MyAppTemplates

Glam documentation

Android Template App

Usage instructions:

On Eclipse:

Make sure that you have configured the Eclipse with Android ADT Plugin and also installed JDK 7 and Android SDK with latest plateforms.
For detail you can visit at: http://developer.android.com/sdk/index.html

After Eclipse is fully setup, open the Eclipse, Go to Files menu then select Import-> Existing project into workspace -> Browse, then select for all the projects (Main Template project and all the related Library files bundled in the Zip provided to you) then click on Finish.
Sometime Eclipse may show errors about Library project missing, for that just Right click on Project -> Properties -> Android-> Scroll down -> Remove all the lib projects that has Red cross -> Click Apply then OK -> Reopen this window and Add back those projects -> Click Apply then OK

Important Class files:

- All the Activity classes can be found in Main package i.e. com.glam
- All the Fragments and UI components can be found in *ui* Package i.e. com.glam.ui
- All the custom code classes can be found in *custom* package i.e. com.glam.custom

- All the Java bean classes are located in *model* package i.e. com.glam.model
- You can put all the Utility classes in *utils* package i.e. com.glam.utils

Photoshop files:

The Photoshop files can be found in the 'PSD' folder. They are layered - to use, simply
open them up in your design tool of choice (i.e. Adobe Photoshop).
==========
Below you will find detailed source code documentation, that will help you in using and making modifications to this template.
Do not hesitate to email at: contact@myapptemplates.com should you run into any issues.

Contents

com.glam.ui

Class Checkout

```
public class Checkout
extends CustomFragment
```

The Class Checkout is the fragment that shows the list products for checkout and show the credit card details as well. You need to load and display actual contents.

-
  - **Nested Class Summary**

Nested Classes

Modifier and Type	Class and Description
private class	**Checkout.CardAdapter** The Class CardAdapter is the adapter for showing products in Card format inside the RecyclerView.

- **Field Detail**
  - ***iList***

    ```
    private java.util.ArrayList<Data> iList
    ```

    The product list.

- **Constructor Detail**
  - ***Checkout***

    ```
    public Checkout()
    ```

- **Method Detail**
  - ***onCreateView***

- `public android.view.View onCreateView(android.view.LayoutInflater inflater,`
- `android.view.ViewGroup container,`

  `android.os.Bundle savedInstanceState)`

  Overrides:

onCreateView in class CustomFragment

- ***onClick***

  `public void onClick(android.view.View v)`

  Specified by:

`onClick` in interface `android.view.View.OnClickListener`

Overrides:

onClick in class CustomFragment

- ***setupView***

  `private void setupView(android.view.View v)`

  Setup the view components for this fragment. You write your code for initializing the views, setting the adapters, touch and click listeners etc.

  Parameters:

`v` - the base view of fragment

- ***loadDummyData***

  `private void loadDummyData()`

  Load dummy product data for displaying on the RecyclerView. You need to write your own code for loading real products from Web-service or API and displaying them on RecyclerView.

- ***onCreateOptionsMenu***
- `public void onCreateOptionsMenu(android.view.Menu menu,`
  `android.view.MenuInflater inflater)`

  Overrides:

`onCreateOptionsMenu` in class `android.support.v4.app.Fragment`

com.glam

Class CheckoutActivity

```
public class CheckoutActivity
extends CustomActivity
```

The Activity CheckoutActivity is just a container class for Checkout fragment to allow checkout screen to be shown separately.

- **Constructor Detail**
  - ***CheckoutActivity***

    ```
    public CheckoutActivity()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    protected void onCreate(android.os.Bundle savedInstanceState)
    ```

    Overrides:

    onCreate in class CustomActivity

  - ***getSupportFragmentManager***

    ```
    public android.support.v4.app.FragmentManager getSupportFragmentM
    anager()
    ```

com.glam.custom

Class CustomActivity

Direct Known Subclasses:

CheckoutActivity, Home, Login, MainActivity, ProductDetail

```
public class CustomActivity
extends android.support.v7.app.ActionBarActivity
implements android.view.View.OnClickListener
```

This is a common activity that all other activities of the app can extend to inherit the common behaviors like setting a Theme to activity.

