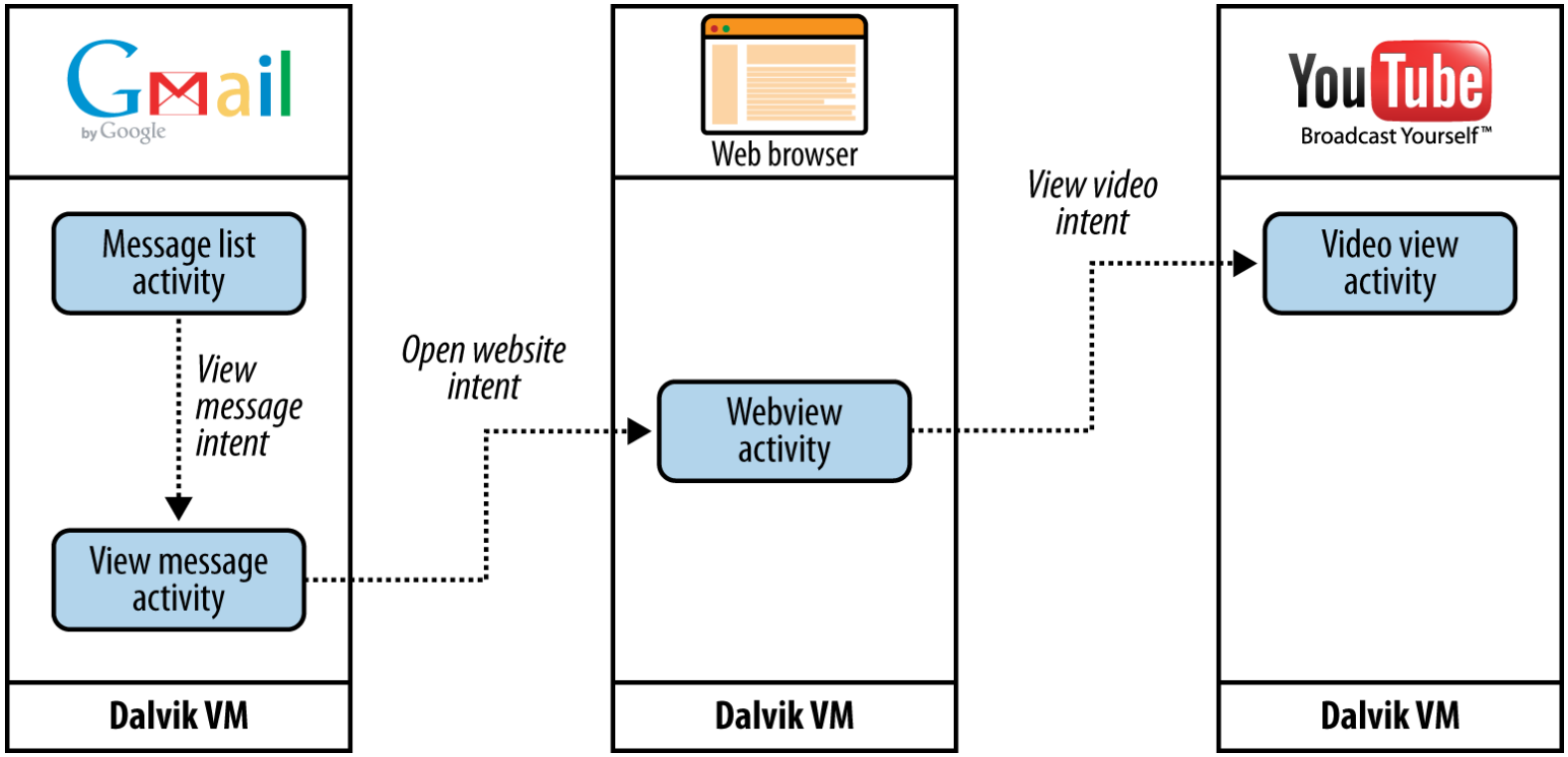
- Intent
  - A bundle of information describing a desired action
    - The data to be acted on
    - The category of component that should perform the action.
- Android locates an appropriate component to respond to the intent, launches a new instance of the component if one is needed, and passes it the intent object.
- Intent Filter
  - Components advertise their capabilities, i.e. the type of intents they can respond to.
  - Describes who can handle the message
  - Specified in the manifest as <intent-filter> elements

Explicit Versus Implicit Intents

- Explicit
  - Open explicitly certain activity
  - Internal messaging between your app
  - Designated target class.

