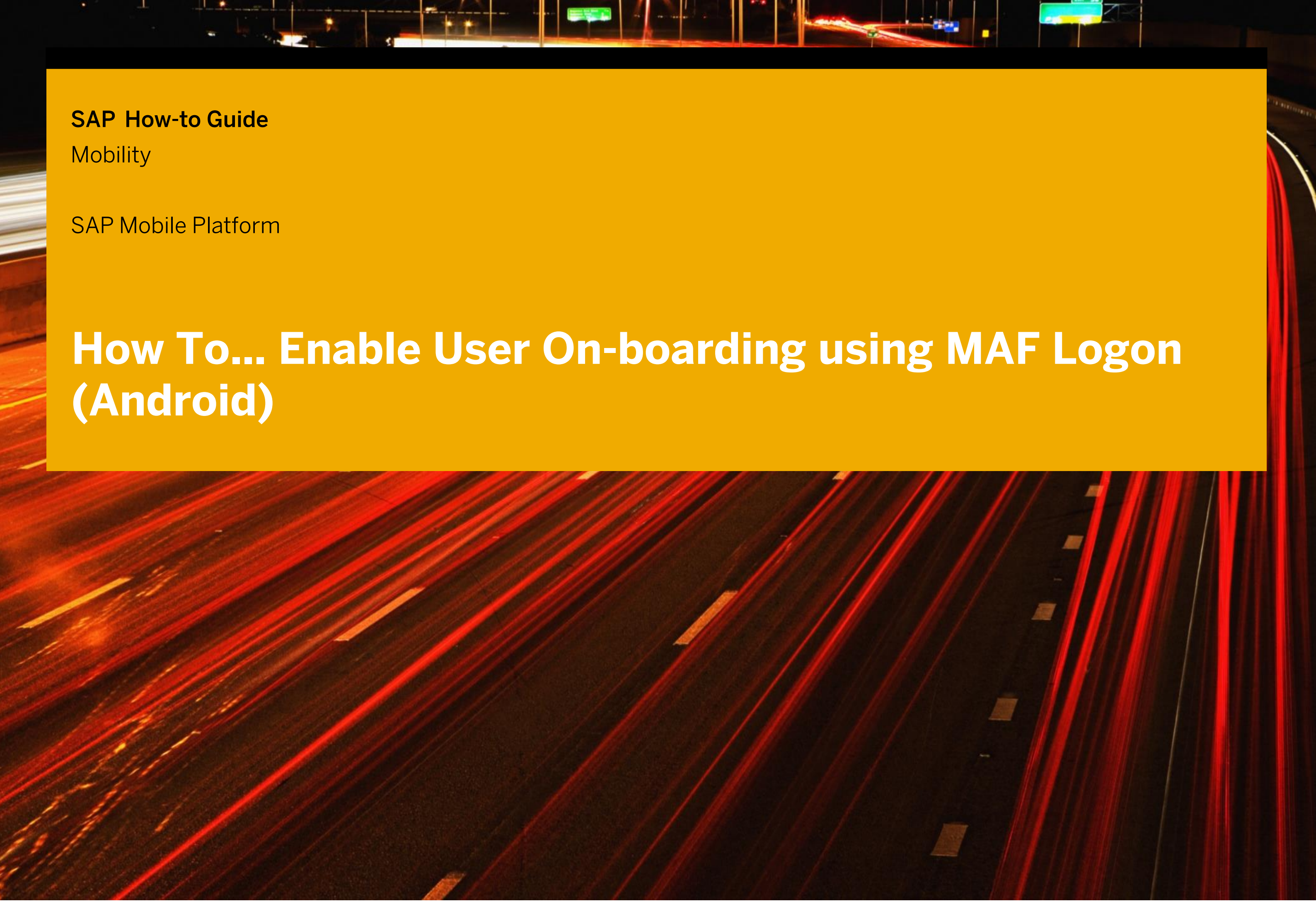


SAP How-to Guide

Mobility

SAP Mobile Platform

How To... Enable User On-boarding using MAF Logon (Android)

Applicable Releases:

SAP Mobile Platform 3.0 SP04

SAP Mobile SDK 3.0 SP05

Version 1.0

The Best-Run Businesses Run SAP™

This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice.

SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.

SAP “How-to” Guides are intended to simplify the product implementtation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.

The Best-Run Businesses Run SAP™

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Icons

Icon	Description
⚠	Caution
💡	Note or Important
✤	Example
⬆	Recommendation or Tip

Table of Contents

1. Business Scenario

Before any communication with the SAP backend systems can take place at all, the app needs to on-board users onto the SAP Mobile Platform. The Mobile Application Framework (MAF) contains the MAF logon component that provides easy integration for applications that use logon UI behavior.

2. Background Information

The goal of this exercise is not to show how to create a project from scratch and dissect every line of code. Instead, it shows the key pieces of code and information needed to integrate MAF Logon component to a new or an existing Android Project in eclipse.

The purpose is to explain how developers can leverage the MAF Logon component to on-board users from scratch.

3. Prerequisites

This exercise has the following prerequisites:

- Java Standard Edition 7
- Eclipse Kepler
- Android SDK with API level 8 and 19 downloaded
- Android ATD Eclipse plugin installed
- To get the most out of this exercise, some experience with Java is recommended.
- SAP Mobile Platform 3.0 SP04

  ⚠ This example assumes you have configured an application in SMP 3.0 called com.sap.flight for more information on how to create an application configuration, please visit Deploying Applications
- SAP Mobile SDK 3.0 SP05

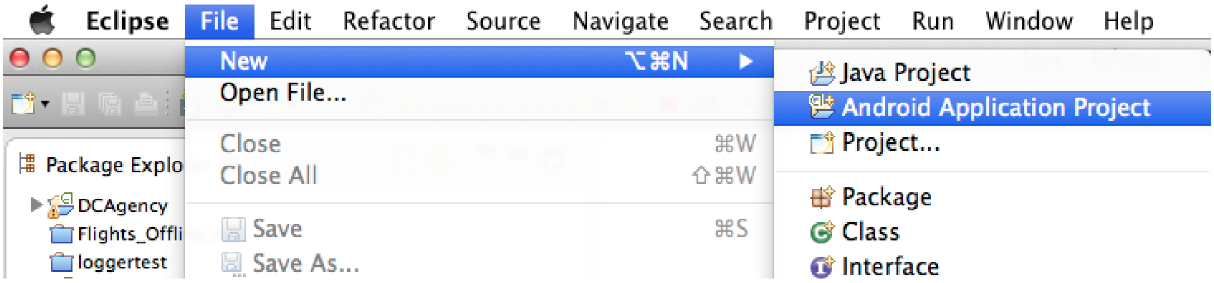
4. Step-by-Step Procedure

The following sections provide a detailed step-by-step procedure on how to code an Android app to on-board/register a user onto the SAP Mobile Platform using the MAF Logon component.

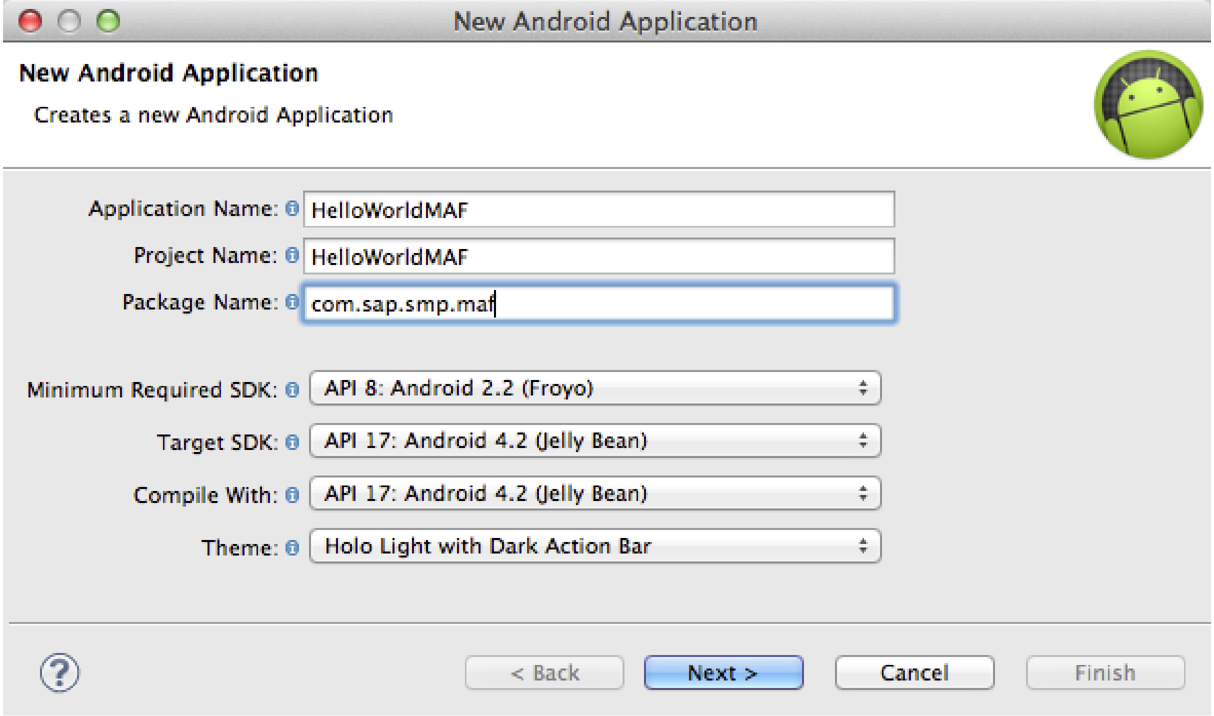
The following section is optional, you can reuse an existing android project for the purpose of this exercise

4.1 Create Android Project

1. Go to File -> New -> Android Application Project



2. Enter the following information to create the android project and click the Next button



Application Name: HelloWorldMAF

Project Name: HelloWorldMAF

Package Name: com.sap.smp.maf

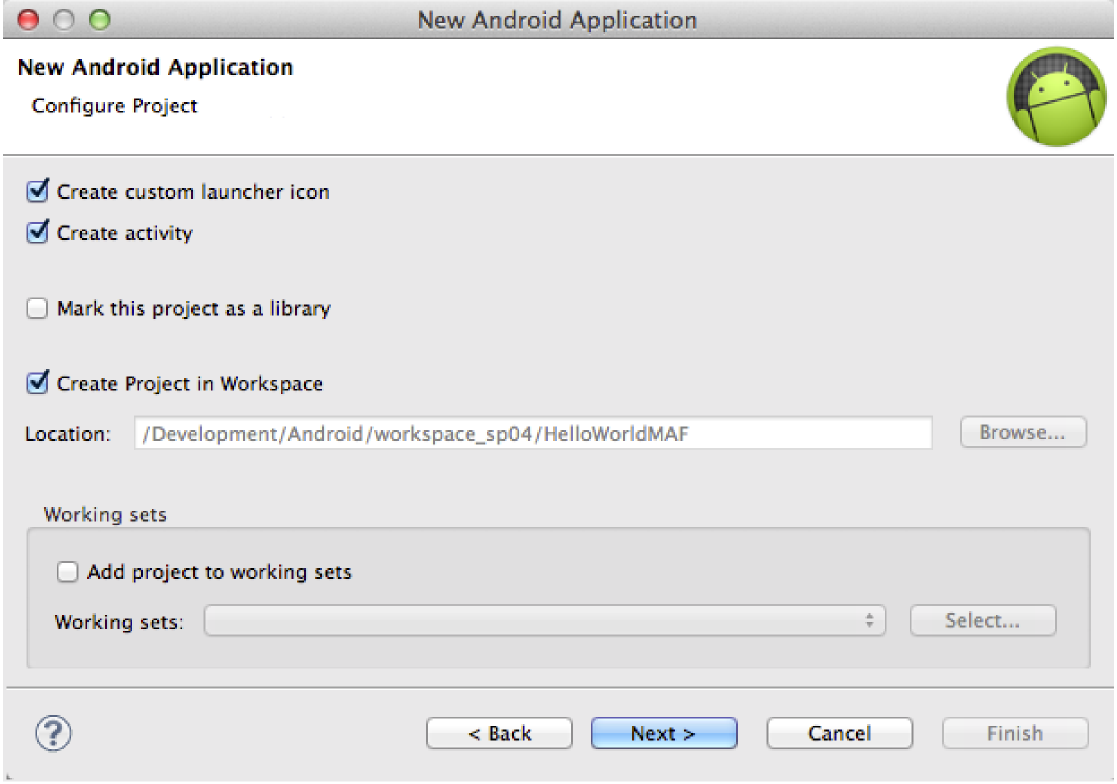
Minimum Required SDK: API 8: Android 2.2 (Froyo)

Target SDK: API 17: Android 4.2 (Jelly Bean)

Compile With: API 17: Android 4.2 (Jelly Bean)

Theme: Holo Light with Dark Action Bar

3. Keep the default values as shown in the image and click Next.



New Android Application

New Android Application

Configure Project

☑ Create custom launcher icon

☑ Create activity

□ Mark this project as a library

☑ Create Project in Workspace

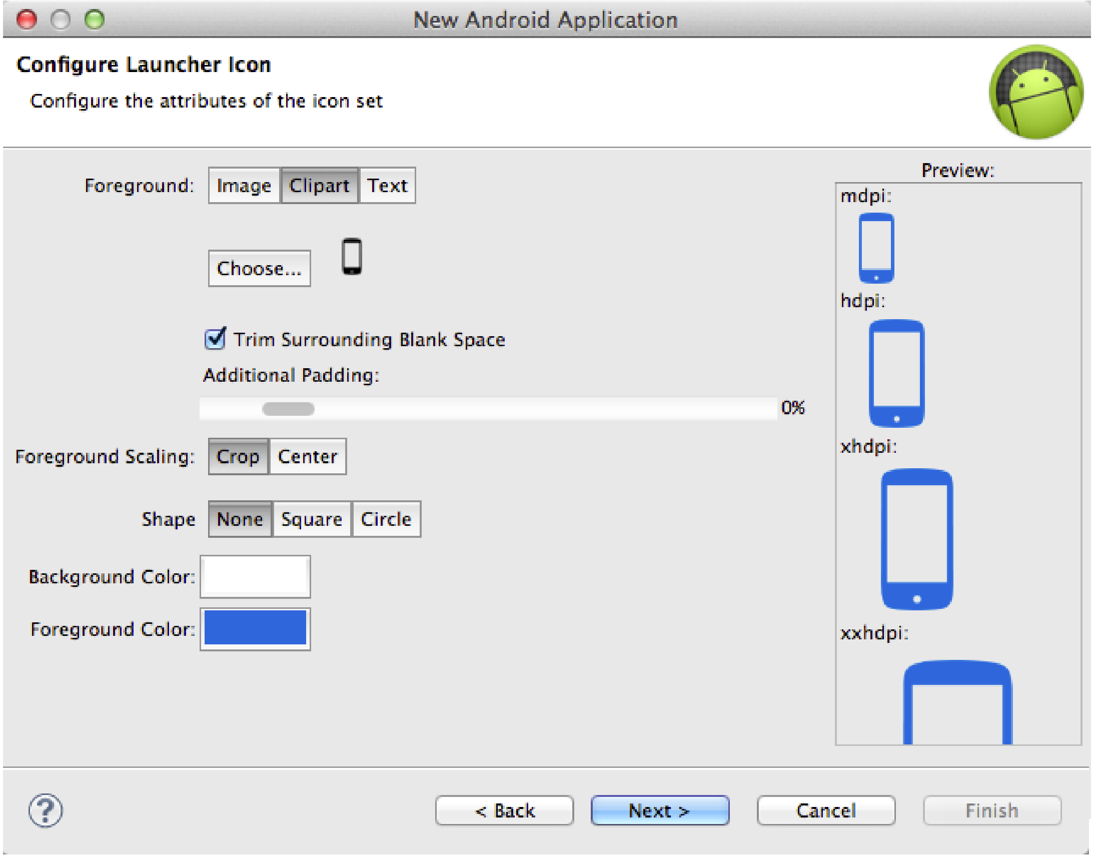
Location: /Development/Android/workspace_sp04/HelloWorldMAF

Working sets

□ Add project to working sets

Working sets:

