IBM MobileFirst Platform compared to "do-it-yourself"

Contents

This document will reveal how the IBM® MobileFirst Platform Developer Edition plug-in compares to unsupported "do-it-yourself" options. This document focuses on tools for the development of web, hybrid and native mobile applications.

The IBM Competitive Project Office compared IBM MobileFirst Platform Developer Edition 6.1 to an option based on Eclipse integrated development environment (IDE), Android software development kit (SDK), and popular frameworks such as jQuery, Jersey (for REST services), Apache Cordova and other frameworks.

You will see some similarities between the solutions. Both the MobileFirst Platform solution and the "do-it-yourself" solutions are free to download and use, are based on similar open source components, allow the creation of different types of mobile applications on different target platforms, and can deploy and debug apps on platform emulators.

However, the similarities end here. MobileFirst Platform offers many advantages in comparison to a "do-it-yourself" option, including:

- **Installation and configuration**. MobileFirst Platform offers a richer, centralized set of documentation to install and configure the various solution components.
- **Design and development**. MobileFirst Platform offers a more-productive set of design and development features. Use wizards, visual user-interface (UI) editors, pre-packaged UI frameworks and JavaScript application programming interface (API) for common UI controls. Add platforms "on-the-fly" (while you continue to promote code reuse).

- **Back-end connectivity**. MobileFirst Platform offers easier back-end services discovery. MobileFirst Platform offers easier creation and easier testing of adapters to intermediate front-end and back-end communication sources and destinations.
- **Deployment in development environments**. The capabilities of MobileFirst Platform enable mobile apps and adapters deployment in embedded development server environments (managed using the IBM MobileFirst Platform Console). Developers can quickly deploy the application for testing purposes before they make the build available for deployment and testing in quality-assurance (QA) and production environments.
- **Debugging and testing**. The debugging and testing features of MobileFirst Platform offer you the advantage of using the Mobile Browser Simulator, SDK emulators and real devices. Use the Mobile Browser Simulator as a fast and more-productive way to test the application, rather than relying on the typical slowness of emulators and the expensive ownership and maintenance of multiple real devices. MobileFirst Platform also has an optional, free component called Mobile Test Workbench for MobileFirst Platform that you can use for automated functional testing of mobile applications on emulators and real devices.

Overview

Take time to delve into the specific project components of interest and discover the similarities and differences between an unsupported "do-it-yourself" (DIY) option for mobile development and IBM MobileFirst Platform solution components. This examination is necessary because the decision to proceed with a DIY mobile development that your IT organization will encounter has an immediate consequence: figuring out what components you require, then installing, configuring, integrating, and maintaining these specific components.

While IBM MobileFirst Platform Developer Edition is built on top of the same well-established open source components as a DIY solution, everything you need is either pre-packaged with the MobileFirst Platform installation, or well-documented and referenced for additional download, installation and configuration.

In addition, MobileFirst Platform extends those open source core capabilities to offer a superior platform for development and delivery of mobile enterprise applications. Ready for immediate use, MobileFirst Platform Developer Edition (Eclipse IDE plug-in) contains the IBM MobileFirst Platform Studio and the IBM WebSphere® Application Server Liberty Profile Edition. These two components offer extra capabilities over the DIY option in the areas of design, development, deployment, debugging and testing of mobile applications (as seen in the *Advantages of IBM MobileFirst Platform* section of this document).

Solution components

Before examining the various components in detail, first consider the major building blocks for each solution.

Figure 1 shows the major building blocks for the DIY solution, consisting of the Eclipse IDE, the Android SDK and development tools, Tomcat server to host backend services, and a REST library to access the backend service.

Figure 2 shows the major building blocks for the MobileFirst Platform solution, consisting of the MobileFirst Platform Studio and Android SDK for development, device runtime library, the IBM MobileFirst Platform Server runtime (which, among other things, uses the WAS Liberty Profile to run adapters responsible for connecting with backend services), and the Tomcat server to host backend services.

