Mobile

5.1 Liferay Screens

5.1.1 Introduction

- Liferay Screens is a collection of fully native mobile components, using all the power of your Liferay platform as an enterprise grade mobile back-end.

(Figure 5.1, page 154) (Figure 5.2, page 155)

Figure 5.1:

- Liferay Screens speeds up and simplifies developing native mobile apps that use Liferay by providing a part of the visual UI and the underlying logic.
- Liferay Screens is based on the concept of screenlets.
- A screenlet is a visual component that you insert into your native app to leverage Liferay Platform's content and services.
- Screenlets are available to log in, create accounts, submit forms, display content, and more.
- Screenlets are independent, so you can use them in a modular fashion.

Figure 5.2:

- Screenlets deliver UI flexibility with Views/Themes that implement elegant User interfaces.
- Screenlets leverage the Liferay Mobile SDK to make server calls.
- The Mobile SDK is a low-level layer on top of the Liferay JSON API.
- Screenlets can be customized to fill needs.
- Writing screenlets requires you to familiarize yourself with Liferay's remote services, writing Mobile SDK calls and constructing the visual UI. This is a lot of effort, but Mobile SDK's details are abstracted from you.
- To use the next Liferay version, you need Liferay Screens 1.4.

5.1.2 Liferay Mobile SDK

- The Liferay Mobile SDK is a framework for building native mobile apps that integrate with your different Liferay Platform instances and their portlets.
- It provides the means for your mobile apps to easily consume Liferay Platform's core web services and the web services of your custom portlets.

- It wraps Liferay JSON web services, takes care of authentication, makes HTTP requests (synchronously or asynchronously), parses JSON results, and handles server side exceptions.

- It is compatible with the next Liferay Platform, and it comes with the Liferay Android SDK and Liferay iOS SDK ready for you to download and use.

5.1.3 Liferay Screens and supported mobile platforms

- Liferay Screens delivers the best experience to mobile users.

- This is achieved by creating native applications for each supported platform.

- Currently, screenlets are available for Android and iOS platforms.

- Both platforms have specific details that are partially abstracted by Screens, but as part of delivering the best experience, you will be developing native applications.

  - Each platform has its own tools and follows its respective best practices. Good prior knowledge is a plus.

5.1.4 Using screenlets in Android Apps

- Using Liferay Screens to enhance your Android app requires a fully-functional Android development environment.

- For a general overview about Android applications development, you can refer to the Android official documentation: `http://developer.android.com/training/index.html`

- In the Liferay Developer Network, you can find all the necessary documentation to use, enhance, and create screenlets. See the chapter on Android apps with Liferay Screens.

- Android Studio is the preferred tool for developing using Screens.

Using screenlets in Android Apps: managing dependencies

- To use Liferay Screens in your Android project, you must install it, install its dependencies, and configure it.
- Liferay Screens is released as an AAR file hosted in jCenter.
- We recommend you use Gradle or Maven to configure your project automatically with Liferay Screens. If you are familiar with any of these build automation systems, you will see that setting dependencies on Screens is similar to any other library.
- Preparing Android projects for Liferay Screens in our documentation guides you step by step through the process of setting a Gradle or Maven environment to start developing with Screens.
  - It is assumed you have a fully-functional Android development environment.
  - Essentially, you need to define the Screens dependency, and the build system will automatically download everything for you.

Using screenlets in Android Apps: communication with a Liferay Platform instance

- You need to configure communication with a Liferay Platform instance.
  - In your project's *res/values* folder, create a new file called `server_context.xml`, where you will define the server address, the `companyId`, and the `groupId` you want to access.

```
<string name="liferay_server">http://10.0.2.2:8080</string>

<integer name="liferay_company_id">10155</integer>
<integer name="liferay_group_id">10182</integer>
```

- Some screenlets have properties that can be configured in this file too. Please, refer to the specific screenlet documentation for details.

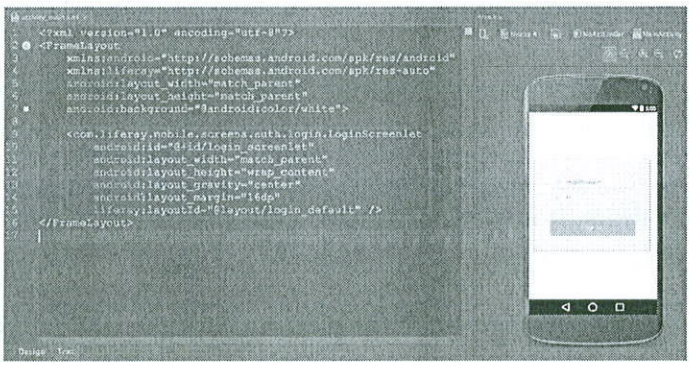
Using screenlets in Android Apps: adding a screenlet

- You can include screenlets in your app like any other component.

- In Android Studio's visual layout editor or your favorite editor, open your app's layout XML file.
- Insert the screenlet in your activity or fragment layout by typing the fully qualified name of the screenlet.
- The following screenshot, for example, shows the Login screenlet inserted in an activity's `FrameLayout`.
- You can see in the attached preview that the screenlet, with all the components (text fields, buttons), has been added.

