Storli Designs >

WalmartProductWheelClassOverview

Walmart Product Wheel

Click here to go to the

[Main Storli Designs Site](https://sites.google.com/site/storlidesignsllc/)

https://sites.google.com/site/storlidesignsllc/

Click here to download the

WalmartProductWheel App

https://drive.google.com/file/d/10Q0RtaAXdXGye-4ielnskH34xZoLtW_p/view

Click here to go to github to download the

WalmartProductWheel Android Studio Project

https://github.com/pstorli/WalmartProductWheel

1) The Walmart Product Wheel app

I would like to outline my process of creating great software for you right now!

The purpose of the **Walmart Product Wheel** app is to displays lists of products and lists of scrolling product details.

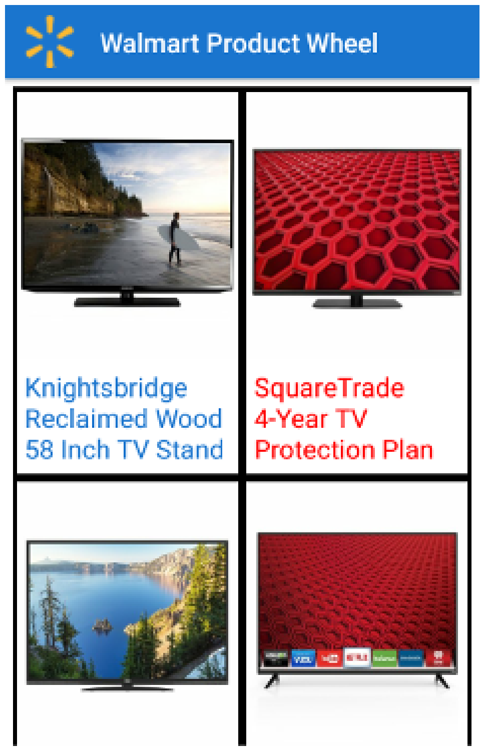
But it also helped me strengthen my Kotlin, layout, rest and json abilities.

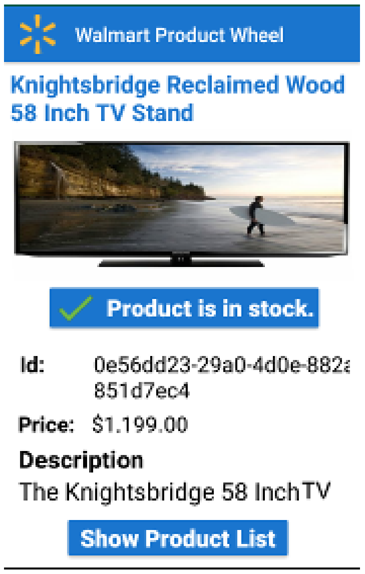
And it can also help other developers learning Kotlin, since it is free, open source and provides examples of certain useful Kotlin programming techniques.