4. Select any icon and click Next.



New Android Application

Configure Launcher Icon

Configure the attributes of the icon set

Foreground: Image [✓ Clipart] Text

☑ Trim Surrounding Blank Space

Additional Padding: 0%

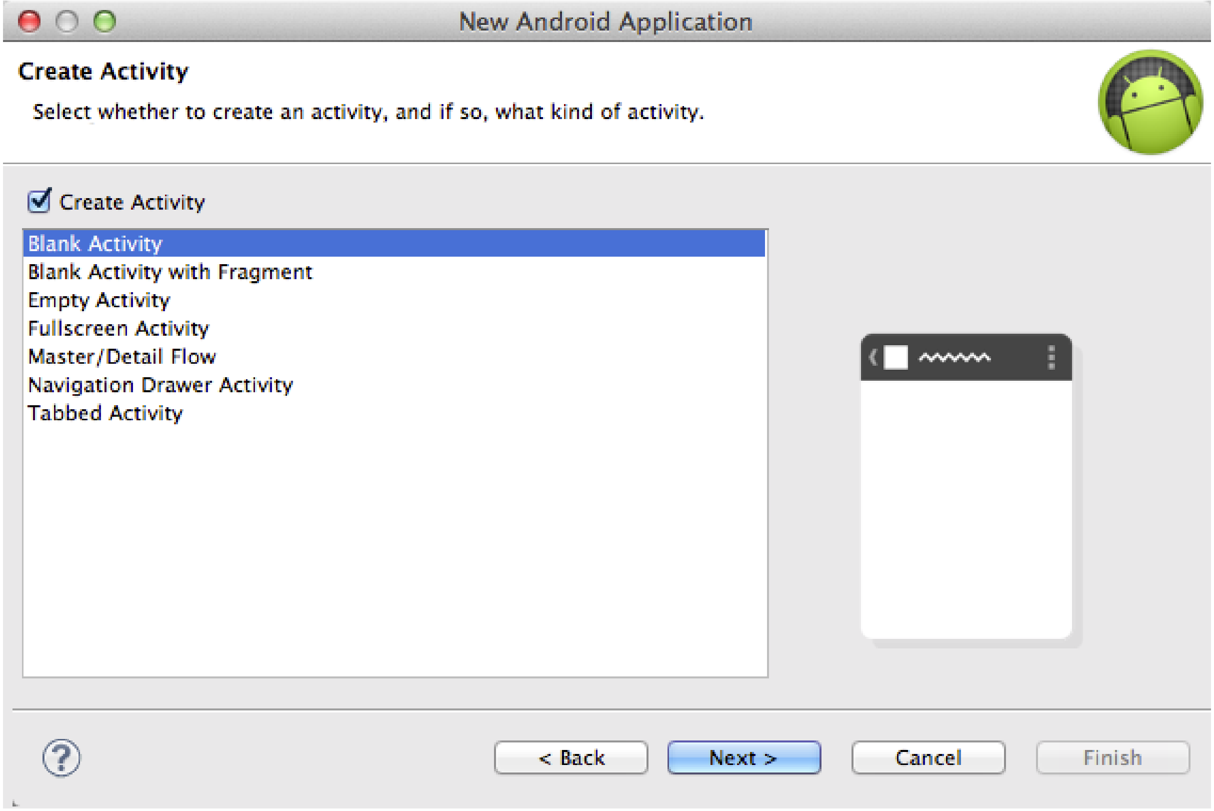
Foreground Scaling: [✓ Crop] Center

Shape [✓ None] Square Circle

Background Color:

Foreground Color:

5. You can choose any template to create an activity, for simplicity purposes we will chose a Blank Activity



New Android Application

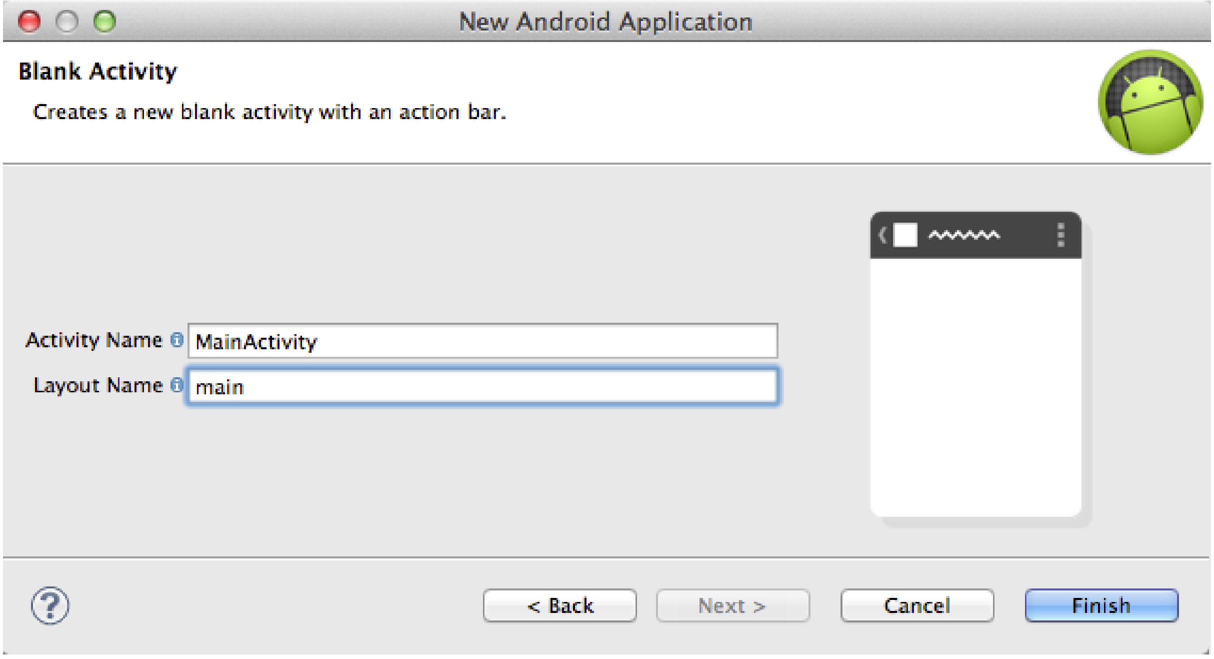
Create Activity

Select whether to create an activity, and if so, what kind of activity.

☑ Create Activity

- Blank Activity
- Blank Activity with Fragment
- Empty Activity
- Fullscreen Activity
- Master/Detail Flow
- Navigation Drawer Activity
- Tabbed Activity

6. Enter the following information to create the activity and its corresponding layout resources and click Finish



New Android Application

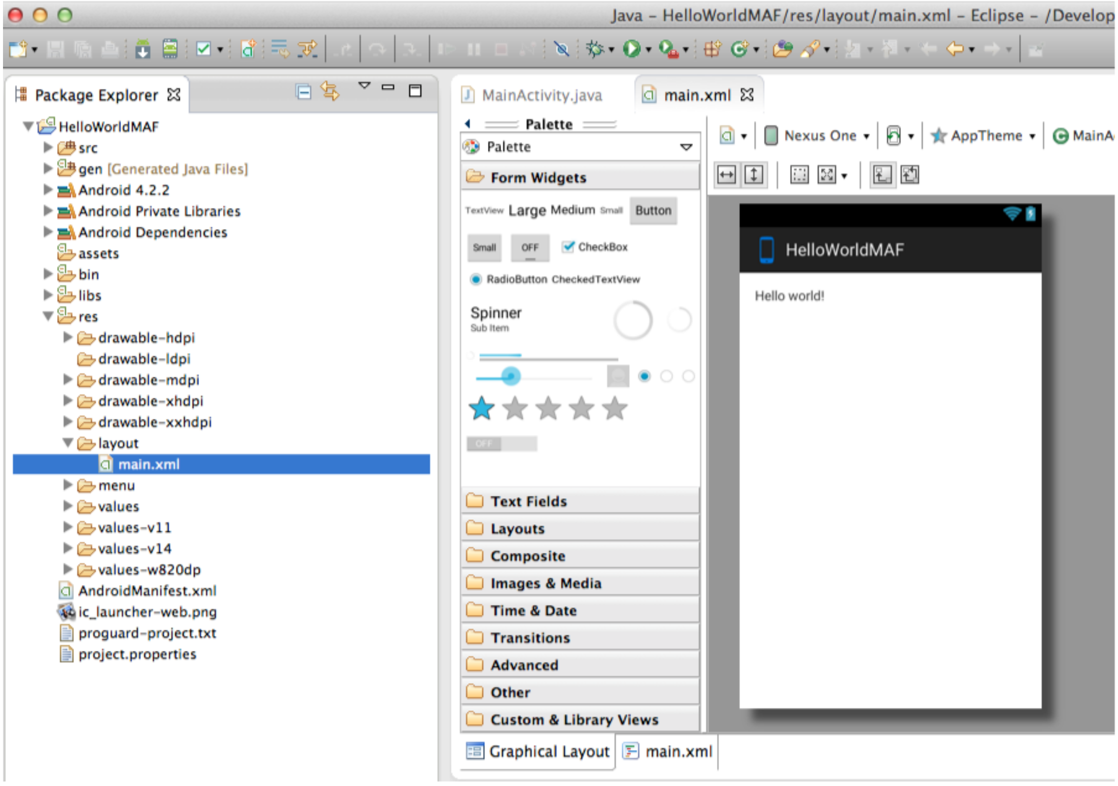
Blank Activity

Creates a new blank activity with an action bar.

Activity Name: MainActivity

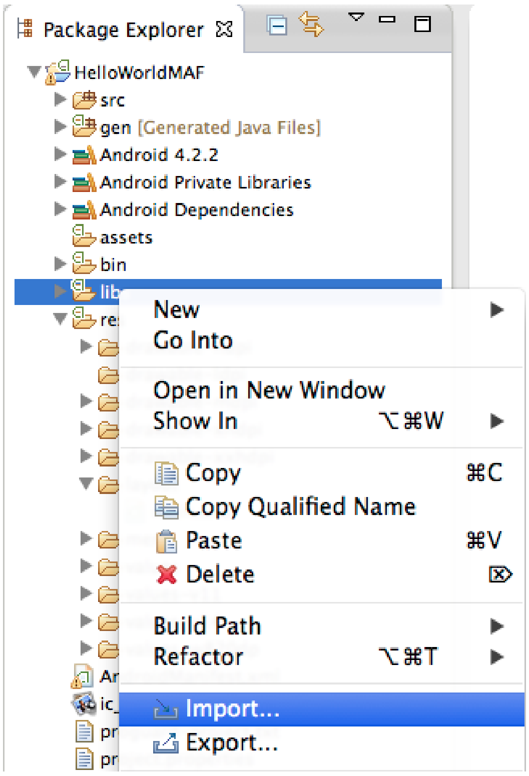
Layout Name: main

7. Your project should look like the image below.

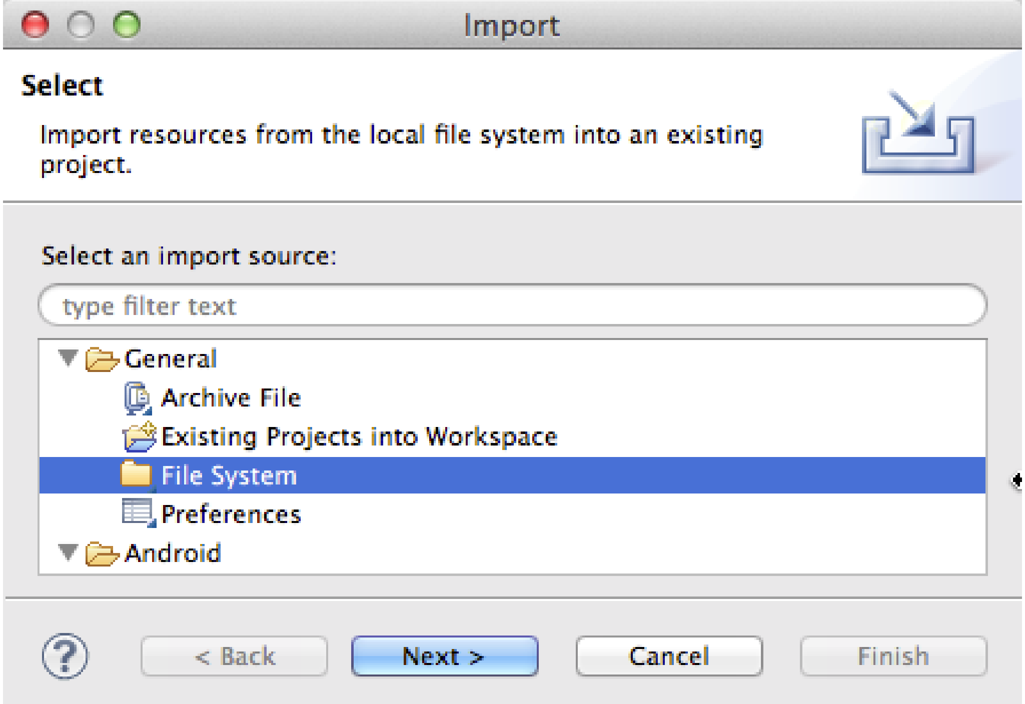


4.2 Add required MAF artifacts

1. Select the libs folder and right-click to open the context menu and select Import



2. Select General -> File System and click Next



Import

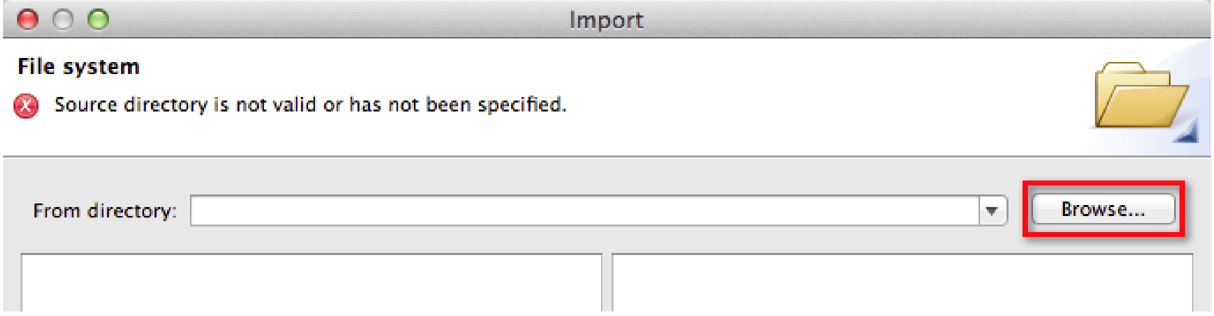
Select

Import resources from the local file system into an existing project.

Select an import source:

type filter text

3. In the Import window, click Browse



Import

File system

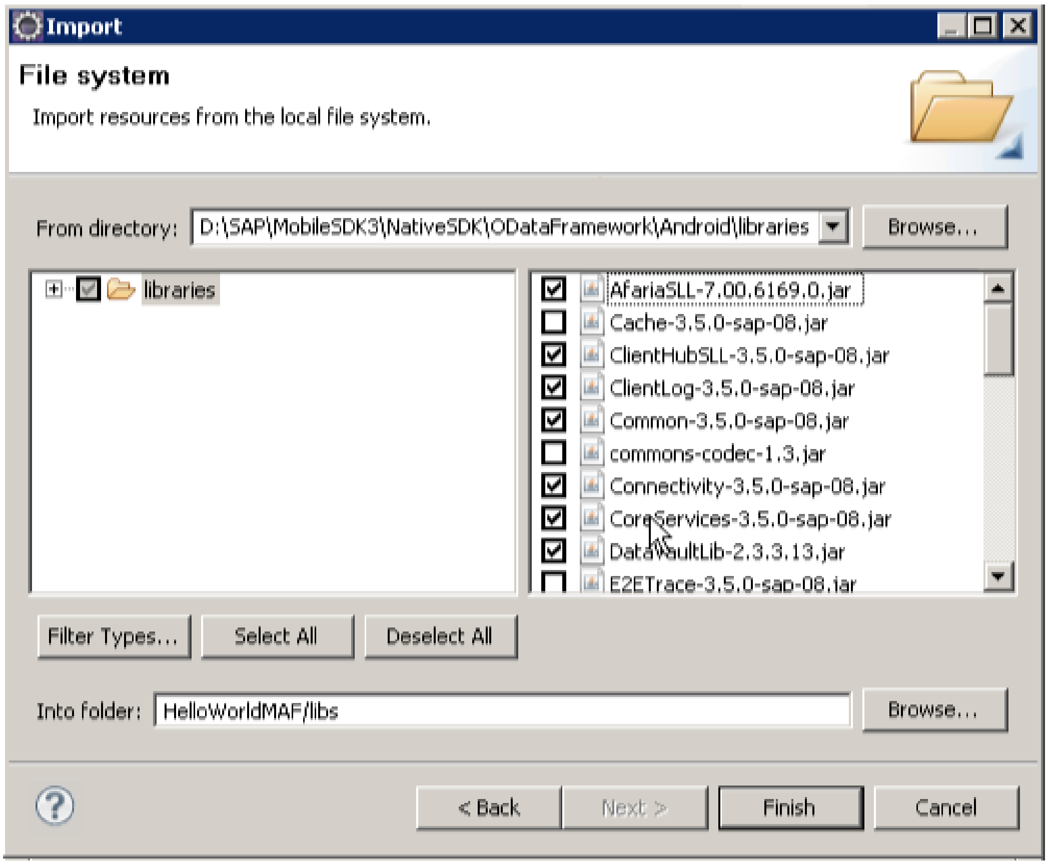
Source directory is not valid or has not been specified.