(Figure 5.3, page 158)

Figure 5.3:



Using screenlets in Android Apps: configure attributes

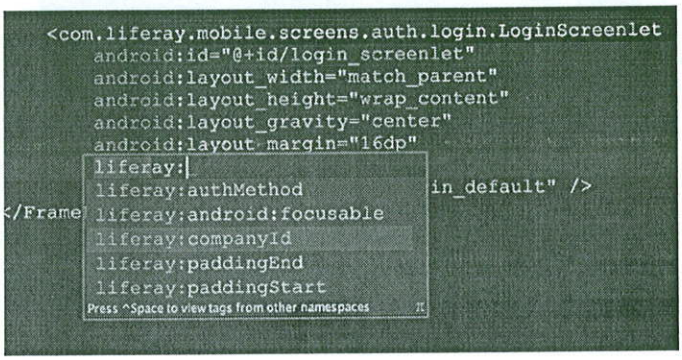
- A screenlet can have configurable attributes.
- In this example, you can select the width and height (like other standard Android components).
- You can also customize some screenlet-specific attributes.
- These attributes have the Liferay namespace.
- For Liferay-provided screenlets, you can find the list and definitions of all attributes in the *Liferay Screens for Android section*:

  `https://dev.liferay.com/develop/reference/-/knowledge_base/6-2/screenlets-in-liferay-screens-for-android`.

- The next screenshot shows the attributes of the Login screenlet being set.

(Figure 5.4, page 159)

Figure 5.4:



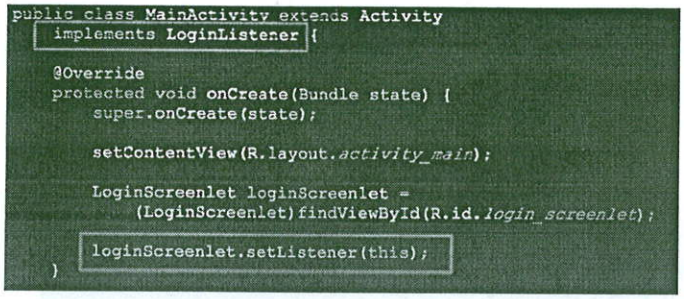
Using screenlets in Android Apps: events and listeners

- Screenlets generate and respond to UI events.
- To configure your app to listen for events the screenlet triggers, you need to implement the screenlet's listener interface in your activity or fragment class.
- The next screenshot shows this to you.
- Refer to the screenlet's documentation to learn its listener interface.
- Make sure to implement all methods required by the screenlet's listener interface.
- Then register your activity or fragment as the screenlet's listener.
- The activity class, for example, in the screenshot below, declares that it implements the Login screenlet's `LoginListener` interface, and it registers itself to listen for the screenlet's events.

(Figure 5.5, page 160)

That's all there is to it! Awesome! Now you know how to use screenlets in your Android apps.

Figure 5.5:



Using Liferay Push in Android Apps

- Liferay Screens supports push notifications in Android apps.
- To use them, you must configure some APIs and modify your app to consume and/or produce push notifications.
- Your first step is to create and configure a Google project to use Google Cloud Messaging. See section *Configure Liferay to use Push notifications* in the LDN for the details. Please note that you have to be registered in Google Cloud Platform. Check the specific terms of service.
- The Liferay Push Client for Android can be found in the Marketplace.
- You also need to configure the Liferay Push Client for Android portal plugin to use the project's GCM API.
- The exact configuration details as well as examples showing how to use it can be found in the Configure Liferay to use Push notifications chapter in the LDN.

5.1.5 Using screenlets in iOS

- Using Liferay Screens to enhance your iOS app requires a fully-functional iOS development environment.
- For a general overview about iOS applications development, you can refer to `https://developer.apple.com/library/ios/referencelibrary/GettingStarted/DevelopiOSAppsSwift/`.
- You can use both Objective-C and Swift to develop your app.

- In the Liferay Developer Network you can find all the necessary documentation to use, enhance, and create screenlets. See the chapter *iOS apps with Liferay Screens*.

Using screenlets in iOS: managing dependencies

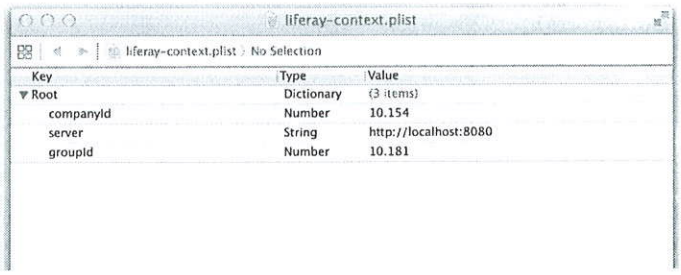
- To use Liferay Screens in your iOS project, you must install it, install its dependencies, and configure it.
- The steps differ depending on the iOS version your project supports.
  - If your app only needs to support iOS versions 8.0 or above, you can prepare it for Screens by adding a single line to your project's Podfile and using CocoaPods to complete configuration. This is possible because Liferay Screens is released as a standard CocoaPods dependency.
  - If you need to support iOS 7, however, then you must manually add Screens to your project. As you've probably guessed, manually adding Screens to your project is more challenging, but it is still not horribly complicated.
- Screens for iOS is written in Swift. You need to use CocoaPods version 0.36 or higher.
- Assuming you have a fully-functional iOS development environment section *Preparing iOS projects for Liferay Screens* guides you step-by-step through the process of setting the environment, to start developing with Screens.

Using screenlets in iOS Apps: communication with a Liferay Platform instance

- You need to configure communication with a Liferay Platform instance.
  - Liferay Screens uses a property list (.plist) file to access your Liferay instance.
  - In your project, create a `liferay-server-context.plist` file, where you will define the server address, the `companyId`, and the `groupId` you want to access.