2) How it was designed (**Project Structure and Classes too)**

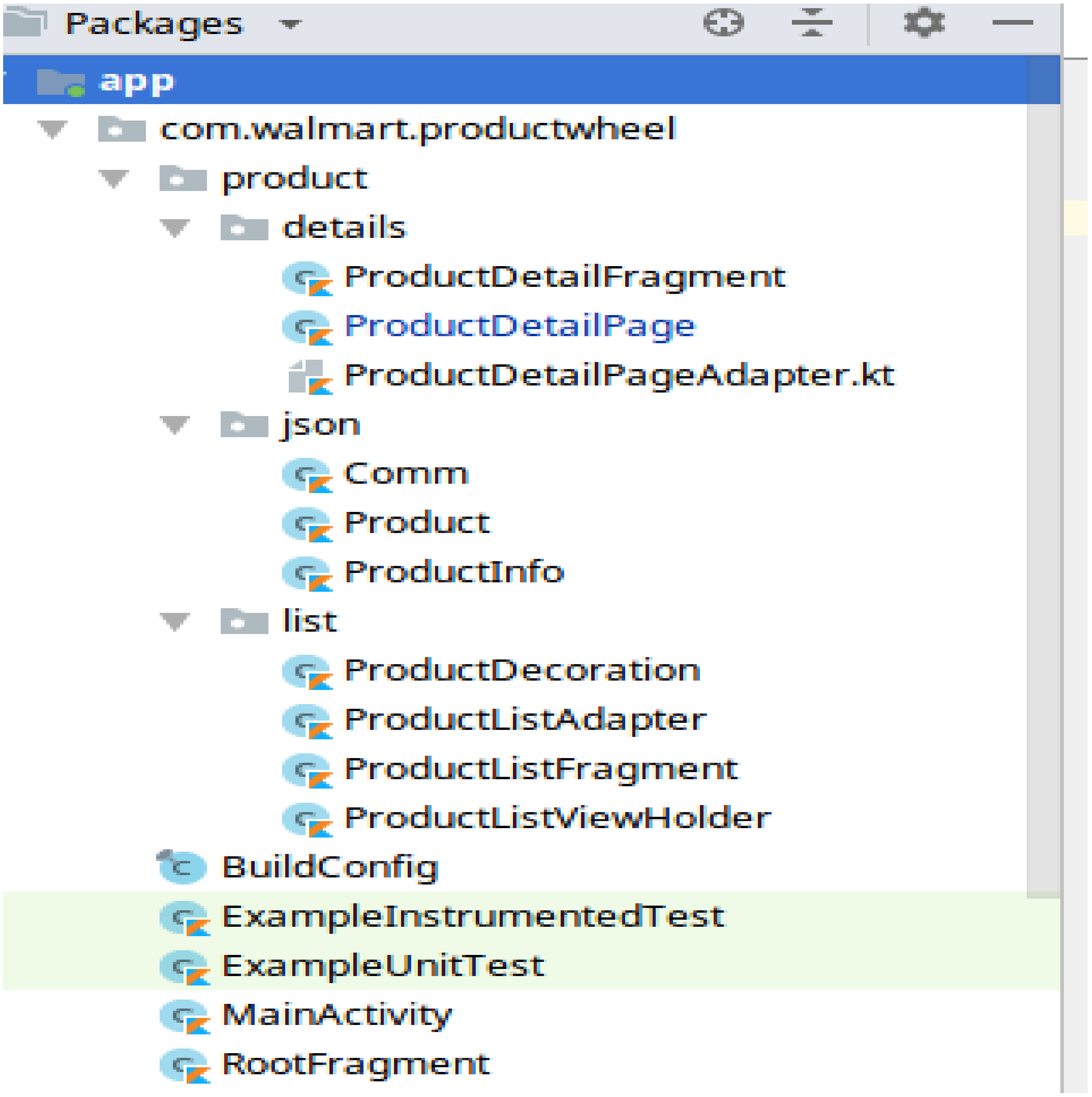
The structure of this app grew out of the requirements presented by Walmart.
“***List of all the products returned by a service call in a Lazy Loading List.***”
.

Overview of Project Structure and Classes

 Walmart Product Wheel Knightsbridge Reclaimed Wood 58 Inch TV Stand SquareTrade 4-Year TV Protection Plan	**Screen 1** So, the **MainActivity,** uses one fragment to “***Display a list that products that lazy load as you scroll.*****” The list uses Android's RecyclerView**
	Screen 2 **The other screen, “*Should display details of the product and be able to swipe to next/previous items on this screen.*”**. The product details fragment allows for this.The

 Walmart Product Wheel Knightsbridge Reclaimed Wood 58 Inch TV Stand Product is in stock. Id: 0e56dd23-29a0-4d0e-882a 851d7ec4 Price: $1,199.00 Description The Knightsbridge 58 Inch TV Show Product List	product details screen uses Android's **ViewPager**

The main package is ***com.product.productwheel***



```
app
  com.walmart.productwheel
    product
      details
        ProductDetailFragment
        ProductDetailPage
        ProductDetailPageAdapter.kt
      json
        Comm
        Product
        ProductInfo
      list
        ProductDecoration
        ProductListAdapter
        ProductListFragment
        ProductListViewHolder
    BuildConfig
    ExampleInstrumentedTest
    ExampleUnitTest
    MainActivity
    RootFragment
```

Based on the requirements there needed to be a list of products then another swipeable list of product details. We also needed a way to go back and forth between list and detail screens.