From directory:

4. Locate your SMP 3.0 SDK installation folder. You will find the jar files in the following folders:
   a. <Client SDK dir>\NativeSDK\ODataFramework\Android\libraries
   b. <Client SDK dir>\NativeSDK\MAFReuse\Android\libraries

5. Select the following jar and click Finish

- AfariaSLL.jar
- ClientHubSLL
- ClientLog.jar
- Common.jar
- Connectivity.jar
- CoreServices.jar
- DataVaultLib.jar
- HttpConvAuthFlows.jar
- HttpConversation.jar
- maflogoncore.jar
- maflogger.jar
- maflogonui.jar
- mafuicomponents.jar
- mafsettingscreen.jar
- MobilePlace.jar
- perflib.jar
- Request.jar
- sap-e2etrace.jar
- SupportabilityFacade.jar



Import

File system

Import resources from the local file system.

From directory: D:\SAP\MobileSDK3\NativeSDK\ODataFramework\Android\libraries

☑ libraries

- ☑ AfariaSLL-7.00.6169.0.jar
- □ Cache-3.5.0-sap-08.jar
- ☑ ClientHubSLL-3.5.0-sap-08.jar
- ☑ ClientLog-3.5.0-sap-08.jar
- ☑ Common-3.5.0-sap-08.jar
- □ commons-codec-1.3.jar
- ☑ Connectivity-3.5.0-sap-08.jar
- ☑ CoreServices-3.5.0-sap-08.jar
- ☑ DataVaultLib-2.3.3.13.jar
- □ E2ETrace-3.5.0-sap-08.jar

Into folder: HelloWorldMAF/libs

6. Save your changes

4.3 Set up dependencies

To use the MAF Login Component you must import a number of Libraries and Resources. MAF Libraries were installed with SMP Client SDK installer into the folder specified by you when you executed the installer. This libraries and resources are already included in the project. Please follow the steps to confirm the project was set up correctly.

MAF is shipped with various resources. These projects have Android nature, but don't ship any source code. Their only purpose is to ship all Android resource files like: images, layouts, localization, XMLs

As a result of the SDK installer you will find the following .apklib files in the installation folder:

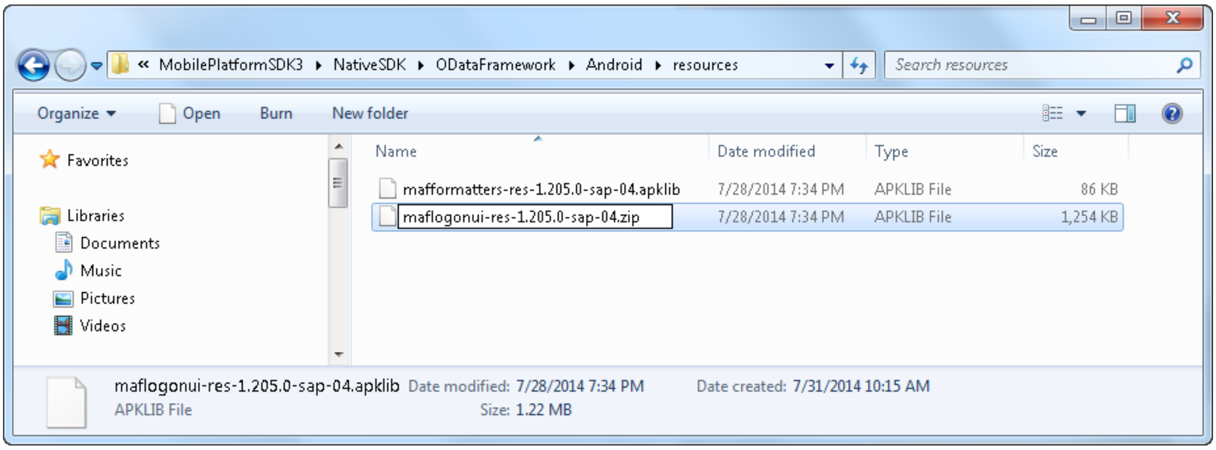
a. <Client SDK dir>\NativeSDK\ODataFramework\Android\resources\maflogonui.akplib

b. <Client SDK dir>\NativeSDK\MAFReuse\Android\resources\mafsettingscreen.apklib

c. <Client SDK dir>\NativeSDK\MAFReuse\Android\resources\mafuicomponents.apklib

1. Open folder <Client SDK dir>\NativeSDK\ODataFramework\Android\resources\

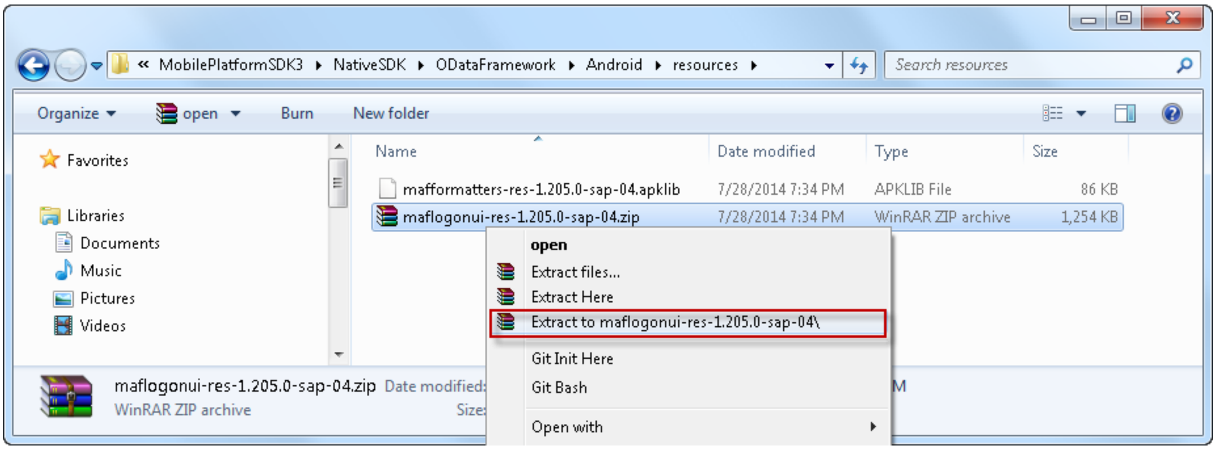
2. The resource archives have .apklib extension (i.e maflogonui-res-xxx.akplib). You will need to change this extension to .zip



Name	Date modified	Type	Size
mafformatters-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	86 KB
maflogonui-res-1.205.0-sap-04.zip	7/28/2014 7:34 PM	APKLIB File	1,254 KB

maflogonui-res-1.205.0-sap-04.apklib Date modified: 7/28/2014 7:34 PM Date created: 7/31/2014 10:15 AM
APKLIB File Size: 1.22 MB

3. Unzip the .zip file into a local folder



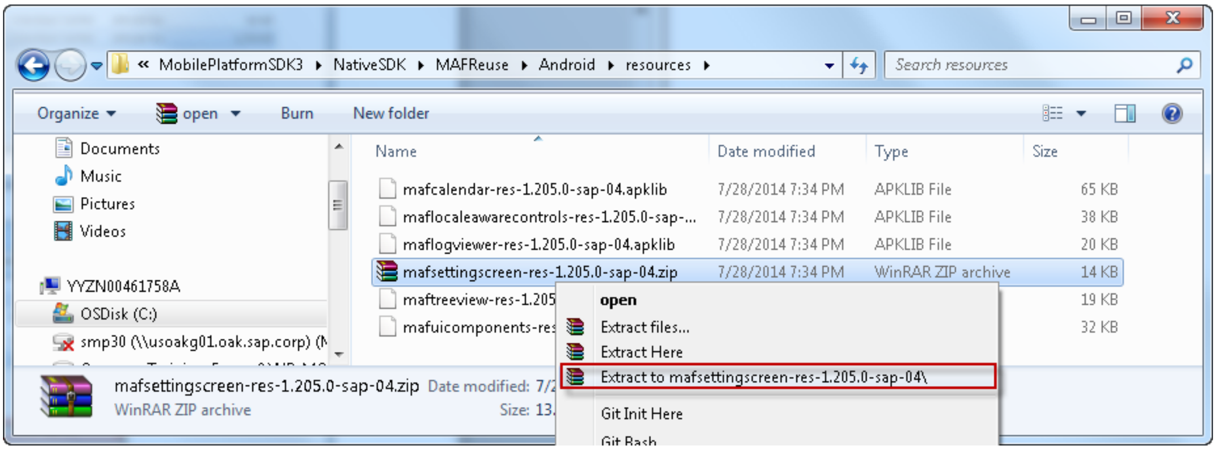
Name	Date modified	Type	Size
mafformatters-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	86 KB
maflogonui-res-1.205.0-sap-04.zip	7/28/2014 7:34 PM	WinRAR ZIP archive	1,254 KB

maflogonui-res-1.205.0-sap-04.zip Date modified:
WinRAR ZIP archive Size:

4. Open folder <Client SDK dir>\NativeSDK\MAFReuse\Android\resources\

The resource archives have .apklib extension (i.e mafsettingscreen-res-xxx.akplib). You will need to change this extension to .zip

5. Unzip the .zip flie into a local folder



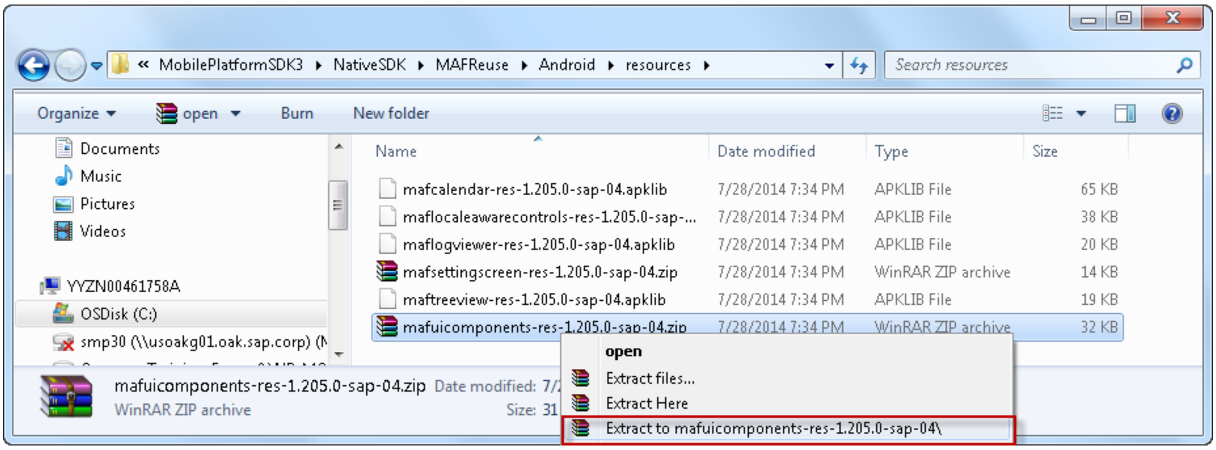
Name	Date modified	Type	Size
mafcalendar-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	65 KB
maflocaleawarecontrols-res-1.205.0-sap-...	7/28/2014 7:34 PM	APKLIB File	38 KB
maflogviewer-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	20 KB
mafsettingscreen-res-1.205.0-sap-04.zip	7/28/2014 7:34 PM	WinRAR ZIP archive	14 KB
maftreeview-res-1.205			19 KB
mafuicomponents-res			32 KB

mafsettingscreen-res-1.205.0-sap-04.zip Date modified: 7/
WinRAR ZIP archive Size: 13.

6. In the same folder <Client SDK dir>\NativeSDK\MAFReuse\Android\resources\ you will find mafuicomponents.apklib

   The resource archives have .apklib extension (i.e mafuicomponents-res-xxx.akplib). You will need to change this extension to .zip

7. Unzip the .zip flie into a local folder

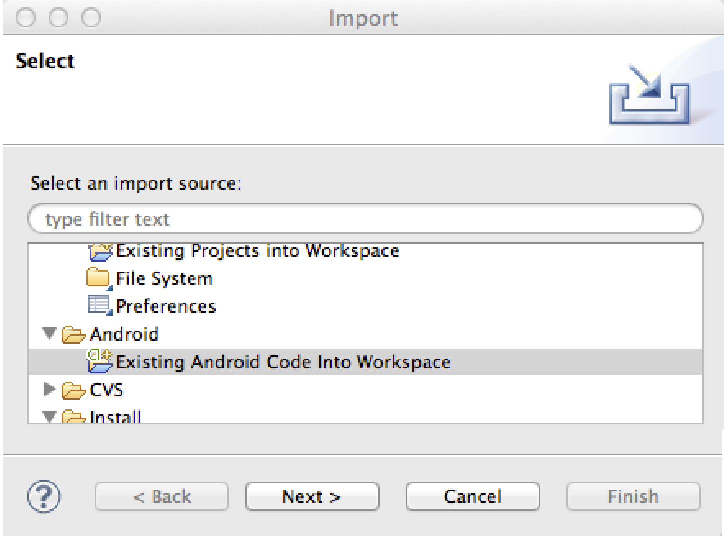


Name	Date modified	Type	Size
mafcalendar-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	65 KB
maflocaleawarecontrols-res-1.205.0-sap-...	7/28/2014 7:34 PM	APKLIB File	38 KB
maflogviewer-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	20 KB
mafsettingscreen-res-1.205.0-sap-04.zip	7/28/2014 7:34 PM	WinRAR ZIP archive	14 KB
maftreeview-res-1.205.0-sap-04.apklib	7/28/2014 7:34 PM	APKLIB File	19 KB
mafuicomponents-res-1.205.0-sap-04.zip	7/28/2014 7:34 PM	WinRAR ZIP archive	32 KB

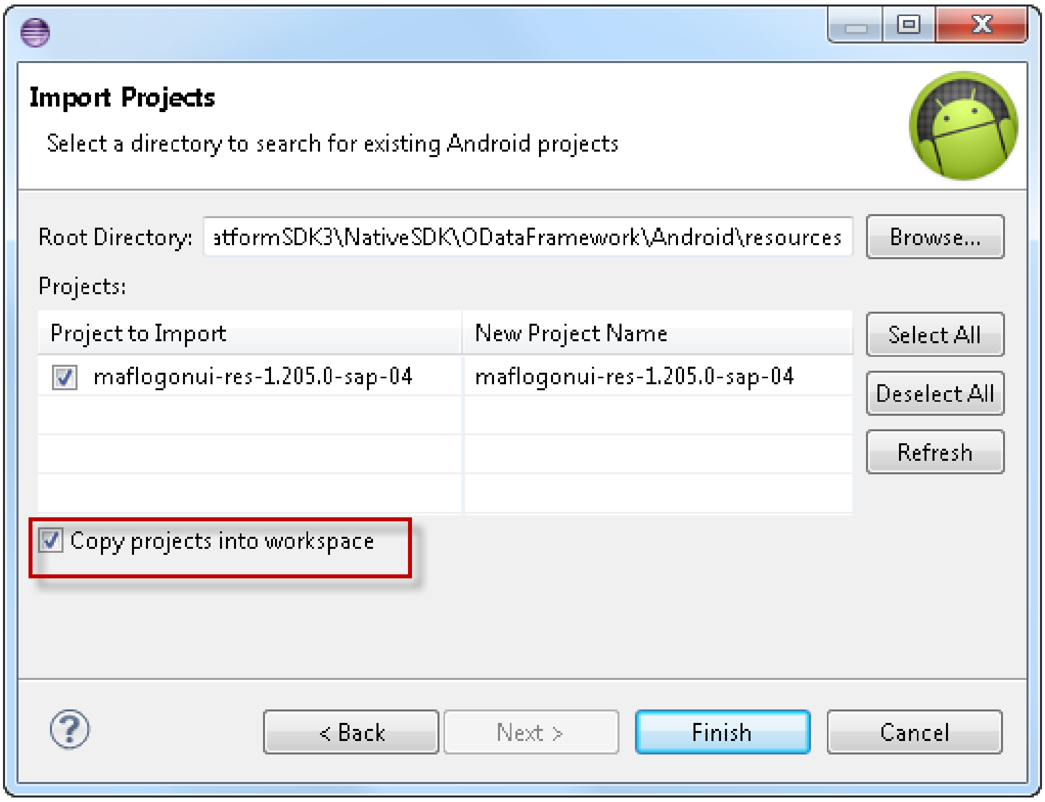
« MobilePlatformSDK3 ▸ NativeSDK ▸ MAFReuse ▸ Android ▸ resources ▸
Search resources
Organize
open
Burn
New folder
Documents
Music
Pictures
Videos
YYZN00461758A
OSDisk (C:)
smp30 (\\usoakg01.oak.sap.corp) (N
mafuicomponents-res-1.205.0-sap-04.zip Date modified: 7/
WinRAR ZIP archive Size: 31
open
Extract files...
Extract Here
Extract to mafuicomponents-res-1.205.0-sap-04\

8. Open eclipse, import the projects by selecting File -> Import menu.

9. The Import window will open, select Existing Android Code into Workspace and click Next.



10. In the Root Directory field, click the Browse button to locate the .zip files in your local folder (i.e. <Client_SDK_dir>\NativeSDK\ODataFramework\Android\resources\maflogonui-res-xxx.akplib) and click Finish.