(Figure 5.6, page 162)

Figure 5.6:

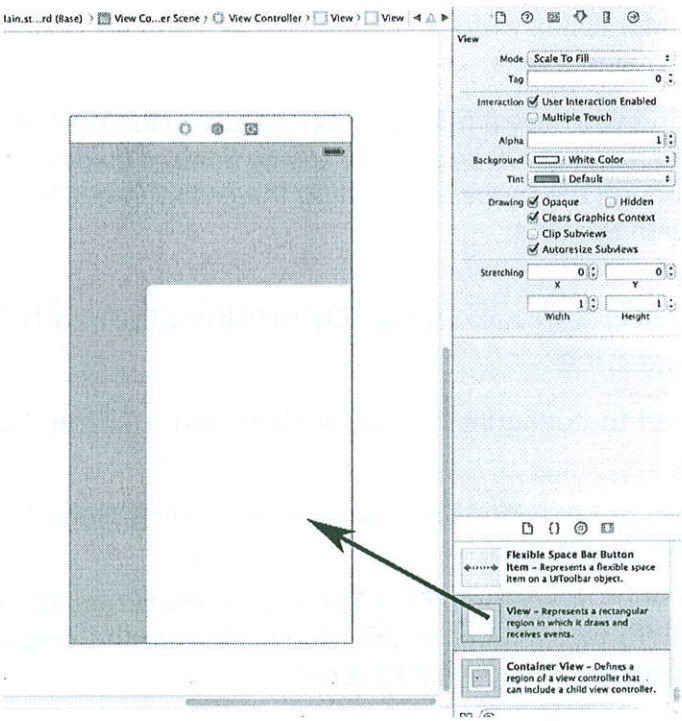


Using screenlets in iOS Apps: adding a screenlet from Swift

- Have your Storyboard in the Interface Builder.
- Insert a new UIView into your Storyboard or XIB file.

(Figure 5.7, page 162)

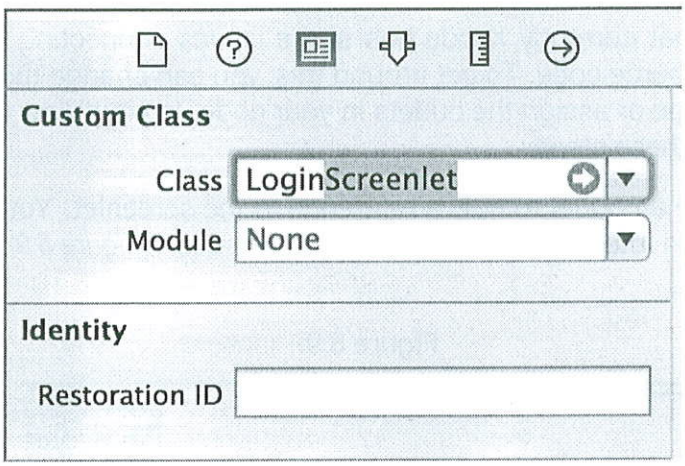
Figure 5.7:



- Next, enter the screenlet's name as the Custom Class. For example, if you're using the Login screenlet, then enter LoginScreenlet as the class.

(Figure 5.8, page 163)

Figure 5.8:

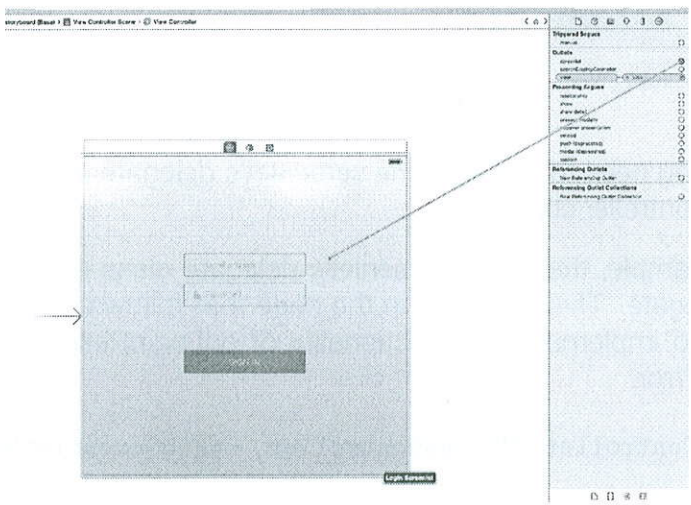


- Now you need to conform the screenlet's delegate protocol in your ViewController class.
- For example, the Login screenlet's delegate class is LoginScreenletDelegate. This is shown in the code that follows. Note that you need to implement the functionality of onLoginResponse and onLoginError.

```
class ViewController: UIViewController, LoginScreenletDelegate {
...

func onLoginResponse(attributes: [String:AnyObject]) {
    // handle succeeded login using passed user attributes
}

func onLoginError(error: NSError) {
    // handle failed login using passed error
}

...
```

- If you're using CocoaPods, you need to import Liferay Screens in your View Controller: `import LiferayScreens`
- Now that the screenlet's delegate protocol conforms in your View-Controller class, go back to Interface Builder and connect the screenlet's delegate to your View Controller. If the screenlet you're using has more outlets, you can assign them as well.
- Note that currently Xcode has some issues connecting outlets to Swift source code. To get around this, you can change the delegate data type or assign the outlets in your code. In your View Controller, follow these steps:
- Declare an outlet to hold a reference to the screenlet. You can connect it in Interface Builder without any issues. *(Figure 5.9, page 164)*

Figure 5.9:



- Assign the screenlet's delegate the `viewDidLoad` method. This is the connection typically done in Interface Builder.

