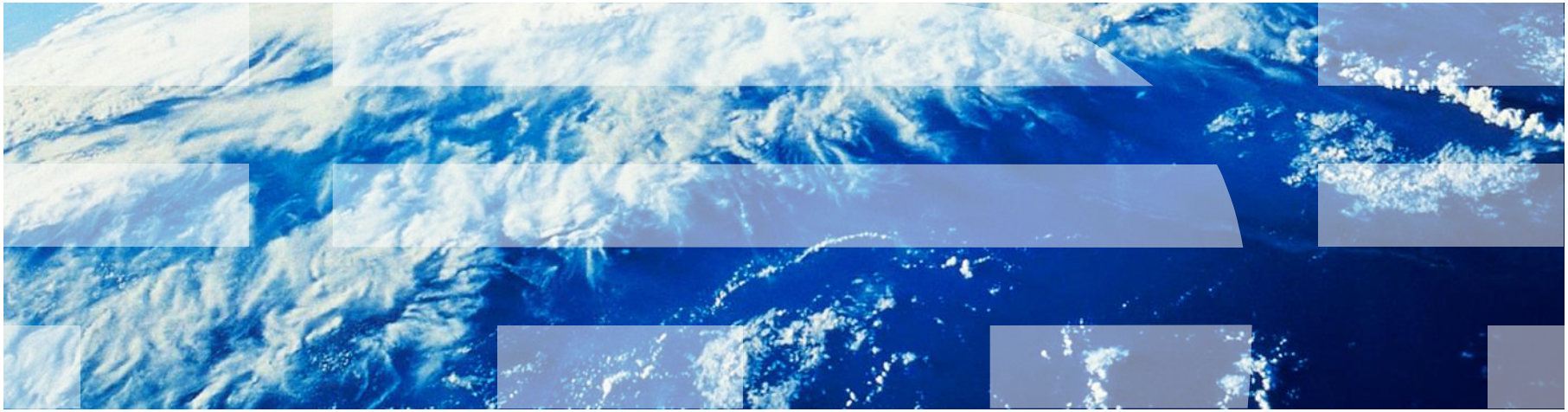
IBM Worklight Foundation V6.2.0
入門

ネイティブ Android アプリケーションでのプッシュ通知



2014 年 6 月 30 日

商標

- IBM、IBM ロゴ、ibm.com および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

IBM® について

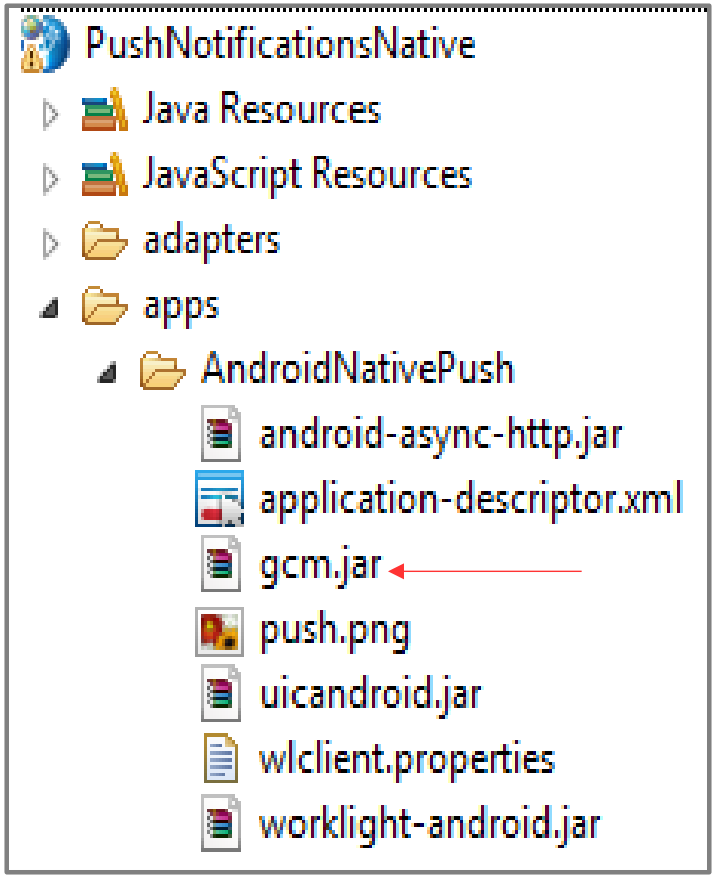
- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

Worklight プロジェクトのセットアップ(1/3)

