MobileFirst Quality Assurance for Hybrid applications

Overview

This tutorial is a continuation of the IBM Mobile Quality Assurance setup step.
If you haven't done it yet, please do so now (../overview/).

This tutorial explains how to install and configure the Mobile Quality Assurance client JavaScript hybrid SDK for MobileFirst Platform Foundation, and how to enable apps to send back bug reports and feedback.

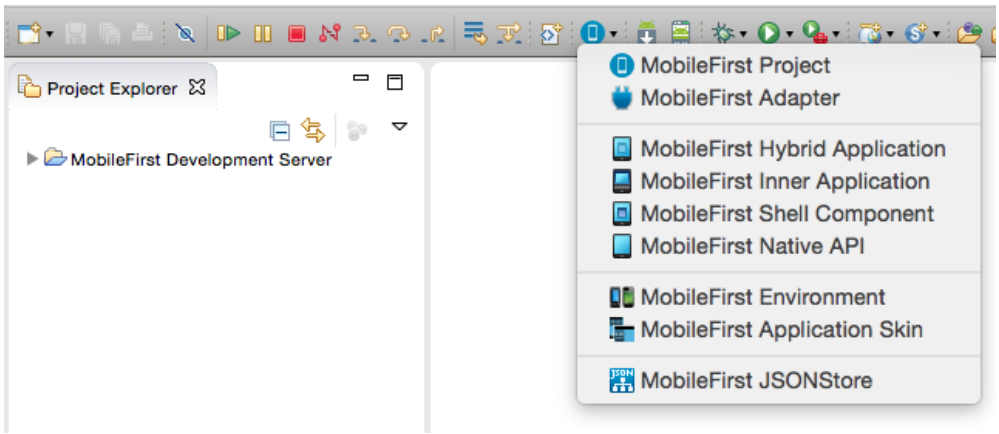
Jump to:

- Creating a simple hybrid app
- Installing the libraries
- Configuring how MQA communicates with your app
- Testing your app
- Further discovery

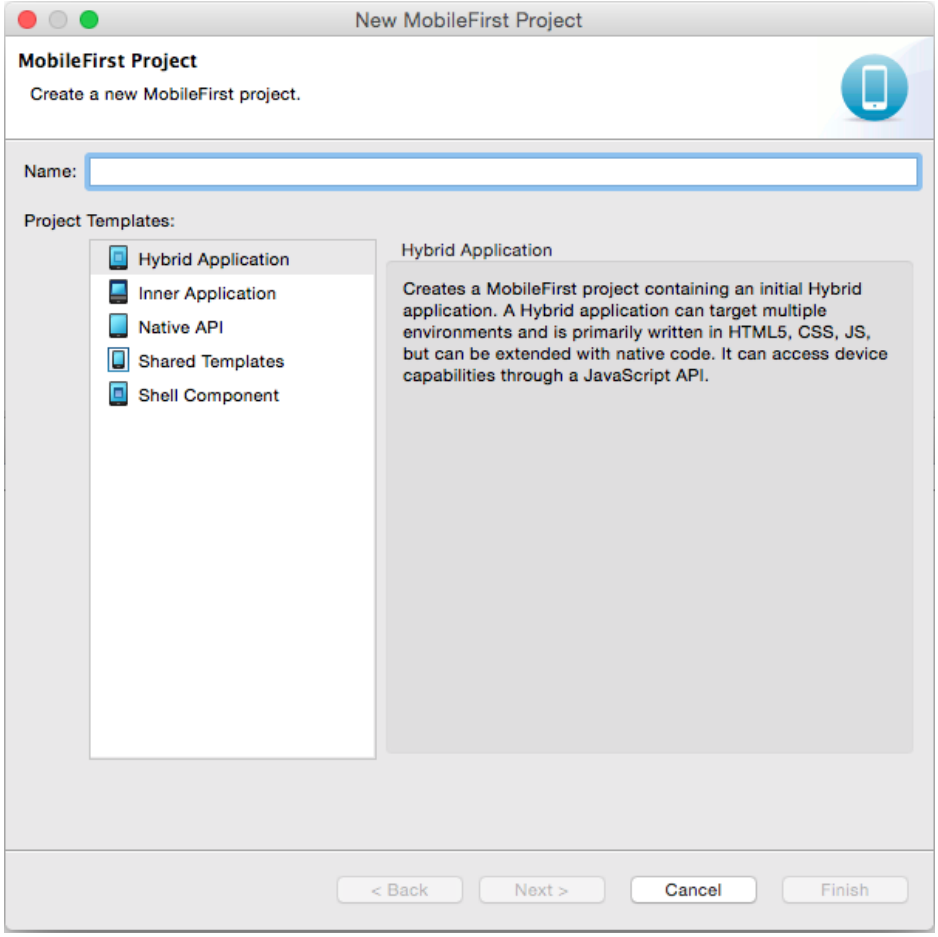
Creating a simple hybrid MobileFirst application

If you do not already have a hybrid app to use for this procedure, create a Hello World app by completing the appropriate procedures after installing MobileFirst Project Studio into your Eclipse IDE:

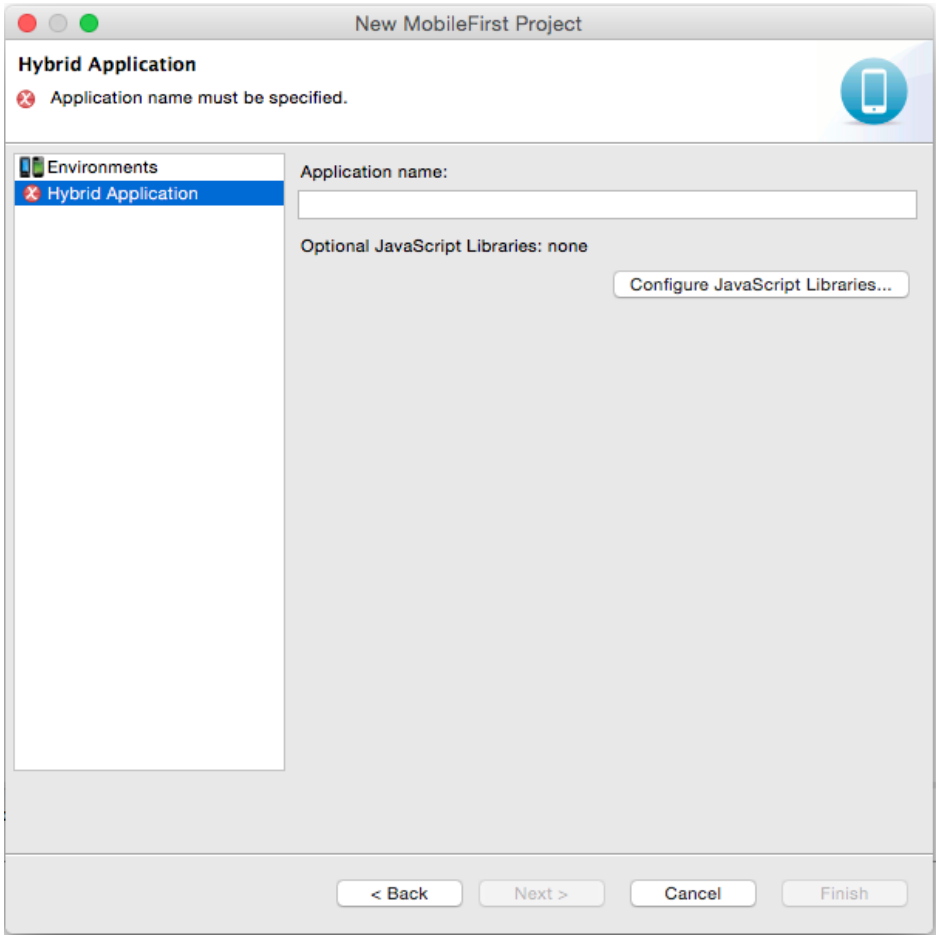
1. Install MobileFirst Studio in your Eclipse IDE by selecting **Eclipse Marketplace** from the Help menu.
2. Locate and install **IBM MobileFirst Platform Studio**.
3. In MobileFirst Studio, select **File > New > MobileFirst Project** to create a new MobileFirst Project from the menu bar.



4. Name your project, for example *HelloWorldProject*, and select the **Hybrid Application** template.



5. Name your application, for example *HelloWorld*.



6. Select **Environments** to select and install the required structure for the platforms, such as Android or iPhone.

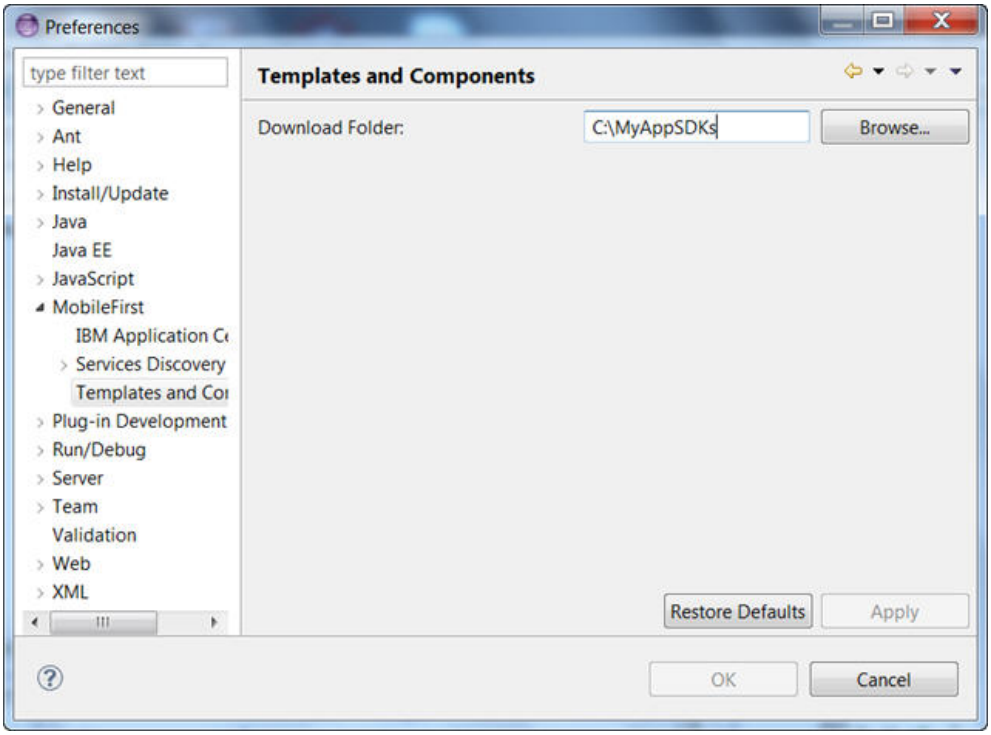
7. Click **Finish** when complete.
8. After the app builds, you can preview it by right-clicking the **Android** platform for your app in the Project Navigation pane and selecting **Run as** > **Preview**.

Installing the libraries

The libraries support both preproduction mode and production mode. Use preproduction mode when you want internal testers to report bugs and feedback with detailed information and have apps under test automatically report crashes. Use production mode when the app is publicly available and you want to gather information after it is released. Complete these steps to install the libraries in preproduction mode:

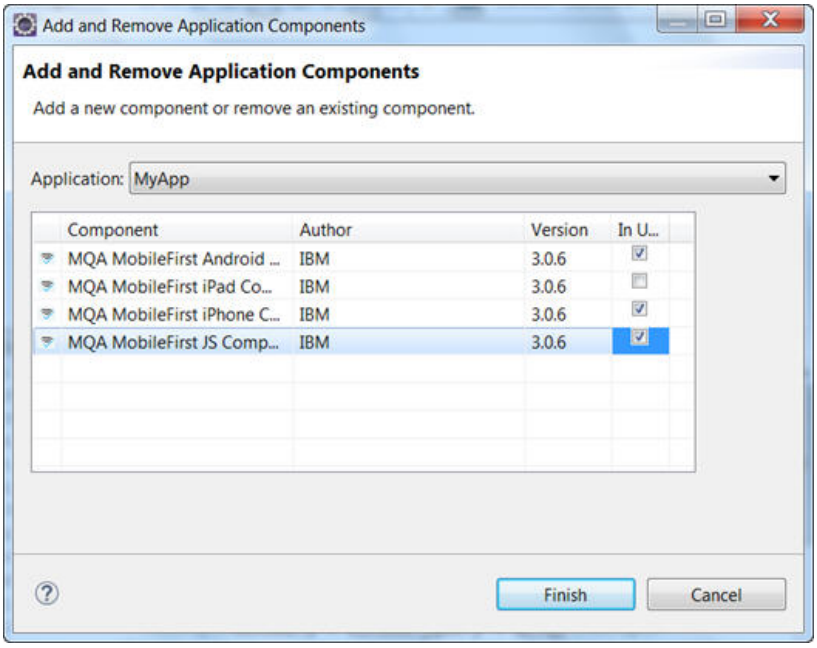
Download the libraries

1. Download the required libraries from Device SDKs for Mobile Quality Assurance for Bluemix (http://www.ibm.com/support/docview.wss?uid=swg27044490):
   - MobileFirst Hybrid JavaScript SDK component (required for all JavaScript hybrid apps)
   - MobileFirst Hybrid Android SDK component (required if your app runs on Android)
   - MobileFirst Hybrid iPhone SDK component (required if your app runs on iPhone)
   - MobileFirst Hybrid iPad SDK component (required if your app runs on iPad)
   - Android SDK for ADT (Eclipse) (required if your app runs on Android)
   - iOS SDK (required if your app runs on iOS)
2. Set your MobileFirst Studio file location preferences.
   When you add the JavaScript components and native platform SDKs to your project, you have to specify where these files are so that MobileFirst Studio can find them.
   1. From the Windows menu in the MobileFirst Studio interface, select **Preferences**.
   2. Click **MobileFirst** > **Templates and Components**
   3. Ensure that the folder where you plan to save the library files is the folder that is specified in the Download Folder field.



Install the Mobile Quality Assurance JavaScript and native platform components

1. From the MobileFirst Studio Project Navigator view in the Design perspective, right-click the HelloWorld project and select **Add/Remove Application Components**.
2. Select the JavaScript component and any operating system components on which you your app runs.



3. Click **Finish**.
4. Click **OK** until you see a confirmation that the components were added.
5. Click **OK** to close the window.

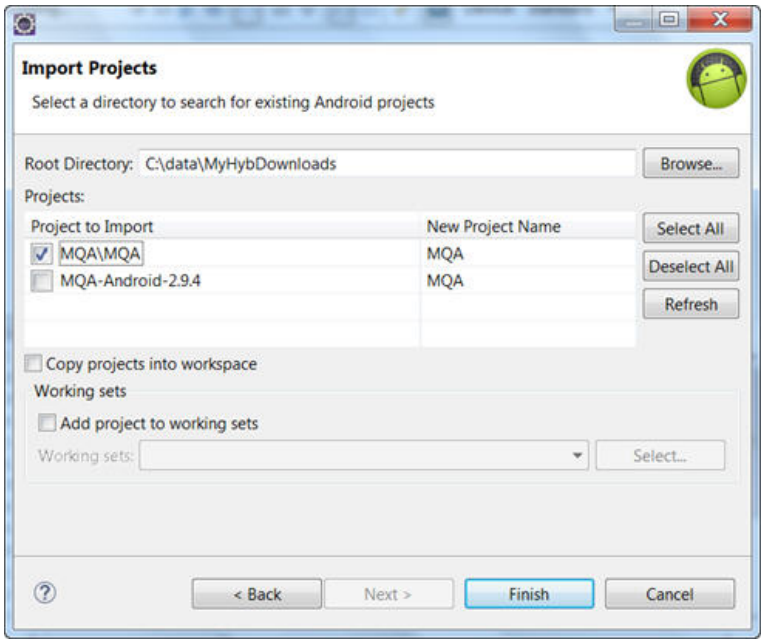
Add the required scripts to the project

In the section of the *project_name/apps/app_name/common/index.html* file, add the following lines:

```
<script src="js/tracekit.js" type="text/javascript"></script>
<script src="js/MQA.js" type="text/javascript"></script>
```

Add the required native SDKs to the project

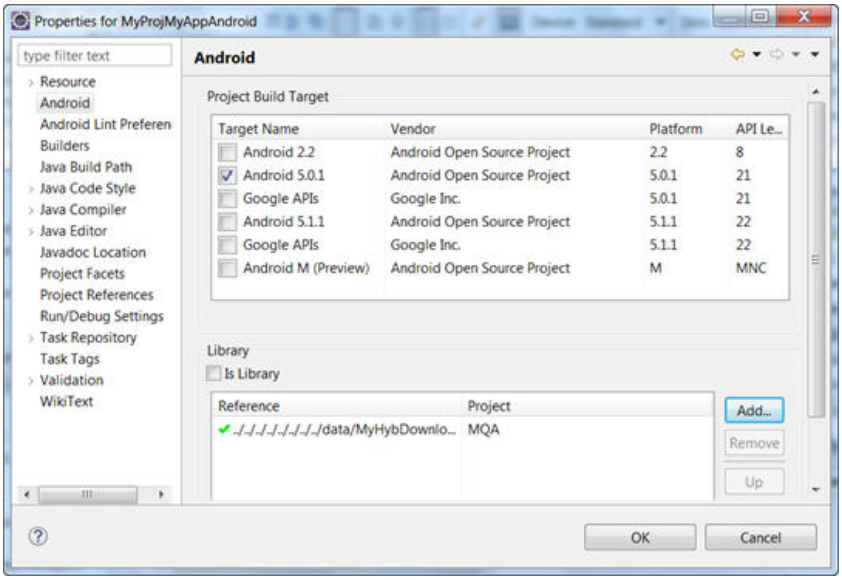
- Android
  1. Extract the native Android SDK for ADT zip file that you downloaded with the JavaScript component.
  2. In the File menu, select **Import...**.
  3. Expand the Android options and select **Existing Android Code into Workspace**.
  4. Browse to the folder where you extracted the Android SDK and select the MQA project.



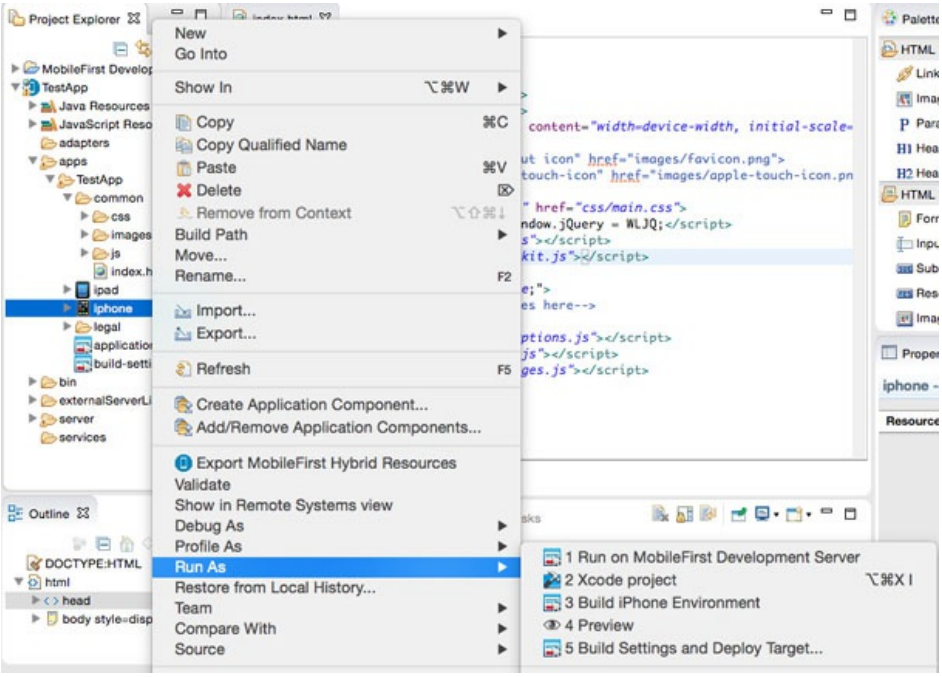
  5. Click **Finish** to import the content.

Link the MQA project your Android project as a library

1. In the Project Explorer, right-click the Android project environment for your project and select **Properties**. The Android project environment name for this example is called *HelloWorldProjectHelloWorldAndroid*.
2. Select **Android** in the navigation.
3. Click **Add** in the library section.
4. Add the Mobile Quality Assurance library, and click **OK**.
5. Click **OK** to close the window and update the environment.



- iOS (requires Xcode)
  1. Extract the native iOS SDK that you downloaded with the JavaScript component.
     Important: If you are using Xcode 7.x on iOS 9 and are using the Mobile Quality Assurance iOS SDK version 2.4.1 or earlier, you must set the Enable Bitcode setting to *No* in the Xcode Build Settings.
  2. If you are using MobileFirst Studio, open the iPhone section of your project in Xcode by right-clicking the platform in the Project Navigator.
  3. Select **Run as...** > **Xcode project**. You must be working on a device with the iOS platform and have Xcode installed to open the file with Xcode.



  4. Add the native iOS platform SDK (Q4M.framework) to the project by dragging it to the Frameworks folder in your Xcode Project Navigator.

Link the required files to your Xcode project:

1. Click your project in the Xcode Project Navigator.
2. Select **Build Phases**.
3. Click **Link Binary With Libraries**.
4. Link the following libraries:
   - `AssetsLibrary.framework`
   - `AudioToolbox.framework`
   - AVFoundation.framework
   - CoreData.framework
   - `CoreLocation.framework`
   - CoreMedia.framework
   - `CoreMotion.framework`
   - `CoreTelephony.framework` (for devices running iOS versions that are earlier than 4.0).
   - `CoreText.framework`
   - CoreVideo.framework
   - MediaPlayer.framework
   - `QuartzCore.framework`
   - `Security.framework`
   - `SystemConfiguration.framework`

Configuring how MQA communicates with your app

For your app to send data to Mobile Quality Assurance, you must have an application key from the MQA service in Bluemix. For more information about application settings, see Registering apps. If you already have an account, you can find your application key by clicking the settings icon (gear) and then selecting App Settings.

Configure MQA to start a new Mobile Quality Assurance session each time your app starts in the JavaScript file

1. Locate the main.js JavaScript file in the `app_name/common/js` folder.
2. Place the following function and required parameters inside the `wlCommonInit()` method immediately after this comment: *//Common initialization code goes here.*

```
MQA.startNewSession(
  {
    mode: "QA",
    //or mode: "MARKET" for production mode
    android: {
      appKey: "your_MQA_Android_appKey" ,
      notificationsEnabled: true
 },
    ios: {
      appKey: "your_MQA_iOS_appKey" ,
      screenShotsFromGallery: true,
        }
 },
     {
    success: function () {console.log("Session Started successfully");},
    error: function (string) { console.log("Session error" + string);}
 }
);
```

> To learn more about starting a Mobile Quality Assurance session, including details about the different session configuration options, see Starting IBM MobileFirst Platform Foundation quality assurance sessions (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_start_worklight_session.html).

3. Build and run your app in an emulator by right-clicking the Android project environment (not the project) and selecting **Run as...** > **Android Application**.

Testing your app

After the environment is set up and working with the emulator, you can load your app on your device and start testing it. Complete the following steps to report a bug or provide feedback during testing:

1. Shake your mobile device lightly.
2. Select **Report a bug** or **Give feedback**.

3. Enter the required information and select **Send** to send the information.
4. View the bug reports and feedback in your Bluemix MQA instance where you retrieved the AppKey.

For more information about reporting bugs, see Reporting bugs (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_ReportingBugs.html). For more information about submitting feedback, see User feedback (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/c_UserFeedback.html).

Further discovery

After you are comfortable with the basics of setting up MQA with your app, you can read content in IBM Knowledge Center to learn and explore some of the other things that you can do. For example:

- Managing users and testers (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_ManagingTesters.html). You can manage the roles and access of your team as they apply to your app and MQA instance.
- Distributing and managing builds (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_DistributingBuilds.html). You can distribute test builds and production builds to the appropriate audience.
- Managing reports (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/t_ManagingReports.html). You can automatically distribute test builds and production builds to your audience.
- User sentiment analytics (http://www.ibm.com/support/knowledgecenter/SSJML5_6.0.0/com.ibm.mqa.uau.saas.doc/topics/UserSentiment.html). You can use MQA to analyze and monitor user comments and information about your app. You can also compare the analysis of those comments with the analysis of the user feedback from similar apps.