To help organize things, ***list*** and ***detail*** packages were created.

The ***layout\activity_main.xml*** screen would hold a container

```
LinearLayout android:id="@+id/container"
```

```
That would be used to switch in and out the product list and produc
detail screens.
```

layout\product_list_fragment.xml and layout\ product_detail_fragment.xml

A ***json*** package was also created to hold classes that help with remote reading of json data from rest calls as well as the Kotlin classes needed to help transform that to/from Kotlin/json.

The classes used are documented below.

Support Classes

com.product.productwheel.json

Comm

The communication class uses ***OkHttpClient*** to access the URL **"https://mobile-tha-server.firebaseapp.com"**
and retrieves a json string of product info ***val products = URL(url).readText()***
then the ***getProductInfo*** method converts the json string into an object of ***ProductInfo*** type by calling ***ObjectMapper().readValue(jsonProductInfo, ProductInfo::class.java)***

Then the product list adapter is notifiyed that more products have been retrieved and need to be displayed ***productListAdapter.notifyDataSetChanged()***

The Comm class also has a method to fetch an image for the indicated product
loadImage (product: Product, productImage: ImageView)
using ***Picasso.get().load(url).into(productImage)***

Product

The product class is used to transfer product related json data to / from kotlin

ProductInfo

The productInfo class is used to transfer json data to / from kotlin
It keeps track of the data's status, page number, page size, total products and an ***Array<Product>?***
to actually hold each product.

com.product.productwheel

BuildConfig

The BuildConfig class has useful constants that you can access syc as:
DEBUG,APPLICATION_ID,BUILD_TYP,FLAVOR,VERSION_CODE and ***VERSION_NAME***

ExampleInstrumentedTest

This autogenerated class is used to help with unit testing. Put your test cases here.

ExampleUnitTest

This autogenerated class is used to help with unit testing. Put your test cases here.

MainActivity

This class keeps the list of products being shown, which have been parsed and downloaded.
This class also has some helper functions to **showFragment**, **addProduct**, **addProducts** or **showSnackbar**

It all starts in its **onCreate** method where the walmart splash icon is added to the title by calling **getSupportActionBar.setIcon.** It then calls **showFragment**, a custom method, that uses the supportFragmentManager to beginTransaction to replace the current fragment on the main screen with the desired new fragment.
See R.id.container in layout/activity_min.xml

In this case the new Fragment set in onCreate is **ProductListFragment**

RootFragment

This class extends Fragment and overrides methods so subclasses don't need to:
onPause(), onDestroy(), onDetach(), onStart() , onStop()

Screen 1 Classes

com.product.productwheel.list

ProductDecoration

Keeps track of the padding between products in the product list.

ProductListAdapter

In method fun onCreateViewHolder the layout
R.layout.*product_list_item is inflated and the* ProductListViewHolder is created.

ProductListFragment

This fragment is used by MainActivity.onCreate to show a list of products (product name and image).
It is responsible for the ***RecyclerView***, the ***GridLayoutManager***, the ***ProductListAdapter*** and the ***recyclerViewOnScrollListener,***
which handles the user scrolling.

```
recyclerView : RecyclerView

lateinit var layoutManager                    : GridLayoutManager
lateinit var productListAdapter                 : ProductListAdapter
val         recyclerViewOnScrollListener = object : RecyclerView.OnScrollListener
```

When the user scrolls and more items are needed, go to the next page by

1. Incrementing the ***MainActivity page count*** and
2. Load products for next page by calling ***MainActivity loadProducts*** (passing in the productL

In method ***onViewCreated*** we

1. Fetch the recycler from ***R.id.productListRecycler***
2. Create layout from ***R.layout.product_list_fragment.xml***
3. Assign layoutManager
4. Add a decorator to enforce the padding between items
5. Access the ***RecyclerView Adapter*** and load the data into it
6. Add the ***recyclerViewOnScrollListener*** and finally
7. Load the first page of products into the list