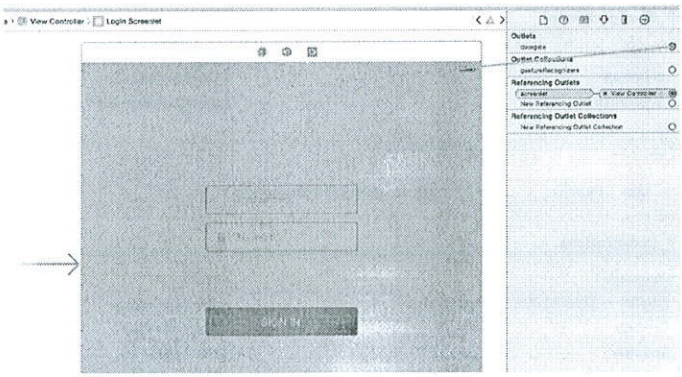
These steps are shown in the following code for Login screenlet's View-Controller.

```
class ViewController: UIViewController,LoginScreenletDelegate {
    @IBOutlet var screenlet: LoginScreenlet?
```

```
override func viewDidLoad() {
    super.viewDidLoad()
    self.screenlet?.delegate = self
}

...
```

(Figure 5.10, page 165)

Figure 5.10:



Using screenlets in iOS Apps: adding a screenlet using Objective-C

- It is also possible to add screenlets using Objective-C.
- The approach is quite different. Please refer to the *Using screenlets from Objective-C* section in the *Using screenlets in iOS Apps* page.

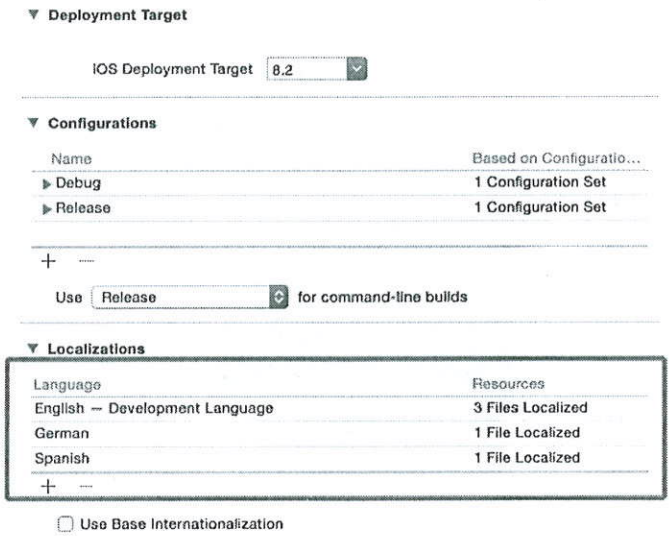
Using screenlets in iOS Apps: Localizing screenlets

- Follow Apple's standard mechanism to implement localization in your screenlet. `https://developer.apple.com/library/ios/documentation/MacOSX/Conceptual/BPInternational/Introduction/Introduction.html`
- Even though a screenlet may support several languages, you must also support those languages in your app.

- Screenlet's support for a language is only valid if your app supports that language. To support a language, make sure to add it as a localization in your project's settings.

(Figure 5.11, page 166)

Figure 5.11:



5.1.6 Available screenlets

- Liferay Screens contains a collection of screenlets that allows you to start working and interact with Liferay Platform within a few steps.
- This is the list of bundled screenlets:
  - LoginScreenlet: Signs users into a Liferay instance.
  - SignUpScreenlet: Registers new users in a Liferay instance.
  - ForgotPasswordScreenlet: Sends emails containing a new password or password reset link to users.
  - UserPortraitScreenlet: Show the user's portrait picture.
  - DDLFormScreenlet: Presents dynamic forms to be filled out by users and submitted back to the server.
  - DDLListScreenlet: Shows a list of records based on a pre-existing DDL in a Liferay instance.

  - **AssetListScreenlet**: Shows a list of assets managed by Liferay's Asset Framework. This includes web content, blog entries, documents, users, and more.
  - **WebContentDisplayScreenlet**: Shows the web content's HTML. This screenlet uses the features available in Web Content Management.

- Each screenlet has its own documentation page, where all the parameters and options are described.

  - Android: `https://dev.liferay.com/develop/reference/-/knowledge_base/6-2/screenlets-in-liferay-screens-for-android`
  - iOS: `https://dev.liferay.com/develop/reference/-/knowledge_base/6-2/screenlets-in-liferay-screens-for-ios`

- If you want to use a particular service available in a Liferay Platform installation, check if the available screenlets allow you to use that particular service.

LoginScreenlet

- The LoginScreenlet authenticates existing platform Users in your app.
- The following authentication methods are supported:

  - Basic: uses User login and password according to HTTP Basic Access Authentication specification (check the method used in your Liferay Platform installation). Depending on the authentication method used by your Liferay instance, you need to provide the User's email address, screen name, or User ID. You also need to provide the User's password.
  - OAuth: implements the OAuth 1.0a specification.

- Credentials can be stored in the device, to enable auto-login on future access.
- This screenlet requires platform configuration.

(Figure 5.12, page 168) (Figure 5.13, page 168)

Figure 5.12:

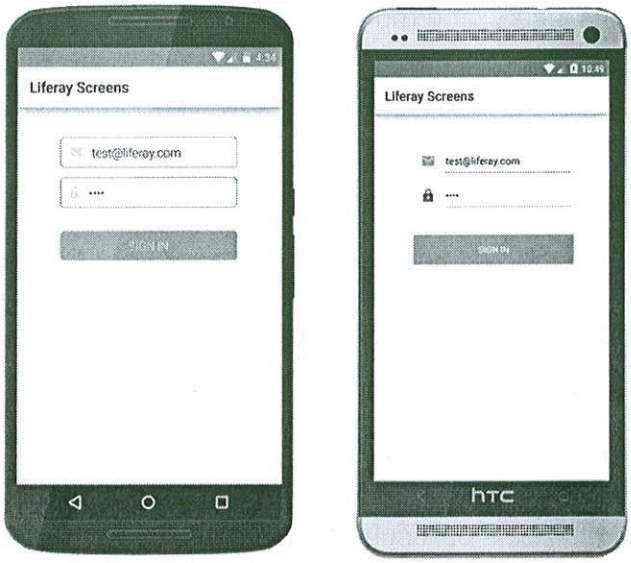
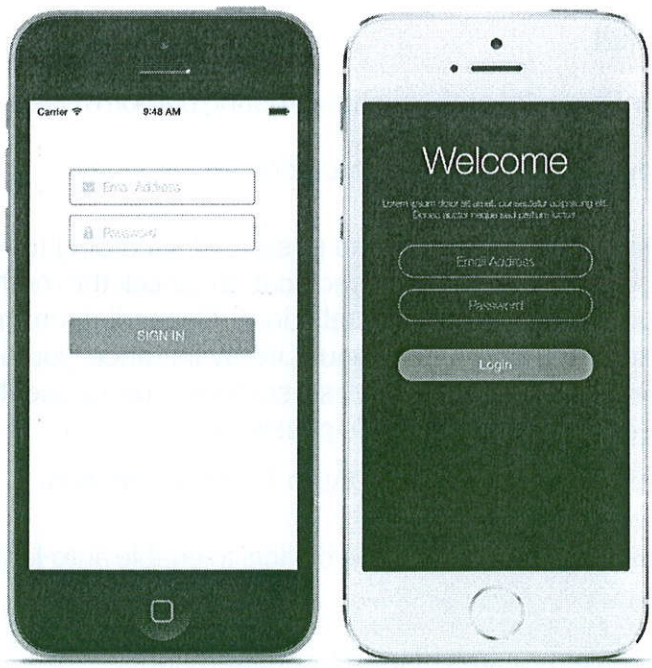


Figure 5.13:

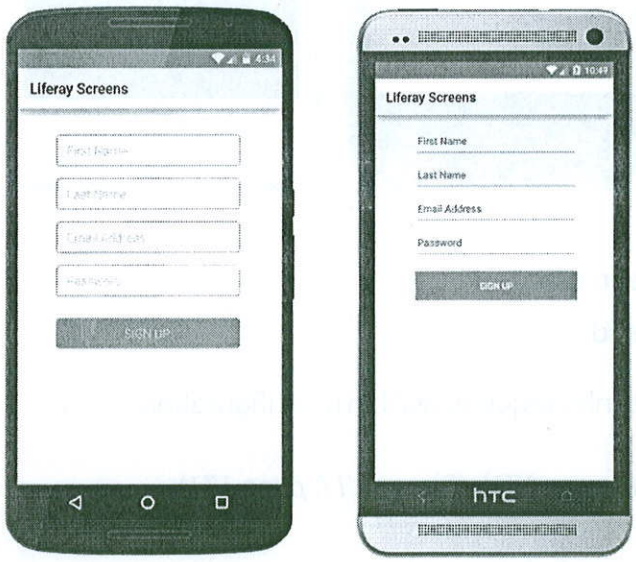


SignUpScreenlet

- The SignUpScreenlet creates a new User in your Liferay instance: a new User of your app can become a new User in your platform.
- You can also use this screenlet to save new Users' credentials on their devices. This enables auto-login for future sessions.
- It also supports navigation of form fields from the device's keyboard.
- This screenlet requires platform configuration.

(Figure 5.14, page 169) (Figure 5.15, page 170)

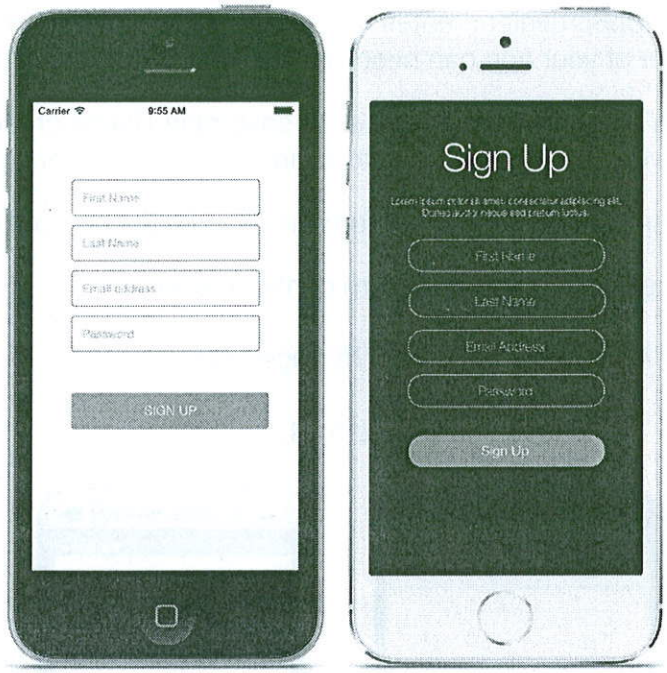
Figure 5.14:



ForgotPasswordScreenlet

- The ForgotPasswordScreenlet sends emails to registered Users with their new passwords or password reset links, depending on the server configuration.
- The available authentication methods are:
  - Email address

Figure 5.15:



  - Screen name
  - User id

- This screenlet requires platform configuration.

(Figure 5.16, page 171) (Figure 5.17, page 171)

UserPortraitScreenlet

- The UserPortraitScreenlet shows the Users profile pictures.
- If a User doesn't have a profile picture, a placeholder image is shown.
- This screenlet allows the profile picture to be edited.
- This screenlet requires special permissions in the Android platform.