- IBM Worklight® Foundation は、ネイティブ Android アプリケーションでプッシュ通知のサブスクライブ、アンサブスクライブ、および受信を行う機能を提供しています。
- Worklight プロジェクトを作成して、Android 対応のネイティブ API を追加します。
- ネイティブ API には以下のファイルが含まれています。
  - **gcm.jar** ファイルには、Android アプリケーションを Google Cloud Messaging (GCM) に登録するために必要なクラスが含まれています。
  - **push.png** は、プッシュ通知の受信時に表示されるアイコン・ファイルです。
- プッシュ通知のセットアップ方法について詳しくは、「IBM Worklight Foundation ユーザー文書」を参照してください。



Worklight プロジェクトのセットアップ(2/3)

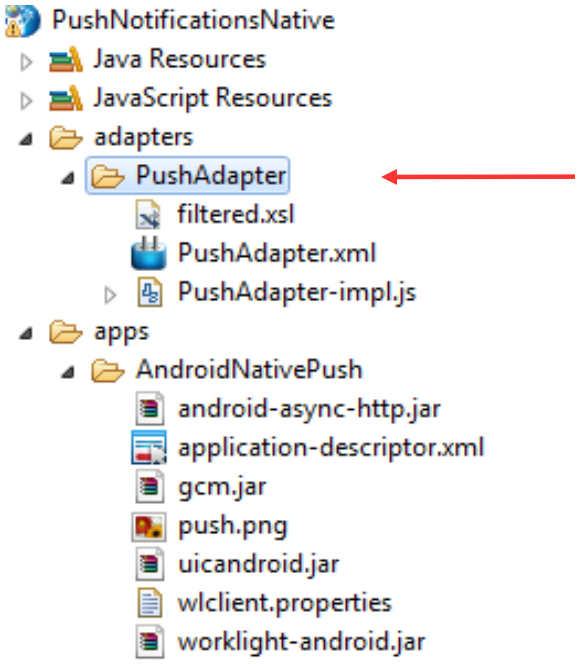
- application-descriptor.xml ファイルを編集して、Google API サーバー・キーおよびプロジェクト ID を <pushSender> タグに追加します。

```
<nativeAndroidApp xmlns="http://www.worklight.com/native-android-descriptor" id="AndroidNativePush" platformVersion="6.0.0" securityTest="MySecur
    <displayName>AndroidNativePush</displayName>
    <description>AndroidNativePush</description>
    <pushSender key="SOME-GCM-KEY" senderId="SOME-GCM-ID"/>
    <publicSigningKey>Replace this text with the public key of the certificate with which you sign the APK. For details see the Worklight Develop
</nativeAndroidApp>
```

- API キーがない場合は、https://code.google.com/apis/console から取得します。

Worklight プロジェクトのセットアップ(3/3)

- 次に、`eventSource` を含むアダプターを追加します。




```
WL.Server.createEventSource({
    name: 'PushEventSource',
    onDeviceSubscribe: 'deviceSubscribeFunc',
    onDeviceUnsubscribe: 'deviceUnsubscribeFunc'
});

function deviceSubscribeFunc(userSubscription, deviceSubscription){
    WL.Logger.debug(">> deviceSubscribeFunc");
}

function deviceUnsubscribeFunc(userSubscription, deviceSubscription){
    WL.Logger.debug(">> deviceUnsubscribeFunc");
}
```

- ネイティブ API およびアダプターをデプロイします。

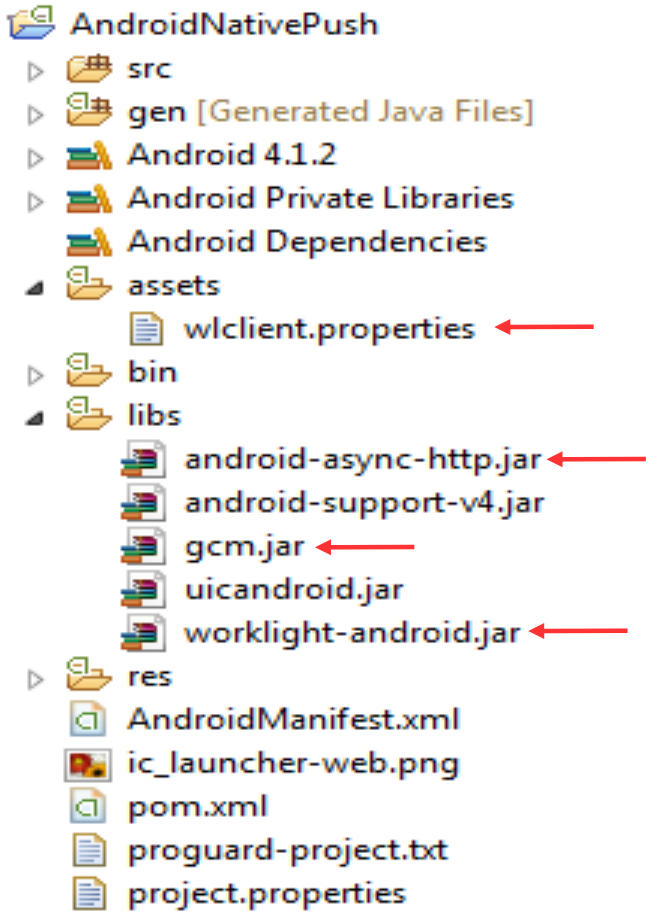
アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