- **Field Detail**
  - ***TOUCH***

    ```
    public static final TouchEffect TOUCH
    ```

    Apply this Constant as touch listener for views to provide alpha touch effect. The view must have a Non-Transparent background.

- **Constructor Detail**
  - ***CustomActivity***

    ```
    public CustomActivity()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    protected void onCreate(android.os.Bundle arg0)
    ```

    Overrides:

`onCreate` in class `android.support.v7.app.ActionBarActivity`

  - ***onOptionsItemSelected***

`public boolean onOptionsItemSelected(android.view.MenuItem item)`

Overrides:

`onOptionsItemSelected` in class `android.app.Activity`

- ***setupActionBar***

  `protected void setupActionBar()`

  This method will setup the top title bar (Action bar) content and display values. It will also setup the custom background theme for ActionBar. You can override this method to change the behavior of ActionBar for particular Activity

- ***onClick***

  `public void onClick(android.view.View v)`

  Specified by:

`onClick` in interface `android.view.View.OnClickListener`

- ***setTouchNClick***

  `public android.view.View setTouchNClick(int id)`

  Sets the touch and click listeners for a view..

  Parameters:

`id` - the id of View

  Returns:

the view

- ***setClick***

  `public android.view.View setClick(int id)`

  Sets the click listener for a view.

  Parameters:

`id` - the id of View

Returns:

the view

- ***getSupportFragmentManager***

```
public android.support.v4.app.FragmentManager getSupportFragmentM
anager()
```

com.glam.custom

Class CustomFragment

Direct Known Subclasses:

Checkout, MainFragment, OnSale, Settings

```
public class CustomFragment
extends android.support.v4.app.Fragment
implements android.view.View.OnClickListener
```

The Class CustomFragment is the base Fragment class. You can extend your Fragment classes with this class in case you want to apply common set of rules for those Fragments.

- **Constructor Detail**
  - ***CustomFragment***

```
public CustomFragment()
```

- **Method Detail**
  - ***onCreateView***
  - `public android.view.View onCreateView(android.view.LayoutInflater inflater,`
  - `android.view.ViewGroup container,`

    `android.os.Bundle savedInstanceState)`

    Overrides:

`onCreateView` in class `android.support.v4.app.Fragment`

  - ***setTouchNClick***

    `public android.view.View setTouchNClick(android.view.View v)`

    Set the touch and click listener for a View.

    Parameters:

`v` - the view

    Returns:

the same view

  - ***onClick***

    `public void onClick(android.view.View v)`

    Specified by:

`onClick` in interface `android.view.View.OnClickListener`

Class Data

```
public class Data
extends java.lang.Object
```

The Class Data is simple Java bean class that holds two members only to represent dummy data for app. You can customize or can write new bean classes as per you needs.

- **Field Detail**
  - ***texts***

    ```
    private java.lang.String[] texts
    ```

    The texts.

  - ***resources***

    ```
    private int[] resources
    ```

    The resources.

- **Constructor Detail**
  - ***Data***
  - ```
    public Data(java.lang.String[] texts,
        int[] resources)
    ```

    Instantiates a new data.

    Parameters:

`texts` - the texts

`resources` - the resources

- **Method Detail**
  - ***getTexts***

    ```
    public java.lang.String[] getTexts()
    ```

    Gets the texts.

    Returns:

    the texts

  - ***setTexts***
    ```

`public void setTexts(java.lang.String[] texts)`

Sets the texts.

Parameters:

`texts` - the new texts

- ***getResources***

  `public int[] getResources()`

  Gets the resources.

  Returns:

the resources

- ***setResources***

  `public void setResources(int[] resources)`

  Sets the resources.

  Parameters:

`resources` - the new resources

com.glam

Class Home

```
public class Home
extends CustomActivity
```

The Activity Home is launched after the Splash screen. It simply show two options for Login and Signup.

- **Constructor Detail**
  - ***Home***

    ```
    public Home()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    protected void onCreate(android.os.Bundle savedInstanceState)
    ```

    Overrides:

onCreate in class CustomActivity

  - ***setupView***

    ```
    private void setupView()
    ```

    Setup the click & other events listeners for the view components of this screen. You can add your logic for Binding the data to TextViews and other views as per your need.

  - ***onClick***

    ```
    public void onClick(android.view.View v)
    ```

    Specified by:

onClick in interface android.view.View.OnClickListener

    Overrides:

onClick in class CustomActivity

  - ***getSupportFragmentManager***

    ```
    public android.support.v4.app.FragmentManager getSupportFragmentM
    anager()
    ```

com.glam.ui

Class LeftNavAdapter

```
public class LeftNavAdapter
extends android.widget.BaseAdapter
```

The Adapter class for the ListView displayed in the left navigation drawer.

- **Field Detail**
  - ***items***

    ```
    private java.util.ArrayList<Data> items
    ```

    The items.

  - ***context***

    ```
    private android.content.Context context
    ```

    The context.

  - ***selection***

    ```
    private int selection
    ```

    The selection.

- **Constructor Detail**
  - ***LeftNavAdapter***
  - ```
    public LeftNavAdapter(android.content.Context context,
                          java.util.ArrayList<Data> items)
    ```

    Instantiates a new left navigation adapter.
    ```

Parameters:

`context` - the context of activity

`items` - the array of items to be displayed on ListView

- **Method Detail**
  - ***isSelection***

    ```
    public int isSelection()
    ```

    Checks if is selection.

    Returns:

    the int

  - ***setSelection***

    ```
    public void setSelection(int selection)
    ```

    Sets the selection.

    Parameters:

    `selection` - the new selection

  - ***getCount***

    ```
    public int getCount()
    ```

  - ***getItem***

    ```
    public Data getItem(int arg0)
    ```

  - ***getItemId***

    ```
    public long getItemId(int position)
    ```

  - ***getView***
  - `public android.view.View getView(int position,`
  - `android.view.View convertView,`
    `android.view.ViewGroup parent)`

com.glam

Class Login

```
public class Login
extends CustomActivity
```

The Activity Login is launched after the Home screen. You need to write your logic for actual Login. You also need to implement Facebook Login if required.

-
  - ## Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
private class	**Login.PageAdapter** The Class PageAdapter is adapter class for ViewPager and it simply holds a Single image view with dummy images.

- ## Field Detail
  - ***pager***

    ```
    private android.support.v4.view.ViewPager pager
    ```

    The pager.

  - ***vDots***

```
private android.widget.LinearLayout vDots
```

The view that hold dots.

- **Constructor Detail**
  - ***Login***

    ```
    public Login()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    protected void onCreate(android.os.Bundle savedInstanceState)
    ```

    Overrides:

`onCreate` in class `CustomActivity`

  - ***setupView***

    ```
    private void setupView()
    ```

    Setup the click & other events listeners for the view components of this screen. You can add your logic for Binding the data to TextViews and other views as per your need.

  - ***initPager***

    ```
    private void initPager()
    ```

    Inits the pager view.

  - ***setupDotbar***

    ```
    private void setupDotbar()
    ```

    Setup the dotbar to show dots for pages of view pager with one dot as selected to represent current page position.

  - ***onClick***

    ```
    public void onClick(android.view.View v)
    ```

    Specified by:

`onClick` in interface `android.view.View.OnClickListener`

Overrides:

onClick in class CustomActivity

- ***getSupportFragmentManager***

  ```
  public android.support.v4.app.FragmentManager getSupportFragmentM
  anager()
  ```

com.glam

Class MainActivity

```
public class MainActivity
extends CustomActivity
```

The Activity MainActivity will launched after the Login and it is the Home/Base activity of the app which holds all the Fragments and also show the Sliding Navigation drawer. You can write your code for displaying actual items on Drawer layout.

- **Field Detail**
  - ***drawerLayout***

    ```
    private android.support.v4.widget.DrawerLayout drawerLayout
    ```

    The drawer layout.

  - ***drawerLeft***

    ```
    private android.widget.ListView drawerLeft
    ```

ListView for left side drawer.

- ***drawerToggle***

  ```
  private android.support.v7.app.ActionBarDrawerToggle drawerToggle
  ```

  The drawer toggle.

- ***mActionBarAutoHideEnabled***

  ```
  private boolean mActionBarAutoHideEnabled
  ```

  The m action bar auto hide enabled.

- ***mActionBarShown***

  ```
  private boolean mActionBarShown
  ```

  The m action bar shown.

- ***mStatusBarColorAnimator***

  ```
  private android.animation.ObjectAnimator mStatusBarColorAnimator
  ```

  The m status bar color animator.

- ***ARGB_EVALUATOR***

  ```
  private static final android.animation.TypeEvaluator
  ARGB_EVALUATOR
  ```

  The Constant ARGB_EVALUATOR.

- ***mActionBarAutoHideSensivity***

  ```
  private int mActionBarAutoHideSensivity
  ```

  The m action bar auto hide sensivity.

- ***mActionBarAutoHideMinY***

  ```
  private int mActionBarAutoHideMinY
  ```

  The m action bar auto hide min y.

- ***mActionBarAutoHideSignal***

  ```
  private int mActionBarAutoHideSignal
  ```

  The m action bar auto hide signal.

- ***toolbar***

  ```
  public android.support.v7.widget.Toolbar toolbar
  ```

  The toolbar.

- **Constructor Detail**
  - ***MainActivity***

    ```
    public MainActivity()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    protected void onCreate(android.os.Bundle savedInstanceState)
    ```

    Overrides:

`onCreate` in class `CustomActivity`

  - ***setupDrawer***

    ```
    private void setupDrawer()
    ```

    Setup the drawer layout. This method also includes the method calls for setting up the Left & Right side drawers.

  - ***onNavDrawerStateChanged***
  - ```
    private void onNavDrawerStateChanged(boolean isOpen,
                                         boolean isAnimating)
    ```

    Called On navigation drawer state changed.

    Parameters:

`isOpen` - true if drawer is open

`isAnimating` - true of drawer is animating

  - ***autoShowOrHideActionBar***
    ```

```
private void autoShowOrHideActionBar(boolean show)
```

Auto show or hide action bar.

Parameters:

`show` - true to show the bar

- ***onActionBarAutoShowOrHide***

```
private void onActionBarAutoShowOrHide(boolean shown)
```

Called On action bar auto show or hide.

Parameters:

`shown` - true is action bar is showing

- ***enableActionBarAutoHide***

```
public void enableActionBarAutoHide(android.support.v7.widget.Rec
yclerView recList)
```

Enable action bar auto hide.

Parameters:

`recList` - the RecyclerView list

- ***onMainContentScrolled***
- `private void onMainContentScrolled(int currentY,`
  `int deltaY)`

Indicates that the main content has scrolled (for the purposes of showing/hiding the action bar for the "action bar auto hide" effect). currentY and deltaY may be exact (if the underlying view supports it) or may be approximate indications: deltaY may be INT_MAX to mean "scrolled forward indeterminately" and INT_MIN to mean "scrolled backward indeterminately". currentY may be 0 to mean "somewhere close to the start of the list" and INT_MAX to mean "we don't know, but not at the start of the list"

Parameters:

`currentY` - the current y

`deltaY` - the delta y

- ***initActionBarAutoHide***

  ```
  public void initActionBarAutoHide()
  ```

  Initializes the Action Bar auto-hide (aka Quick Recall) effect.

- ***setupLeftNavDrawer***

  ```
  private void setupLeftNavDrawer()
  ```

  Setup the left navigation drawer/slider. You can add your logic to load the contents to be displayed on the left side drawer. It will also setup the Header and Footer contents of left drawer. This method also apply the Theme for components of Left drawer.

- ***setupContainer***

  ```
  private void setupContainer(int pos)
  ```

  Setup the container fragment for drawer layout. This method will setup the grid view display of main contents. You can customize this method as per your need to display specific content.

  Parameters:

`pos` - the new up container

- ***onPostCreate***

  ```
  protected void onPostCreate(android.os.Bundle savedInstanceState)
  ```

  Overrides:

`onPostCreate` in class `android.app.Activity`

- ***onConfigurationChanged***

```
public void onConfigurationChanged(android.content.res.Configurat
ion newConfig)
```

Specified by:

`onConfigurationChanged` in interface `android.content.ComponentCallbacks`

Overrides:

`onConfigurationChanged` in class `android.support.v7.app.ActionBarActivity`

- ***onOptionsItemSelected***

```
public boolean onOptionsItemSelected(android.view.MenuItem item)
```

Overrides:

`onOptionsItemSelected` in class `CustomActivity`

- ***getSupportFragmentManager***

```
public android.support.v4.app.FragmentManager getSupportFragmentM
anager()
```

com.glam.ui

Class MainFragment

```
public class MainFragment
extends CustomFragment
```

The Class MainFragment is the base fragment that shows the list of various products. You can add your code to do whatever you want related to products for your app.

-
  - ## Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
private class	**MainFragment.CardAdapter** The Class CardAdapter is the adapter for showing products in Card format inside the RecyclerView.
private class	**MainFragment.PageAdapter** The Class PageAdapter is adapter class for ViewPager and it simply holds a RecyclerView with dummy images.

- ## Field Detail
  - ***iList***

    ```
    private java.util.ArrayList<Data> iList
    ```

    The product list.

- ## Constructor Detail
  - ***MainFragment***

    ```
    public MainFragment()
    ```

- ## Method Detail
  - ***onCreateView***
  - ```
    public android.view.View onCreateView(android.view.LayoutInflater
     inflater,
    ```
  - ```
                                  android.view.ViewGroup container,

    android.os.Bundle savedInstanceState)
    ```
    Overrides:

    onCreateView in class CustomFragment

  - ***onClick***

    ```
    public void onClick(android.view.View v)
    ```
    Specified by:

`onClick` in interface `android.view.View.OnClickListener`

Overrides:

onClick in class CustomFragment

- ***setupView***

  ```
  private void setupView(android.view.View v)
  ```

  Setup the view components for this fragment. You write your code for initializing the views, setting the adapters, touch and click listeners etc.

  Parameters:

`v` - the base view of fragment

- ***initPager***

  ```
  private void initPager(android.view.View v)
  ```

  Inits the pager view.

  Parameters:

`v` - the root view

- ***loadDummyData***

  ```
  private void loadDummyData()
  ```

  Load dummy product data for displaying on the RecyclerView. You need to write your own code for loading real products from Web-service or API and displaying them on RecyclerView.

- ***onCreateOptionsMenu***
- ```
  public void onCreateOptionsMenu(android.view.Menu menu,
                                  android.view.MenuInflater inflater)
  ```

  Overrides:

`onCreateOptionsMenu` in class `android.support.v4.app.Fragment`
  ```

com.glam.ui

Class OnSale

```
public class OnSale
extends CustomFragment
```

The Class OnSale is the fragment that shows the products in GridView.

-
  - **Nested Class Summary**

Nested Classes

Modifier and Type	Class and Description
private class	**OnSale.CardAdapter** The Class CardAdapter is the adapter for showing products in Card format inside the RecyclerView.

- **Field Detail**
  - ***iList***

    ```
    private java.util.ArrayList<Data> iList
    ```

    The product list.

- **Constructor Detail**
  - ***OnSale***

    ```
    public OnSale()
    ```

- **Method Detail**
  - ***onCreateView***
  - `public android.view.View onCreateView(android.view.LayoutInflater inflater,`
  - `android.view.ViewGroup container,`

    `android.os.Bundle savedInstanceState)`

    Overrides:

onCreateView in class CustomFragment

  - ***onClick***

    `public void onClick(android.view.View v)`

    Specified by:

`onClick` in interface `android.view.View.OnClickListener`

    Overrides:

onClick in class CustomFragment

  - ***setupView***

    `private void setupView(android.view.View v)`

    Setup the view components for this fragment. You write your code for initializing the views, setting the adapters, touch and click listeners etc.

    Parameters:

`v` - the base view of fragment

  - ***loadDummyData***

    `private void loadDummyData()`

    Load dummy product data for displaying on the RecyclerView. You need to write your own code for loading real products from Web-service or API and displaying them on RecyclerView.

  - ***onCreateOptionsMenu***
  - `public void onCreateOptionsMenu(android.view.Menu menu,`
    `android.view.MenuInflater inflater)`

    Overrides:

`onCreateOptionsMenu` in class `android.support.v4.app.Fragment`

com.glam

Class ProductDetail

```
public class ProductDetail
extends CustomActivity
```

The Activity ProductDetail is launched when user select a product item from product list or grid views in other sections of the app. Currently it shows Dummy details of product with dummy pics. You need to write your own code to load and display actual contents.

-
  - ## Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
private class	**ProductDetail.PageAdapter** The Class PageAdapter is adapter class for ViewPager and it simply holds a Single image view with dummy images.

- ## Field Detail
  - ***pager***

```
private android.support.v4.view.ViewPager pager
```

The pager.

- ***vDots***

  ```
  private android.widget.LinearLayout vDots
  ```

  The view that hold dots.

- **Constructor Detail**
  - ***ProductDetail***

    ```
    public ProductDetail()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    protected void onCreate(android.os.Bundle savedInstanceState)
    ```

    Overrides:

    onCreate in class CustomActivity

  - ***setupView***

    ```
    private void setupView()
    ```

    Setup the click & other events listeners for the view components of this screen. You can add your logic for Binding the data to TextViews and other views as per your need.

  - ***initPager***

    ```
    private void initPager()
    ```

    Inits the pager view.

  - ***setupDotbar***

    ```
    private void setupDotbar()
    ```

    Setup the dotbar to show dots for pages of view pager with one dot as selected to represent current page position.

  - ***onCreateOptionsMenu***

```
public boolean onCreateOptionsMenu(android.view.Menu menu)
```

Overrides:

`onCreateOptionsMenu` in class `android.app.Activity`

- ***onOptionsItemSelected***

```
public boolean onOptionsItemSelected(android.view.MenuItem item)
```

Overrides:

`onOptionsItemSelected` in class `CustomActivity`

- ***getSupportFragmentManager***

```
public android.support.v4.app.FragmentManager getSupportFragmentM
anager()
```

com.glam.ui

Class Settings

```
public class Settings
extends CustomFragment
```

The Class Settings is the fragment that shows various settings options.

- **Constructor Detail**
  - ***Settings***

```
public Settings()
```

- **Method Detail**
  - ***onCreateView***

- public android.view.View onCreateView(android.view.LayoutInflater inflater,
- android.view.ViewGroup container,

android.os.Bundle savedInstanceState)

Overrides:

onCreateView in class CustomFragment

- ***onClick***

public void onClick(android.view.View v)

Specified by:

onClick in interface android.view.View.OnClickListener

Overrides:

onClick in class CustomFragment

com.glam

Class SplashScreen

```
public class SplashScreen
extends android.app.Activity
```

The Class SplashScreen will launched at the start of the application. It will be displayed for 3 seconds and than finished automatically and it will also start the next activity of app.

- **Field Detail**
  - ***isRunning***

```
private boolean isRunning
```

Check if the app is running.

- **Constructor Detail**
  - ***SplashScreen***

    ```
    public SplashScreen()
    ```

- **Method Detail**
  - ***onCreate***

    ```
    public void onCreate(android.os.Bundle savedInstanceState)
    ```

    Overrides:

`onCreate` in class `android.app.Activity`

  - ***startSplash***

    ```
    private void startSplash()
    ```

    Starts the count down timer for 3-seconds. It simply sleeps the thread for 3-seconds.

  - ***doFinish***

    ```
    private void doFinish()
    ```

    If the app is still running than this method will start the Login activity and finish the Splash.

  - ***onKeyDown***
  - `public boolean onKeyDown(int keyCode, android.view.KeyEvent event)`

    Specified by:

`onKeyDown` in interface `android.view.KeyEvent.Callback`

    Overrides:

`onKeyDown` in class `android.app.Activity`

com.glam.utils

Class TouchEffect

```
public class TouchEffect
extends java.lang.Object
implements android.view.View.OnTouchListener
```

The Class TouchEffect is a Base Touch listener. It can be attached as touch listener for any view. It simply set alpha value for whole view to create a Touch like effect on View

- **Constructor Detail**
  - ***TouchEffect***

    ```
    public TouchEffect()
    ```

- **Method Detail**
  - ***onTouch***
  - ```
    public boolean onTouch(android.view.View v,
                           android.view.MotionEvent event)
    ```

    Specified by:

`onTouch` in interface `android.view.View.OnTouchListener`
    ```