(Figure 5.18, page 172) (Figure 5.19, page 172)

Figure 5.16:

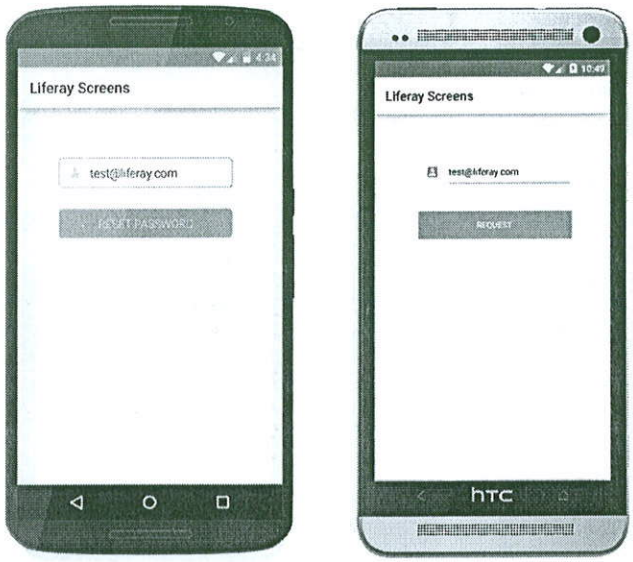


Figure 5.17:

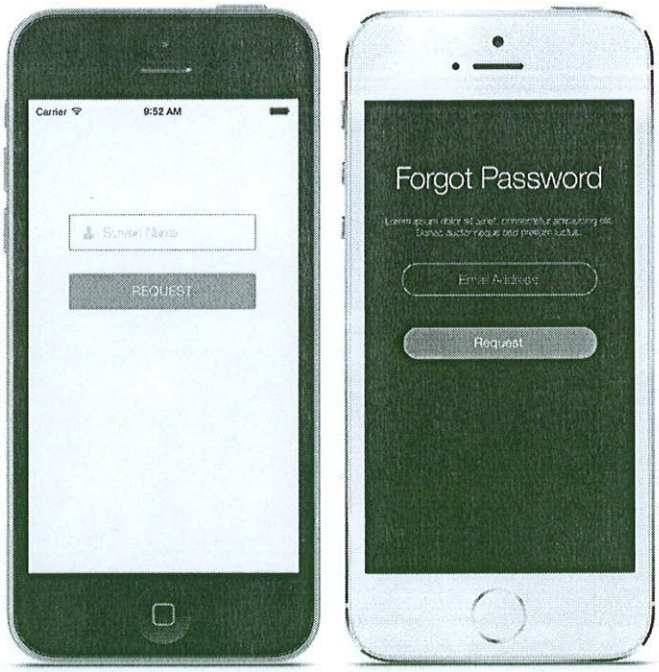


Figure 5.18:

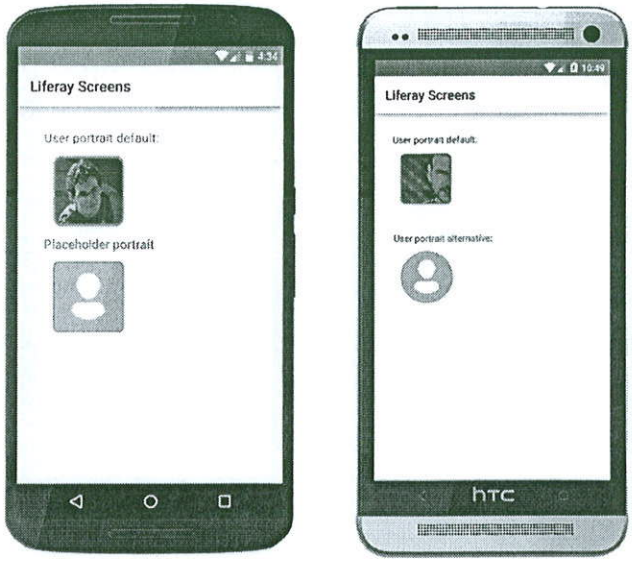
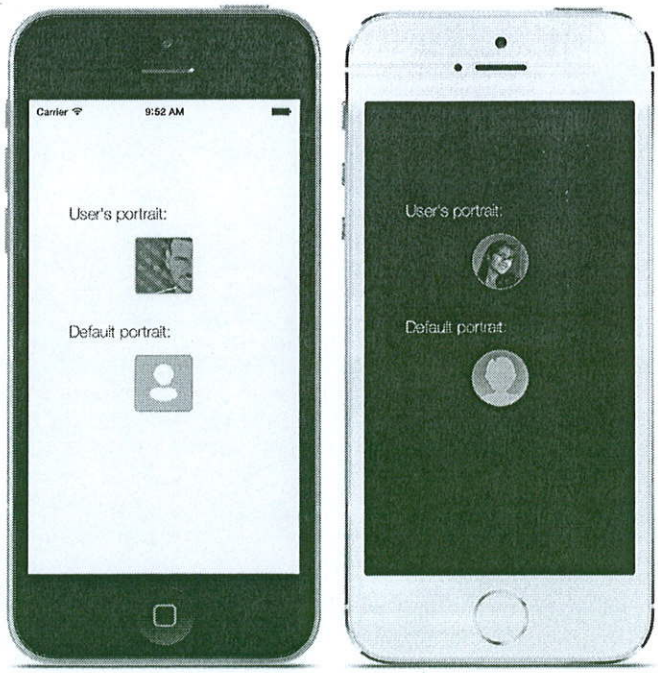


Figure 5.19:

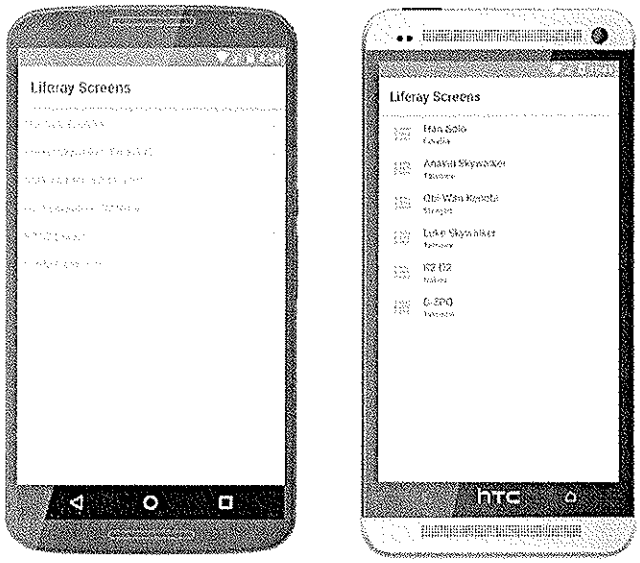


DDLFormScreenlet

- The DDLFormScreenlet shows a set of fields that can be filled in by the User. Initial or existing values can be shown in the fields.
- Fields of the following data types are supported:
  - Boolean: A two state value typically represented by a checkbox.
  - Date: A formatted date value. The format depends on the device's current locale.
  - Decimal, Integer, and Number: A numeric value.
  - Documents & Media: A file stored on the device. It can be uploaded to a specific platform repository.
  - Radio: A set of options to choose from. A single option must be chosen.
  - Select: A selection box of options to choose from. A single option must be chosen.
  - Text: A single line of text.
  - Text Area: Multiple lines of text.
- The DDLFormScreenlet also supports the following features:
  - Stored records can support a specific workflow.
  - A Submit button can be shown at the end of the form.
  - Required values and validation for fields can be used.
  - Users can traverse the form fields from the keyboard.
  - Supports i18n in record values and labels.
- There are also a few limitations that you should be aware of when using DDLFormScreenlet:
  - Nested fields in the data definition aren't supported.
  - Selection of multiple items in the Radio and Select data types isn't supported.
- This screenlet has different capabilities and requirements for each platform. Please check documentation for further details.
- This screenlet requires platform configuration.

(Figure 5.20, page 174) (Figure 5.21, page 175)

Figure 5.22:

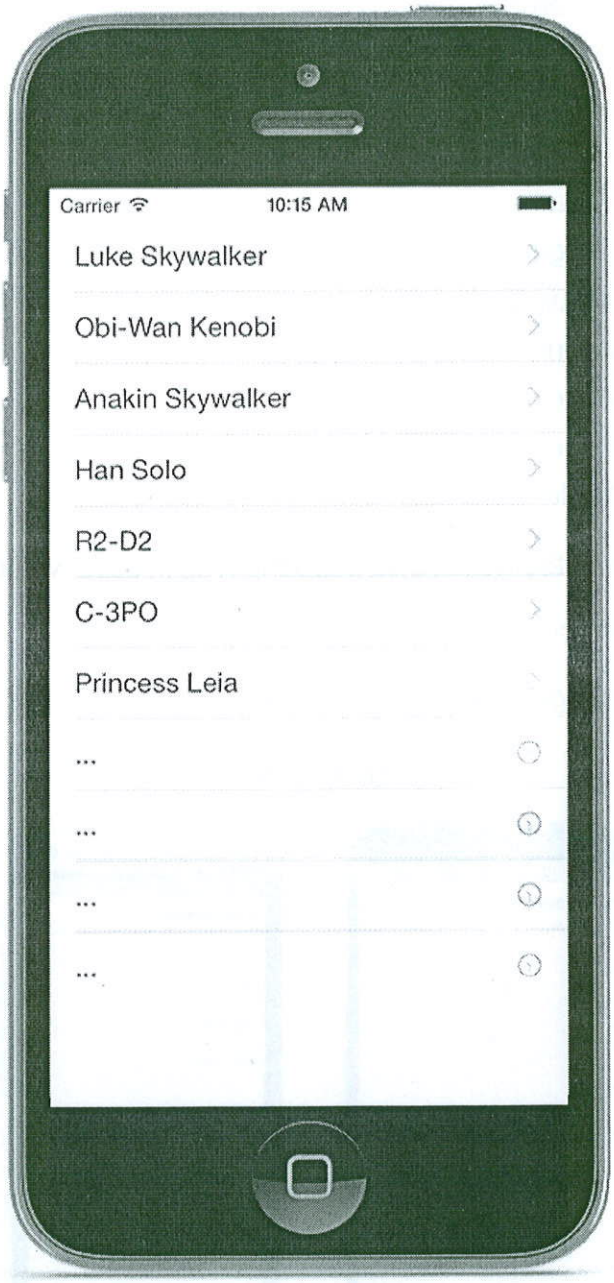


  - Layout
  - Organization
  - User
  - UserGroup
  - BlogsEntry
  - BookmarksEntry
  - BookmarksFolder
  - CalendarEvent
  - DLFileEntry
  - DLFileEntryMetadata
  - DLFileEntryType
  - DLFileRank
  - (continues)

- The AssetListScreenlet can show assets belonging to the following classes:

  - DLFileShortcut

Figure 5.23:



- DLFileVersion

- DDLRecord
- DDLRecordSet
- JournalArticle (Web Content)
- JournalFolder
- MBMessage
- MBThread
- MBCategory
- MBDiscussion
- MBMailingList
- WikiPage
- WikiPageResource
- WikiNode

- The AssetListScreenlet also supports i18n in asset values.
- This screenlet requires platform configuration.

