解 説 2

エンタープライズ・モバイル・アプリケーション基盤としてのIBM Worklight

近年、スマートフォンやタブレットなどのモバイル・デバイスの急速な普及を背景として、モバイル・デバイス向けのサービス提供のニーズが高まっています。日本市場においては、現時点では一般消費者向け（B to C）のサービスのモバイル対応が先行していますが、いわゆるコンシューマライゼーションによりモバイル・デバイスがビジネス・ユースにも浸透し、企業向けシステム（B to B/B to E）にもモバイル対応が求められるであろうことは想像に難くありません。

このような背景を踏まえ、本記事ではモバイル・デバイスをエンタープライズ・システムに適用するに当たっての、アプリケーション・アーキテクチャーの観点から見た課題と、IBM Worklight（以下、Worklight）を中心とした解決策について解説します。本稿で取り上げている課題はすべてエンタープライズ・システム向けモバイル・アプリケーションにおいて一般的なものであり、対応すべきレベルの差異はありますが、どれもおろそかにしてよいものではありません。これらに包括的に対応するためには、個別のソリューションではなく、統合的なアプリケーション基盤の導入が効果的です。

❶ モバイルWebアプリケーションとネイティブ・アプリケーションの違いと使い分け

モバイル・デバイス上で利用するアプリケーションの提供形態は、大きく分けて「モバイル Web アプリケーション（以下、モバイル Web）」と「ネイティブ・アプリケーション（以下、アプリ）」の 2 種類があります。それぞれの特徴は**表 1** の通りです。

どちらもメリット・デメリットがあり、どちらか一方が絶対的に優位ということはありません。ケース・バイ・ケースでどちらの方式を採用するか検討すべきでしょう。

モバイル・デバイスに搭載されている Web ブラウザーは、PC 向け Web ブラウザーよりも HTML の新しい仕様である HTML5 への対応が進んでおり、Web アプリケーション（HTML/JavaScript）でもオフライン・キャッシュやモバイル・デバイスのネイティブ機能の一部を利用することができるようになっています［1］。例えば、Geolocation API による位置情報の取得などはモバイル Web アプリケーションでも利用可能です。

しかし、モバイル Web では実現できない機能や制約事項が多いのも事実です。例えば、本記事執筆時点（2012 年 12 月）では、モバイル Web では以下のような要件には対応できない、もしくは制限付きの対応となります。

- カメラで撮影した画像を加工する。
- 電子コンパスを利用して方位を取得する（iOS5.0 以降のみ独自拡張で対応）。
- DRM（Digital Rights Management）で保護されたビデオやオーディオを再生する。
- NFC（Near Field Communication）や Bluetooth などを利用した通信を行う。

表 1. モバイル Web とアプリの特徴

	モバイルWeb	アプリ
動作プラットフォーム	Webブラウザー	モバイルOS
クロスプラットフォーム対応	可能	不可　プラットフォーム別に開発
インストール	不要	必要
オフライン起動	（原則）不可　サーバーに接続して初期HTMLをロードする必要あり	可能
端末固有機能利用（例：プッシュ通知）	（原則）不可　GPSなど一部はブラウザーでサポートされる	可能
開発に必要なスキル	HTML/CSS/JavaScript	ネイティブSDKのスキル　iOS : Objective-C Android : Java
開発環境	PC向けWebアプリケーションと同じ	ネイティブSDK
アプリケーション・ストアでの配布	不可	可能
アプリ更新	サーバー上のリソースを更新するのみ	AppStore、Google Playに更新版アプリケーションを登録し、再インストール　もしくは、MDM（モバイル・デバイス管理）の機能を利用

- アドレス帳にアクセスする。
- デバイスの詳細情報（機器 ID、OS 種別やバージョンなど）を取得する。

❷ ハイブリッド・アプリケーション

前出の通り、モバイル Web で要件を満たせない場合は必然的にアプリによる実装を選択することになります。しかし、アプリにも以下のような課題があります。

- 開発スキルの特殊性：モバイル・デバイスごとに開発言語や API、開発環境が異なっており、それぞれに対応するためのスキルを保持したエンジニアが必要となる。従来のエンタープライズ・システム開発において主流であった Java/Web 系のスキル・セットとはギャップが大きい。
- プラットフォームの多様性：現時点では日本市場において主要なプラットフォームとして iOS（Apple）と Android（Google）があり、今後は Windows8（Microsoft）も普及する可能性がある。グローバル市場においてはほかに BlackBerry（RIM）なども利用されている。

特に、複数のプラットフォームに対して機能的に同等のアプリケーションを提供する場合、まったく異なるコード・ベースをプラットフォームの数だけ用意する必要があり、初期開発コストだけでなく継続的にメンテナンスしていくための負担が大きくなります。