プッシュ通知用のネイティブ・アプリケーションのセットアップ(1/4)

- Android アクティビティー・プロジェクトを作成して、以下のファイルを Worklight プロジェクトから Android プロジェクトにコピーします。

Worklight プロジェクト	Android アクティビティー・プロジェクト
wlclient.properties	assets/wlclient.properties
gcm.jar	libs/gcm.jar
worklight-android.jar	libs/worklight-android.jar
android-async-http.jar	libs/android-async-http.jar
push.png	drawable*/push.png



プッシュ通知用のネイティブ・アプリケーションのセットアップ(2/4)

- ネイティブ Android プロジェクトの `wlclient.properties` ファイルを編集して、以下のフィールドに適切な値を入力します。
  - **wlServerHost** – Worklight Server のホスト名または IP アドレス
  - **wlServerPort** – Worklight Server が listen するポート
  - **wlServerContext** – Worklight Server のコンテキスト・ルート
  - **GcmSenderId** – Google API コンソールを使用して取得したプロジェクトのプロジェクト番号。

```
wlServerProtocol = http
wlServerHost = 169.254.25.40
wlServerPort = 10080
wlServerContext = /PushNotificationsNative/
wlAppId = AndroidNativePush
wlAppVersion = 1.0
wlEnvironment = Androidnative
wlUid = wY/mbnwKTDDYQUvuQCdSgg==
wlPlatformVersion = 6.2.0.00.20140520-1620
#languagePreferences = Add locales in order of preference (e.g. en, fr, fr-CA)
#For Push Notifications,uncomment below line and assign value to it
#GcmSenderId =
```

プッシュ通知用のネイティブ・アプリケーションのセットアップ(3/4)

- Android プロジェクトの AndroidManifest.xml ファイルに、次のアクセス権を追加します。

```
<permission android:name="com.worklight.androidnativepush.permission.C2D_MESSAGE" android:protectionLevel="signature" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="com.worklight.androidnativepush.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

- メイン AndroidNativePush アクティビティーに、launchMode 属性を追加します。その値を singleTask に設定します。

```
<activity
    android:name="com.worklight.androidnativepush.AndroidNativePush"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar"
    android:launchMode="singleTask">
```

プッシュ通知用のネイティブ・アプリケーションのセットアップ(4/4)

- 通知のメイン AndroidNativePush アクティビティーに、インテント・フィルターを追加します。

```
<intent-filter>
    <action android:name="com.worklight.androidnativepush.AndroidNativePush.NOTIFICATION" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

- GCMIntentService を追加し、通知の RECEIVE および REGISTRATION についてインテント・フィルターを追加します。

```
<service android:name="com.worklight.wlclient.push.GCMIntentService" />

<receiver android:name="com.worklight.wlclient.push.WLBroadcastReceiver" android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.worklight.androidnativepush" />
    </intent-filter>

    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.worklight.androidnativepush" />
    </intent-filter>
</receiver>
```

アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

ネイティブ・プッシュ通知 API

- 最初のステップとして、WLClient クラスのインスタンスを作成します。

```
final WLClient client = WLClient.createInstance(this);
```

- WLPush クラスからのすべてのプッシュ通知操作を実行します。
  - getPush – このメソッドを使用して、WLPush クラスのインスタンスを WLClient インスタンスから取得します。
    WLPush push = WLClient.getPush();
  - WLOnReadyToSubscribeListener – Worklight Server への接続時に、アプリケーションは GCM サーバーにそのアプリケーション自体を登録してプッシュ通知の受信を試行します。

```
client.getPush().setOnReadyToSubscribeListener(listener);
client.connect(listener);
```

  - 登録が完了すると、WLOnReadyToSubscribeListener の onReadyToSubscribe メソッドが呼び出されます。

```
@Override
public void onReadyToSubscribe() {
    WLClient.getInstance().getPush().registerEventSourceCallback("myAndroid", "PushAdapter","PushEventSource", this );
```

