

*Lab 2: Using Worklight Server, Application Center and
Environment Optimization – Lab Exercise*



Table of Contents

Lab 2 Using the Worklight Server, Application Center, and Environment Optimizations.....	3
2.1 Building and testing on the Android platform	3
2.1.1 Adding a new Worklight environment.....	3
2.1.2 Building and testing the Android application	4
2.1.3 Optimizing for the Android environment.....	6
2.2 Deploying to an external Worklight Server	12
2.2.1 Deploying to an external Worklight Server instance.....	12
2.3 Using the Application Center.....	16
2.4 Summary.....	27

Lab 2 Using the Worklight Server, Application Center, and Environment Optimizations

In Lab 1 you familiarized yourself with building and testing a mobile application using Worklight Studio. In Lab 2 you are going to continue to work with the application in the Worklight Studio, but you will also work directly with the Worklight Server and the Application Center.

In this lab, you will learn how to do the following:

- Create a new Worklight environment in Worklight Studio
- Build and test an application for Android environments
- Use environment optimization capabilities in Worklight
- Use the direct update feature of Worklight Server
- Deploy a Worklight application and adapter to an external Worklight Server instance
- Deploy an application to the Application Center
- Use the Application Center client to install an application on an emulator

2.1 Building and testing on the Android platform

IBM Worklight supports the deployment of mobile applications to multiple different operating system platforms. In Lab 1, you saw how to build and deploy applications to the iOS platform. In this section of the lab, you will use the Worklight Studio to add support for the Android platform.

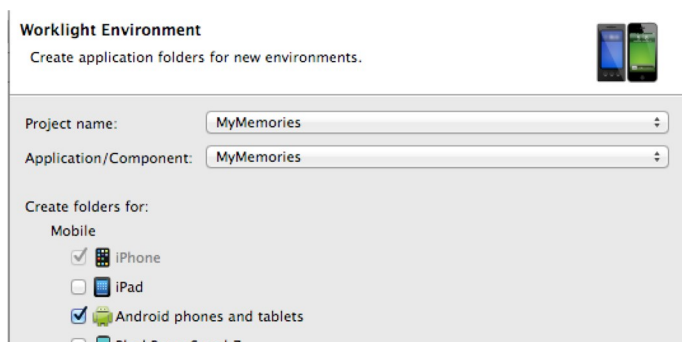
Adding support for an additional platform that is supported by Worklight is an easy task. You start by adding a new Worklight environment for the desired platform, and then you can use the appropriate native tools to produce a device installable for the target platform. In the case of Android, Worklight integrates directly with the Android SDK, which means that you can build the device installable application and launch it on an emulator without leaving the Worklight Studio.

2.1.1 Adding a new Worklight environment

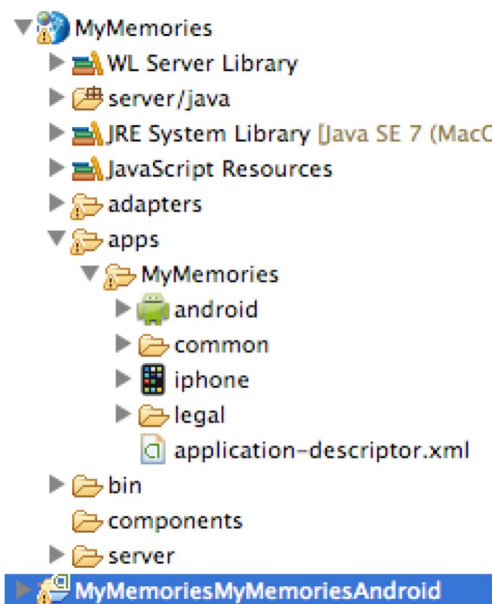
In this part of the lab, you will add a new Worklight environment for Android.

- __1. Add the Android environment
 - __a. If not already expanded, expand the **MyMemories** Worklight project.
 - __b. Expand the **apps** folder and right-click on the **MyMemories** application folder.
 - __c. Click on **New->Worklight Environment**.

- ___d. Select the **Android phones and tablets** checkbox and click **Finish**.



- ___e. After clicking Finish, you will see that a new folder with the name **android** appears under the **MyMemories** application. In addition, a top-level project called **MyMemoriesMyMemoriesAndroid** now exists.



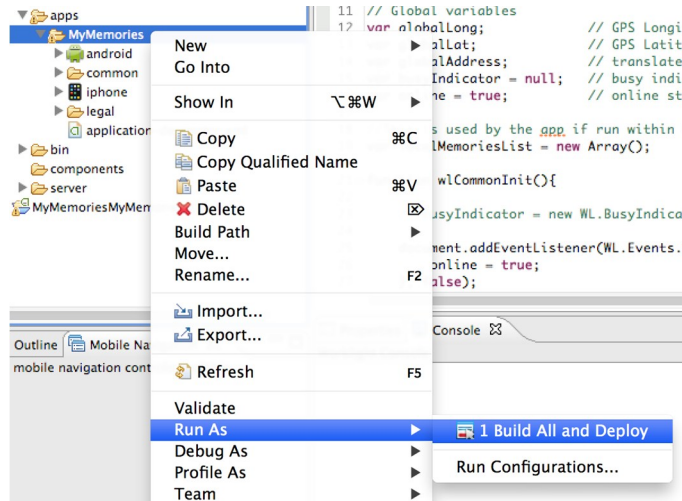
- ___f. The **android** environment folder under the **MyMemories** application folder will hold web and native content specific to the Android platform. The **MyMemoriesMyMemoriesAndroid** project was created automatically because the Android Development Tools (ADT) Eclipse Plugin has been installed into this environment. You can use this project to develop any native (Java for Android) code for your application.

2.1.2 Building and testing the Android application

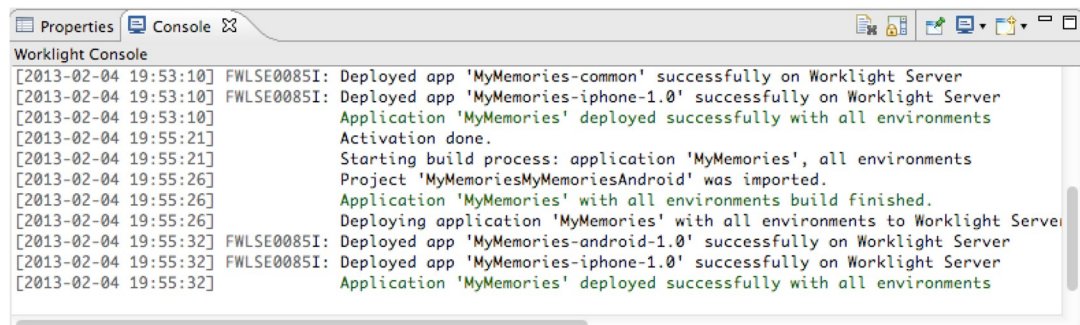
In this part of the lab, you will test the MyMemories application on an Android Virtual Device (AVD).