Import Projects

Select a directory to search for existing Android projects

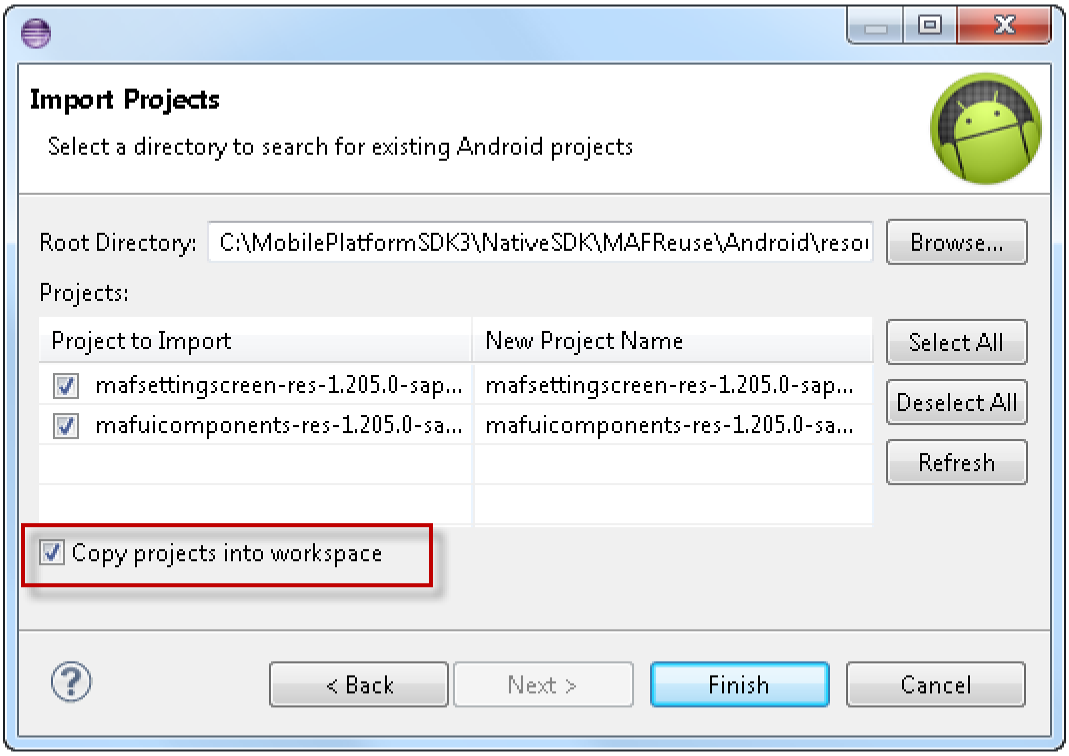
Root Directory: atformSDK3\NativeSDK\ODataFramework\Android\resources

Projects:

Project to Import	New Project Name
☑ maflogonui-res-1.205.0-sap-04	maflogonui-res-1.205.0-sap-04

☑ Copy projects into workspace

11. Repeats steps 12-13 to import the other MAF project resources:, mafsettingscreen and mafuicomponents



Import Projects

Select a directory to search for existing Android projects

Root Directory: C:\MobilePlatformSDK3\NativeSDK\MAFReuse\Android\resou

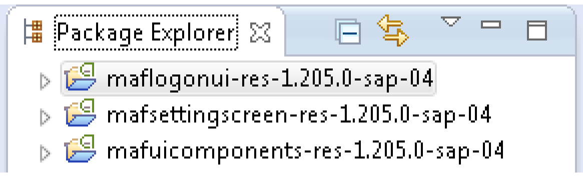
Projects:

Project to Import	New Project Name
☑ mafsettingscreen-res-1.205.0-sap...	mafsettingscreen-res-1.205.0-sap...
☑ mafuicomponents-res-1.205.0-sa...	mafuicomponents-res-1.205.0-sa...

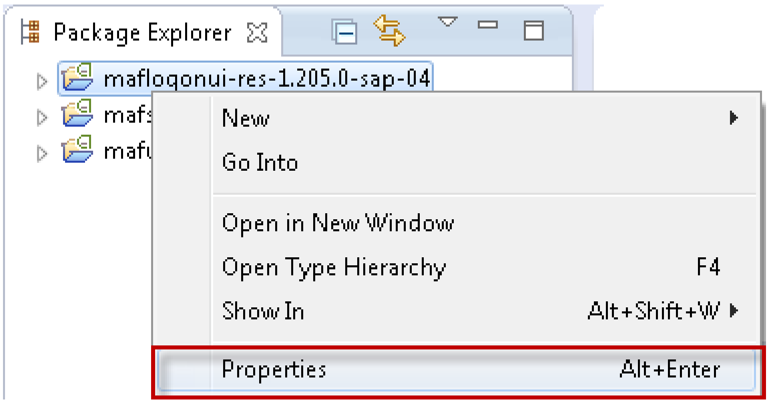
☑ Copy projects into workspace

.

12. After importing the three MAF resources, in the Package Explorer view, check your project, it should look like the image below.

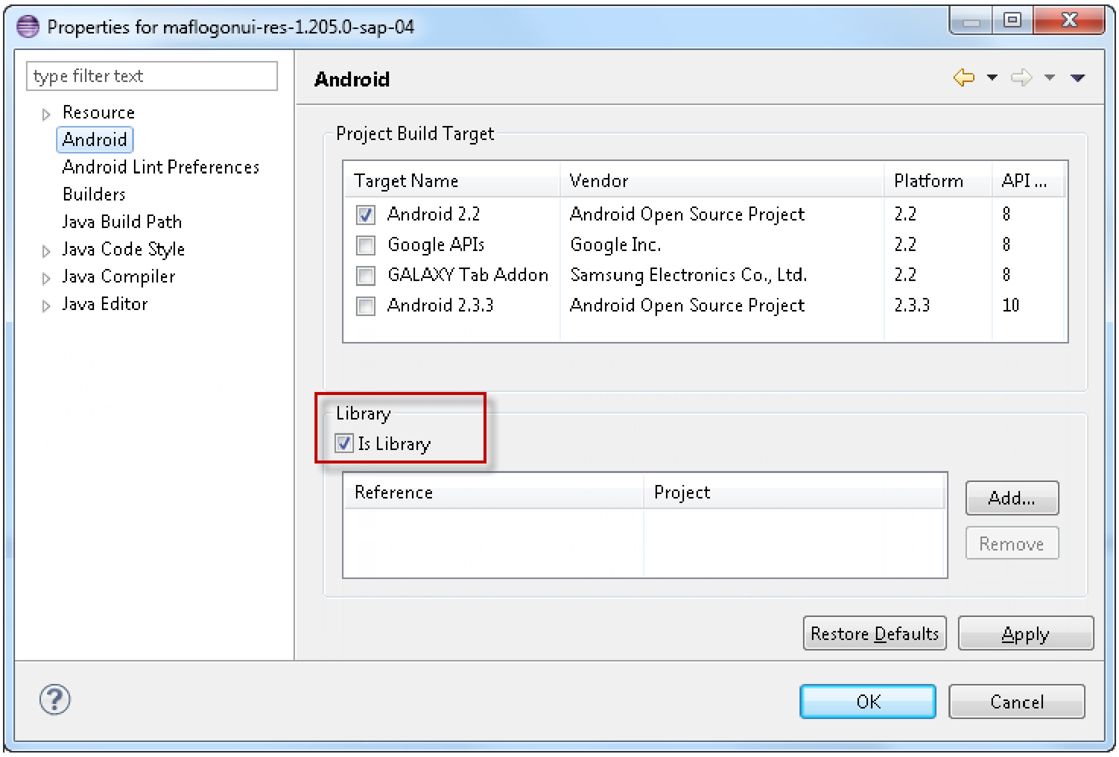

Package Explorer
maflogonui-res-1.205.0-sap-04
mafsettingscreen-res-1.205.0-sap-04
mafuicomponents-res-1.205.0-sap-04

13. Select maflogonui-res-xxx project. Right click and select the project properties from the context menu. These MAF resources projects need to be marked as Android library project.



14. In the project properties you need to select Android and then mark the project as Library.

⚠ Remember to repeat this step for all three of the resource projects: maflogonui, mafsettingscreen and mafuicomponents.



Properties for maflogonui-res-1.205.0-sap-04

Android

Project Build Target

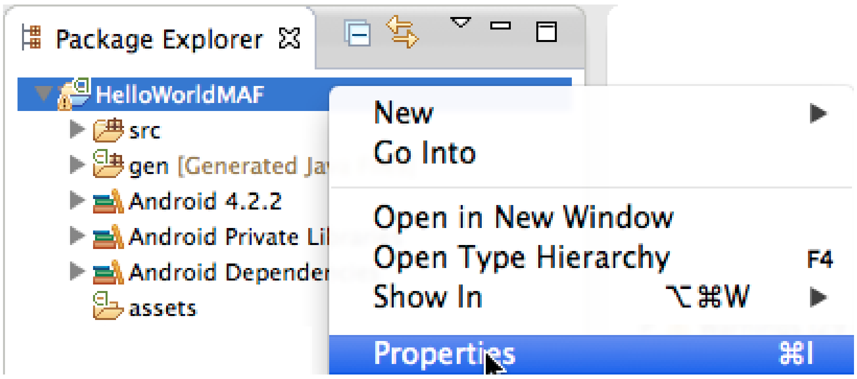
	Target Name	Vendor	Platform	API ...
☑	Android 2.2	Android Open Source Project	2.2	8
□	Google APIs	Google Inc.	2.2	8
□	GALAXY Tab Addon	Samsung Electronics Co., Ltd.	2.2	8
□	Android 2.3.3	Android Open Source Project	2.3.3	10

Library

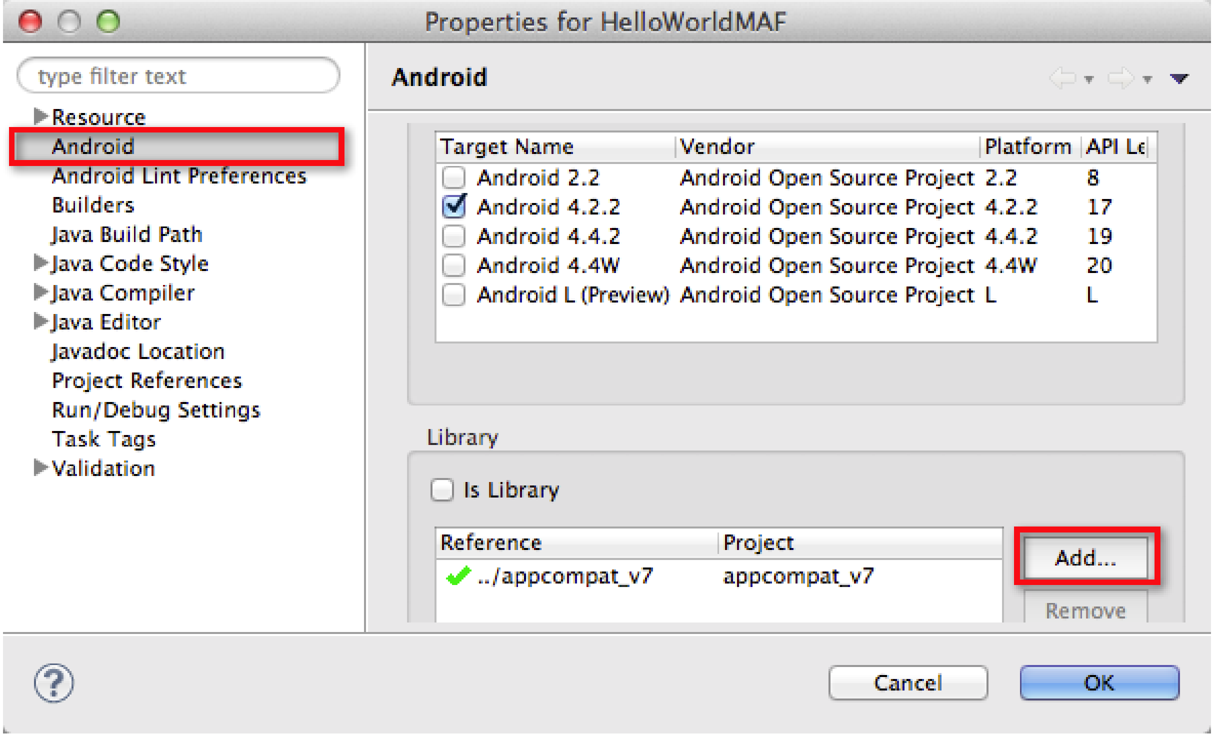
☑ Is Library

Reference	Project

15. Select HelloWorldMAF and open the properties to set up the reference to the MAF resources



16. Select Android. In the Library section you need to click the Add button.

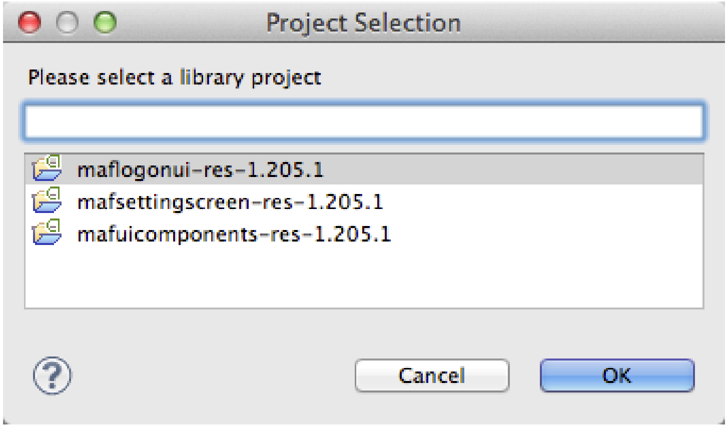


Target Name	Vendor	Platform	API Le
□ Android 2.2	Android Open Source Project	2.2	8
☑ Android 4.2.2	Android Open Source Project	4.2.2	17
□ Android 4.4.2	Android Open Source Project	4.4.2	19
□ Android 4.4W	Android Open Source Project	4.4W	20
□ Android L (Preview)	Android Open Source Project	L	L

□ Is Library

Reference	Project
✓ ../appcompat_v7	appcompat_v7

17. The Project Selection modal view will be shown. Select each of the maf resources and click OK

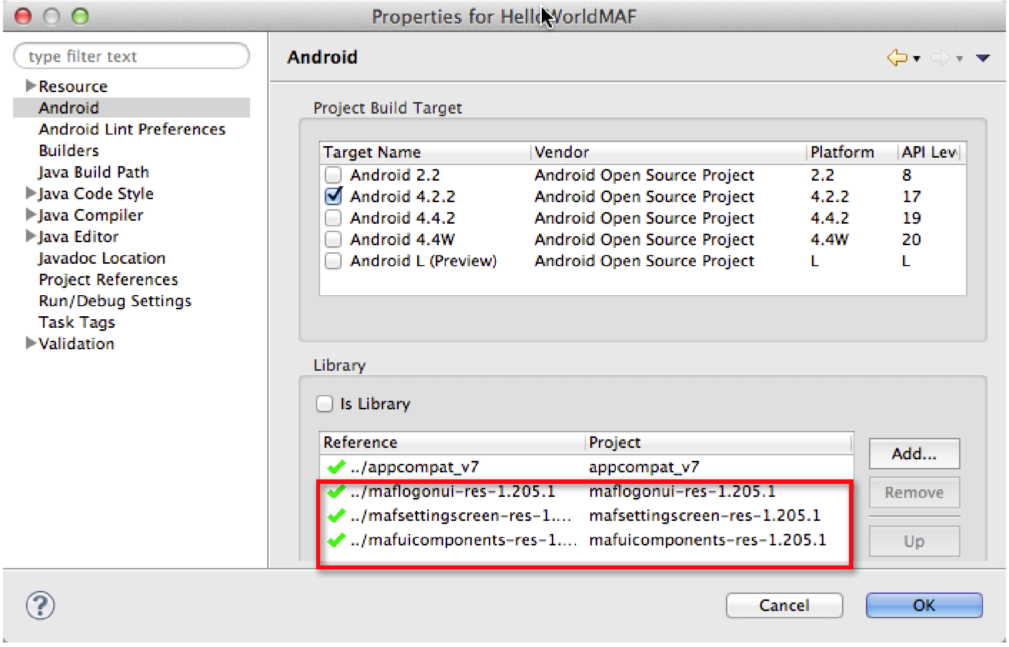


Project Selection

Please select a library project

- maflogonui-res-1.205.1
- mafsettingscreen-res-1.205.1
- mafuicomponents-res-1.205.1

18. Once you have added the 3 resources, click OK



Properties for HelloWorldMAF

Android

Project Build Target

Target Name	Vendor	Platform	API Lev
□ Android 2.2	Android Open Source Project	2.2	8
☑ Android 4.2.2	Android Open Source Project	4.2.2	17
□ Android 4.4.2	Android Open Source Project	4.4.2	19
□ Android 4.4W	Android Open Source Project	4.4W	20
□ Android L (Preview)	Android Open Source Project	L	L

Library

□ Is Library

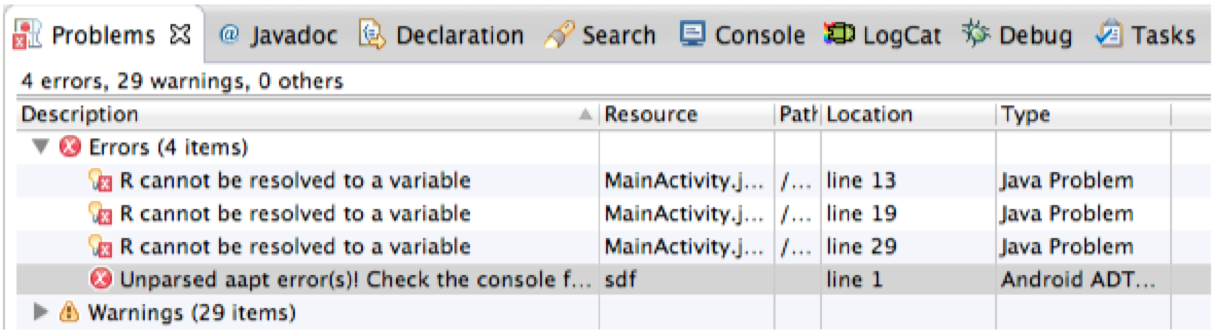
Reference	Project
✓ ../appcompat_v7	appcompat_v7
✓ ../maflogonui-res-1.205.1	maflogonui-res-1.205.1
✓ ../mafsettingscreen-res-1....	mafsettingscreen-res-1.205.1
✓ ../mafuicomponents-res-1....	mafuicomponents-res-1.205.1

If your project has dependencies to the Android Support Library package (i.e appcompat_v7), when you add the MAF resources, you may encounter some errors.