WLPush.registerEventSourceCallback

- WLPush.registerEventSourceCallback メソッドを使用して、特定のイベント・ソースに別名を登録します。

```
WLClient.getInstance().getPush().registerEventSourceCallback("myAndroid", "PushAdapter","PushEventSource", this );
```

- このステートメントは以下のパラメーターを使用します。
  - myAndroid は別名です。
  - PushEventSource は別名が呼び出すイベント・ソースです。
  - PushAdapter はイベント・ソースが定義されているアダプターです。
- 通常、このメソッドは onReadyToSubscribe コールバック関数で呼び出されます。

その他の WLPush メソッド

- **isPushSupported()**
  - プッシュ通知がデバイスでサポートされているかどうかを示します。

```
WLClient client = WLClient.getInstance();
boolean supported = client.getPush().isPushSupported();
```

- **isSubscribed()**
  - デバイスがプッシュ通知にサブスクライブしているかどうかを示します。

```
WLClient client = WLClient.getInstance();
boolean bIsSubscribed = client.getPush().isSubscribed("myAndroid");
```

アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

Android アクティビティーのライフサイクル・メソッドのオーバーライド

- アクティビティー・クラスで、Android アクティビティーのライフサイクルを定義するメソッドを以下のようにオーバーライドします。
  - **onPause()** は、アプリケーションが一時停止した際に通知を通知バーで受信するように、WLPush インスタンスの setForeground(false) メソッドを呼び出す必要があります。

```
@Override
protected void onPause() {
    super.onPause();
    if (push != null)
        push.setForeground(false);
}
```

  - **onResume()** は、通知をアプリケーションのコールバックで受信するように、WLPush インスタンスの setForeground(true) メソッドを呼び出す必要があります。

```
@Override
protected void onResume() {
    super.onResume();
    if (push != null)
        push.setForeground(true);
}
```

  - **onDestroy()** は、アプリケーションの終了時に受信側からリーク例外が回避されるように、WLPush インスタンスの unregisterReceivers メソッドを呼び出す必要があります。

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (push != null)
    push.unregisterReceivers();
}
```

アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

プッシュ通知へのサブスクライブ

- **WLPush.subscribe(alias, pushOptions, responseListener)**
  - alias – デバイスがサブスクライブする必要がある別名。
  - pushOptions – タイプ WLPushOptions のオブジェクト。
  - responseListener – サブスクライブの完了時に呼び出されるタイプ WLResponseListener のオブジェクト。

```
WLClient client = WLClient.getInstance();
client.getPush().subscribe("myAndroid",new WLPushOptions(), new MyListener(MyListener.MODE_SUBSCRIBE));
```

- **MyListener.java**
  - WLResponseListener を実装します。
  - onSuccess – サブスクライブが成功した場合に呼び出されます。
  - onFailure – サブスクライブが失敗した場合に呼び出されます。

プッシュ通知からのアンサブスクライブ

- **WLPush.unsubscribe(alias, responseListener)**
  - alias – デバイスがサブスクライブしている別名。
  - responseListener – サブスクライブの完了時に呼び出されるタイプ WLResponseListener のオブジェクト。