- ___1. Building and deploying to the Worklight Server

- __a. Before testing the application on an AVD, you need to build and deploy the updated application to the Worklight Server.
- __b. Right-click on the **MyMemories** application folder. Click **Run As->Build All and Deploy**.



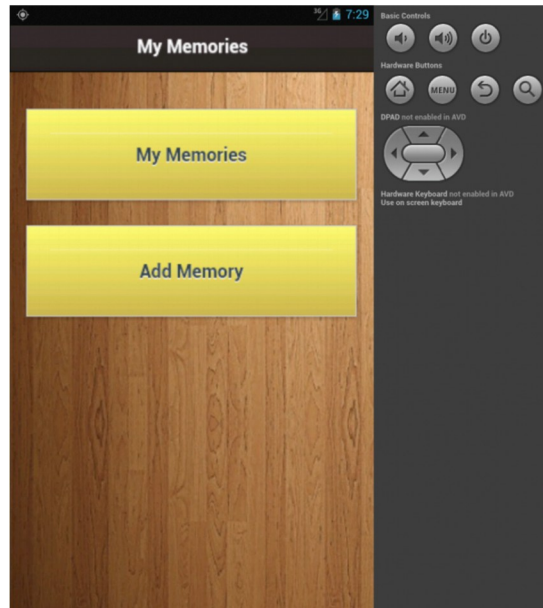
- __c. This will build the application with the new Android environment and deploy the updated artifacts to the embedded Worklight Server. When the build and deploy is complete, you should see success messages in the console view in the Worklight Studio.



__2. Running the application on an AVD

- __a. Right-click on the **MyMemoriesMyMemoriesAndroid** project. Click **Run As->Android Application**.
- __b. This will open the **Android Device Chooser** window. Select the **Launch a new Android Virtual Device** button and then click on the **appropriate** AVD. Click **OK**.
- __c. An AVD instance will launch. It may take several minutes for the instance to initialize. After initialization, you will see the lock screen. Slide the lock to unlock the instance.

- ___d. The MyMemories application will automatically open and start on the AVD instance. If the MyMemories application is not open after unlocking, wait a few moments as the application installation completes. Once the application is installed and opened, you should see the home screen for the MyMemories application (**Note:** The layout of your AVD may vary from the below image, depending on which target platform and API level are configured for you emulator).



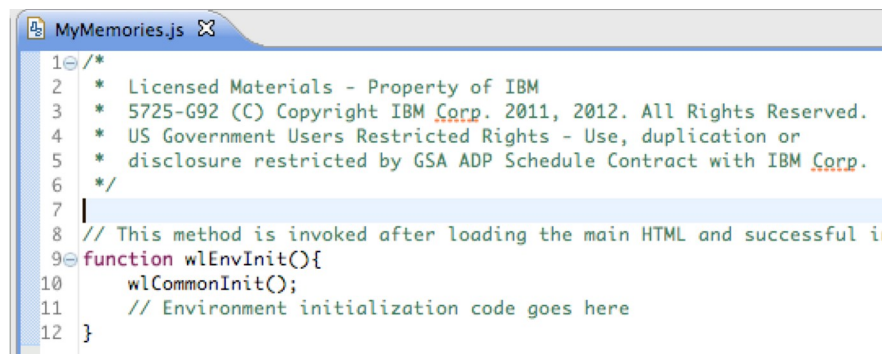
- ___e. You can now test the application on the AVD instance. Navigate to different screens by clicking on any of the icons or navigational buttons.
- ___f. When done testing the application exit the application on the AVD. Do not close the AVD instance as you will be using the application in the next section.

2.1.3 Optimizing for the Android environment

In this part of the lab, you will learn how to make optimizations for the MyMemories application when running on an Android platform. Additionally, you will discover the Direct Update feature of the Worklight runtime.

- ___1. Introducing a JavaScript optimizations
- ___a. You will be introducing JavaScript that is specific to the Android platform. In particular, you will use the Worklight client-side JavaScript API to dynamically create menu items. The menu items will provide an alternative means of navigation within the application so that you can get to any screen from any other screen in the application. The ability to create menu items is only applicable on the Android and Windows Phone platforms as other platforms (such as iOS) do not have the concept of menus.

- ___b. If not already there, open the Worklight Studio. Navigate to the **MyMemories->apps->MyMemories->android->js** directory. This directory holds all application JavaScript that is specific to the Android platform. The environment specific JavaScript directories follow a sparse model whereby developers only write code that overrides or augments what is in the common directory. In this way you can avoid writing duplicate code.
- ___c. Locate the **MyMemories.js** file. The same file also exists in the js directory under the common folder. Currently the project does not contain environment specific code, so the MyMemories.js file under the **android** folder is mostly empty. It contains a single function called **wlEnvInit** that is used to call the common initialization code when the application starts.



```

1  /*
2  * Licensed Materials - Property of IBM
3  * 5725-G92 (C) Copyright IBM Corp. 2011, 2012. All Rights Reserved.
4  * US Government Users Restricted Rights - Use, duplication or
5  * disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
6  */
7
8  // This method is invoked after loading the main HTML and successful i
9  function wlEnvInit(){
10     wlCommonInit();
11     // Environment initialization code goes here
12 }

```

- ___d. During the next few steps, you will be inserting custom code into the MyMemories.js file. If you do not want to edit the code as instructed, you can simply copy the contents of the **Lab2.snippets.txt** file into the MyMemories.js file. If you choose to do this, it is advisable to follow the next steps to get a better understanding of the new code.
- ___e. You will start by declaring a global JavaScript variable that is an array instance. Place the following line of code just before the declaration of the wlEnvInit function:

```
var itemList = new Array();
```

- ___f. Next you will modify the wlEnvInit function. After the wlCommonInit() line in the wlEnvInit function, copy and paste the following code:

```
WL.OptionsMenu.init({opacity: "0.9"});
```

```

    WL.OptionsMenu.addItem('homePage', function()
    {changePage("#homePage");}, 'Go to Home', {image:'', enabled :
true});

    itemList.push('homePage');

    WL.OptionsMenu.addItem('createMem', function()
    {changePage("#cameraPage");}, 'Add Memory', {image:'', enabled :
true});

```

```

    itemList.push('createMem');

    WL.OptionsMenu.addItem('listMem', function()
    {changePage("#listPage");}, 'My Memories', {image:'', enabled :
    true});

    itemList.push('listMem');

    WL.OptionsMenu.addItem('closeApp', function() {WL.App.close();},
    'Exit', {image:'', enabled : true});

    itemList.push('closeApp');

    WL.OptionsMenu.setEnabled(true);

    WL.OptionsMenu.setVisible(true);

    $(document).on('pagechange', handlePageChange);

```

- __g. The majority of the code above deals with using the WL.OptionsMenu API. This API is provided by Worklight and allows you to programmatically add menu items from within your application code. After creating each menu item using the WL.OptionsMenu.addItem API, the code enables the menu and makes it visible using the WL.OptionsMenu.setEnabled and WL.OptionsMenu.setVisible APIs.
- __h. When creating the menu items, a function called changePage is referenced. You need to add that function to the JavaScript file. Copy and paste the following into the MyMemories.js file:

```

function changePage(targetPage) {

    console.log('Changing page to ' + targetPage);

    $.mobile.changePage(targetPage);

}

```

- __i. The changePage function is called when a menu item is clicked and uses the changePage functionality built into jQuery Mobile to navigate between screens in the application.
- __j. The last line added to the w!Env!nit function was \$(document).on('pagechange', handlePageChange). This line registers a callback handler on a page change event. This means that anytime a user navigates from one screen to another, the handlePageChange function is called. This function will take care of ensuring the correct menu items are enabled based on the screen that is currently active in the application. Copy and paste the following two function declarations to the MyMemories.js file:


```
function handlePageChange(arg, obj) {

    enableAll();

    var pageId = $.mobile.activePage.attr('id');

    var disableItem = null;

    if(pageId == 'homePage') {
        disableItem = WL.OptionsMenu.getItem('homePage');
    }
    else if(pageId == 'cameraPage') {
        disableItem = WL.OptionsMenu.getItem('createMem');
    }
    else if(pageId == 'listPage') {
        disableItem = WL.OptionsMenu.getItem('listMem');
    }

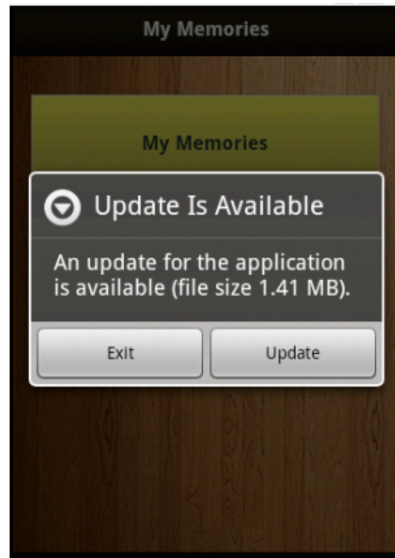
    if(disableItem != null) {
        disableItem.setEnabled(false);
    }
}

function enableAll() {

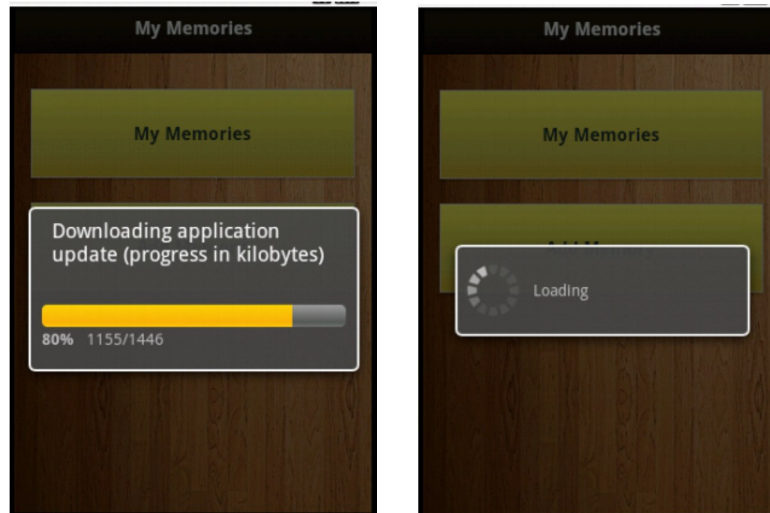
    itemList.forEach(function(item)
    {WL.OptionsMenu.getItem(item).setEnabled(true);});
}
```

}

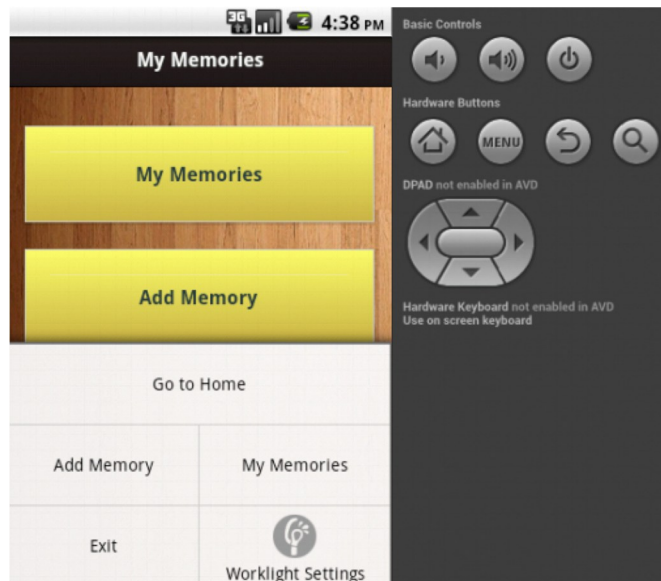
- ___k. Now that the changes are done, you need to build the updated version of the application. To do this, right-click on the **MyMemories** application folder and then click **Run As->Build All and Deploy**. This will build the application package with the changes, and it will deploy the updated application to the Worklight Server. It is important to note that this step does not update the application running on the AVD instance.
- ___l. In order to test the changes, the application contents installed on the AVD instance do need to be updated. You could build and deploy the application to the AVD instance as you did in the previous section of the lab, but since the changes were only made to JavaScript you can also take advantage of the Direct Update capability of the Worklight platform.
- ___m. Go back to the open AVD instance. If the MyMemories application is still open on the AVD instance, exit the application. Return to the application listing on the AVD instance, and open the MyMemories application again.
- ___n. Once the application is open, you should see a prompt to update the application.



- ___o. Click the **Update** button to begin the update process. Once clicked, the Worklight Device runtime will download the updated application contents from the Worklight Server.



- __p. Once retrieved, the existing application contents are cleared from the disk cache, and the new contents are unpacked. When complete, the application is restarted from the new contents.
- __q. To test your changes, you need to bring up the menu on the AVD instance. You can do this by pressing the menu icon on the AVD instance. You should see a menu that has an item for each screen in the application.



- __r. Click on **My Memories**, **Add Memory**, or **Go Home** to navigate to any of the main application screens. You can continue to use the menu to navigate between screens or you can click the **Exit** menu item to close the application.
- __s. When you are satisfied with the changes you can close the AVD instance.

NOTE: In order to run this application on a real android device, you must include the permissions for location and external storage in the AndroidManifest.xml file using these lines (this is not part of the lab, but important to remember for dealing with Android device access permissions).

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

2.2 Deploying to an external Worklight Server

Up to this point you have been using the Worklight Developers Edition which includes a Worklight Server instance embedded in the Eclipse process, and wizards which make local deployment simple. This is ideal for development and unit testing environments, but as you move through the application lifecycle, you will eventually need to deploy the application to an externally deployed Worklight Server instance for testing, staging and production.

Deploying to an external server is accomplished through the server's console, by uploading the application artifacts from the appropriate locations within the eclipse project. The following steps will demonstrate how to prepare and move an app onto a Worklight server. The high level summary is as follows

- Update the server url in the application's application-descriptor.xml file
- Rebuild the application
- Start the external server
- Upload the *.wlapp and *.wladapter files (server-side components of a worklight app) from the Worklight console

2.2.1 Deploying to an external Worklight Server instance

- __1. Update the Worklight application for external deployment
 - __a. Before you deploy the application to the external Worklight Server, you need to update the application-descriptor.xml to take into account the new location of the server instance.
 - __b. In the Worklight Studio, open the **MyMemories->apps->MyMemories->application-descriptor.xml** file.
 - __c. Scroll to the bottom of the file and locate the **worklightServerRootURL** element. This value needs to be updated to point to the external Worklight Server instance. The external instance is also running on your lab machine, but it is running on a different port and uses a different context root. To point the application at this external instance, update the value of that element to **http://{local.IPAddress}:9080/worklight**.

```

application-descriptor.xml
<application xmlns="http://www.worklight.com/application-aescriptor" id="mymemory" platformver
<displayName>MyMemory</displayName>
<description>MyMemory</description>
<author>
  <name>application's author</name>
  <email>application author's e-mail</email>
  <homepage>http://mycompany.com</homepage>
  <copyright>Copyright My Company</copyright>
</author>
<height>460</height>
<width>320</width>
<mainFile>MyMemory.html</mainFile>
<thumbnailImage>common/images/thumbnail.png</thumbnailImage>
<iphone bundleId="com.MyMemory" version="1.0">
  <worklightSettings include="true"/>
  <security>
    <encryptWebResources enabled="false"/>
    <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg, jpeg, gi
  </security>
</iphone>
<android version="1.0">
  <worklightSettings include="true"/>
  <security>
    <encryptWebResources enabled="false"/>
    <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg, jpeg, gi
    <publicSigningKey>Replace this text with the public key of the certificate with wh
  </security>
</android>
<worklightServerRootURL>http://${local.IPAddress}:9080/worklight</worklightServerRootURL>
</application>

```

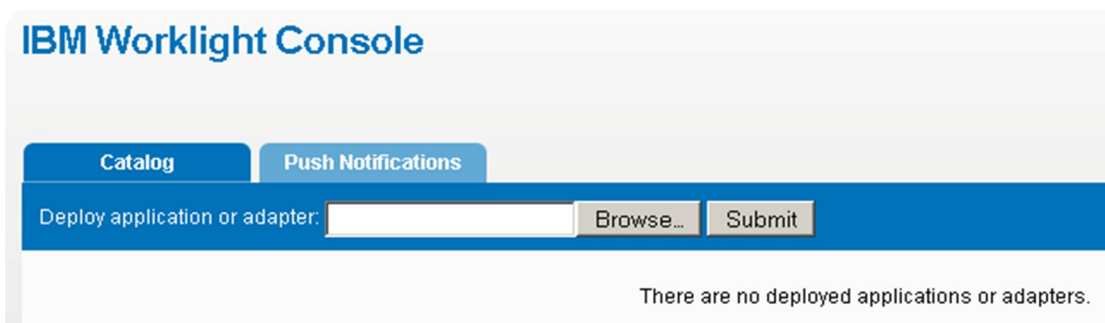
- ___d. Now you need to rebuild the application with the change to the application-descriptor.xml. To do this, right-click on the **MyMemories** application folder and then click on **Run As->Build All and Deploy**.
- ___e. The application is now ready to be deployed to the external Worklight Server instance.

___2. Deploying the application and adapter

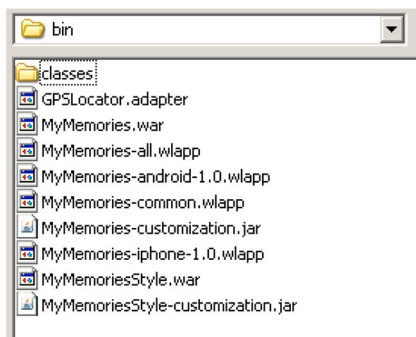
- ___a. Start the external Worklight server by double clicking "StartLiberty.bat" on the desktop:

Name	Date Modified	Size	Type
AppCenter	6/21/2012 7:13 AM	1 KB	Internet Shortcut
Embedded Worklight Console	7/19/2012 2:05 AM	1 KB	Internet Shortcut
Liberty Worklight Console	6/21/2012 7:12 AM	1 KB	Internet Shortcut
StartLiberty.bat	1/18/2013 7:46 AM	1 KB	MS-DOS Batch File
StopLiberty.bat	1/18/2013 7:46 AM	1 KB	MS-DOS Batch File
Worklight Studio	1/21/2013 9:15 PM	1 KB	Shortcut

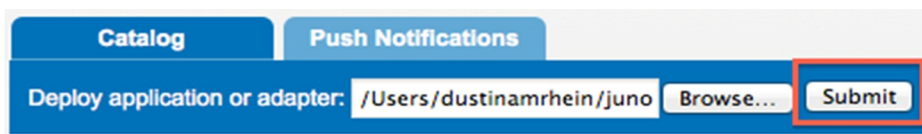
- __b. You will use the Worklight Console to deploy to the external Worklight Server instance. The Worklight Console provides a browser-based interface with which you can manage the Worklight Server.
- __c. Open the Worklight Console by opening the <http://localhost:9080/worklight/console> URL in a web browser (You can also click on “Liberty Worklight Console” in the above picture) and login as wladmin / wladmin. You will be taken to the catalog view of the Worklight Console.



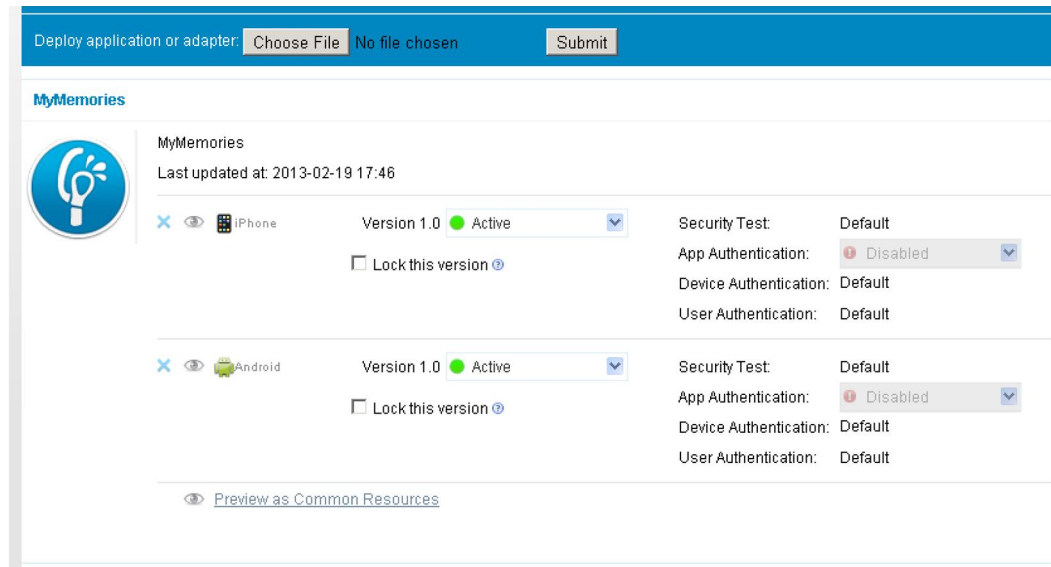
- __d. You will use the file upload dialog to deploy the application and adapter files. Click in the input box or click on the **Browse** button. Browse to the **<WORKSPACE_HOME>/MyMemories/bin** directory. You should see contents similar to the following:



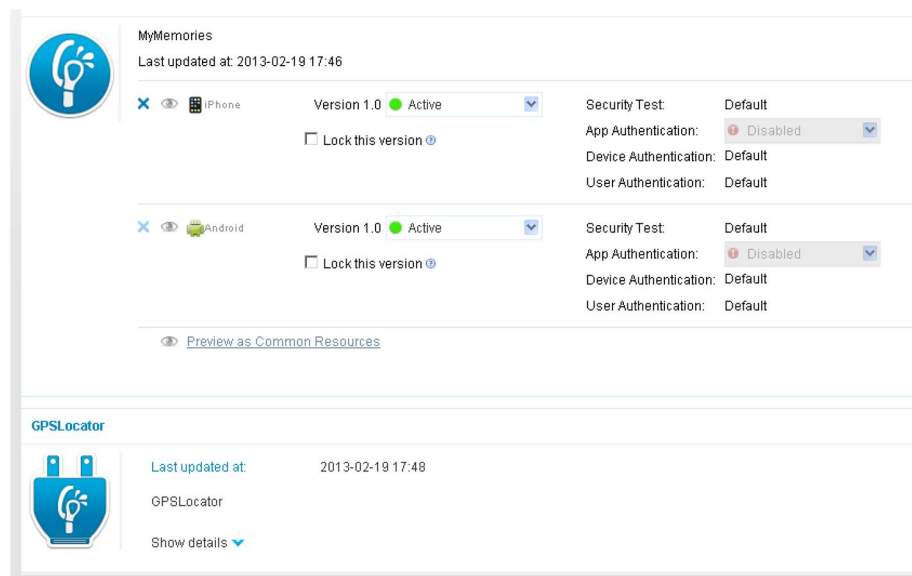
- __e. First select the **MyMemories-all.wlapp** file and then click **Submit** in the file upload dialog.



- __f. The upload will take a few moments, and when done you will see a success message in the Worklight Console. You should also see the MyMemories application in the catalog view now.



- __g. Click the input box or the **Browse** in the file upload dialog again. If not already there, navigate to the **<WORKSPACE_HOME>/MyMemories/bin** directory. Select the **GPSLocator.adapter** file and click the **Submit** button. When the upload completes, you should see another success message and the GPSLocator adapter should appear in the catalog view.



- __h. The application and adapter are now deployed.

__3. Testing the deployment

- __a. Earlier you updated the application-descriptor.xml of the **MyMemories** application to point to the external Worklight Server instance. Now you need to test connectivity to this instance.

- ___b. To do this, you can simply launch the application on an AVD instance like you did earlier in the lab. From the Worklight Studio, right-click on the **MyMemoriesMyMemoriesAndroid** project. Click **Run As->Android Application**.
- ___c. The **Android Virtual Device Chooser** window will appear again, and as you did earlier select the **Launch a new Android Virtual Device** button. Next click on the **MyMemories** AVD and then click **OK**.
- ___d. After the instance initializes, unlock the home screen as you did earlier. The **MyMemories** application should either already be opened or should open in a few moments. You should be able to test the application as you did earlier. If you encounter any errors related to network connectivity or otherwise, please notify an instructor.

2.3 Using the Application Center

In this section you will learn how to use the Application Center. The Application Center is an enterprise app store that is an included part of the IBM Worklight Platform. You can use the Application Center to facilitate the distribution of your mobile applications within the organization. You will learn how to use both the server-side and client-side components of the Application Center in this section.

- ___1. Deploying an application to the Application Center
 - ___a. First you need to verify that the Application Center is running. **Open a different browser window than the browser you used for the Worklight Console.** Make sure that you use different browser types. For instance, if you used Internet Explorer for the Worklight Console, use Firefox for the Application Center. This prevents possible issues with authentication and session confusion.
 - ___b. Open the <http://localhost:9080/applicationcenter> URL in the browser. When the login prompt appears, enter **appcenteradmin** as the username and **admin** as the password and click **OK**.
 - ___c. After logging in, you are taken to the applications view. Here you can see all the applications that are currently deployed to this Application Center instance.

IBM Worklight Application Center Applications Devices Users / Groups Welcome Administrator IBM

You are in: Applications

Application Management

Available Applications

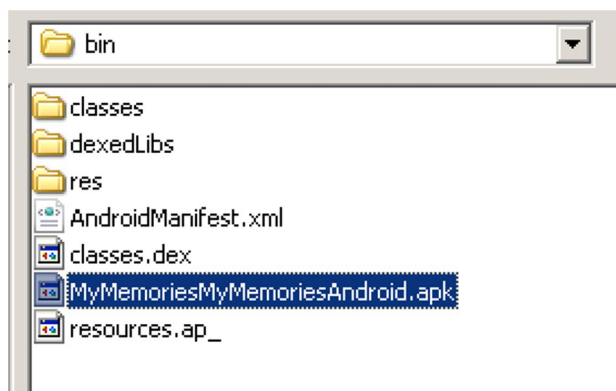
Add Application Display: [List Icon]

1 of 1 Page 1 Previous | Next

Sort by: Label ^ | OS | Update Date Show inactive

IBM App Center
 Android (com.ibm.appcenter)
 Access control: unrestricted
 version 1 | 1/2/13 | ☆☆☆☆☆ (0)

- __d. Currently the only application deployed is the **IBM App Center** application. This is the client-side component of the Application Center. It is a mobile application that is developed using Worklight and provided by IBM. You will work with this application later.
- __e. Click the **Add Application** button above the application listing.
- __f. Click inside the file upload dialog to begin the upload process. The Application Center accepts device installable archives for either Android (.apk files) or iOS (.ipa files). In this case, you will be uploading the Android archive for the MyMemories application.
- __g. Navigate to the `<WORKSPACE_HOME>/MyMemories/apps/MyMemories/android/native/bin` directory. Select the **MyMemoriesMyMemories.apk** file.



- __h. Open the file, and the upload will begin automatically. When the file upload completes, you will see a success message.

Application File

Upload an application file with file extension apk or an ipa.

* File:

 File MyMemoriesMyMemoriesAndroid.apk uploaded

__i. Click the **Next** button to proceed to the details page.

Application Details

Package, Version and Label must be set in the uploaded application package and cannot be modified afterwards.

Package:	com.MyMemories	Identifies the application
Internal Version:	1	Internal version number used to compare versions
Commercial Version:	1.0	Version displayed on the mobile device
Label:	MyMemories	Label of the application as defined by the developer
Author:	appcenteradmin	User who has uploaded this application
Description:	<input type="text" value="IBM POT MyMemories application"/>	(2048 characters maximum)
Recommended:	<input type="checkbox"/>	This application will be listed as a recommended application on the mobile device
Installer:	<input type="checkbox"/>	Indicates whether this application is an installer
Active:	<input checked="" type="checkbox"/>	An active application can be installed on a device
Ready for production:	<input type="checkbox"/>	Indicates whether this application is ready for production

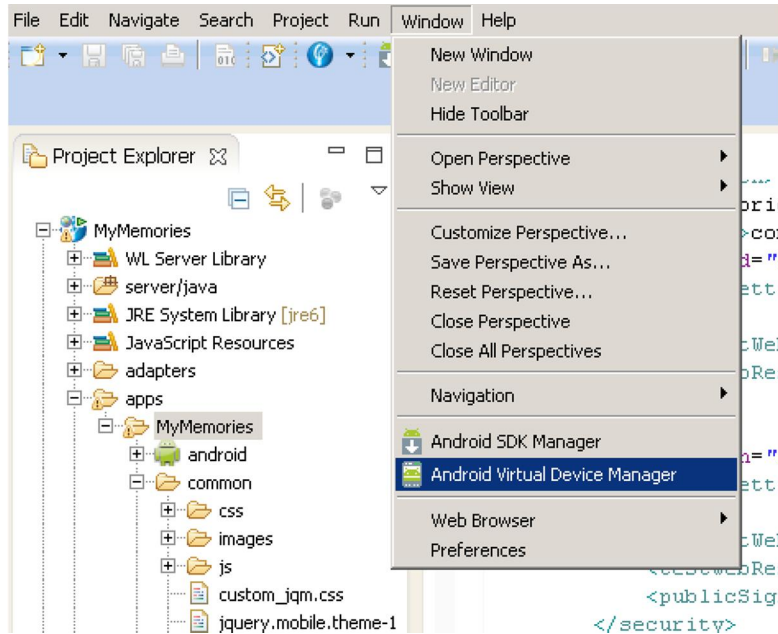
__j. On the Application Details page you will see the package identifier, version information, label, application description, and more. Some of this information (package, version, and label) is determined based on the contents of the uploaded application and is not editable. Other information, such as the description, can be changed. Enter a description in the **Description** field.

__k. There are also four checkboxes on the details page:

- ___i. Recommended: Indicates whether the application will show up in the recommended application view on the mobile client.
 - ___ii. Installer: Indicates whether the application is an installer. If an application is an installer, users can install it by going to a special URL in their device's web browser. As an example, the **IBM App Center** application is an installer.
 - ___iii. Active: Indicates whether the application should be active and eligible for installation on user devices.
 - ___iv. Ready for production: Indicates whether the application is ready for production.
- ___l. Leave the default checkbox selections and click **Finish**. You should now see the MyMemories application on the applications page.

The screenshot displays the 'Application Management' interface. At the top, there is a search bar and a title 'Application Management'. Below this, the section 'Available Applications' is visible. A button labeled 'Add Application' is on the left, and a 'Display' dropdown menu is on the right. The main area shows two application cards: 'IBM App Center' and 'MyMemories'. Each card includes an icon, the application name, the package name (e.g., 'com.ibm.appcenter'), access control ('unrestricted'), and version information. At the bottom, there are pagination controls showing '1 - 2 of 2' items, 'Page 1', and 'Jump to page 1 of 1'.

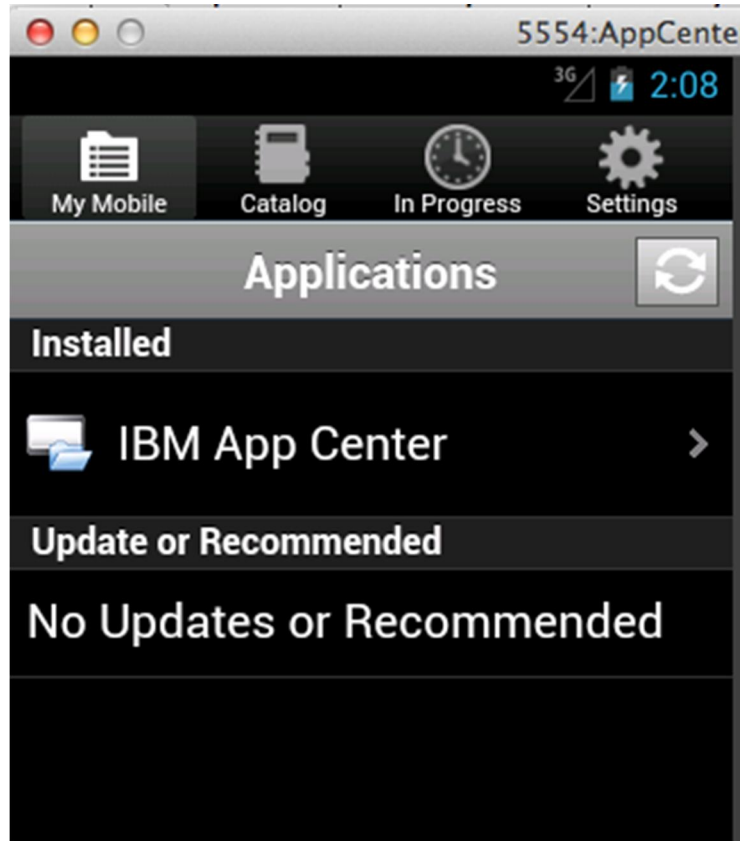
- ___2. Using the Application Center mobile client
- ___a. Now that the MyMemories application is deployed to the Application Center, you can learn how to use the client-side component of the Application Center to interact with the server-side component.
 - ___b. You need to open a new AVD instance. To do this, return to the Worklight Studio and click **Window** on the top toolbar. Click on the **Android Virtual Device Manager** link.



- ___c. In the Android Virtual Device Manager window, select the **AppCenter** AVD and then click the **Start** button to start an instance of this AVD. After the instance initializes, unlock the screen if necessary. Navigate to the application listing. If you are unsure of how to do this on the AVD, ask an instructor. (Note: This AVD instance does not currently have the MyMemories application installed. You will install it using the Application Center mobile client during this section).
- ___d. Locate and open the **IBM App Center** application.
- ___e. The application will open to the **Settings** page. The Settings page allows you to specify the location of the server-side Application Center instance with which the client should interact. It also allows you to enter credentials that will be used to authenticate the client with the Application Center instance. Use **appcenteradmin** as user name and **admin** as the password. Update the Server address with the actual **IP address** of your virtual machine (do not use localhost – find the IP address from a Windows Command Prompt in the VMware image using the 'ipconfig' command or use the default Android emulator host address of 10.0.2.2). Once the form is completed, click **Connect**.

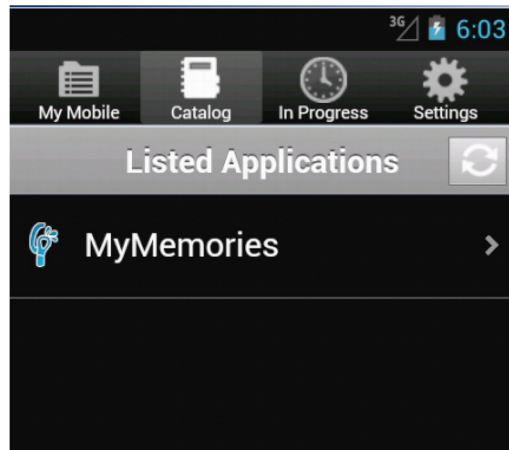


- __f. You will be taken to the home screen of the mobile application where you can see all the applications that were currently installed on the device from the Application Center.

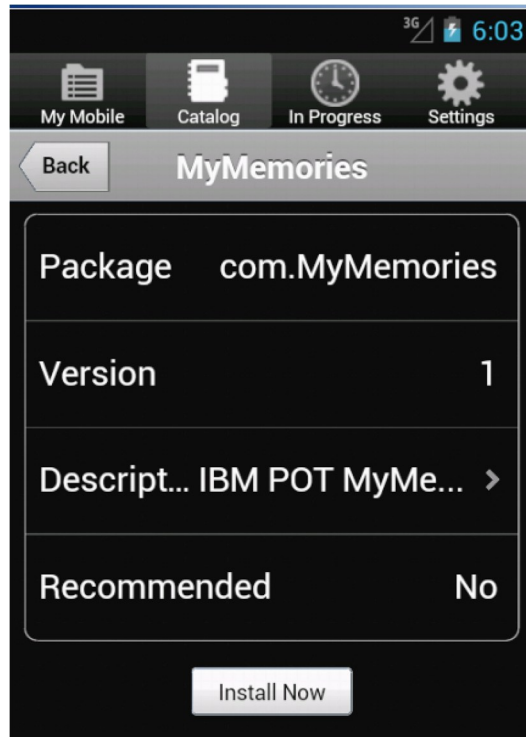


- ___g. Currently the **IBM App Center** application appears in the listing. It was installed and saved in this instance prior to the lab. A special URL that allows access to installer applications in the Application Center was used to load the IBM App Center application on the device.

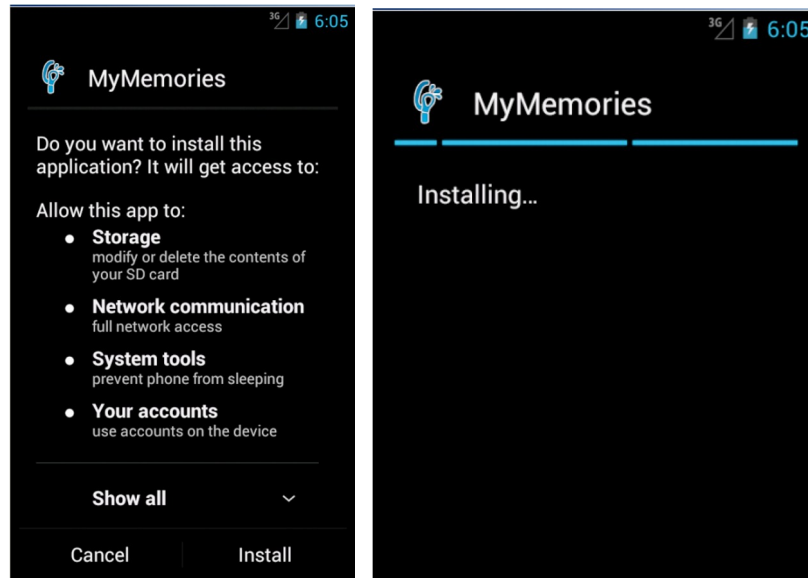
- ___h. Click on the **Catalog** icon in the tab bar at the top of the screen. This screen displays the list of applications in the Application Center to which the authenticated user has access. In this case, you will see the **MyMemories** application that you uploaded earlier.



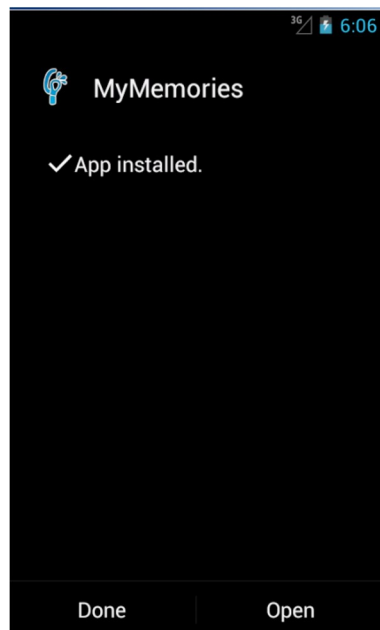
- __i. Click on the **MyMemories** link to go to the application details screen.



- __j. On the details page, you can see the package name, version, description, and whether or not the application is recommended. In this case the value of the **Recommended** field is **No** simply because when you uploaded the application to the server-side Application Center instance, you did not check the **Recommended** checkbox.
- __k. You can also initiate the installation of the application from the details page. Click on **Install Now** to begin the installation of the **MyMemories** application.
- __l. You will briefly be taken to the **In Progress** screen. This screen shows all applications in the progress of being installed. After a very short time, you will see the installation prompt from the Android operating system.

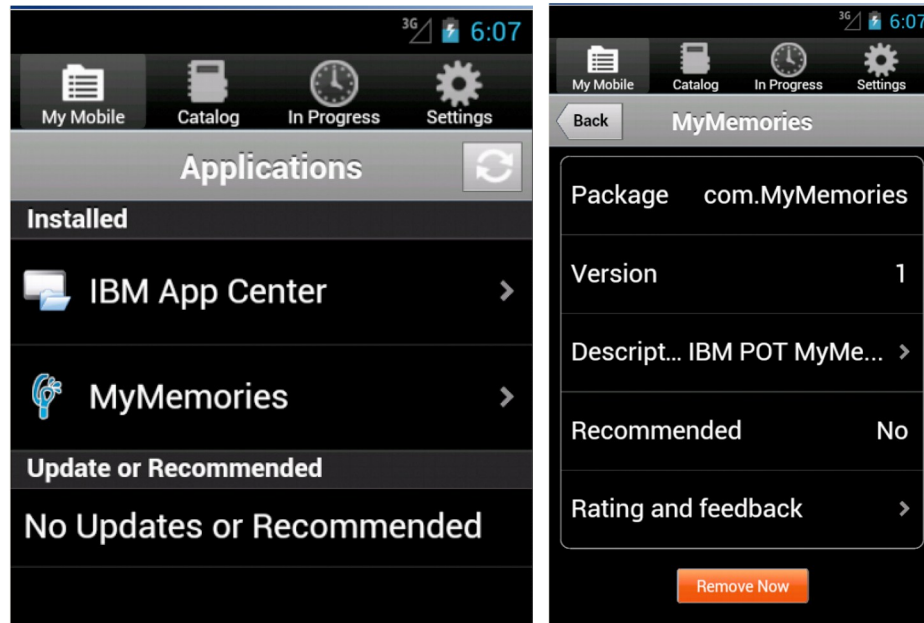


- ___m. Click **Install** to initiate the installation process. Once you click the install button, the application archive is downloaded from the Application Center and then installed on the device. When the install completes, you will see the success screen.

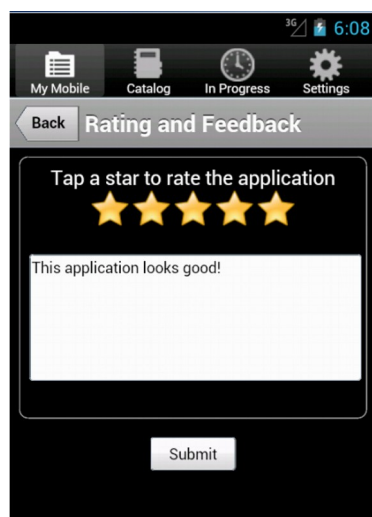


- ___n. Now click **Open** to open the **MyMemories** application that was just installed. You will see the same MyMemories application that you used previously in this lab, and you can test it in the same manner as you did earlier in the lab.
- ___o. Return to the application listing on your AVD instance and open the **IBM App Center** application again.

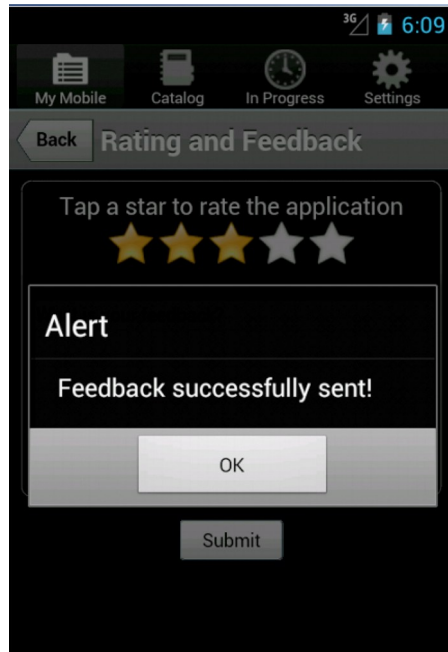
- ___p. In addition to enabling users to install and uninstall applications, the mobile client for the Application Center also allows users to rate and provide feedback for applications. After the application opens, click on the **My Mobile** icon in the top tab bar. You should now see the **MyMemories** application under the list of installed applications. Click on it.
- ___q. This is the details page for the installed applications. It looks similar to the details page you saw for the **MyMemories** application before installing it, but there are two new items. There is a link to provide ratings and feedback, and there is a button to initiate the removal of the application.



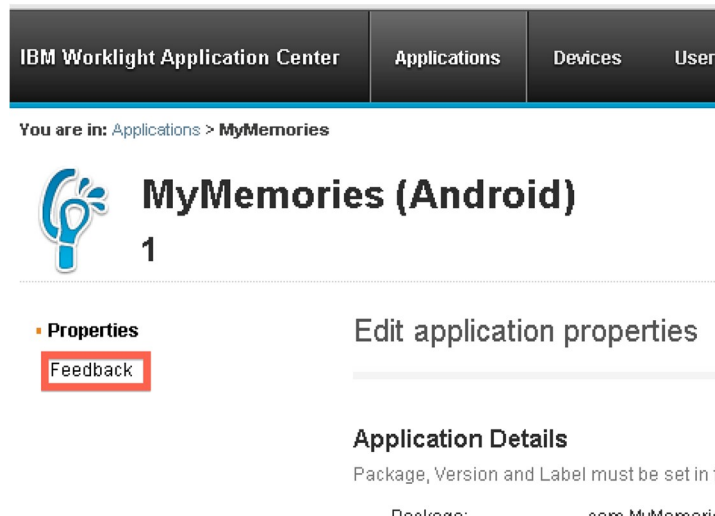
- ___r. Click on the **Rating and feedback** link. Select a rating from one to five stars, and then enter comments in the feedback section.



- __s. Click the **Submit** button to send the rating and feedback to the server-side Application Center instance. You will see an alert indicating the feedback submission was a success.




- __t. Now return to the Application Center in the browser. If you closed the window, the URL is <http://localhost:9080/applicationcenter>. If prompted to authenticate, enter **appcenteradmin** as the username and **admin** as the password.
- __u. Click on the **MyMemories** application on the main page. On the details page, click on the **Feedback** link on the left side of the screen.



- __v. On the feedback page, you should see the rating and the comments you provided using the mobile client.

You are in: Applications > MyMemories

 **MyMemories (Android)** 1

Properties Application Feedback

Feedback

Average rating: ★★★★★

1 of 1 Page 1

Sort by: [Rating](#) | [Creation Date](#) ▾

appcenteradmin commented on 2/19/13 6:08 PM ★★★★★

This application looks good!

___w. You have completed this section of the lab. You can close the AVD instance you used for this portion of the lab.

2.4 Summary

Congratulations! You have completed Lab 2.

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.