- Implicit
  - External messaging between apps
  - Open some Activity which offers a certain service
    - You don’t know which activity of which app
  - Android will at run time resolve the best class suited to performing the action.
    - Eg ACTION_DIAL, ACTION_VIEW

Intents



Application Context

- You will make reference to this when developing applications.
- Refers to application environment
  - Allows sharing of data and resources between the building blocks *(activities, services, content providers, broadcast receivers, data, files,…)*
  - Has it's own Linux user ID, Linux process, and VM
  - Uniquely identified based on the application package name
  - Created when first component of the app starts up.

Event Listeners

- Tutorial
  - Create UI component in the layout file
  - Within the activity, create a variable to represent that component

```
TextView myTextView=(TextView) findViewById(R.id.textView);
```

- The next step was to implement logic which would be executed when the user interacted with that component ➔ Android Event Handling
- How did we do this?

Event Listeners

- Events represent a response to user's interaction with input controls, such as press of a button or touch on a screen.
- In order to do this we implemented **onClickListener** interface and **registered** a listener to our button.

myButton.setOnClickListener(this)

- Android framework places occurring events into a queue, which is based on first-in first-out logic.
- When an event happens, the registered event listener should implement a corresponding callback method (event handler) to handle the event.

Event Listeners

- Some of the most used Event Handlers with the respective Event Listeners are:
  - onClick() – onClickListener(): for clicking or focusing upon a view component
  - onLongClick() – onLongClickListener(): for clicking or holding or focusing upon a view component, for more than one second.
  - onTouch() – onTouchListener(): performing a touch action, including motion gesture on the screen
  - onKey() – onKeyListener(): presses or releases a hardware key on the device.
  - onFocusChange() – onFocusChangeListener(): for navigating onto or away from an item

Event Listeners

- There are different ways of registering listeners, each with there own pros and cons:
  - Implement the Event Listener interface in the Activity class
  - Use an anonymous inner Event Listener class
  - Declare it in the layout XML file.

Event Listeners

- **Implement Event Listener interface in the Activity class.**
- Recommended for a single control of that listener.
- It has poor scale for multiple controls, because the listener cannot take arguments.
- A large range of multiple controls requires further programming, in order to check which control fires the event.

```
public class MainActivity extends Activity
                        implements View.OnClickListener{

…

button = (Button) findViewById(R.id.button);
button.setOnClickListener(this);

@override
public void onClick(View view) {
    int id = view.getId();

    if(id == R.id.button){
    …
    }
}

}
```

Event Listeners

- **Use an anonymous inner Event Listener Class**
- Used for a single control.
- Works poorly for multiple controls with long code in the Event Handler callback, because it can be difficult to maintain the code.

Event Listeners

```
// Create an anonymous implementation of OnClickListener
private OnClickListener mListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.button);
    // Register the onClick listener with the
    // implementation above
    button.setOnClickListener(mListener);
    ...
```

Event Listeners

- **Declare it in the layout XML file**
- Doesn't require an implementation of Event Listener interface.
- Define the Event Handler in the layout of the Activity, specifically into the android:onClick attribute of the View component.
- After that, simply implement the respective function into the Activity.
- Simple and Flexible – different methods for multiple controls.
- Disadvantages:
  - Isn't always clear for developers which handler corresponds to which control
  - The handler method only takes view as an argument and returns void. This can't be changed.

Event Listeners

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/copy"
    android:id="@+id/button"
    android:onClick="onClick"/>


public void onClick(View view) {
    int id = view.getId();

    if(id == R.id.button){
   // implement logic
   }
```

Recap

- You have created single screen applications:
  - Create widgets
  - Link them to event listeners
  - Handle user interactions
- The next step is to add a variety of widgets.
- Then we can look at linking multiple Activities together.

Key Android Classes

- Activity
- View
- Intent
- Service
- BroadcastReceiver
- Context
- OnClickListener