(Figure 5.24, page 178) (Figure 5.25, page 179)

Figure 5.24:

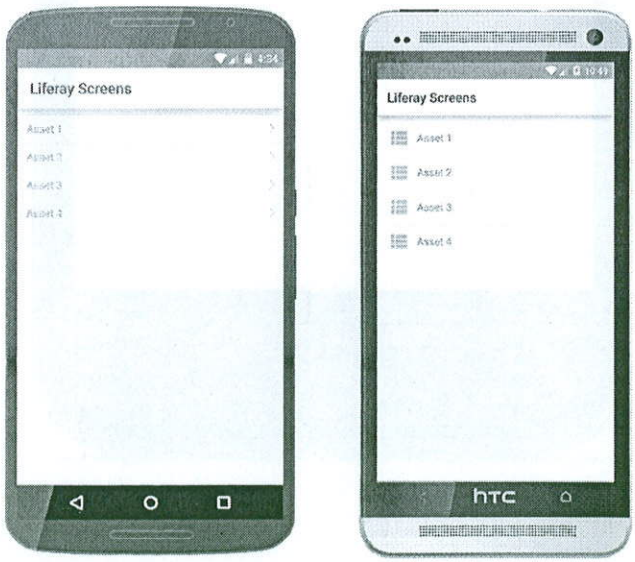
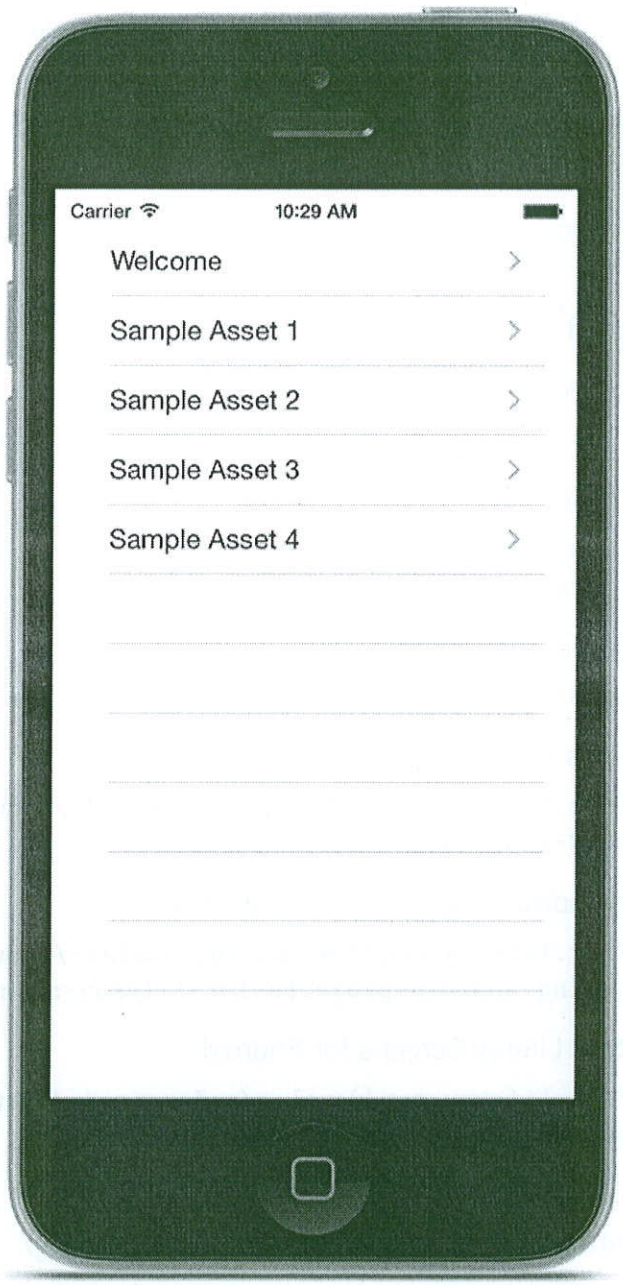


Figure 5.25:



WebContentDisplayScreenlet

- The WebContentDisplayScreenlet shows web content elements in your app, rendering the web content's inner HTML.
- The screenlet also supports i18n, rendering contents differently depending on the device's locale.

(Figure 5.26, page 181)

5.1.7 Available resources

- Liferay Screens:

  `https://www.liferay.com/en/products/liferay-screens`

- Liferay Mobile SDK:

  `https://www.liferay.com/es/community/liferay-projects/liferay-mobile-sdk`

- Android official documentation:

  `http://developer.android.com/training/index.html`

- Android apps with Liferay Screens:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/android-apps-with-liferay-screens`

- Preparing Android projects for Liferay Screens:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/preparing-android-projects-for-liferay-screens`

- Screenlets in Liferay Screens for Android:

  `https://dev.liferay.com/develop/reference/-/knowledge_base/6-2/screenlets-in-liferay-screens-for-android`

- Google Cloud Messaging:

  `https://developers.google.com/cloud-messaging/`

- Configure Liferay to use Push notifications:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/using-liferay-push-in-android-apps#configuring-to-use-liferay-push-notifications`

Figure 5.26:

- Liferay Push Client for Android:

`https://www.liferay.com/es/marketplace/-/mp/application/48438926`

- Configure Liferay to use Push notifications:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/using-liferay-push-in-android-apps`

- Start Developing iOS Apps (Swift):

  `https://developer.apple.com/library/ios/referencelibrary/GettingStartedDevelopiOSAppsSwift/`

- iOS apps with Liferay Screens:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/ios-apps-with-liferay-screens`

- Preparing iOS projects for Liferay Screens:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/preparing-ios-projects-for-liferay-screens`

- Using screenlets in iOS Apps:

  `https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/using-screenlets-in-ios-apps`

- Screenlets in Liferay Screens for iOS:

  `https://dev.liferay.com/develop/reference/-/knowledge_base/6-2/screenlets-in-liferay-screens-for-ios`