# MobFox SDK Integration Guide
## Version 3.2 (i 1.5)

**If any questions arise during integration, don't hesitate to e-mail us at support@mobfox.com**

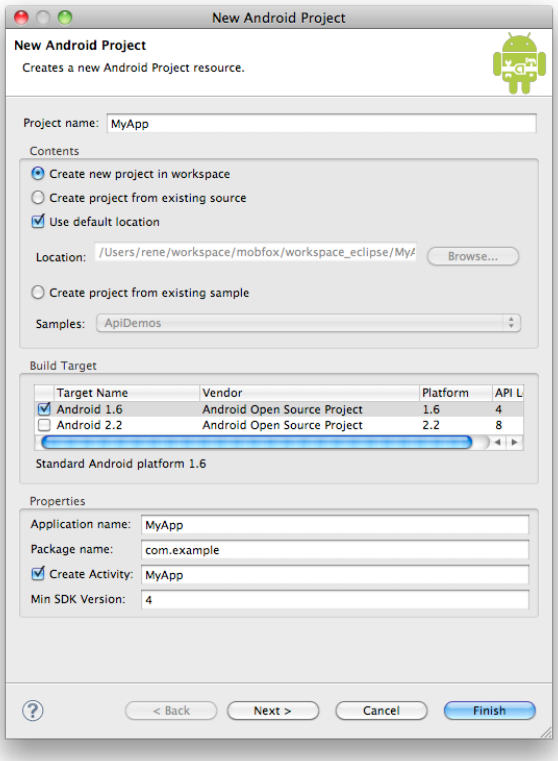
## 1. Quick Integration

**1. Start Eclipse**

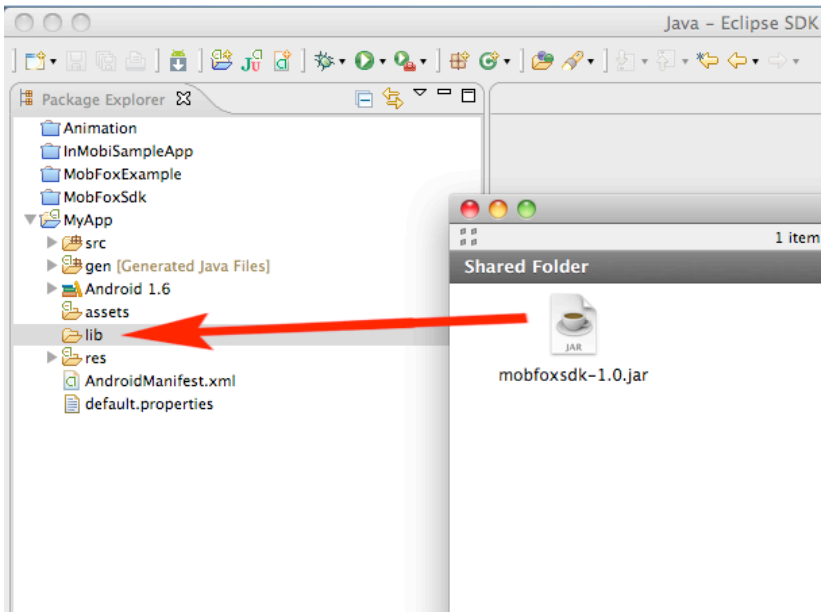
**2. Create a new Android Project:**

File -> New -> Android Project

**3. Enter your Project Name, select the Android Version (minimum is 1.6), and enter the Application Properties:**



4. Now create a new **source folder** in your Project called 'lib'. Then drag the mobfoxsdk-1.5.jar into the 'lib' folder.




*Also add the MobFox library to your java build path! (under Project properties)*

### 5. Open the AndroidManifest.xml and add the following permissions:

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

**Optionally, you can also add the following permission:**

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
```

## Why our SDK needs those permissions:

**1. INTERNET**
This permission is obviously needed to connect to our ad-server and fetch an ad.

**2. READ_PHONE_STATE**
Used to access a unique identifier. This Permission is **Optional** and the phone's Android ID will automatically be used for unique identification in case this permission is not added.

### 6. Now, add the following activity:

```
<activity android:name="com.mobfox.sdk.InAppWebView"/>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.mobfox.example"
      android:versionCode="1"
      android:versionName="1.0">

    <uses-sdk android:minSdkVersion="4" />

    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<!--    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission> -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>

    <application android:icon="@drawable/icon" android:label="@string/app_name" android:debuggable="true">
        <activity android:name=".MobFoxExample"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.mobfox.sdk.InAppWebView"/>
    </application>
</manifest>
```

## 7. Open the layout/main.xml and include the MobFoxView:

```
<com.mobfox.sdk.MobFoxView
        android:id="@+id/mobFoxView"
        android:layout_width="fill_parent"
        android:layout_height="50dp"
        publisherId="<enter your publisher id here>"
/>
```

## Info

MobFox ads are 50 device independent pixels high (dip or dp). By defining the ad's size in dip, the ad will be scaled properly across the different Android screen sizes.

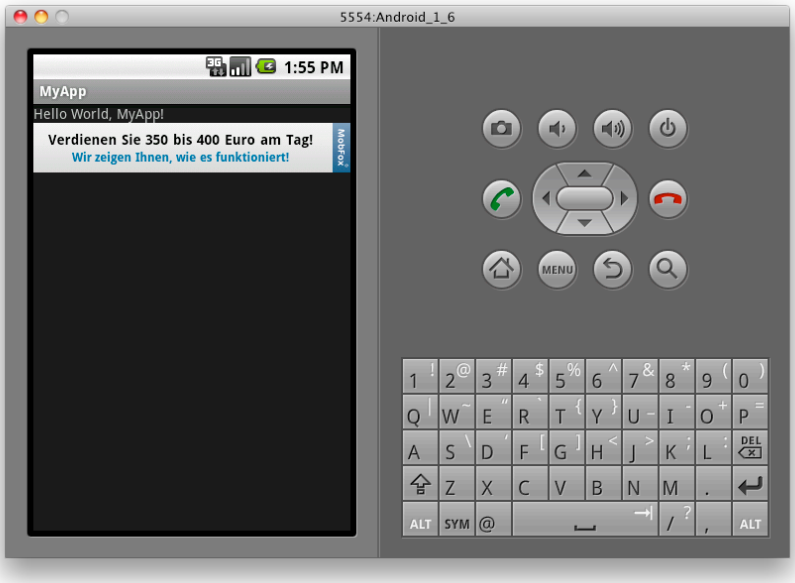
main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />


<com.mobfox.sdk.MobFoxView
    android:id="@+id/mobFoxView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    publisherId="f9205cd278ce092837a01cc4ca40af6a"
/>
</LinearLayout>
```

**That's all!**

Now, run your application and you should see a MobFox Test-Banner!



### All Options for the MobFoxView

```
<com.mobfox.sdk.MobFoxView
        android:id="@+id/mobFoxView"
        android:layout_width="fill_parent"
        android:layout_height="50dp"
        publisherId="<enter your publisher id here>"
        mode="test/live"
        includeLocation="true/false"
        animation="true/false"/>
```

- ***publisherId***: your publisher Id provided by MobFox (http://account.mobfox.com)
- ***mode: test*** for test mode, and ***live*** for live mode
- ***includeLocation: true*** the location is included in the ad request (even better eCPM!)

  If set to true the following permission must be added to the AndroidManifest.xml:

  ```
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
  ```

- **animation**: **true** the banners are animated (fade-in and refresh animations)

### The MobFoxView can also be created dynamically in your view with the following code:

```
String publisherId = "...";
boolean includeLocation = false;
boolean animation = true;
MobFoxView mobfoxView = new MobFoxView(this, publisherId, includeLocation, animation);
layout.addView(mobfoxView);
```

**or**

```
String publisherId = "...";
boolean includeLocation = false;
Mode mode = Mode.Test;
boolean animation = true;
MobFoxView mobfoxView = new MobFoxView(this, publisherId, test, includeLocation, animation);
```

### The MobFoxView also needs to know when the activity is paused and resumed, so override the onPause and onResume in your Activity class an call the proper methods on the MobFoxView object:

```
@Override
protected void onPause() {
        super.onPause();
        mobfoxView.pause();
}

@Override
protected void onResume() {
        super.onResume();
        mobfoxView.resume();
}
```

**If you want to avoid that an ad request is performed when the phone is rotated and the orientation changes then you have to add following to your app:**

**Insert the following attribute to the Activity's node in the AndroidManifest.xml**
android:configChanges="keyboardHidden|orientation"

**Then override the following method in your Activity and pause and resume the MobFoxView:**

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
  super.onConfigurationChanged(newConfig);
  mobfoxView.pause();
  mobfoxView.resume();
}
```

**If no ad is available or the ad request has failed a notification is sent from MobFoxView to the registered BannerListener:** (noAdFound means that there is currently no ad available for the ad request – you can try to request an ad from a secondary network if this happens)

```
MobFoxView mobfoxView = new MobFoxView(this, "...", false);
mobfoxView.setBannerListener(new BannerListener() {

        @Override
        public void bannerLoadFailed(RequestException cause) {
                // do something
        }

        @Override
        public void noAdFound() {
                // do something
        }
});
```

**Everytime a new ad is received, the registered BannerListener will receive a bannerLoadSucceeded notification from MobFoxView. You can use this notification to show the view only after receiving an ad.**

```
MobFoxView mobfoxView = new MobFoxView(this, "...", false); mobfoxView.setBannerListener(new
BannerListener() {
@Override public void bannerLoadSucceeded() {
// do something
}
@Override public void bannerLoadFailed(RequestException cause) {
// do something
}
@Override public void noAdFound() {
// do something
}
});
```