The Android Support Library package is a set of code libraries that provide backward-compatible versions of Android framework APIs as well as features that are only available through the library APIs. Each Support Library is backward-compatible to a specific Android API level. This design means that your applications can use the libraries' features and still be compatible with devices running Android 1.6 (API level 4) and up

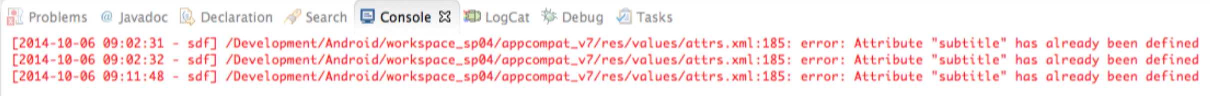
Please check the Appendix A – Fixing Problems with Android Project section at the end of this guide to remove the dependency to the Android Support Library package

Some sample errors messages you may get



4 errors, 29 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (4 items)				
R cannot be resolved to a variable	MainActivity.j...	/...	line 13	Java Problem
R cannot be resolved to a variable	MainActivity.j...	/...	line 19	Java Problem
R cannot be resolved to a variable	MainActivity.j...	/...	line 29	Java Problem
Unparsed aapt error(s)! Check the console f...	sdf		line 1	Android ADT...
Warnings (29 items)				



```
[2014-10-06 09:02:31 - sdf] /Development/Android/workspace_sp04/appcompat_v7/res/values/attrs.xml:185: error: Attribute "subtitle" has already been defined
[2014-10-06 09:02:32 - sdf] /Development/Android/workspace_sp04/appcompat_v7/res/values/attrs.xml:185: error: Attribute "subtitle" has already been defined
[2014-10-06 09:11:48 - sdf] /Development/Android/workspace_sp04/appcompat_v7/res/values/attrs.xml:185: error: Attribute "subtitle" has already been defined
```

4.4 Review AndroidManifest.xml

1. Ensure that AndroidManifest.xml contains the INTERNET permission:

```
    <!-- allow connections to Internet Services. -->
    <uses-permission android:name="android.permission.INTERNET" />
```

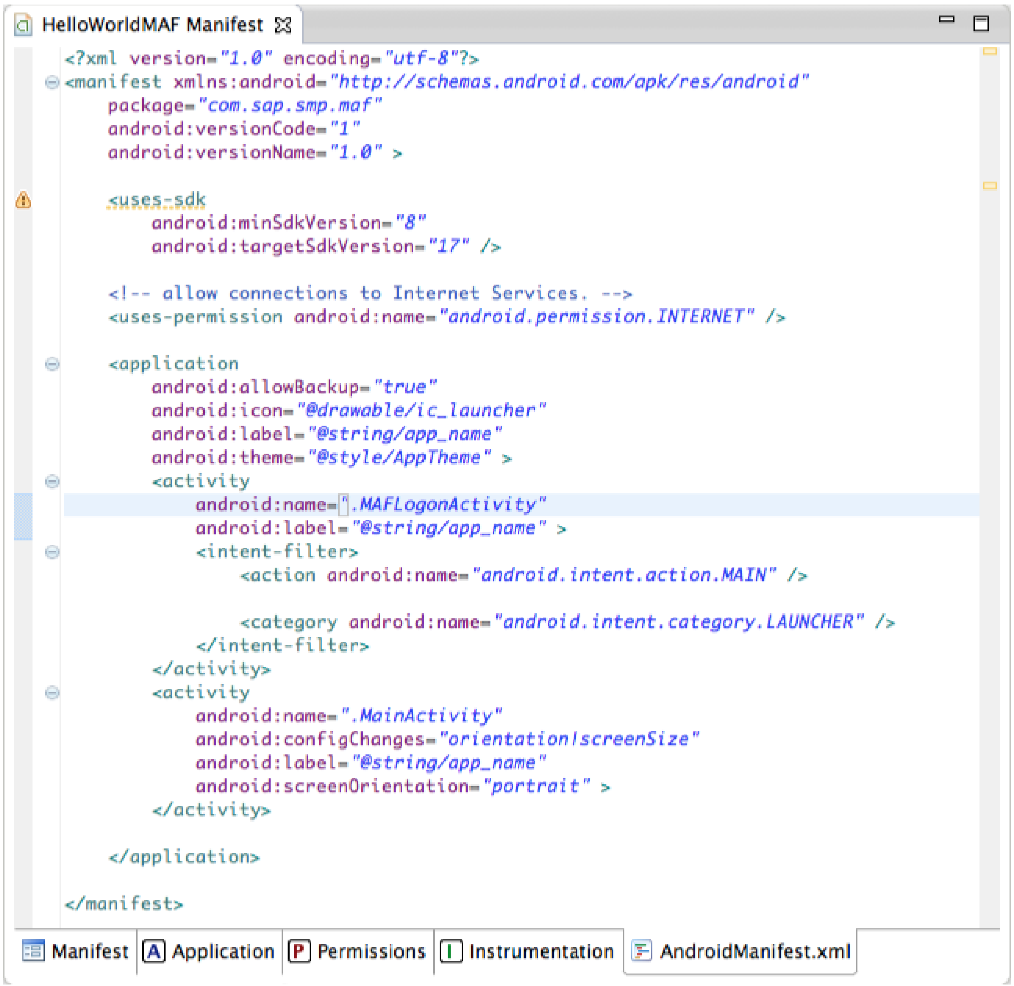
2. Define as main activity the MAFLogonActivity. This is a new activity we are going to create during this exercise that will reuse the MAF Logon component.

```
       <activity
             android:name=".MAFLogonActivity"
             android:label="@string/app_name" >
             <intent-filter>
                 <action android:name="android.intent.action.MAIN" />
                 <category android:name="android.intent.category.LAUNCHER" />
             </intent-filter>
         </activity>
```

3. Define the MainActivity as a second screen that will be displayed after the registration finishes.

```
   <activity
         android:name=".MainActivity"
         android:configChanges="orientation|screenSize"
         android:label="@string/app_name"
         android:screenOrientation="portrait" >
     </activity>
```

4. Your manifest file should look like the image below.

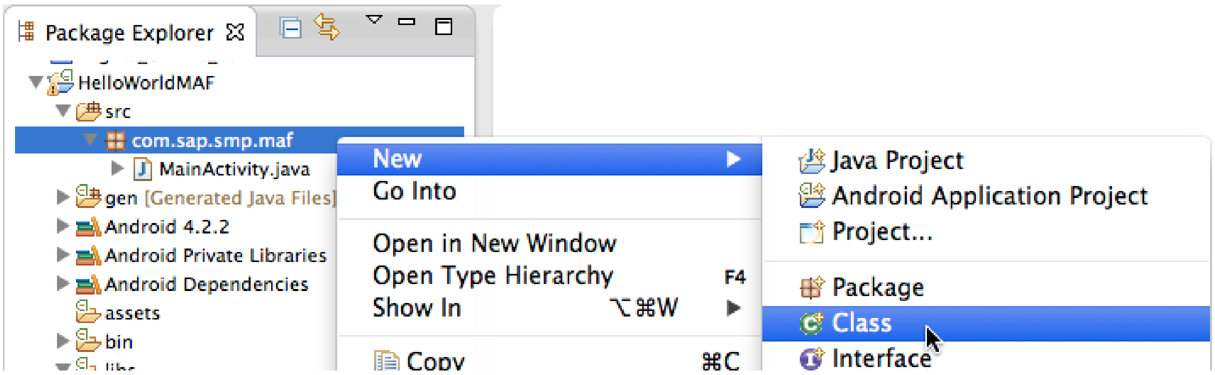


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sap.smp.maf"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <!-- allow connections to Internet Services. -->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MAFLogonActivity"
            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=".MainActivity"
            android:configChanges="orientation|screenSize"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
        </activity>

    </application>

</manifest>
```

4.5 Implement MAFLogonActivity

Now that we have successfully set up the Android project, we can use the MAF Logon resources in our project.

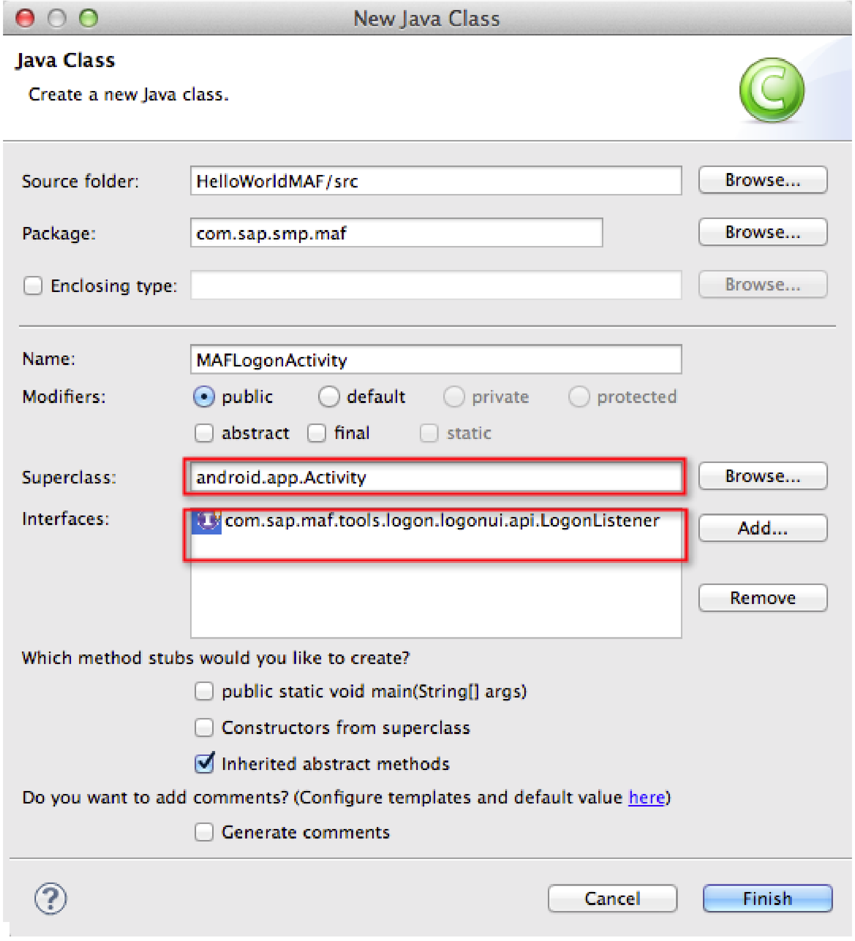
1. Go to src -> com.sap.smp.maf
2. Right-click on the com.sap.smp.maf package to open the context menu
3. Select New -> Class



4. Enter the following information to create an Activity and click Finish
   a. Name = MAFLogonActivity

b. Superclass = android.app.Activity

c. Interface = com.sap.maf.tools.logon.logonui.api.LogonListener



Source folder: HelloWorldMAF/src
Package: com.sap.smp.maf
□ Enclosing type:
Name: MAFLogonActivity
Modifiers: ● public ○ default ○ private ○ protected
□ abstract □ final □ static
Superclass: android.app.Activity
Interfaces: com.sap.maf.tools.logon.logonui.api.LogonListener
Which method stubs would you like to create?
□ public static void main(String[] args)
□ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
□ Generate comments

5. Define the following class variables

```
private final String TAG = MAFLogonActivity.class.getSimpleName();
private LogonUIFacade mLogonUIFacade;
private Context mContext;
```

6. Implement the onCreate method to initialize the LogonUIFacade with the following code

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // get an instance of the LogonUIFacade
    mLogonUIFacade = LogonUIFacade.getInstance();
    // set context reference
    mContext = this;
    this.showLogonScreen();
}
```

7. Define to the showLogonScreen method with the following code.

⚠ This example assumes you have configured an application in SMP 3.0 called com.sap.flight

for more information visit Deploying Applications

```
private void showLogonScreen() {
    //TODO 1-2 COMPLETE showLogonScreen in MAFLogonActivity ************
```

```
        // initialize the LogonUIFacade with context and application id
        // This example assumes “com.sap.flight” exists in
        // SMP 3.0 Management Cockpit
         mLogonUIFacade.init(this, mContext, “com.sap.flight”);

         // ask LogonUIFacede to present the logon screen
         // set the resulting view as the content view for this activity
         setContentView(mLogonUIFacade.logon());
    }
```

`setContentView` method is used to present the logon screen from the MAFLogonActivity

8. Locate onLogonFinished and complete the method with the following code. This method redirects to the main screen when the registration finishes

   This callback method handles BOTH success and error cases.

```
@Override
public void onLogonFinished(String message, boolean isSuccess, LogonContext
lgContext) {
      //TODO 1-3 COMPLETE onLogonFinished in MAFLogonActivity ******
             Log.d(TAG, "onLogonFinished: "+message);
             //Check if it finished successfully
             if (isSuccess) {
                 try {
                     //For debugging purposes will log the app connection id and
                     // the end point url.
                     // In a productive app, remember to remove these logs
                     String appConnID = LogonCore.getInstance().getLogonContext()
                             .getConnId();
                     Log.d(TAG, "onLogonFinished: appcid:"+ appConnID);
                     Log.d(TAG, "onLogonFinished: endpointurl:"+
                                         lgContext.getEndPointUrl());

                 } catch (LogonManagerException e) {
                     Log.e(TAG, e.getLocalizedMessage(), e);
                 } catch (LogonCoreException e) {
                     Log.e(TAG, e.getLocalizedMessage(), e);
                 }

                 // Navigate to the Main menu screen
                 Intent goToNextActivity = new Intent(this, MainActivity.class);
                 startActivity(goToNextActivity);
                 finish();
             }

    }
```

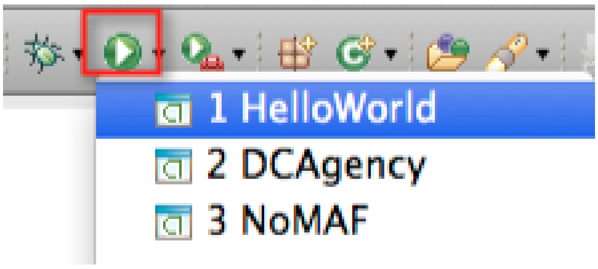
7. Save the changes

8. Please find the MAFLogonActivity in Appendix B - MAFLogonActivity.java

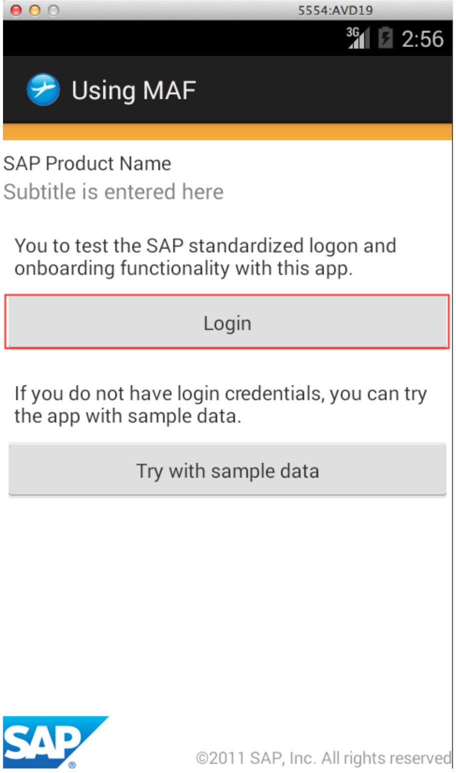
4.6 Run the Application

If you haven’t created the Debug Configuration, please review Appendix C – Creating Debug Configuration

9. In Eclipse, go to the Run icon and select the corresponding HelloWorld configuration


1 HelloWorld
2 DCAgency
3 NoMAF

8. Click on the Login button



5554:AVD19

2:56

Using MAF

SAP Product Name

Subtitle is entered here

You to test the SAP standardized logon and onboarding functionality with this app.