```
WLClient client = WLClient.getInstance();
client.getPush().unsubscribe("myAndroid",new MyListener(MyListener.MODE_UNSUBSCRIBE));
```

- **MyListener.java**
  - WLResponseListener を実装します。
  - onSuccess – アンサブスクライブが成功した場合に呼び出されます。
  - onFailure – アンサブスクライブが失敗した場合に呼び出されます。

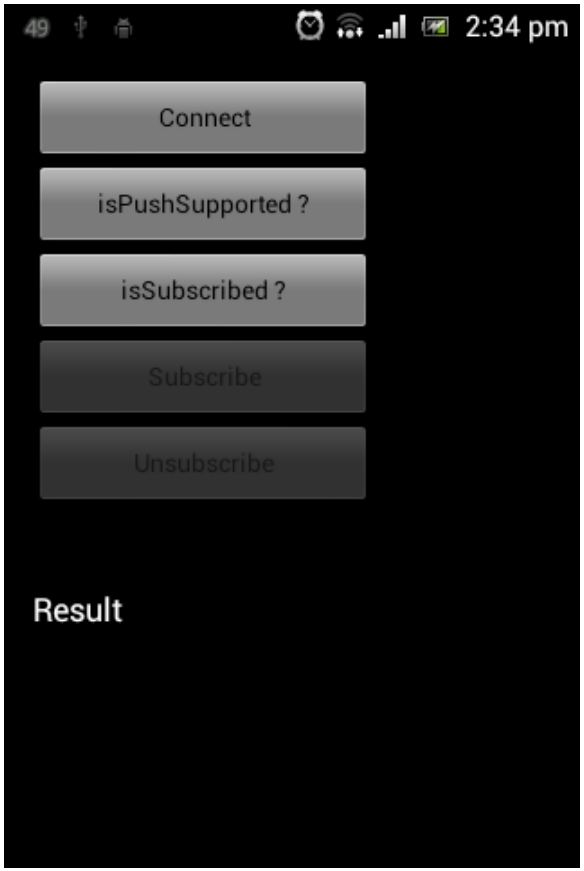
アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

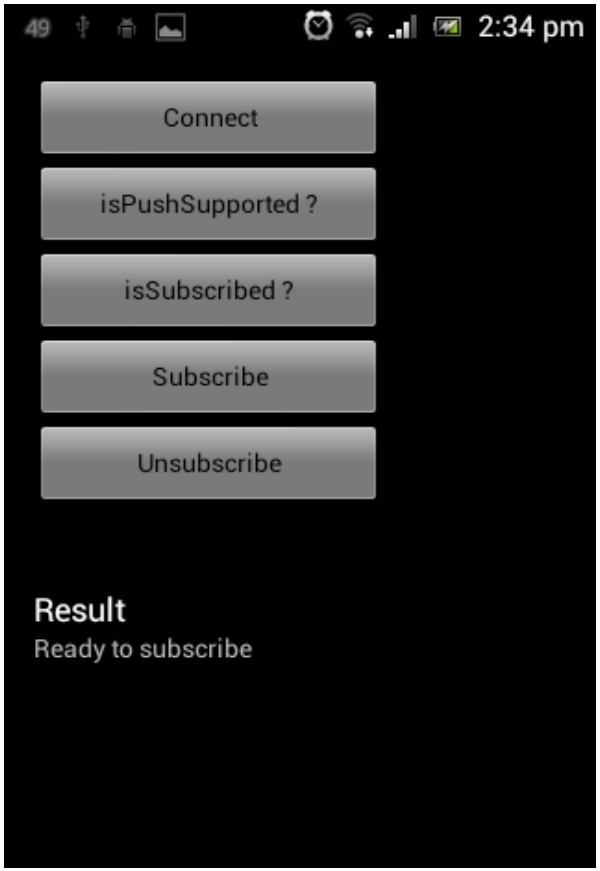
Android プロジェクトの実行

- Android プロジェクトを右クリックし、**「実行 (Run As)」> 「Android アプリケーション (Android Application)」**を選択します。

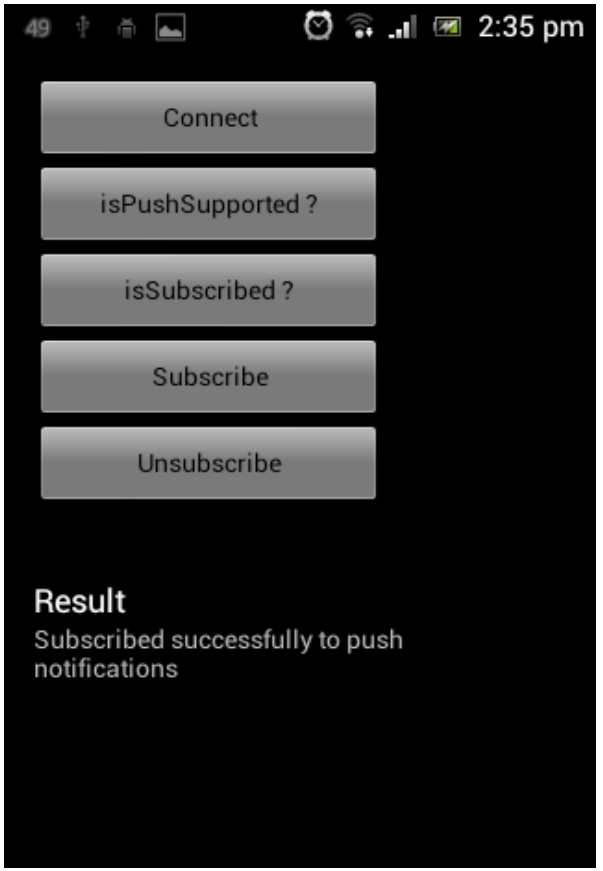
「接続 (Connect)」をクリックします。



onReadyToSubscribe リスナーが呼び出されるまで待機します。



次に、**「サブスクライブ (Subscribe)」**をクリックします。



アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

プッシュ通知の送信 (1/2)

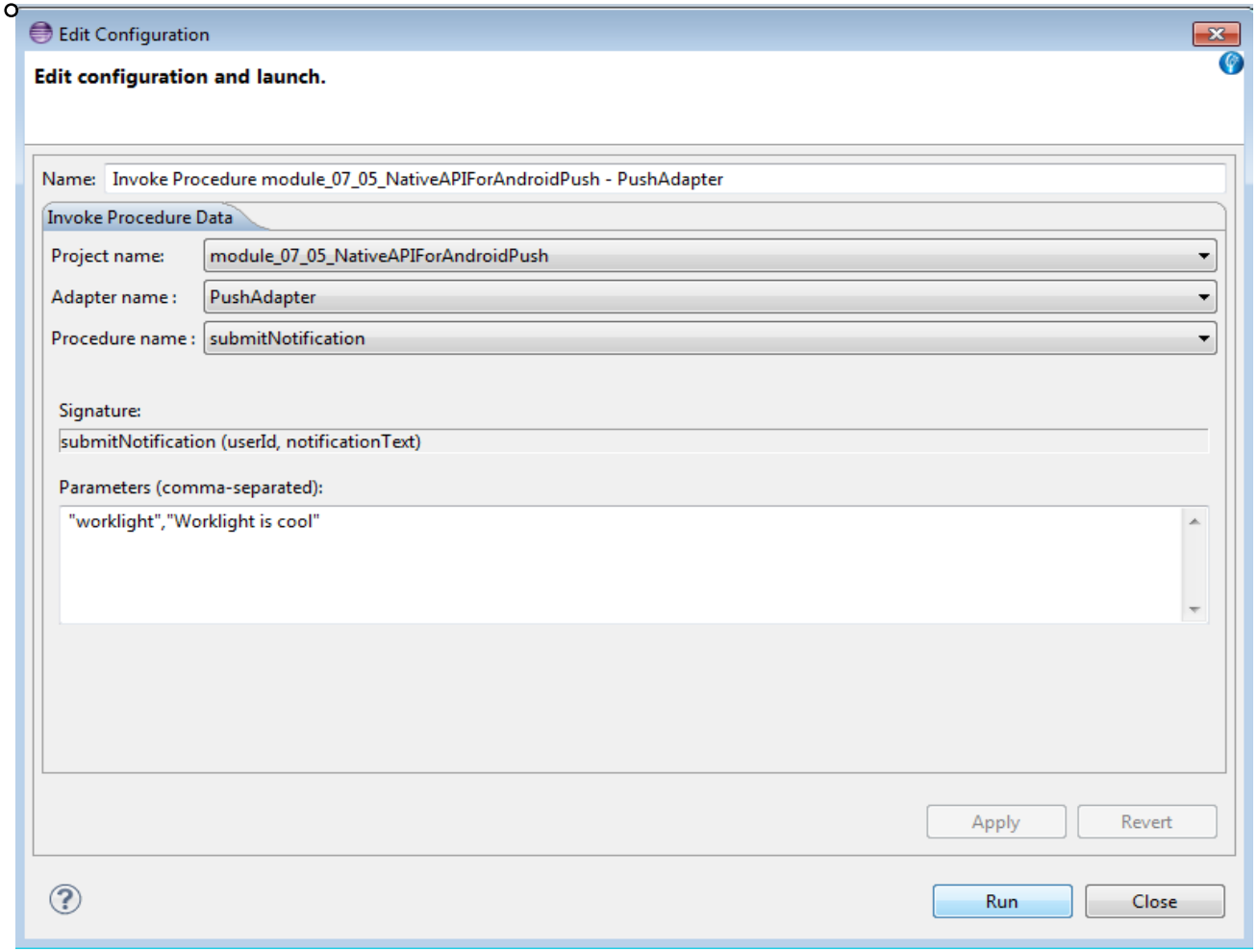
- `PushAdapter` には、`submitNotification()` プロシージャーが含まれます。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for user :: " + userId };
    }

    WL.Logger.debug("submitNotification >> userId :: " + userId + ", text :: " + notificationText);

    WL.Server.notifyAllDevices(userSubscription, {
        badge: 1,
        sound: "sound.mp3",
        activateButtonLabel: "ClickMe",
        alert: notificationText,
        payload: {
            foo : 'bar'
        }
    });

    return { result: "Notification sent to user :: " + userId };
}
```

