Android - Using native pages

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.0/adding-native-functionality/android-using-native-pages-hybrid-applications.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

This tutorial explains how to integre native and web "pages" in an Android application by using the WL.NativePage.show() API to open a native page from JavaScript.

With this method, data can be sent from JavaScript to the opened native page. You can specify a callback to be called after the native page closes.

This tutorial covers the following topics:

- Connecting to the plugin from the JavaScript code
- · Creating a native page
- · Returning control to the web view
- Sample application

Connecting to the plugin from the JavaScript code

1. Implement the WL.NativePage.show() method to open the native page:

```
function openNativePage() {
  var params = {
    nameParam : $('#nameInput').val()
  };
  WL.NativePage.show(nativePageClassName, backFromNativePage, params)
  ;
}
```

- nativePageClassName: The name of a native Android Class to start.
- o backFromNativePage: A callback function to call when the native page closes.
- o params: An optional custom parameters object to pass to the native code.
- 2. To handle the callback function:

```
function backFromNativePage(data) {
   alert("Received phone number is: " + data.phoneNumber)
;
}
```

The backFromNativePage(data) function passes data back to the web part of an application after the native page closes.

Creating a native page

In Android, the native page must be implemented as an Android Activity, or extend an existing Activity.

1. Declare the native page in the AndroidManifest.xml file, as you would any Activity. For example:

```
<activity android:name=".HelloNative"></activity>
```

2. To retrieve custom data parameters that are passed from the web view, use an Intent object:

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   String name = getIntent().getStringExtra("nameParam")
;
```

Returning control to the web view

When the native page switches back to the web view, the finish() function is called for the Activity. You can pass data back to the web view by using an Intent object. For example:

Java:

```
String phoneNumber = editText.getText().toString();
Intent phoneNumberInfo = new Intent();
phoneNumberInfo.putExtra("phoneNumber", phoneNumber);
setResult(RESULT_OK, phoneNumberInfo);
finish();
```

JavaScript:

```
function backFromNativePage(data) {
   alert("Received phone number is: " + data.phoneNumber)
;
}
```

Sample application

Click to download

(http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/UsingNativePagesInHybridAppsProject.zip) the Studio project.