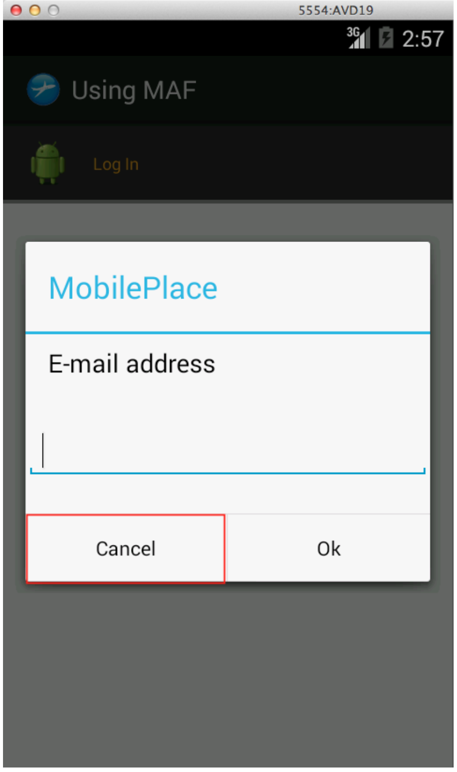
Login

If you do not have login credentials, you can try the app with sample data.

Try with sample data

©2011 SAP, Inc. All rights reserved

9. Click on Cancel



5554:AVD19

2:57

Using MAF

Log In

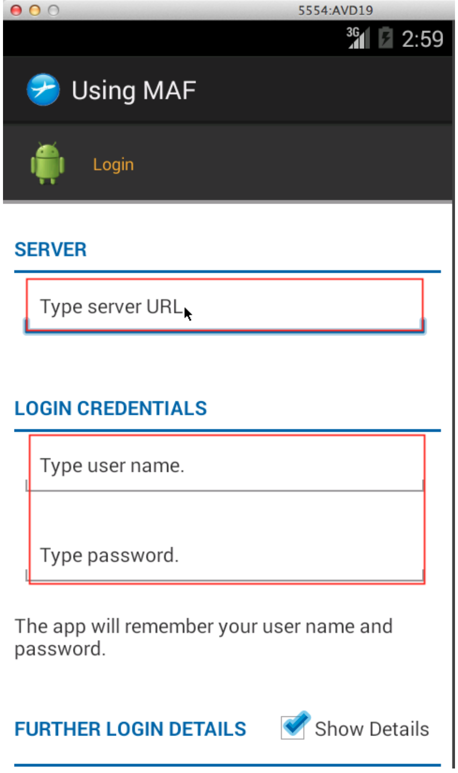
MobilePlace

E-mail address

Cancel

Ok

10. Enter the IP address of the SMP 3.0 server and backend credentials



5554:AVD19

2:59

Using MAF

Login

SERVER

Type server URL.

LOGIN CREDENTIALS

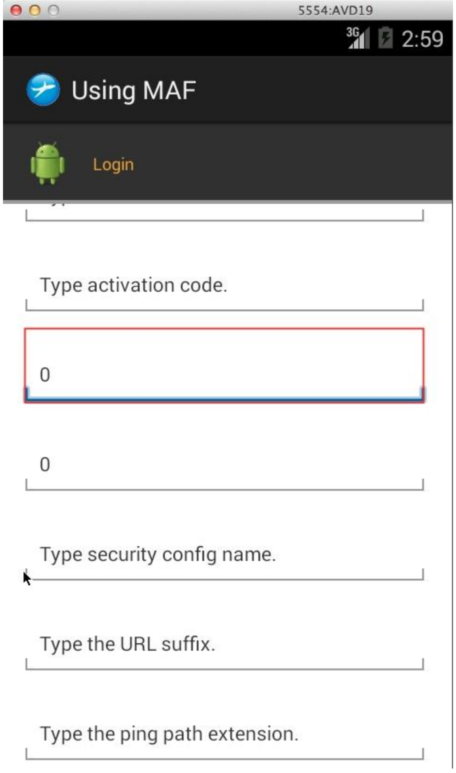
Type user name.

Type password.

The app will remember your user name and password.

FURTHER LOGIN DETAILS ☑ Show Details

11. Scroll down to enter port (8080 is default)



5554:AVD19

2:59

Using MAF

Login

Type activation code.

0

0

Type security config name.

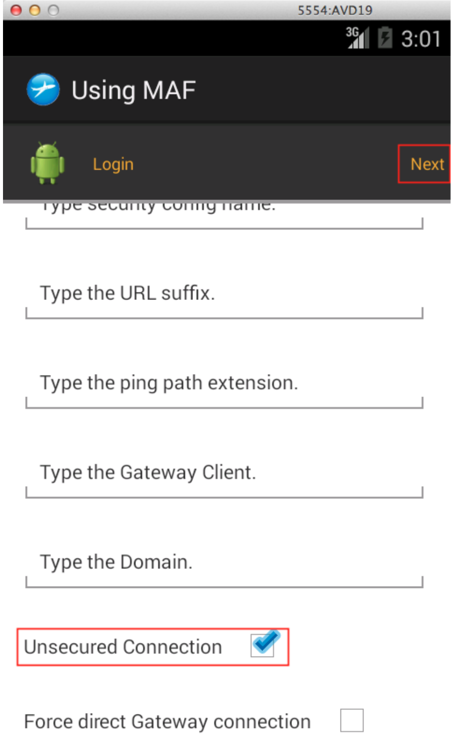
Type the URL suffix.

Type the ping path extension.

12. Keep scrolling until the end of the screen to

13. Enter a passcode or uncheck the Use

check the Unsecured option and click on Next



5554:AVD19

3:01

Using MAF

Login Next

Type security config name.

Type the URL suffix.

Type the ping path extension.

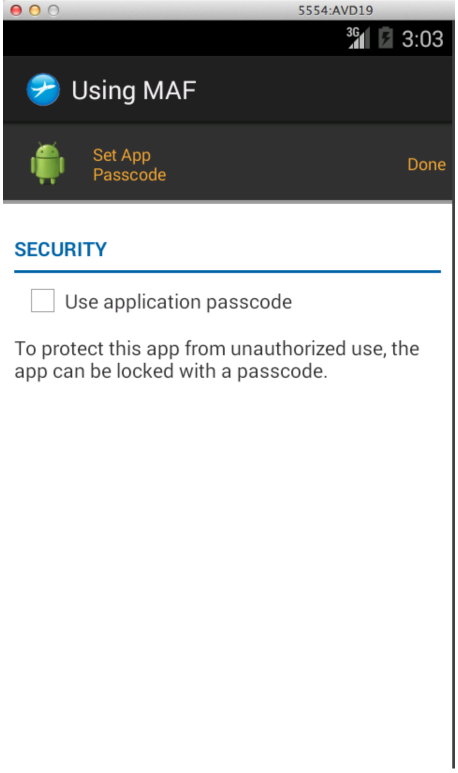
Type the Gateway Client.

Type the Domain.

Unsecured Connection ☑

Force direct Gateway connection □

application passcode and click on Done



5554:AVD19

3:03

Using MAF

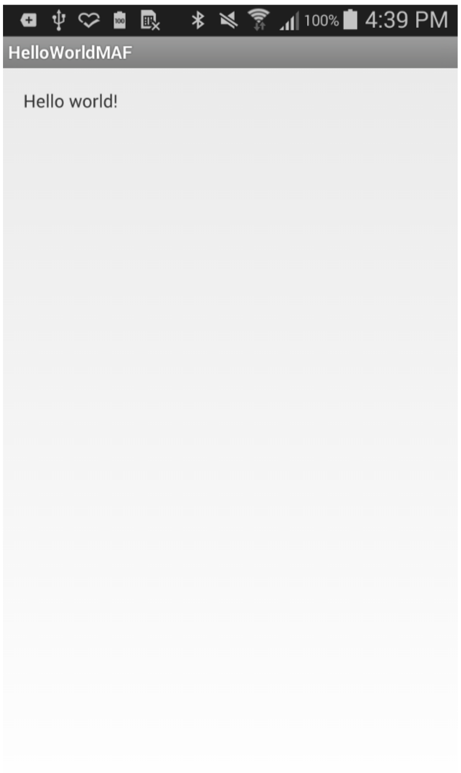
Set App Passcode Done

SECURITY

□ Use application passcode

To protect this app from unauthorized use, the app can be locked with a passcode.

14. The main screen will be displayed



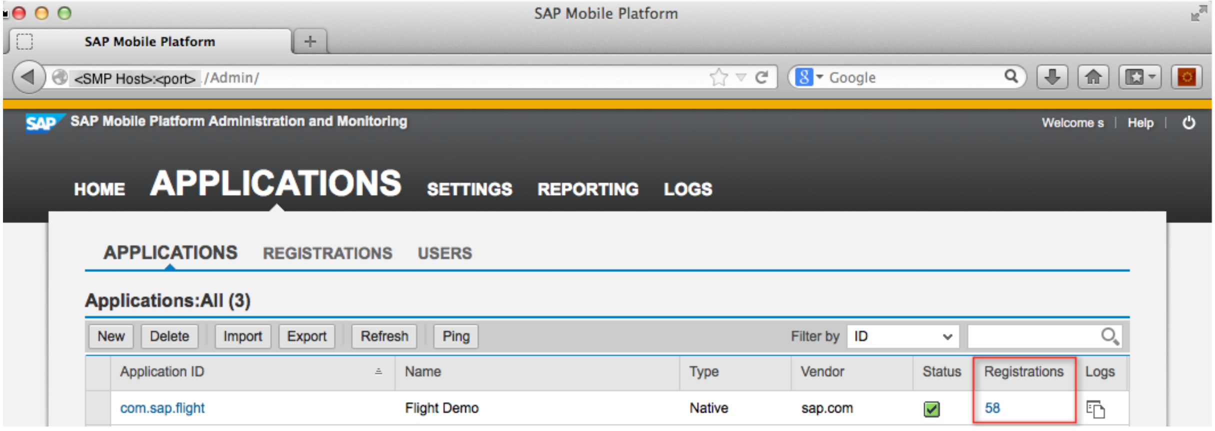
100% 4:39 PM

HelloWorldMAF

Hello world!

4.7 Verifying the registration

To verify that the user was registered successful, you can open the SAP Management Cockpit and check the number if the number of registrations increases when you register your device



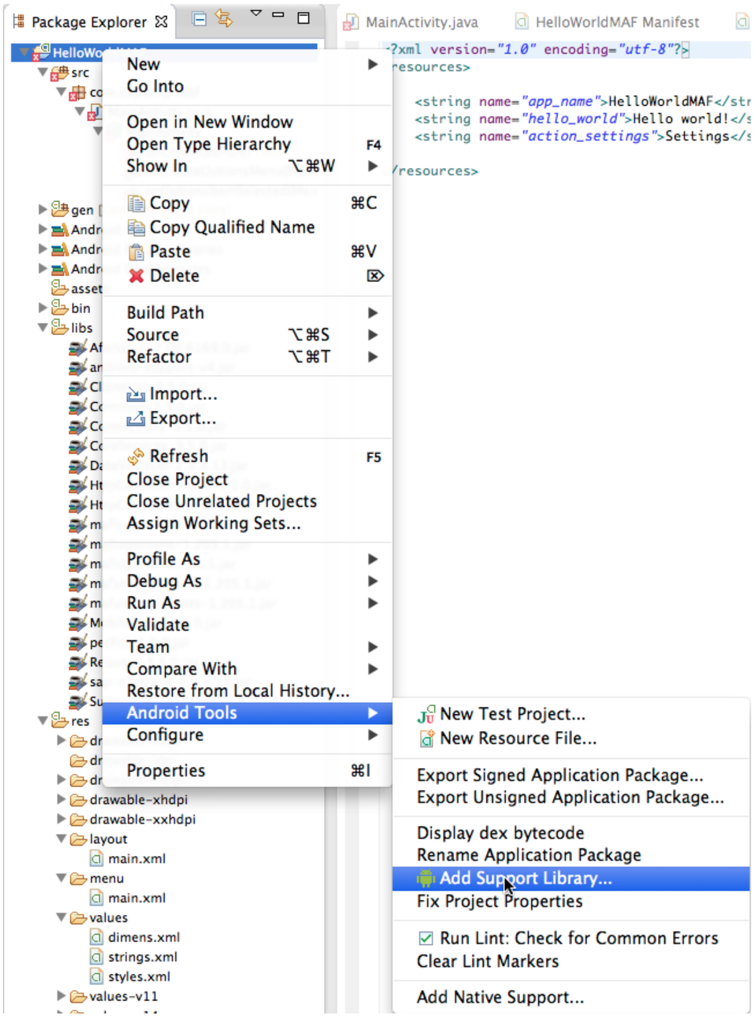
Applications:All (3)

Application ID	Name	Type	Vendor	Status	Registrations	Logs
com.sap.flight	Flight Demo	Native	sap.com	☑	58	

5. Appendix

Appendix A – Fixing Problems with Android Project

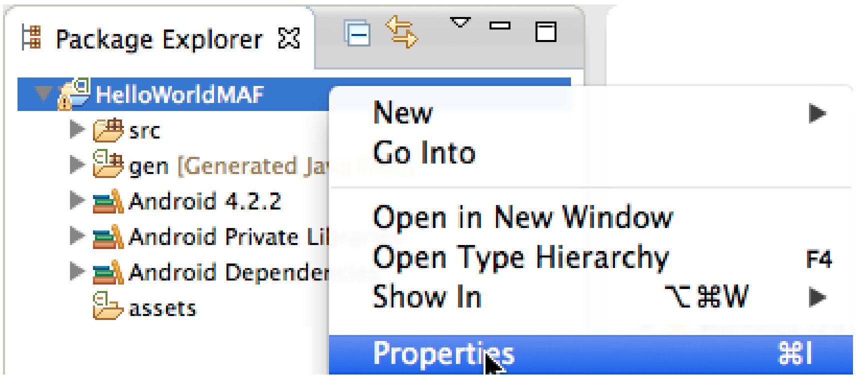
1. Right-click on the project to open the context menu.
2. Select Android Tools -> Add Support Library... to add the support library as a jar file in the project



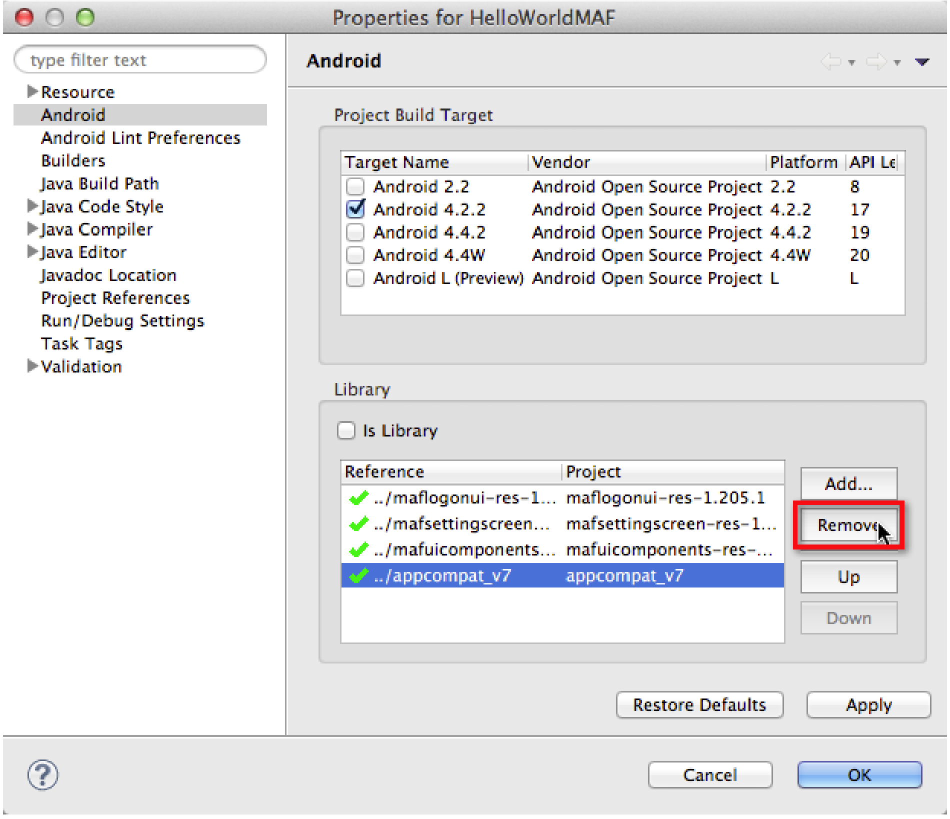
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">HelloWorldMAF</str
    <string name="hello_world">Hello world!</s
    <string name="action_settings">Settings</s