- アダプターを右クリックして、**「実行 (Run As)」>「Worklight プロシージャーの呼び出し (Invoke Worklight Procedure)」**を選択します。

プッシュ通知の送信 (2/2)

- `submitNotification` を呼び出して、プッシュ通知を送信します



プッシュ通知の受信 (1/3)

- プッシュ通知を受信すると、`onReceive` メソッドが `WLEventSourceListener` インスタンスで呼び出されます。

```
public class MyListener implements WLOnReadyToSubscribeListener, WLResponseListener, WLEventSourceListener{
```

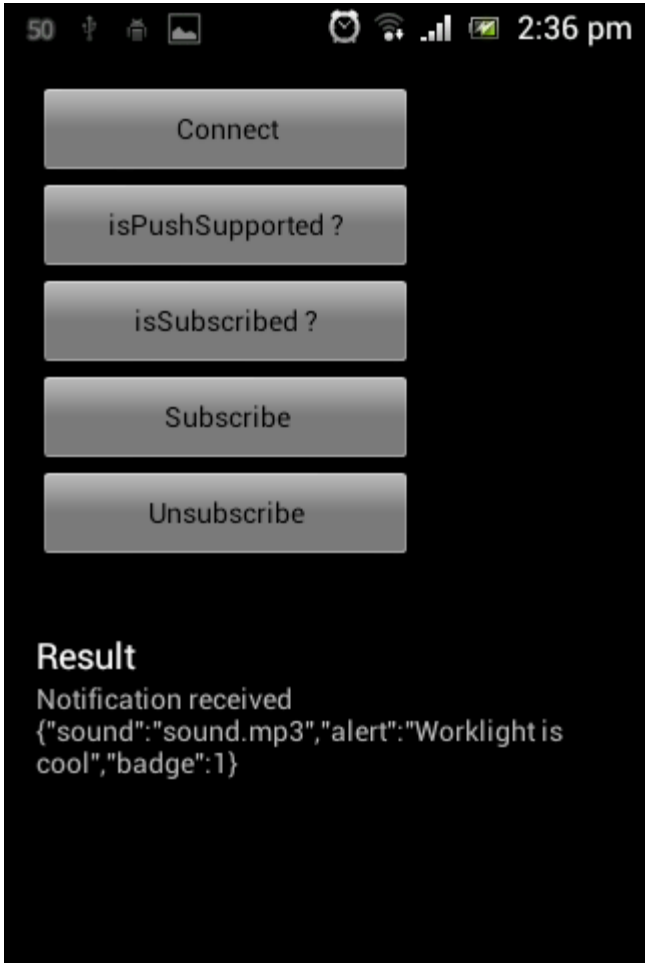
```
@Override
public void onReceive(String arg0, String arg1) {
    AndroidNativePush.updateTextView("Notification received  " + arg0);
}
```

- `WLEventSourceListener` インスタンスが、`registerEventSourceCallback` コールバック中に登録されます。

```
WLClient.getInstance().getPush().registerEventSourceCallback("myAndroid", "PushAdapter","PushEventSource", this );
```

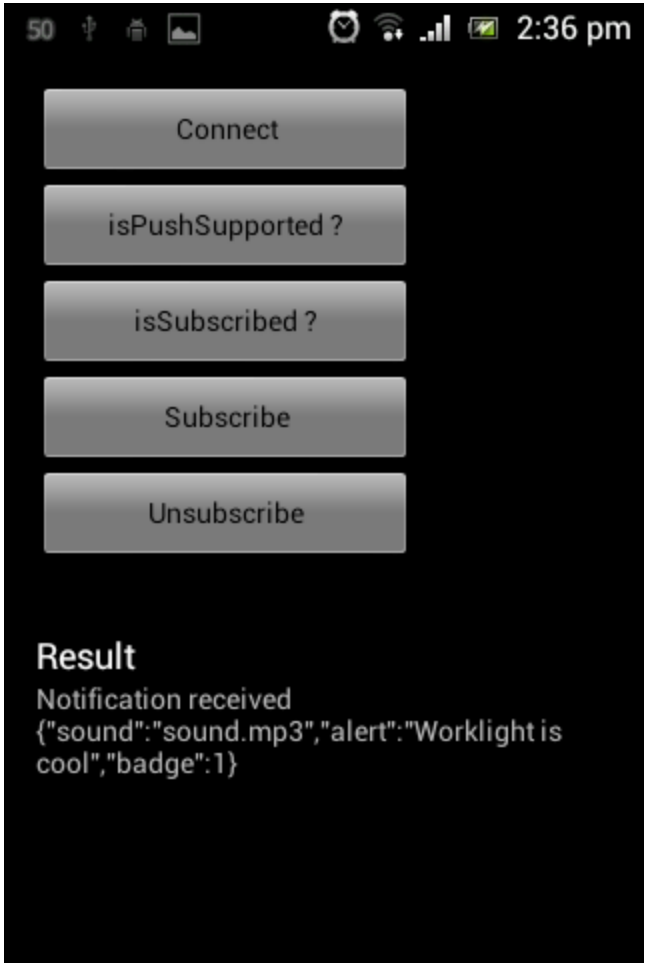
プッシュ通知の受信 (2/3)

