Developing Android Apps with Liferay Screens

Logging into Liferay Portal

Before you start, you have to create your user account using this page:
https://screensdemo.liferay.com/web/screens-challenge/signup

In this section, you will use a LoginScreenlet within the newly created Android project for logging into Liferay Portal with a dummy account.

1. In Android Studio, open res/layout/activity_main.xml file. Select Text tab at the bottom to edit the XML
2. Delete the <TextView> element.
3. Add a new <com.liferay.mobile.screens.auth.login.LoginScreenlet> element to the activity_main.xml file.
4. Define the android:layout_height and android:layout_width attributes as match_parent.
5. Define the android:id attribute as @+id/login_default.
6. Define the liferay:layoutId attribute as @layout/login_default which uses a predefined layout.
7. The completed definition should look like:

```
<com.liferay.mobile.screens.auth.login.LoginScreenlet
   android:id="@+id/login_default"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   liferay:layoutId="@layout/login_default" />
```

8. If the word “liferay” appears in red in the XML above, set the cursor on it and type Alt+Enter. It will include the namespace in the XML header.
9. Run the project and see your brand new screenlet in the simulator
    a. Try to sign in using wrong user and password: you will see how the screenlet shows a default error message
    b. If you type correct credentials the screenlet will silently sign you into the portal. Notice the success message shown in the logs in Android Studio

Listening to the Screenlet’s events

In this section, you will learn how you can be notified when anything interesting happens inside the screenlet. For instance, you can get your code called when the login is successfully completed.

1. If we want to perform an action when the login operation is successfully completed, we need first to add the listener to the screenlet. When anything interesting happens inside the screenlet, it will notify to the listener object right away.
2. Locate the Login activity, get the screenlet using findViewById method, and set the listener:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   LoginScreenlet loginScreenlet = (LoginScreenlet) findViewById(R.id.login_default);
   loginScreenlet.setListener(this);
}
```

4. Make sure the class MainActivity implements the interface LoginListener by adding “implements LoginListener” to the end of the class line:

```
public class MainActivity extends ActionBarActivity implements LoginListener {
```

5. Create the required onLoginSuccess and onLoginFailure methods and add log to show the result

```
@Override
public void onLoginSuccess(User user) {
   System.out.println(“Login is OK: “ + user.getAttributes());
}

@Override
public void onLoginFailure(Exception e) {
   System.out.println(“Login failed: “ + e);
}
```

5. Launch the application and login using the same credentials used above that was used to login to the portal

Navigating to a new screen

1. Change the onLoginSuccess method of MainActivity, and use an intent to open the new activity (called SecondActivity)

```
@Override
public void onLoginSuccess(User user) {
   System.out.println(“Login is OK: “ + user.getAttributes());

   startActivity(new Intent(this, SecondActivity.class));
}
```

Displaying a Dynamic Form from Liferay

In this section, you will use a dynamic form screenlet to allow the user to send some data to the portal

1. In Android Studio, open SecondActivity
2. Delete the <TextView> element.
3. Add a new <com.liferay.mobile.screens.ddl.form.DDLFormScreenlet> element to the activity_view_content.xml file.
4. Define the android:layout_height and android:layout_width attributes as match_parent.
5. Define the liferay:structureId attribute as the Dynamic Definition id: 22354
6. Define the liferay:recordSetId attribute as the DDL List id: 22356
7. Define the liferay:repositoryId attribute, to upload files in the DDL form, as the repository id: 22339
8. Define the liferay:folderId attribute as the repository id: 22709
9. Define the liferay:layoutId attribute as @layout/ddl_form_default which uses a predefined layout.
10. The completed definition should look like:

```
<com.liferay.mobile.screens.ddl.form.DDLFormScreenlet
   android:layout_height="match_parent"
   android:layout_width="match_parent"
   liferay:structureId="22354"
   liferay:recordSetId="22356"
   liferay:repositoryId="22339"
   liferay:folderId="22709"
   liferay:layoutId="@layout/ddl_form_default" />
```

11. That's it! Launch the application, login using the same credentials used above and you'll see how the form will appear ready to be filled in.
    a. Leave some fields empty, and press Submit button. You'll see how validation fails because of required fields.
    b. Fill all values and press Submit button. You'll see a success message.
12. Now go back to the portal, and check the just added record in the Dynamic Data List

https://screensdemo.liferay.com/web/screens-challenge

Theming

In this section, you finally will learn how to modify the look & feel of your screenlet and meet your app visual requirements. For this, we use the concept of "viewset": a library of classes and layouts with a visual representation of the screenlets. You can change one screenlet layout and the look & feel will be changed, without changing the behavior.

1. The project already have an additional viewset installed called "material" (because it follows the Google's Material guidelines). You don't have to install it again, but just notice it's a single line in the app's gradle file:

```
dependencies {
    ...
    compile 'com.liferay.mobile:liferay-screens:1.0.+'
    compile 'com.liferay.mobile:liferay-material-viewset:1.0.+'
}
```

2. In Android Studio, open your "activity_main.xml" file and choose the "Text" view (a tab at the bottom)
3. In the LoginScreenlet XML tag, change the "layoutId" attribute to use the new viewset one (notice we use **@layout/login_material** instead of **@layout/login_default**)