ProductListViewHolder

Is responsible for returning how many products are being displayed by implementing getItemCount as well as binding the product and productListViewHolder by calling productListViewHolder.bindView (product)

Also, when the user clicks a product in the list:

1. The current product position is updated: ***MainActivity.instance.pc***
2. The showFragment method in MainActivity is called to show the product details: ***MainActivi***

Screen 2 Classes

com.product.productwheel.details

ProductDetailFragment

This fragment is used by ***productListViewHolder.onClick()*** to show a left / right swipeable list of product detail screens.

In the ***onCreateView*** method, the ***viewPager*** is retrieved from ***R.id.productPager*** *and* ***addOnPageChangeListener*** is added to be notified ***onPageSelected***.

Also, if the ***showProdListBtn*** is clicked, then the product list is displayed again by calling
MainActivity.instance.showFragment(ProductListFragment())

ProductDetailPage

Holds and Shows all of the product details.

In ***onCreateView*** the view is inflated from ***R.layout.product_detail_page***

In the bind method, the product is retrieved by calling
```
val product = MainActivity.instance.getProduct (page)
```

Then the products image is asked to load by calling

```
MainActivity.instance.comm.loadImage (product, productImage)
```

The text color is set to `R.color.green_lt` if the product is in stock or `R.color.red` if the product is out of stock,

and the correct in stock or out of stock icon is displayed **`R.drawable.check` or `R.drawable.red_x`** by calling `inStockBtn.setCompoundDrawablesWithIntrinsicBounds ( R.drawable.check, 0, 0, 0);`

The widgets are then loaded with the product details.

ProductDetailPageAdapter

Creates **ProductDetailPage**s and sets their page number when

```
override fun getItem (position: Int): Fragment? {
```

is called.

Also returns the current product count by calling

```
MainActivity.instance.getProductCount ()
```

External classes and packages used and why.

3) Why Kotlin?

Kotlin is rapidly replacing Java as the language of choice for Android Development is easy to learn and has several improvements over Java such as:

- **A more concise clean look due to being able to specify constants (val) and variables (var) by name only**
- **Apps run faster than java apps**
- **Leverages the language design expertise built into Java**
- **Completely interoperable with Java**
- **And more!**
-

4) Improvements Still Needed

As I create an app, I log milestones in readme.txt in the root of the project.
I keep track of bugs I have fixed or features that I have added.

I also keep track of outstanding issues.

I have posted the ***readme.txt*** here for you to see.
Scroll down to Know Issues to see what still needs to be done,

and Happy Coding!

:) Pete

```
Name: Walmart Product Wheel

Author: Pete Storli
Date: 13 Jan 2019 - 15 Jan 2019
Email: pstorli@gmail.com
Version "1.0.0.0"

Issues Resolved:

  0000 Inital Creation
  0000 Added product list and product detail fragments.
  0000 Added click on product in list and details for peoduct will be shown.
  0000 All UI functioning correctly.
  0000 Loading product list from url now.
  0000 Added lazy loading to product list
  0000 Adding padding to product list items
  0000 Added product image to product details
  0000 Hid short and long descriptions if text is blank.
  0000 Added routine to convert short and long text from html to span
  0000 Made product list text color red if out of stock, black if selected otherwise walmart blue
  0000 Added snackbar when product in stock button is pressed. Use red text if out of stock, in stock

Known Issues:
  0000 When you go from the ProductDetail page to the ProductList page,
       app needs to scroll to current item, more work is needed to complete this feature.

  0000 Added code to handle onSaveInstanceState and onRestoreInstanceState
       When an onOrientation change happens, onSaveInstanceState and onRestoreInstanceState are calle
       but more work is needed to complete this feature and get the app ddata persisted. (Started)

Notes:

  https://www.walmartbrandcenter.com/downloads.aspx
```

Add files

Comments

Pete Storli

Add a comment