</resources>
```

Remove dependency to appcompat_v7 resources.

3. Right-click on the project to open the context menu
4. Select Properties



5. Select appcompat_v7 and click on Remove.



Target Name	Vendor	Platform	API Le
☐ Android 2.2	Android Open Source Project	2.2	8
☑ Android 4.2.2	Android Open Source Project	4.2.2	17
☐ Android 4.4.2	Android Open Source Project	4.4.2	19
☐ Android 4.4W	Android Open Source Project	4.4W	20
☐ Android L (Preview)	Android Open Source Project	L	L

Reference	Project
✓ ../maflogonui-res-1…	maflogonui-res-1.205.1
✓ ../mafsettingscreen…	mafsettingscreen-res-1…
✓ ../mafuicomponents…	mafuicomponents-res-…
✓ ../appcompat_v7	appcompat_v7

Manually remove all dependencies to appcompat_v7 resources.

6. Open res -> values -> styles.xml and replace

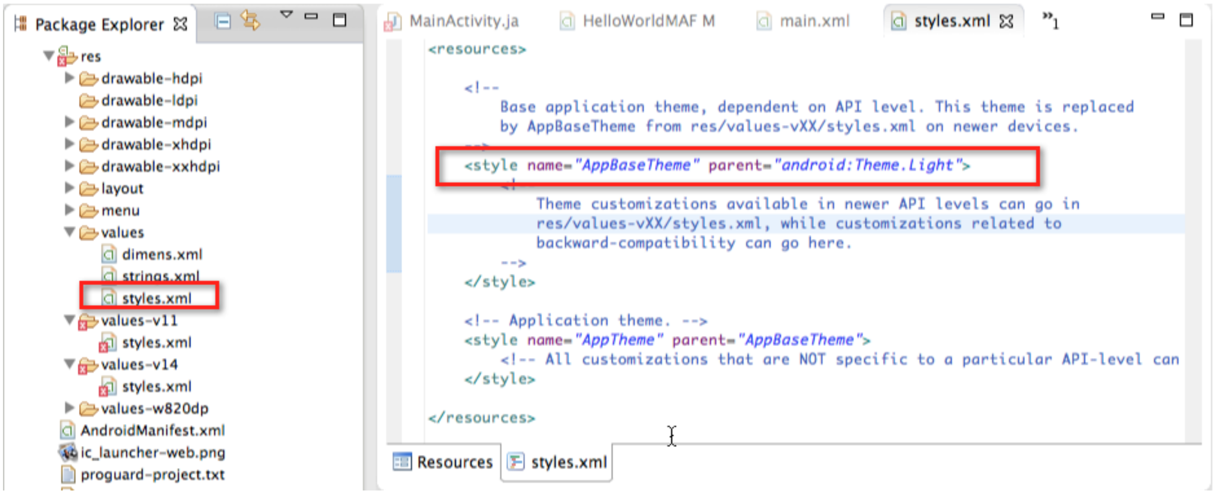
This line of code that reference the appcompat_v7 resources

```
    <style name="AppBaseTheme" parent="Theme.AppCompat.Light">
```

For this one

```
    <style name="AppBaseTheme" parent="android:Theme.Light">
```

7. Styles.xml should look lik the image below



```
<resources>

    <!--
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Light">
        <!--
            Theme customizations available in newer API levels can go in
            res/values-vXX/styles.xml, while customizations related to
            backward-compatibility can go here.
        -->
    </style>

    <!-- Application theme. -->
    <style name="AppTheme" parent="AppBaseTheme">
        <!-- All customizations that are NOT specific to a particular API-level can
    </style>

</resources>
```

8. Open res -> values-v11 -> styles.xml and replace

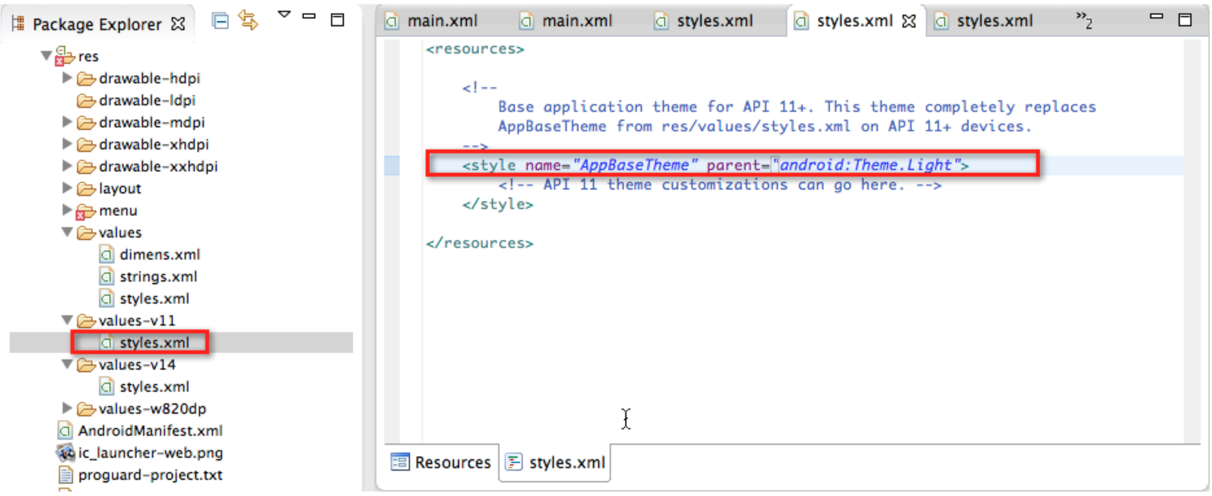
   This line of code that reference the appcompat_v7 resources

```
<style name="AppBaseTheme" parent="Theme.AppCompat.Light">
```

   For this one

```
<style name="AppBaseTheme" parent="android:Theme.Light">
```

9. res -> values-v11 -> styles.xml should look like the image below



```
<resources>

    <!--
        Base application theme for API 11+. This theme completely replaces
        AppBaseTheme from res/values/styles.xml on API 11+ devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Light">
        <!-- API 11 theme customizations can go here. -->
    </style>

</resources>
```

10. Open res -> values-v14 -> styles.xml and replace

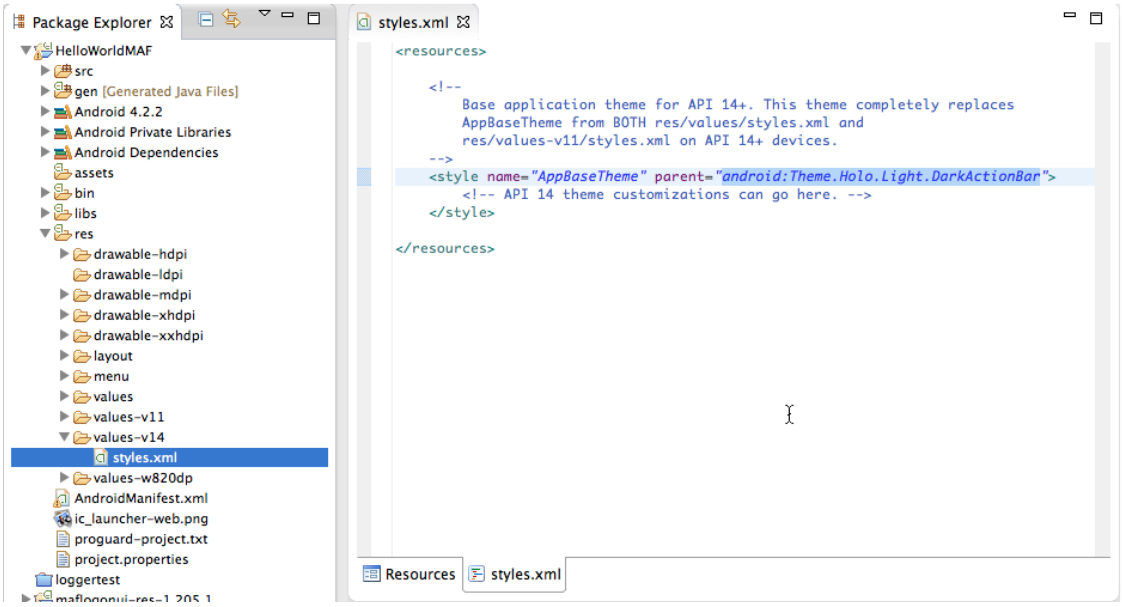
    This line of code that reference the appcompat_v7 resources

```
<style name="AppBaseTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

    For this one

```
<style name="AppBaseTheme" parent="android:Theme.Holo.Light.DarkActionBar">
```

11. res -> values-v14 -> styles.xml should look like the image below

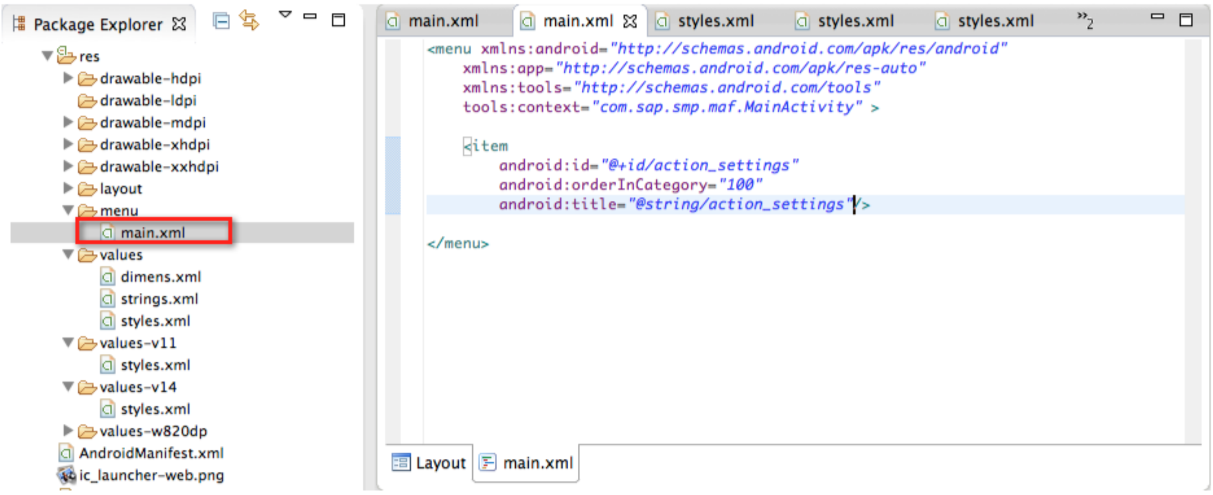


```
<resources>

    <!--
        Base application theme for API 14+. This theme completely replaces
        AppBaseTheme from BOTH res/values/styles.xml and
        res/values-v11/styles.xml on API 14+ devices.
    -->
    <style name="AppBaseTheme" parent="android:Theme.Holo.Light.DarkActionBar">
        <!-- API 14 theme customizations can go here. -->
    </style>

</resources>
```

12. Open Res -> menu -> main.xml
13. You will get an error: No resource identifier found for attribute 'showAsAction' in package 'com.sap.smp.maf'
14. Remove the showAsAction from the item properties.

```
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never"/>
```

14. Res -> menu -> main.xml should look like the image below



```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.sap.smp.maf.MainActivity" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"/>

</menu>
```

15. Open src -> com.sap.smp.maf -> MainActivity.java
16. Replace

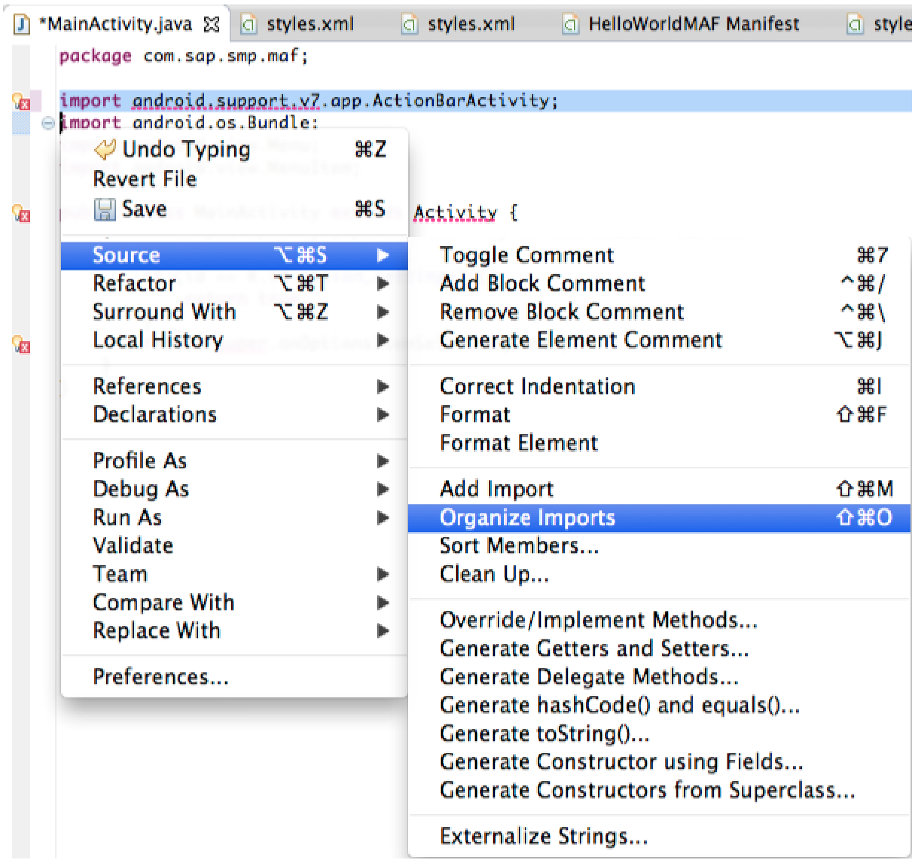
    This line of code

```
public class MainActivity extends ActionBarActivity {
```

for this one

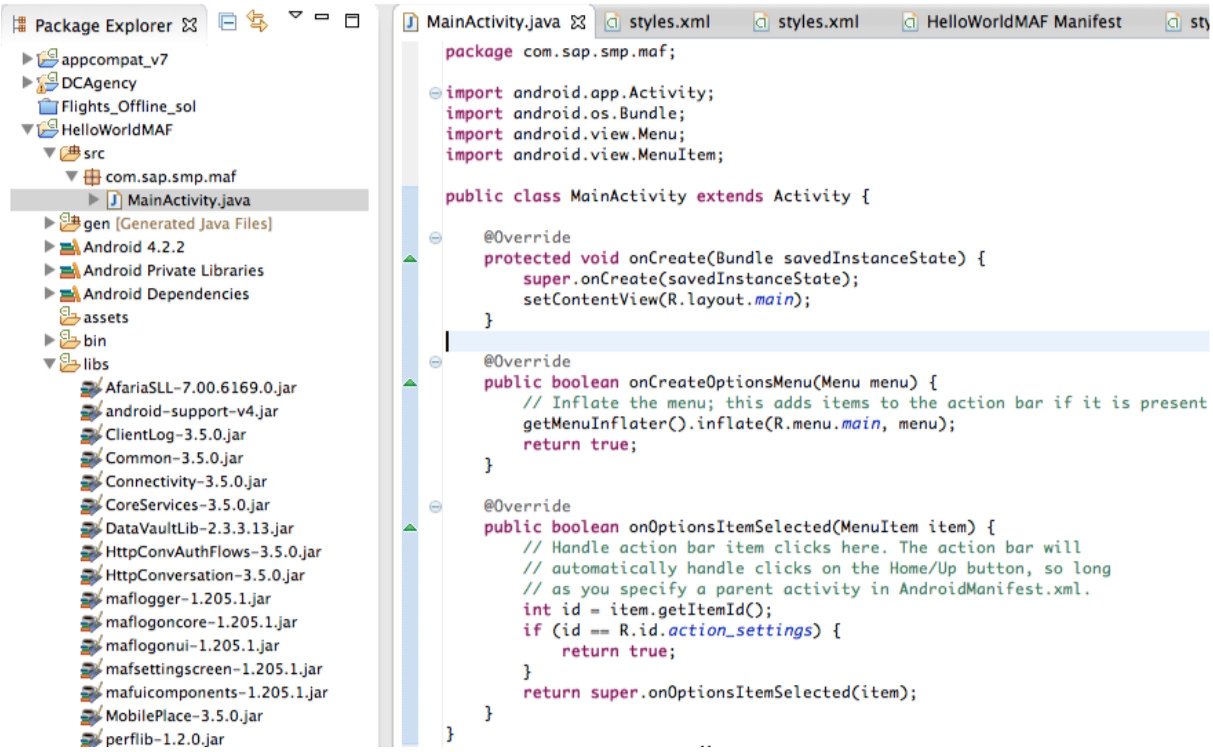
```
    public class MainActivity extends Activity {
```

