Developing for Android Wear: Part 2 (Wear-Generated Notifications and Wearable Apps)

Justin Munger
GDG Chicago
August 12, 2014

What, No Data Layer API? :-O

- Ended up being too large of a topic to fit in scope of this talk
- Happy to cover it in a future meeting if there is sufficient interest
- Added in section on Wear-Generated Notifications to make up for missing Data Layer API content
- Quick demo of Data Layer API at end of talk

What is Android Wear?

- Wearable computer in a watch form factor
- Runs Android OS 4.4W (API Level 20)
- Pairs to phone/tablet through Bluetooth
- Uses Android Wear app and Google Play Services to sync between phone/tablet and Wear device
- Current devices available (as of July 2014) are LG G Watch and Samsung Gear Live

Why Android Wear?

- Removes the distractions caused by the immersive nature of mobile devices
- Delivers only the essential information needed in a timely manner
- Allows partial interaction with phones/tablets to perform certain necessary actions

Android Wear Design

Suggest - The Context Stream

- Vertical list of cards
- Cards display contextually-relevant (time- and/or location-based) information
- One card displayed at a time
- Optional background photos give additional context
- Cards pushed to user by Android Wear device and/or paired phone/tablet

Demand - The Cue Card

- Allows user to interact with Wear device through voice or tapping suggested options
- Contains pre-defined voice commands that applications can hook into
- Also allows for custom voice commands to start functionality
- Similar to how a person interacts with Google Glass and even phones/tablets

What Does This Presentation Cover?

- Wear-generated notifications
- Android Wear apps

Wear-Generated Notifications

Wear-Generated Notifications

- Generated on the Android Wear device
- Used for contextual notifications
- Can use current user context (time/location/sensor input) to trigger notifications to user
- Allows for more complex notification interaction (such as taking action directly on a card)

How to Get Notifications on Wear

- Create notifications the same way that you would for phones and tablets
- Use Notification and NotificationManager to create and post notifications
- Doesn’t require Compat classes if notification is only displayed on Wear device
- Wear-specific functionality is added using Notification.WearableExtender()

Creating a Notification on Wear

- Get an instance of NotificationManager
- Create a Notification using the Notification.Builder methods
- Add Wear-specific functionality using NotificationManager.WearableExtender
- Call notify() on NotificationManager, passing in unique notification ID and Notification object

Basic Notification

- Contains the basic elements that typical notifications have:
  - Small icon
  - Content title
  - Context text
- Title and text are left aligned on the card

Content Action Notification

- Allows action to be taken directly from the notification
- Specify a PendingIntent that contains the Activity to launch when tapping notification
- Add it to a Notification using addAction() in Notification.WearableExtender()
- Indicate tappable notification action by setting setContentAction() with Action index

Display Intent Notification

- Custom UI for a notification
- Use an Activity to host the notification layout
- Activity must be defined in AndroidManifest.xml with specific settings
- Use the WearableExtender to add the custom UI using setDisplayIntent() method
- Activity is displayed when notification is moved out of peek state

Custom Size Notification

- Create notifications of five different sizes from extra-small to full-screen
- Uses an Activity to host the notification layout
- Use the WearableExtender to add the custom UI using setDisplayIntent() method
- Set custom size using . setCustomSizePreset()

Wearable Apps

It’s Android But It’s Not Android

- Android Wear devices has access to most of the APIs that regular Android devices have
- However, some functionality is not optimized for Wear devices, mainly UI widgets
- Other functionality is not available on Wear devices, such as direct network access

What Kind of UI Do We Use Then?

- Unofficial Google library for Android Wear UI
- UI made for small form-factor and to support square/round displays
- Add the following dependency in the project's build.gradle file:

```
dependencies {
    compile 'com.google.android.support:wearable:+'
}
```

Examples of Wear UI Components

- Wear-specific UI
  - WearableListView
  - GridViewPager
  - CardFragment
- Support UI
  - DismissOverlayView
  - ConfirmationActivity
- Square/Round Helper UI
  - WatchViewStub
  - BoxInsetLayout
  - InsetActivity

Wear-specific UI

WearableListView

- Wear-specific ListView implementation
- Allows for vertical scrolling of items
- Items lock into place to facilitate easier scrolling on a small screen
- Subclasses RecyclerView
- Uses Material Design concepts to display content

WearableListView.Adapter

- Responsible for creating and delivering content to WearableListView
- Subclasses RecyclerView.Adapter
- Override onCreateViewHolder() to deliver inflated layout
- Override onBindViewHolder() to populate inflated layout with content

GridViewPager

- Allows 2D scrolling of content
- Displays content in rows and columns
- GridViewPager is non-symmetric
- Each row can have different column count
- Scrolling up and down between rows moves back to the first column of the row
- Background images have fade transitions and parallax effects when scrolling

GridViewAdapter

- Supplies GridViewPager with custom View
- Overrides:
  - getRowCount() - number of rows
  - getColumnCount() - number of columns
  - instantiateItem() - create custom View and add to ViewGroup
  - destroyItem() - destroy View
  - getBackground() - supply background image using ImageReference

FragmentGridViewAdapter

- Supplies GridViewPager with Fragment-based UI to display
- Can use with CardFragment
- Overrides:
  - getRowCount() - number of rows
  - getColumnCount() - number of columns
  - getFragment() - create Fragment-based UI and add to ViewGroup
  - getBackground() - supply background image using

CardFragment

- Fragment-based UI that renders a card
- Instantiated with static method, specifying title, text, and icon
- Can specify custom layout by subclassing and overriding onCreateContentView()

Support UI

DismissOverlayView

- Gives an dismissal option for a full-screen Wear app when a right swipe can’t be used
- Long-pressing the UI brings up an X in a red circle
- Tapping the X dismisses the activity

Setting up DismissOverlayView

- Create a style that sets android: windowSwipeToDismiss=false
- Set theme of the hosting Activity to the style
- Add DismissOverlayView as the last view in top-level view hierarchy
- Get reference to DismissOverlayView in Activity

Setting up DismissOverlayView

- Create GestureDetector and create new SimpleOnGestureListener, overriding onLongPress()
- Show DismissOverlayView in onLongPress()
- Override onTouchEvent of Activity and pass event to GestureDetector
- WHEW! That’s a lot of work! :-O There must be an easier way

ConfirmationActivity

- Display an animation to confirm an action
- Show Success, Open On, and Failure animations
- Can customize text underneath animations
- Launched from an activity rather than subclassed

Setting Up ConfirmationActivity

- Add the activity to the application manifest
- Create an intent, explicitly specifying ConfirmationActivity
- Add intent extras for the animation type and animation text
- Start the intent

Square/Round Helper UI

WatchViewStub

- Use one activity layout to specify distinct rectangular and round layouts
- Widget will determine the proper layout to load based on the form factor
- Retrieve references to UI elements by implementing WatchViewStub. OnLayoutInflatedListener() and overriding onLayoutInflated()

BoxInsetLayout

- Automatically determines form factor and adjusts UI accordingly
- Specified in XML layout as root element
- No additional code required to use aside from XML layout element

InsetActivity

- Combines BoxInsetLayout and DismissOverlayView into one component
- No extra XML layout elements required
- Move normal layout inflation and UI initialization from onCreate() to onReadyForContent()
- Sometimes works, sometimes doesn’t

Questions?

Additional Resources

- Android Wear Training: http://developer.android.com/training/building-wearables.html
- Google I/O Dev Bytes Wearables Videos: http://goo.gl/yqWsq4
- Demo code: http://github.com/justinkmunger/DevelopingForAndroidWear

Thank You!