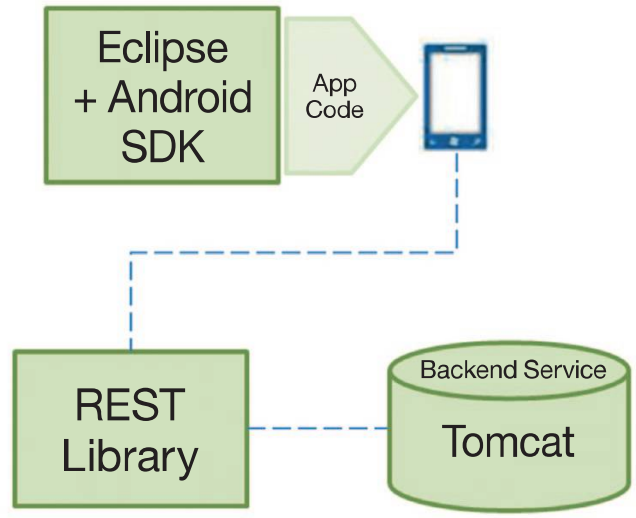

Figure 1. Major blocks of the DIY solution

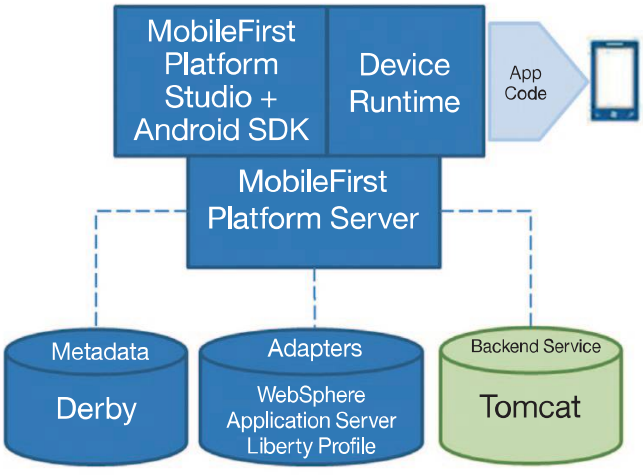

Figure 2. Major blocks of the IBM MobileFirst Platform solution

The IBM MobileFirst Platform Developer Edition 6.1 is a lightweight, simplified free version of IBM MobileFirst Platform Enterprise Edition. This edition contains full code fidelity and most of the components that the enterprise version of MobileFirst Platform possesses, along with access to IBM world-class, worldwide technical support.

Note: Other versions of the MobileFirst Platform family are: IBM MobileFirst Platform Consumer Edition and IBM MobileFirst Platform Enterprise Edition.

The IBM Competitive Project Office team compared IBM MobileFirst Platform Developer Edition v6.1 to a DIY option based on Eclipse, Android SDK, and other popular components. Both environments that the IBM team compared have:

- Eclipse Juno v4.2
- Android Development Toolkit (ADT) plug-in for Eclipse v22.3.0
- Android SDK v22.3
- JDK 1.7.0_45 and JRE7
- Apache Tomcat 7.0.47
- Jersey 1.17.1 (implementation of JAX-RS or JSR 311 spec)
- JQuery Mobile 1.3.2 library
- PhoneGap 2.9.0

The MobileFirst Platform Developer Edition is installed as a plug-in to the Eclipse IDE. The development environment is MobileFirst Platform Studio. MobileFirst Platform Studio extends Eclipse's basic tools with various features provided by the plug-in and MobileFirst Platform Studio integrates with the various SDKs of the supported devices.

Among other features, MobileFirst Platform is delivered with an embedded development server (IBM WebSphere Application Server Liberty Profile) to deploy applications and adapters.

Only JQuery core is part of the MobileFirst Platform Studio installation—JQuery Mobile must be downloaded and installed separately. IBM Dojo Mobile tools are also available to install, but it is not in scope of this paper.

The Cordova framework is available with the MobileFirst Platform installation. PhoneGap was used in the DIY solution; however, Cordova and PhoneGap have similar content. Cordova is an open source project under the Apache organization. PhoneGap is a distribution of Cordova provided by Adobe.

Types of mobile applications

As part of this research, the IBM team created sample mobile apps to experiment and compare each solution from a development perspective. Below, you will find a definition of the common types of mobile applications.

Mobile web apps

In general, these are websites that look and feel like mobile applications. Mobile web apps can be built in HTML5 as single page applications, and simulate moving from device page to page with the use of anchors. These apps run on a browser on the device, but there could be no visible browser buttons or bars, which would make it hard to distinguish it from a hybrid or native app. Users access the app by navigating to a URL and can add a bookmark to that page on their device's browser.