- `onReceive` メソッドにより、受信した通知が画面に表示されます。
- アプリケーションが実行中でない場合、通知アイコンが、画面の上部にある通知バーに表示されます。



プッシュ通知の受信 (3/3)

- このトレーニング・モジュールのサンプルは、IBM Worklight 文書 Web サイト (http://www.ibm.com/mobile-docs) の「入門」ページにあります。
- このサンプルには以下の 2 つのプロジェクトが含まれます。
  - **PushNotificationsNative.zip** には、Worklight Server にデプロイする Worklight ネイティブ API が含まれます。
  - **AndroidNativePush.zip** には、Worklight ネイティブ API ライブラリーを使用して Worklight Server と通信するネイティブ Android アプリケーションが含まれます。



アジェンダ

- Worklight プロジェクトのセットアップ
- プッシュ通知用のネイティブ・アプリケーションのセットアップ
- ネイティブ・プッシュ通知 API
- Android アクティビティー・ライフサイクルのメソッドのオーバーライド
- プッシュ通知へのサブスクライブとプッシュ通知からのアンサブスクライブ
- Android プロジェクトの実行
- プッシュ通知の送信および受信
- タグ・ベース通知とブロードキャスト通知

タグ・ベース通知

- タグはユーザーが関心のあるトピックを表し、選択した関心に従って通知を受けられる機能を提供します。
- この通知タイプは、タグによりフィルタリングされたメッセージをデバイスが送信および受信できるようにします。
- タグ・ベース通知の受信を開始するには、デバイスがまずアプリケーションでプッシュ通知タグにサブスクライブする必要があります。
- タグは *application-descriptor.xml ファイル*で次のように定義されています。

```
<tags>
    <tag>
        <name>PushTag1</name>
        <description>About PushTag1</description>
    </tag>
    <tag>
        <name>PushTag2</name>
        <description>About PushTag2</description>
    </tag>
</tags>
```

- この通知は、アプリケーションでタグにサブスクライブしているすべてのデバイスをターゲットにしています。

タグ・ベース通知

- クライアント・サイド API:
  - `WLPush.subscribeTag(tagName,options)`

    指定されたタグ名にデバイスをサブスクライブします。
  - `WLPush.unsubscribeTag(tagName,options)`

    指定されたタグ名からデバイスをアンサブスクライブします。
  - `WLPush.isTagSubscribed(tagName)`

    指定されたタグ名にデバイスがサブスクライブされているかどうかを返します。

タグ・ベース通知

タグ・ベース通知についての詳細は、「IBM Worklight Foundation ユーザー文書」の「タグ・ベース通知」を参照してください。

ブロードキャスト通知

- ブロードキャスト通知は、プッシュが使用可能な Worklight アプリケーションではどれでも、デフォルトで使用可能になっています。予約されたタグ `Push.ALL` へのサブスクリプションが、すべてのデバイスに対して作成されます。
- ブロードキャスト通知は、予約されたタグ `Push.ALL` からデバイスをアンサブスクライブすることにより、使用不可にできます。

ブロードキャスト通知

ブロードキャスト通知についての詳細は、「IBM Worklight Foundation ユーザー文書」の「ブロードキャスト通知」を参照してください。

タグ・ベース通知とブロードキャスト通知の共通 API (1/2)

- クライアント・サイド API:
  - *WLNotificationListener* は、通知の受信時に通知されるコールバック・メソッドを定義します。

```
client.getPush().setWLNotificationListener(listener)
```

  - デバイスがプッシュ通知を受信したときに *WLNotificationListener* の `onMessage(props,payload)` メソッドが呼び出されます。
    - `props` - プラットフォームの通知プロパティーを含む JSON ブロック。
    - `payload` - Worklight Server から送信されるその他のデータを含む JSON ブロック。このブロックは、タグ通知およびブロードキャスト通知のタグ名も含んでいます。タグ名は `<tag>` エレメントに表示されます。ブロードキャスト通知では、デフォルトのタグ名は `Push.ALL` です。

タグ・ベース通知とブロードキャスト通知の共通 API (2/2)

- サーバー・サイド API:
  - `WL.Server.sendMessage(applicationId,notificationOptions)`
    このメソッドは、以下の 2 つの必須パラメーターを使用します。
    - `applicationId` - Worklight アプリケーションの名前。
    - `notificationOptions` - メッセージ・プロパティーを含む JSON ブロック。

  指定されたターゲット・パラメーターに基づいた通知を送信します。
  - メッセージ・プロパティーの完全なリストは、「IBM Worklight Foundation ユーザー文書」を参照してください。

特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました