CPET 499/565 Mobile Computing Systems

Lecture

Oct. 24, 2012

Android Application Framework

- Provided in android.jar file
- Android SDK is made up of the following packages

Top-Level Package	Purpose
android.*	Android application fundamentals
dalvik.*	Dalvik Virtual Machine support classes
java.*	Core classes and generic utilities for networking, security, math, etc
javax.*	Java extension classes: encryption support, parsers, SQL, etc
junit.*	Unit testing support
org.apache.http.*	HTTP protocol
org.json	JavaScript Object Notation (JSON) support
org.w3c.dom	W3C Java bindings for the Document Object Model core (XML and HTML)
org.xml.sax.*	Simple API for XML (SAX) support for XML
org.xmlpull.*	High-performance XML parsing

- Google APIs Add-On - an extension to the Android SDK, https://developers.google.com/android/add-ons/google-apis/
  - The Maps external library
  - The USB Open Access Library
  - A sample Android application called MapsDemo
  - Full Maps library documentation
- Android documentation references, http://developer.android.com/index.html

Android Applications

- Android Applications are programs, with functionalities and data, for performing tasks, display information to the screen, and act upon data from a variety of sources

Application Fundamentals, http://developer.android.com/guide/components/fundamentals.html

Android Terminology

- **Context**, http://developer.android.com/reference/android/content/Context.html
  - **public abstract class Context extends Object**
  - It allows access to application-specific resources and classes
  - The central command center for an Android application-level operations such as
    - Launching activities
    - Broadcasting Intents
    - Receiving Intents
  - All application specific functionality can be accessed through the Context
- **Activity**, http://developer.android.com/reference/android/app/Activity.html
  - **public class Activity extends ContextThemeWrapper implements ComponentCallBacks** …
  - An activity represents a single screen with a user interface
  - An android application is a collection of tasks, each of which is called an Activity

- **Intent**, http://developer.android.com/reference/android/content/Intent.html
  - **public class Intent extends Object implements Parcelable Cloneable**
  - An abstract description of an operation to be performed.
  - An Intent is recognized as a request to do something with late runtime binding between the code in different applications.
  - The Android OS uses an asynchronous messaging mechanism to match task requests with the appropriate Activity
- **Service**, http://developer.android.com/guide/components/services.html
  - An application component for performing long-running, background operations that do not provide a user interface.
  - Tasks that do not require user interaction can be encapsulated in a service.
  - Most useful when the operations are lengthy (offloading time consuming processing) or need to be done regularly (such as checking a server for new mail)

Using the Application Context

- **Context class**, http://developer.android.com/reference/android/content/Context.html
- The application Context is the central location for all top-level application functionalities.
  - Retrieving the Application Context
  - Retrieving the Application Resources
  - Accessing Application Preferences
  - Accessing other Application Functionalities
    - Launch **Activity** instances
    - Retrieve assets packaged with the application
    - Request a system service (for example: a location service)
    - Manage private application files, directories, and databases
    - Inspect and enforce application permission
- public abstract class Context extends Object
  - Inherited Methods from class: java.lang.Object
  - Constants
  - Public constructors – Context()
  - Public Methods
    - **getApplicationContext()** method – retrieving the Application Context
    - **getResources()** method – retrieving Application Resources
    - **getSharedPreferences()** method – retrieve Application Preferences
    - ... etc

Performing Application Tasks with Activities

- **Activity class**, http://developer.android.com/reference/android/app/Activity.html
- An Example – A simple game application might have the following 5 Activities
  - Startup/Splash Activity
    - Main Menu Activity
      - Game Play Activity
      - High Score Activity
      - Help/About Activity
- Lifecycle of an Android Activity
- More Examples

- Using Activity Callbakcs to manage application state and resources
- Initializing static Activity data in onCreate()
- Initializing and retrieving Activity data in onResume()
- Stopping, saving, and releasing Activity data in onPause()
- Avoiding Activity objects being Killed
  - Under low-memory operation, OS can kill the process for any Activity that has been paused, stopped, or destroyed.
- Saving Activity state into a bundle with onSaveInstanceState()
- Destroy static Activity data in onDestroy()