Pros:

- They look like native apps but are cheaper to build
- Deployed changes are immediately available to users
- They are a more powerful cross-platform, scalable and affordable solution when compared to native apps

Cons:

- They give access to only basic device functions such as location and media files
- Mobile browser can take up screen space

These types of mobile applications are not in scope for this paper since their development is not similar to any web application.

Hybrid apps

In simple terms, hybrid apps are a combination of native apps and web apps. Like native apps, hybrid apps can leverage many device features available, and the apps can be download from an app store. Like web apps, hybrid apps are built using web technologies such as HTML, JavaScript, and CSS. The app is rendered in an embedded browser on the device, so there is no URL to reference.

Company leaders decide to build hybrid apps as wrappers for an existing web application so that the apps can have a presence in an app store. Without spending too much effort re-developing the app, a hybrid app makes possible cross-platform development, since the same HTML code can be reused on different mobile operating systems.

Pros:

- Interface with devices native functions and hardware (geo-location, camera, etc.) for advanced features
- HTML is available for rendering pages on mobile browsers

Cons:

- Must follow app store process for approval, quality and security
- Requires manual download, installation, and maintenance by users

Native app

Native apps are available from an app store and are installed on the device. Since these apps are developed for a specific platform, they can access all the device features, such as the camera, sensors (including GPS, accelerometer, and compass), the list of contacts, and more. User interface gestures can also be incorporated in the app. These apps can use the device's notification system and can work offline.

Pros:

- Can interface with the device's native functions and hardware
- Faster than mobile web apps

Cons:

- Must follow app store process for approval, quality and security
- Require manual download, installation and maintenance by users
- Costly and time-consuming to develop
- Costly long-term investment due to fragmentation (for example, multiple languages and versions for multiple operating systems)

IBM MobileFirst Platform Approach

MobileFirst Platform supports development of the common mobile application types above, with a variation for hybrid apps: *Hybrid Apps -Web* and *Hybrid Apps–Mixed*.

With *Hybrid Apps –Mixed*, you can create applications that use a container to access device capabilities, but you can also use other native, platform-specific components such as libraries, or specific user-interface elements to enhance the mobile application.

The various mobile application development models supported by MobileFirst Platform are seen on Figure 3:

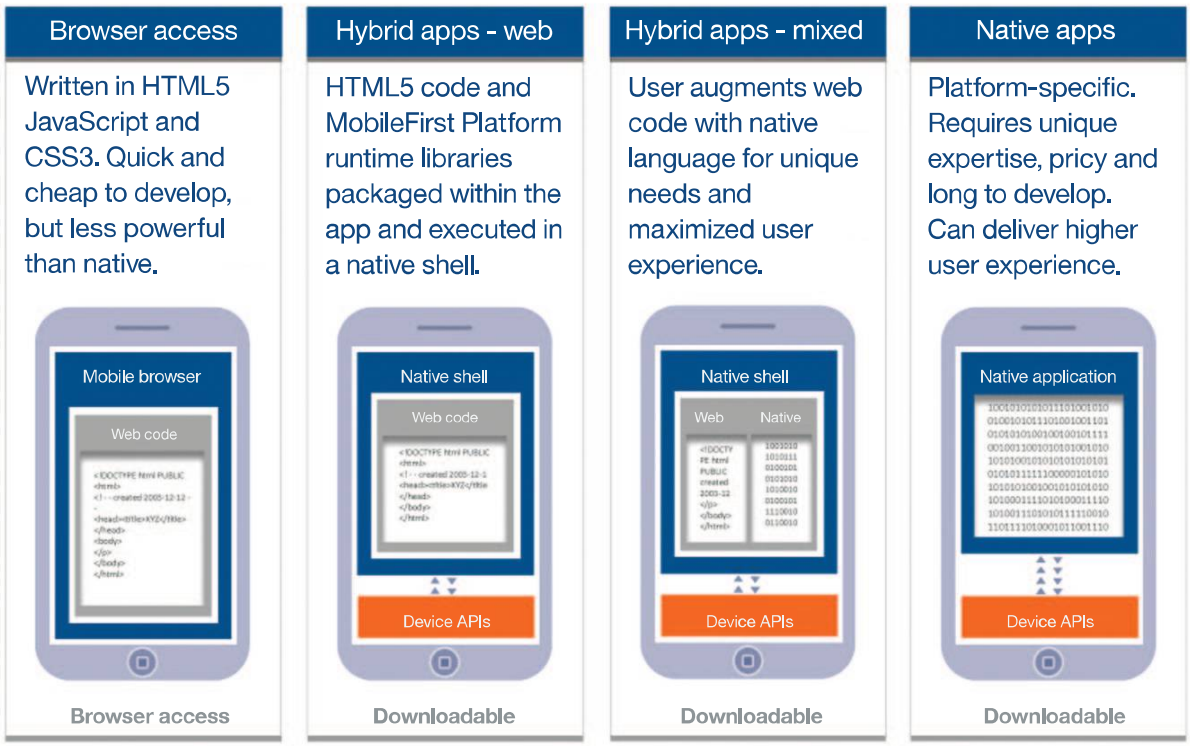

Figure 3. Mobile application development models supported by MobileFirst Platform

How to decide which type to use

There is no single answer. Leaders at each organization must weigh in factors such as those on Table 1 and decide which type is best for their business, target audience and development expertise.

Aspect	Web development	Hybrid development	Hybrid mixed development	Native development
Easy to learn	Easy	Medium	Medium	Hard
Application performance	Slow	Moderate	Moderate	Fast
Device knowledge required	None	Some	Some	A lot
Development lifecycle (build,test,deploy)	Short	Medium	Medium	Long
Application portability to other platforms	High	High	Medium	None
Support for native device functionality	Some	Most	All	All
Distribution with built-in mechanisms	No	Yes	Yes	Yes
Ability to write extensions to device capabilities	No	Yes	Yes	Yes

Table 1. Comparison of mobile development approaches

A sample application scenario

For the purposes of using MobileFirst Platform and the DIY option from a development standpoint, the IBM team created sample mobile applications for a fictitious company called JKE. The sample app provides access to banking activities such as a list of account transactions and account balance; the app also makes it possible for users to find nearby charities and to donate dividends from their accounts to charities of choice.

Two versions of the application were created in each solution: a hybrid app (using technologies such as Cordova, JavaScript, HTML5, and CSS) and a native app (using Java, targeted for the Android platform). Samples of the app user interface (on Android emulator) are seen on Figure 4.

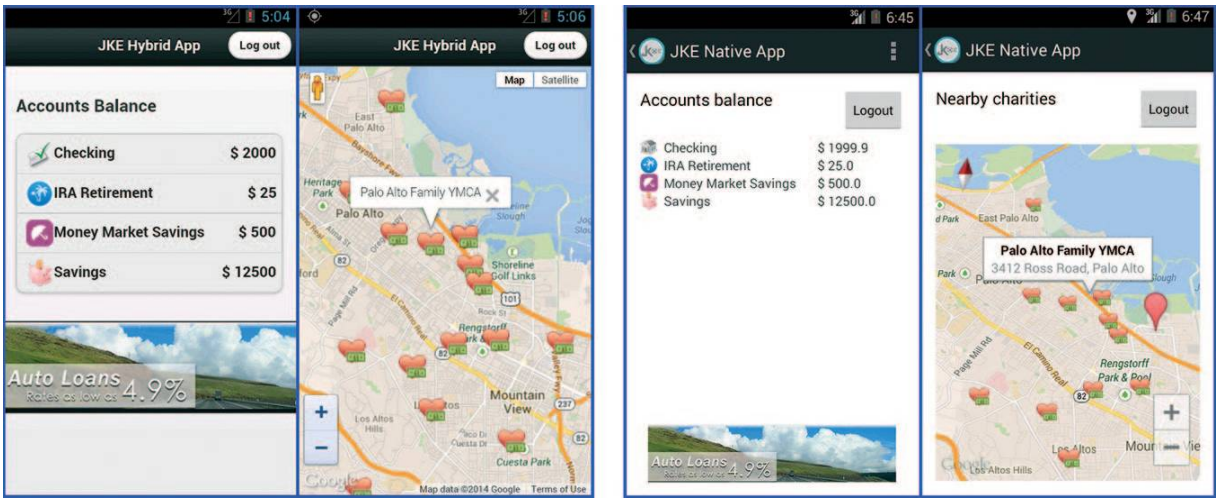

Figure 4. The company's hybrid app (left) and the company's native app (right) on Android emulator

The app communicates with a backend REST service (JKE Service) that authenticates the user and retrieves account information, such as transactions and balance for each account.

The service is deployed to Tomcat server in all scenarios. However, Tomcat does not have a native JAX-RS implementation to support REST services as does WebSphere Application Server Liberty Profile. To make Tomcat work for that purpose, the Jersey library jar files were manually added to the Tomcat installation folder. Alternatively, the Jersey jar files could have been packaged with the JKE Service war file for deployment to Tomcat.

WebSphere Application Server Liberty Profile, on the other hand, is capable of hosting REST services without any further configuration. On the MobileFirst Platform environment, the use of Tomcat could have been omitted altogether and the JKE Service could have been deployed directly to the Liberty Profile, without the need to add Jersey library files to that solution, since Liberty Profile natively supports REST services. However, the IBM team wanted to exercise a scenario with heterogeneous environments, in which a company might have legacy services running elsewhere, and the MobileFirst Platform solution is used to plug into that backend with the use of adapters.

To access the backend service from the frontend, the IBM team used a few different technologies. In the DIY scenario, different technologies were used: JavaScript Ajax calls to access the service from the hybrid app, and Apache HTTP calls from the native app. In contrast, the MobileFirst Platform solution offers the ability to create an adapter to the backend service, which in this case was used by both the hybrid and native applications.

Native apps can also use adapters to access backend services. The IBM team used MobileFirst Platform to create and deploy adapters using the MobileFirst Platform Library API project. This API provides the interface for native apps to call the adapters procedures (originally written in JavaScript) from the native (Java) code. The IBM team created two distinct MobileFirst Platform projects: one for the adapters and one for the native app. For hybrid apps, since the adapter's code and app code are both based on JavaScript, the adapters and app code can be on the same MobileFirst Platform project. In this example, the native app accesses the JKE Legacy Service running on Tomcat, using a MobileFirst Platform adapter running on WebSphere Application Server Liberty Profile server.

The geo-location service used to find charities also varied based on the environments. For hybrid apps in both environments, the IBM team used Cordova to return the coordinates that find and draw charities (markers) on the map. For the native apps, the IBM team used the Android geo-location API in the open source environment, and MobileFirst Platform geo-location API in the MobileFirst Platform environment.

Similarities between the two options

In previous sections, you have seen many similarities between the two solutions. Both are free for download and use (note that we are focusing on MobileFirst Platform Developer Edition, which is a lightweight, simplified version of MobileFirst Platform Enterprise Edition).

Both solutions are based on common open source components available from well-known sources, as seen in the *Solution Components* section above.

As seen on *JKE sample application* section above, both solutions rely on similar technologies to allow the creation of web, hybrid, or native mobile apps. Both solutions offer technologies to access backend services. Both solutions rely on the specific platform SDKs (such as Android or iOS), including development of applications and deployment to emulators for debugging and testing purposes.

Table 2 summarizes the technologies used for each type of app in each environment.

	DIY scenario		MobileFirst Platform	
	Hybrid app	**Native app**	**Hybrid app**	**Native app**
Frontend technologies	HTML, JavaScript, CSS, JQuery Mobile	Java	HTML, JavaScript, CSS, JQuery Mobile	Java
Backend servers	Tomcat	Tomcat	Tomcat WebSphere Application Server Liberty Profile	Tomcat WebSphere Application Server Liberty Profile
Access to backend service	JavaScript Ajax	Apache HTTP API	MobileFirst Platform adapter	MobileFirst Platform adapter
Geo-location	Cordova	Android geo-location API	Cordova	MobileFirst Platform geo-location API
Maps and places	Maps API (Google)	Maps API (Android)	Maps API (Google)	Maps API (Android)

Table 2. Technologies used to create sample mobile apps

Advantages of IBM MobileFirst Platform

MobileFirst Platform Developer Edition offers many advantages when compared to a DIY solution.

Installation and configuration

The DIY solution requires users to pick and choose what open source components will be installed and configured together. The DIY approach requires users to know the sources for download and to understand the compatibility between different component versions, references for documentation, tutorials, discussion forums, and more.

MobileFirst Platform requires you to download and install some components in preparation for the environment setup; however, there is a clearly documented path that instructs users on what components to install, in what order, and where to find the components.

In addition, the installation of the MobileFirst Platform plug-in to an existing instance of Eclipse includes the MobileFirst Platform Studio component and the MobileFirst Platform Server component. These components add advanced features for design, development and delivery of mobile applications. The MobileFirst Platform Server component uses the WebSphere Application Server Liberty Profile for the deployment of adapters and applications.

Learn more about the MobileFirst Platform Developer Edition installation and configuration information.

Design and development

MobileFirst Platform makes the design and development of mobile applications easier and more productive by providing the following capabilities:

- Project creation wizard creates the initial structure for the application, including packaging structure for UI and logic components, backend adapters and application deployment to MobileFirst Platform server.
- Design perspective, including "what you see is what you get" (WYSIWYG), and drag-and-drop editors are provided for controls. In web and hybrid apps, the MobileFirst Platform approach supports HTML controls and jQuery Mobile and Dojo Mobile widgets, and in Android native apps, MobileFirst Platform supports Java UI controls.
- Embedded Dojo tool kit package and tools that you can use to develop your mobile web applications.
- Encapsulated JQuery that you can add to the project with a single line of code.
- JavaScript API to invoke common UI controls regardless of the environment, and automatically render these controls in a native way for each platform.
- "Skins" to provide support for multiple form factors in a single executable file for devices of the same operating-system family.
- Framework to enable the translation of applications into other languages.
- You gain the ability to add environments "on the fly." Expect code separation between supported environments (which promotes code reuse).
- Environment optimization framework. The core logic and design guidelines of the app are written using web technologies (for example, HTML, CSS and Java Script) and the core logic and design guidelines are shared by all environments. Environment-specific optimization can be added when required.
- The Cordova framework is integrated into all mobile environments. The MobileFirst Platform framework uses the Cordova library. MobileFirst Platform exposes the Cordova APIs so that developers can access native device functions through those services.
- You gain the ability to enhance hybrid applications with a native UI. Use the MobileFirst Platform API to mix web pages and native pages on the same app (or use Cordova to access native functions from hybrid apps).
- MobileFirst Platform client Java Script API bridges to native mobile-platform APIs and allows for dynamic HTML load and other elements.
- You can optimize mobile web app performance with "minification" and concatenation (to reduce the number of files), and with HTML5 application cache.

While the DIY solution provides basic capabilities for developing mobile applications, the approach does not offer the kind of enhanced experience for the developer that MobileFirst Platform offers, including WYSIWYG editors, code reuse, and all other features itemized above.

Backend connectivity

In the DIY option, a developer may use open frameworks to access backend REST services such as JavaScript Ajax calls for web and hybrid apps, and Apache HTTP API for native app. This situation requires different technologies for different types of applications, requiring different skills from a developer.

In addition, the calls to the backend server reside on the client, which is not an optimal way to architect an application. For example, to avoid latency issues (while the client waits for the server to respond to a request), a developer needs to guarantee the implementation of asynchronous calls, dealing with service unavailability, and so on.

In the MobileFirst Platform solution, you can still use open technologies as mentioned above to access backend services, but MobileFirst Platform also provides the ability to create adapters specifically for that purpose. The creation of adapters in MobileFirst Platform can be fully automated for SOAP services (just point to the service WSDL file and follow steps on a wizard). For other backend types, such as REST services, you still must declare and implement the procedures to access backend services operations, but MobileFirst Platform creates the structure of the adapters as a starting point.

Adapters are defined in JavaScript but adapters can also be invoked by Java code. The same implementation of an adapter can be used by web, hybrid, and native applications. Adapters also provide the added benefit of intermediating communication between the client and backend service; adapters are deployed to the MobileFirst Platform server available with the solution (WebSphere Application Server Liberty Profile) or other supported servers (see the section of this paper titled *Deployment in development environment*).

Other advantages of using adapters when compared to common open technologies are:

- Backend service discovery (for SOAP or SAP services). Auto-generate adapter with procedures based on service WSDL file.
- Server side scripting to enhance adapter capabilities (pre- and post-logic, processing done in one transaction, mashups from different sources).
- The ability to use Java in adapters (invokes Java code from adapter procedure, and invokes adapter procedure from Java code).
- Native MobileFirst Platform Client API to manage authentication and back-end access, and to benefit from more server functionality.
- The ability to use encrypted cache mechanism for storing sensitive data on the client side.
- The ability for the native Android applications to communicate with a MobileFirst Platform Server by using the MobileFirst Platform native API library.

Deployment in development environment

MobileFirst Platform includes an embedded development server (WebSphere Application Server Liberty Profile) for deployment of adapters and applications. You can also deploy adapters to the MobileFirst Platform server and test the adapter procedures before using the code in an application.

It is possible to have the MobileFirst Platform server use other servers such as Apache Tomcat for deployment of adapters and applications on QA and production environments. This option requires additional configuration done by hand or using Ant scripts. For the purposes of this paper, the IBM team used the provided WebSphere Application Server Liberty Profile.

In the DIY solution, deployment can be done by manually deploying (installing) the application to the platform emulator for debugging and manual testing. A more-scalable form of deployment, such as deploying applications to a private app store, was not in the scope of this paper; however, there seems to be no DIY solution available for that purpose. It is worth mentioning that the full enterprise versions of MobileFirst Platform provide a component for that purpose: the IBM MobileFirst Platform Application Center enables company teams to set up an enterprise app store to help govern the distribution and management of pre-release and production-ready mobile applications.

Debugging and testing

MobileFirst Platform provides a debugger API. You can print log messages to the log for the environment that you use. APIs are multiplatform; the output destination changes according to the platform upon which that application runs.

MobileFirst Platform provides a Mobile Browser Simulator for previewing and debugging apps. MobileFirst Platform also simulates Cordova APIs. The advantage is that you can preview the application on a browser, as opposed to previewing on an emulator, thus creating a much faster way to load and preview the app.

MobileFirst Platform also includes an optional component called Mobile Test Workbench for MobileFirst Platform. This component makes it possible for you to automate the functional testing of Android and iOS mobile applications that are built in MobileFirst Platform.

The DIY solution does not have a debugger API. A developer must use common ways to publish messages to the console or to the device screen as a way to debug an application. The DIY solution has no web browser simulator either, so debugging and testing must happen on a platform emulator, which is typically slower than the mobile browser simulator, but which is acceptable for manual testing or automated testing. However, the DIY solution does not offer an immediate-use functional testing tool as the MobileFirst Platform solution does; the DIY solution would require installing and configuring a separate test tool, which can be guaranteed to seamlessly integrate with the integrated development environment.

Conclusions

This document showed how MobileFirst Platform Developer Edition is better than a do-it-yourself solution, with many more features available for immediate use. And remember: MobileFirst Platform Developer Edition is available for use at no charge. And best yet, this IBM solution can be scaled up to full versions of Platform for enterprise deployment of mobile applications.

Take a “test drive” for yourself. Download MobileFirst Platform Developer Edition today.

Appendix

Comparison of development features found in IBM MobileFirst Platform Developer Edition and a "do-it-yourself" (DIY) option (in scope for this paper: Android development).

Category	Feature	Sub-feature	Supports feature (Y/N)		Category ranking 4 stars = advanced 1 star = basic		Comment
			MobileFirst Platform	DIY	MobileFirst Platform	DIY	
Target platforms					****	****	Both solutions require installing and configuring Android SDK (or other desired platforms when supported)
	Android phone		Y	Y			
	Android tablet		Y	Y			
	Embedded Web page		Y	Y			
	Mobile Web app		Y	Y			
	Standard Web		Y	Y			
Application type (Channel)					****	***	MobileFirst Platform supports inner shell approach which helps compartmentalize skill sets and responsibilities while developing enterprise apps.
	Web		Y	Y			
		HTML 5	Y	Y			
		HTML 5 single page (SPA)	Y	Y			
		Basic HTML	Y	Y			
	Mobile Web		Y	Y			
	Hybrid		Y	Y			
	Native		Y	Y			
	Web site (normal)		Y	Y			
	Shell/inner app approach		Y	N			
	Uses Container for Hybrid		Y	N			
Native API support					****	****	Based on installed SDKs
	Android		Y	Y			

Category	Feature	Sub-feature	Supports feature (Y/N)		Category ranking 4 stars = advanced 1 star = basic		Comment
			MobileFirst Platform	DIY	MobileFirst Platform	DIY	
Tool platform					****	****	
	Windows		Y	Y			
	Mac OS X		Y	Y			
	Linux		Y	Y			
IDE					****	**	MobileFirst Platform increases developer's productivity by supporting multiple targets in the same project and easy GUI creation.
	Developer-level		Y	Y			
	Eclipse-based		Y	Y			
	Target platform emulators		Y	Y			
	Integrated debug		Y	Y			
	Single project for multiple targets		Y	N			
	Drag-and-drop GUI builder		Y	N			
	Can use Cordova (PhoneGap)		Y	Y			
	Version management		Y	Y			3rd party tools that integrate with Eclipse
	Full WYSIWYG		Y	N			DIY does not have HTML drag-n-drop visual editor (it has preview though); it has visual drag-n-drop editor for Java UI elements (Android native app)
	OOTB Templates		Y	Y			DIY: it depends on templates available with the platform SDK
	Code reuse		Y	N			MobileFirst Platform allows addition of platforms on the fly and does a better job in separating the common code from the platform-specific code
Languages					****	****	
	Java		Y	Y			
	JavaScript		Y	Y			

Category	Feature	Sub-feature	Supports feature (Y/N)		Category ranking 4 stars = advanced 1 star = basic		Comment
			MobileFirst Platform	DIY	MobileFirst Platform	DIY	
Script Libraries					****	***	DIY: requires installing each library; MobileFirst Platform: some libraries are prepackaged with the tool
	jQuery		Y	Y			Included in MobileFirst Platform; separate install in DIY
	Dojo Mobile		Y	Y			Included in MobileFirst Platform; separate install in DIY
	Sencha Touch		Y	Y			Separate install
	Included platform libraries written in JavaScript		Y	N			MobileFirst Platform has a JavaScript library to increase productivity and allow for creating platform-independent apps more easily
Mobile 3rd party frameworks					****	***	
	JQuery Mobile		Y	Y			Requires separate installation
	Dojo Mobile		Y	Y			Requires separate installation
	Sencha Touch		Y	Y			Requires separate installation
	Node.js		Y	Y			Requires separate installation
	Cordova (PhoneGap)		Y	Y			Cordova is pre-packaged with MobileFirst Platform; requires separate installation on DIY
Simulation					****	*	MobileFirst Platform uses a Mobile Browser Simulator and platform emulator that is faster to load and use for developer's tests
	Via browser		Y	N			
	Device recognition		Y	N			
	Simulates Cordova		Y	N			
	USes device SDK emulator		Y	Y			

Category	Feature	Sub-feature	Supports feature (Y/N)		Category ranking 4 stars = advanced 1 star = basic		Comment
			MobileFirst Platform	DIY	MobileFirst Platform	DIY	
Back-end connec-tivity					****	*	DIY has to use 3rd party libraries to access backend services, and the implementation is fully left for the developer to provide
	Direct adapters/ connectors						
		SAP	Y	N			
		HTTP	Y	N			
		JDBC	Y	N			
		SOAP Web services	Y	N			
		REST XML	Y	N			
		REST JSON	Y	N			
		JMS/MQ	Y	N			
		Database	Y	N			
		Cast Iron	Y	N			
		MQTT	Y	N			
	Code-free backend service creation		Y	N			
	Simulator for backend services		Y	N			
	SMS support		Y	N			
	Middleware integration			N			
		IIB (WMB)	Y	N			
		MQ	Y	N			
		Cast Iron	Y	N			
		BPM integration	Y	N			
Trial or free Dev. License			Y	Y	****	****	

For more information

To learn more about MobileFirst Platform Developer Edition, please contact your IBM representative or IBM Business Partner, or visit the following website: **ibm.com**/developerworks/mobile/

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We'll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit: **ibm.com**/financing

About the author

Ricardo Balduino

Ricardo Balduino is a senior software engineer at the IBM Competitive Project Office. His focus is on the complete software development lifecycle processes and tools. His goal: to help organizational leaders to adopt the tools and best practices that support better, more-predictable software development.

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
October 2014

WSW14226-USEN-01