17. Organize imports



```
package com.sap.smp.maf;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
                                            Activity {
```

18. MainActivity.java should look like



```
package com.sap.smp.maf;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

19. Save the changes

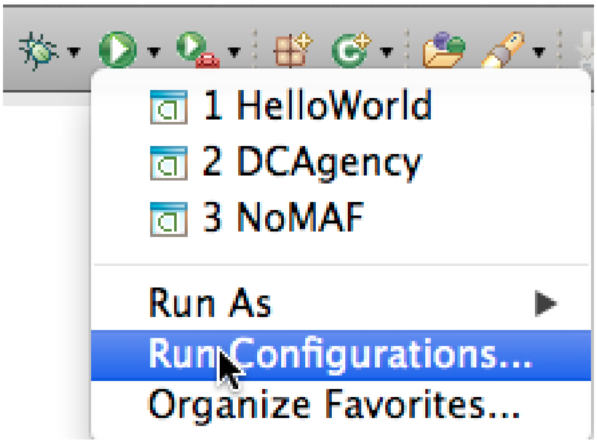
Appendix B – MAFLogonActivity.java

```
package com.sap.smp.maf;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;

import com.sap.maf.tools.logon.core.LogonCore;
import com.sap.maf.tools.logon.core.LogonCoreContext;
import com.sap.maf.tools.logon.core.LogonCoreException;
import com.sap.maf.tools.logon.logonui.api.LogonListener;
import com.sap.maf.tools.logon.logonui.api.LogonUIFacade;
import com.sap.maf.tools.logon.manager.LogonContext;
import com.sap.maf.tools.logon.manager.LogonManager.LogonManagerException;

public class MAFLogonActivity extends Activity implements LogonListener {
    private final String TAG = MAFLogonActivity.class.getSimpleName();
    private LogonUIFacade mLogonUIFacade;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // get an instance of the LogonUIFacade
        mLogonUIFacade = LogonUIFacade.getInstance();
        // set context reference
        mContext = this;
        this.showLogonScreen();
    }

    private void showLogonScreen() {
        //TODO 1-2 COMPLETE showLogonScreen in MAFLogonActivity ************
      // initialize the LogonUIFacade with context and application id
        mLogonUIFacade.init(this, mContext, "com.sap.flight");

        // ask LogonUIFacede to present the logon screen
        // set the resulting view as the content view for this activity
        setContentView(mLogonUIFacade.logon());
    }

    @Override
    public void objectFromSecureStoreForKey() {
        // TODO Auto-generated method stub

    }

    @Override
    public void onApplicationSettingsUpdated() {
        // TODO Auto-generated method stub

    }

    @Override
    public void onBackendPasswordChanged(boolean arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onLogonFinished(String message, boolean isSuccess, LogonContext lgContext) {
        //TODO 1-3 COMPLETE onLogonFinished in MAFLogonActivity ******
               Log.d(TAG, "onLogonFinished: "+message);
               //Check if it finished successfully
               if (isSuccess) {
                   try {
                       //For debugging purposes will log the app connection id and
                       // the end point url.
                     // In a productive app, remember to remove these logs
                       String appConnID = LogonCore.getInstance().getLogonContext()
                               .getConnId();
```

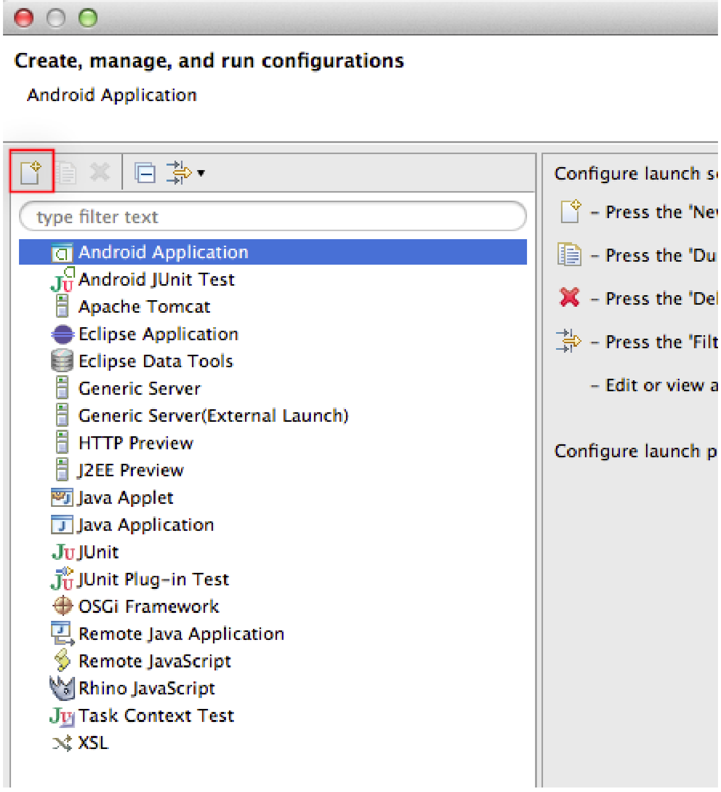
```
                        Log.d(TAG, "onLogonFinished: appcid:"+ appConnID);
                        Log.d(TAG, "onLogonFinished: endpointurl:"+ lgContext.getEndPointUrl());

                    } catch (LogonManagerException e) {
                        Log.e(TAG, e.getLocalizedMessage(), e);
                    } catch (LogonCoreException e) {
                        Log.e(TAG, e.getLocalizedMessage(), e);
                    }

                    // Navigate to the Main menu screen
                    Intent goToNextActivity = new Intent(this, MainActivity.class);
                    startActivity(goToNextActivity);
                    finish();
                }

    }

    @Override
    public void onRefreshCertificate(boolean arg0, String arg1) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onSecureStorePasswordChanged(boolean arg0, String arg1) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onUserDeleted() {
        // TODO Auto-generated method stub

    }

    @Override
    public void registrationInfo() {
        // TODO Auto-generated method stub

    }

}
```

Appendix C – Creating Debug Configuration

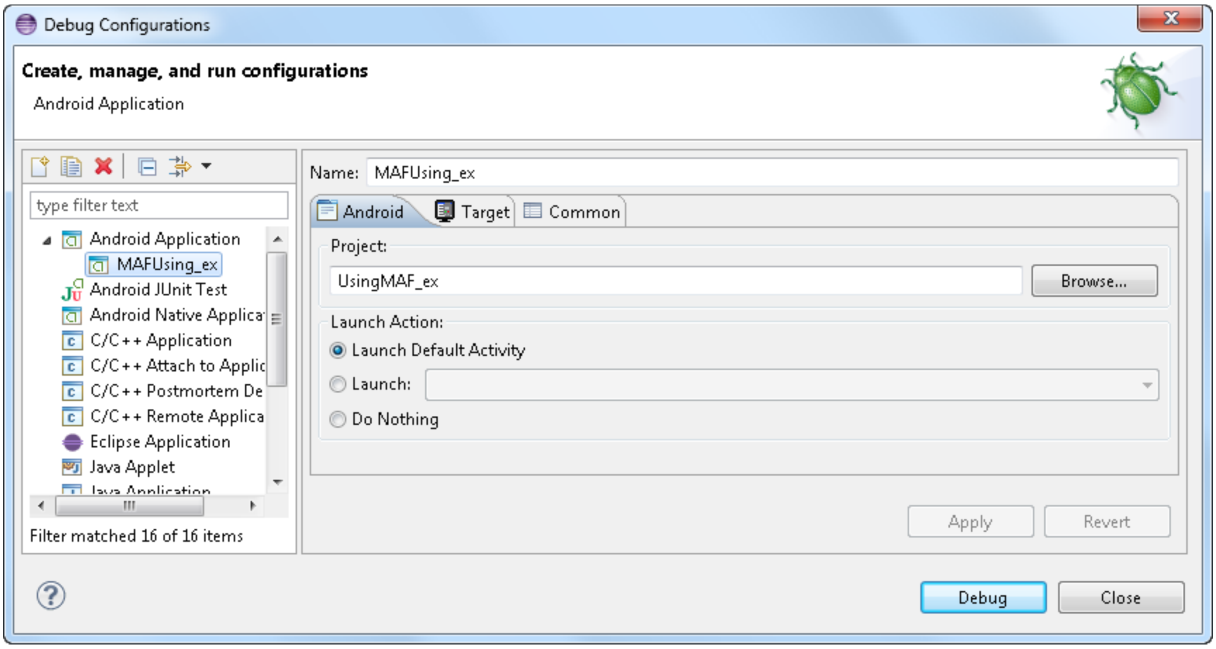
1. In Eclipse, go to the Run icon and select Run Configurations



2. Select Android Application and click on the Create icon



3. Select the Android tab and select the corresponding HelloWorldMAF project



Debug Configurations

Create, manage, and run configurations

Android Application

Name: MAFUsing_ex

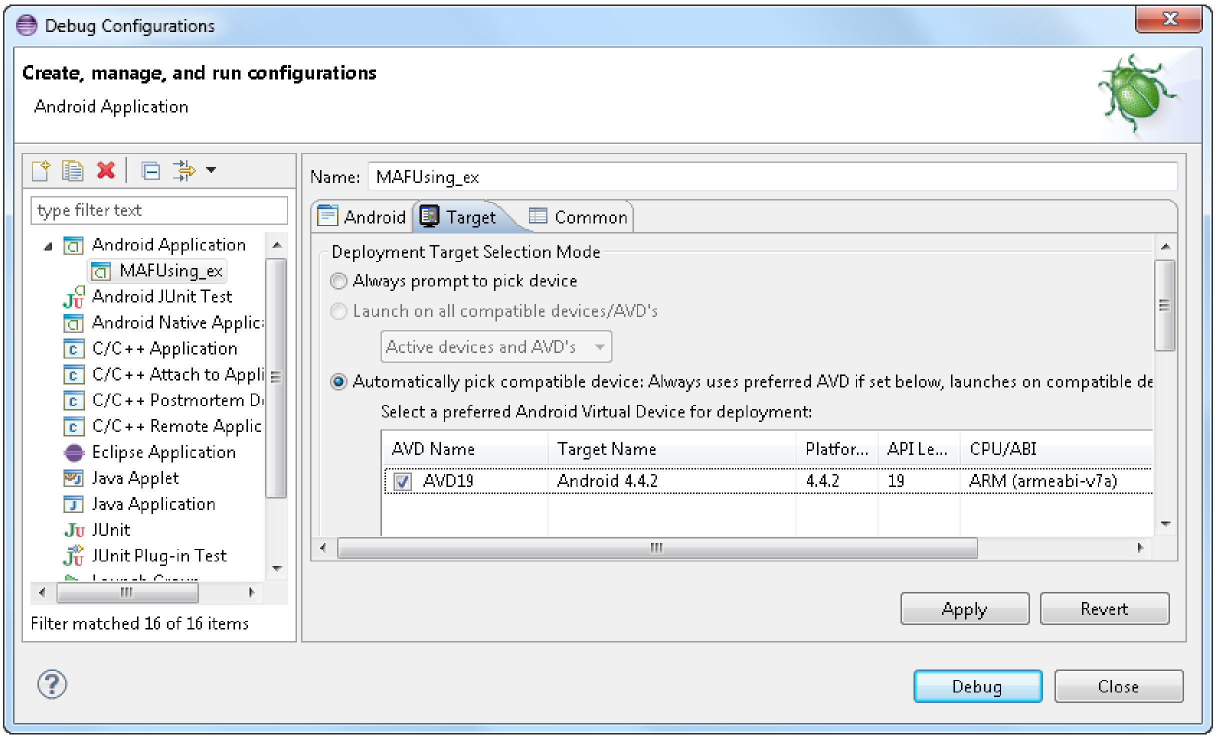
Project: UsingMAF_ex

Launch Action:

- ● Launch Default Activity
- ○ Launch:
- ○ Do Nothing

Filter matched 16 of 16 items

4. Select the Target tab and make sure you have one Android Virtual Device (AVD) selected. Then click on Apply and then on Debug.



Debug Configurations

Create, manage, and run configurations

Android Application

Name: MAFUsing_ex

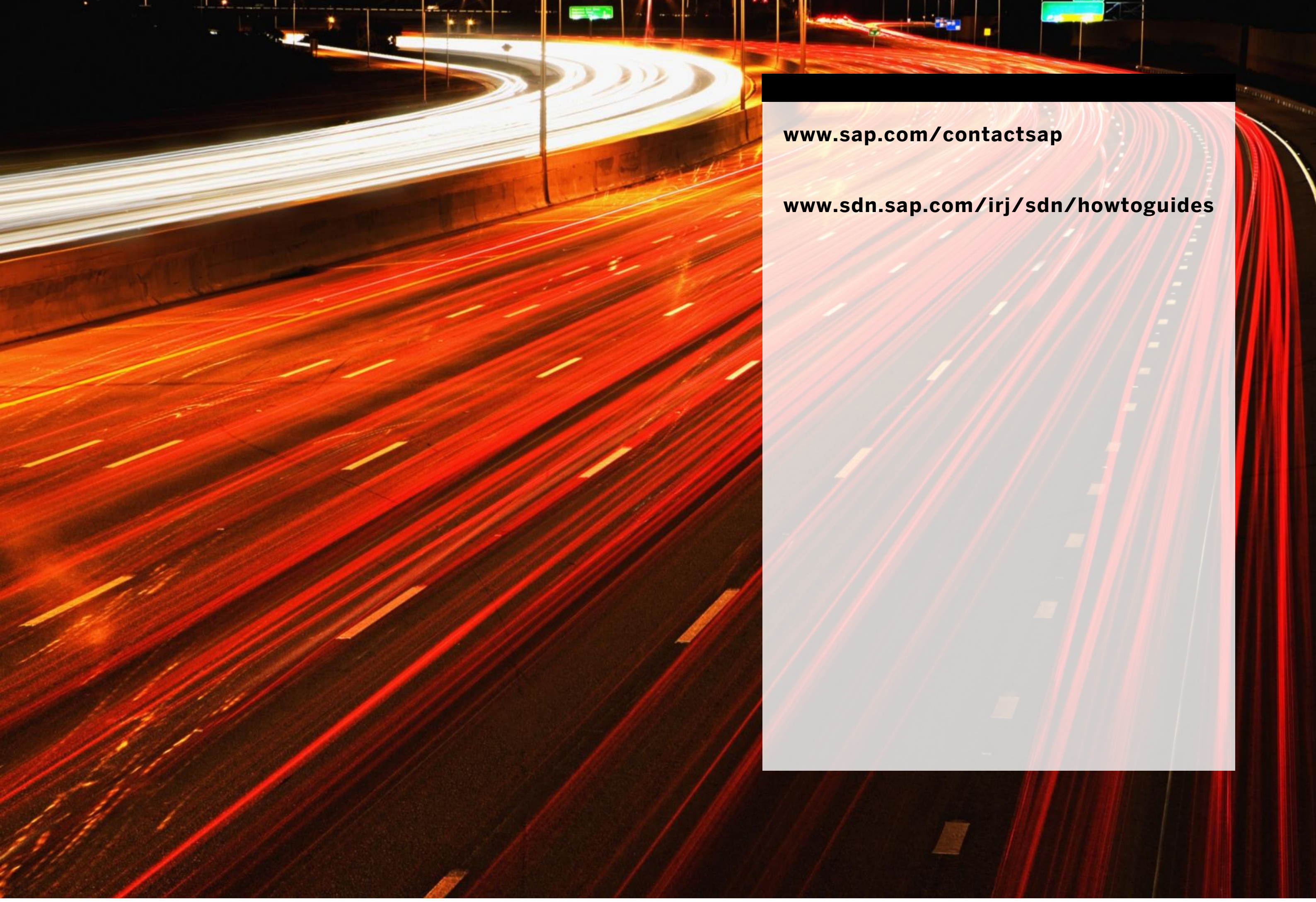
Deployment Target Selection Mode

- ○ Always prompt to pick device
- ○ Launch on all compatible devices/AVD's
- Active devices and AVD's
- ● Automatically pick compatible device: Always uses preferred AVD if set below, launches on compatible de

Select a preferred Android Virtual Device for deployment:

AVD Name	Target Name	Platfor...	API Le...	CPU/ABI
☑ AVD19	Android 4.4.2	4.4.2	19	ARM (armeabi-v7a)

Filter matched 16 of 16 items



www.sap.com/contactsap

www.sdn.sap.com/irj/sdn/howtoguides

The Best-Run Businesses Run SAP™