```
<com.liferay.mobile.screens.auth.login.LoginScreenlet
   android:id="@+id/login_default"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   liferay:layoutId="@layout/login_material" />
```

4. Run the app. You'll see our beloved LoginScreenlet, but with a brand-new skin
5. Go back to Android Studio and open "activity_second.xml".
6. Now let's change the viewset to DDLFormScreenlet. In this case, you have to set the viewset for the screenlet, but also for each field type. Type the next changes (highlighted in red)

```
<com.liferay.mobile.screens.ddl.form.DDLFormScreenlet
   android:layout_height="match_parent"
   android:layout_width="match_parent"
   liferay:structureId="22354"
   liferay:recordSetId="22356"
   liferay:repositoryId="22339"
   liferay:folderId="22709"
   liferay:layoutId="@layout/ddl_form_material"
   liferay:checkboxFieldLayoutId="@layout/ddlfield_checkbox_material"
   liferay:dateFieldLayoutId="@layout/ddlfield_date_material"
   liferay:selectFieldLayoutId="@layout/ddlfield_select_material" />
```

7. Run the app, type right credentials, and you'll see how the form will appear with the new look and feel applied.

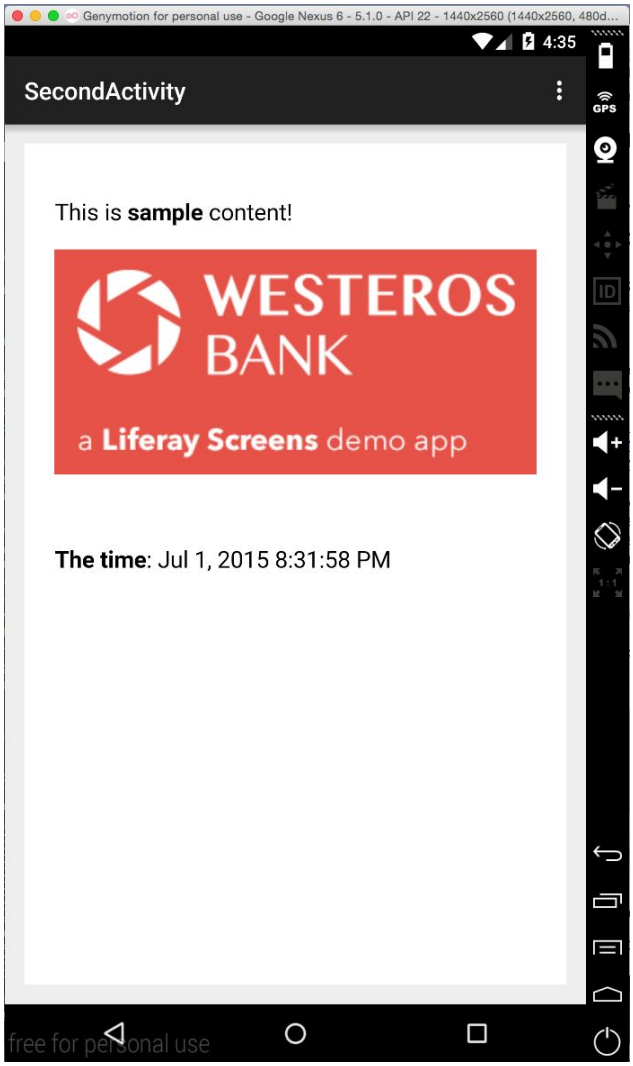
Post-Event Follow-up Activity

In this section, you will use the skills learned above to change the app to display web content from Liferay's web content management system. Don't worry, the content is already created, all you have to do is insert the screenlet and go!

1. In Android Studio, open SecondActivity
2. Delete the <com.liferay.mobile.screens.ddl.form.DDLFormScreenlet> element.
3. Add a new <com.liferay.mobile.screens.ddl.form.WebContentArticleScreenlet> element to the activity_view_content.xml file.
4. Define the android:layout_height and android:layout_width attributes as match_parent
5. Define the liferay:groupId attribute as the repository id: **22339**
6. Define the liferay:articleId attribute as the repository id: **41713**
7. Define the liferay:layoutId attribute as @layout/webcontentdisplay_default which uses a predefined layout.
8. The completed definition should look like:

```
<com.liferay.mobile.screens.webcontentdisplay.WebContentDisplayScreenlet
   android:layout_height="match_parent"
   android:layout_width="match_parent"
   liferay:articleId="41713"
   liferay:groupId="22339"
   liferay:layoutId="@layout/webcontentdisplay_default" />
```

9. That's it! Launch the application, login using the same credentials used above and you'll see how the example content will appear:



10. Take a screenshot and post it to the Screens forum thread created for this event. Include your comments, questions, and other feedback! The first 10 people to do this will receive a small gift from Liferay! Note: in order to post, you will need to register for a free account at liferay.com if you do not yet have one.