Using Activity callbacks to manage Application state and resources

```
public class MyActivity extends Activity{
    protected void onCreate(Bundle savedInstanceState);
        // Initialize static Activity data
    protected void onStart();
    protected void onRestart();
        // Bring activity to Foreground
    protected void onResume();
        //Bring activity to Foreground
        // Appropriate place for placing/starting Audio, Video, and Animators
    protected void onPause();
        // Pushed down the current Activity to the Activity Stack
        // Should stop any Audio, Video, and Animators
        //Deactivate resources such as a database Cursor object
        //Last chance for clean-up any resources it does not needed while in the background
        //Need to save any uncommitted data here, in case the application does not resume
    protected void onStop();
    protected voidonDestroy();
    }
```

Managing Activity Transitions with Intents

- **public class Intent**, http://developer.android.com/reference/android/content/Intent.html
- Can be used with startActivity() to launch an Activity, and appropriate finish() methods
- Examples
  - sendBroadcast(Intent intent) to send it to any interested BroadcastReceiver components
  - startService(Intent) or bindServiceIIntent, ServiceConnection, int) to communicate with a background Service
- Other Examples
  - Transitioning between Activities with Intents
  - Launching a new Activity by class name
  - Creating Intents with action and data (action/data pair)
  - Launching an Activity belonging to another application
    - Customer Relationship Management (CRM) app
  - Passing additional information using Intents
  - Organizing Activities and Intents in an application using Menus

Launching an Activity belonging to another application

- Customer Relationship Management (CRM) launch the **Contacts** application
  - to browse the **Contact database**
  - Choose a **specific contact**
  - Return that contact's **unique ID**
- Launch Phone_Dialer app with a specific number
  Uri number = Uri.parse(tel:2604816339);
  Intent dial = new Intent(Intent.ACTION_DIAL, number)
  startActivity(dial);

Intents List: Invoking Google applications on Android Devices (Target Application/Intent URI), http://developer.android.com/guide/appendix/g-app-intents.html

- Browser (view, web search)
- Dialer (call)
- Google Maps (view)
- Google Streetview
- etc

Working with Services

- Services, http://developer.android.com/guide/components/services.html
- An application component that can perform long-running operations in the background and does not provide a user interface
- Examples (Background processing/tasks)
  - Handle network transactions
  - Play music
  - Perform file I/O
  - Interact with content provider
- Launching two forms of services
  - Started:
    - A service is "started" when an application component (such as an activity) starts it by calling **startService()** method, http://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)
  - Bound:
    - A service is "bound" when an application component binds to it by calling **bindService()**
- Other Examples of service implementations
  - Routinely check updates: weather, email, or social network app
  - A photo or Media app that keeps its data in SYNC online (package and upload new content in the background when the service is idle)
  - A video-editing app might offload heavy processing to a queue on its service (to avoid affecting overall system performance for non-essential tasks)

Android Programming Exercises

- **1st Programming Exercise**, Hello World, http://developer.android.com/training/basics/firstapp/index.html
- **2nd Programming Exercise**, an Activity (a single screen with a text field and a button), http://developer.android.com/training/basics/firstapp/building-ui.html
- **3rd Programming Exercise,** Starting Another Activity, http://developer.android.com/training/basics/firstapp/starting-activity.html
- **4th Programming Exercise,** Managing the Activity Lifecycle, (download the activity demo), http://developer.android.com/training/basics/activity-lifecycle/index.html
    - Starting an Activity, http://developer.android.com/training/basics/activity-lifecycle/starting.html
    - Pausing and Resuming an Activity, http://developer.android.com/training/basics/activity-lifecycle/pausing.html
    - Stopping and Restarting an Activity, http://developer.android.com/training/basics/activity-lifecycle/stopping.html
    - Recreating an Activity, http://developer.android.com/training/basics/activity-lifecycle/recreating.html

References

[1] Android documentation references, http://developer.android.com/index.html

[2] Lauren Darcey and Shane Conder, Android Wireless Application Development, 2nd Edition, Addison Wesley, 2011

[3] Reto Meier, Professional Android 4 Application Development, 2012, John Wiley & Sons, Inc