これらの課題に対処するアプローチとして注目されているのが「ハイブリッド・アプリケーション」です。ハイブリッド・アプリケーションの基本的なアーキテクチャーは次のようになります。

- Web ブラウザー・コンポーネントを内部に埋め込んだアプリの外枠（シェル）を用意することで、HTML や JavaScript といった Web リソースを実行する。
- シェルには Web ブラウザー・コンポーネント内部とモバイル・デバイスのネイティブ API を仲介する層が設けられており、JavaScript からネイティブ API を実行する。

ハイブリッド・アプリケーションは、HTML や JavaScript といったモバイル Web と同じスキル・セットで開発が可能であり、アプリケーション・コードも複数プラットフォームで共通化することが容易です。また、ネイティブ API の呼び出しが可能であるため、モバイル Web における制約を超えてプラットフォームの機能をフルに利用することが可能であり、さらに成果物（バイナリー）はアプリと同様にアプリケーション・ストアでの配布が可能であることから、実用上はアプリと同等に扱うことができます。

Worklight［2］はハイブリッド・アプリケーションの開発および実行基盤として、アプリケーション開発から実行・管理までをトータルでサポートします。ハイブリッド・アプリケーションを実現するためのコア・コンポーネントとしてはオープンソースの Apache Cordova（PhoneGap）［3］をベースとして利用しており、Apache Cordova のエコシステムで蓄積されているナレッジや、プラグインなどの資産を活用することが可能です。Worklight アプリケーションのソフトウェア・スタックの概要は**図 1** のようになります。

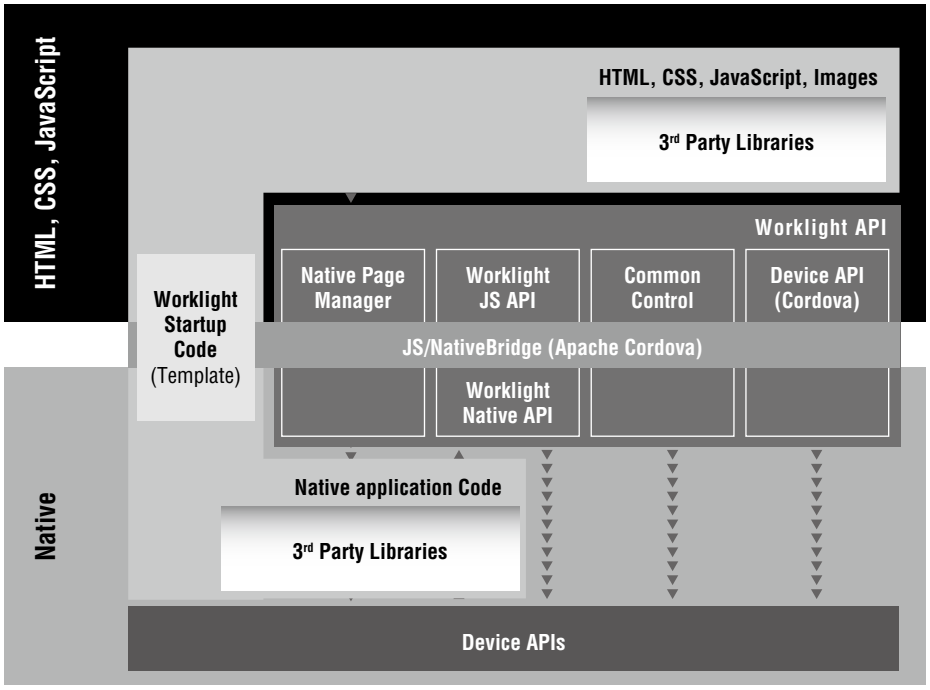

図1. IBM Worklight アプリケーションのソフトウェア・スタック概要

❸ バックエンド・システムとの連携

モバイル・アプリケーションの高機能化に伴い、アプリケーション単体でサービスが完結することはまれで、何らかのバックエンド・システムと連携してサービスを構成することが多くなってきています。例えば、カメラ・アプリケーションで撮影した画像をバックエンド・システムで保管したり、SNS に投稿して共有したりという使い方はごく当たり前ですし、エンタープライズ・システムにおいては CRM や ERP などとの連携は不可欠でしょう。

しかし、モバイル・アプリケーション提供者は一般に小規模なベンチャー企業が多く、モバイル・アプリケーションの開発にはたけていても、バックエンド・システムの構築および運用の経験に乏しく、実際の構築には大きな負担がかかるというケースが多く見受けられます。このような状況を受け、最近ではモバイル・アプリケーション提供者に対してバックエンド・システムのみを提供するというサービスも登場してきています。このようなサービス提供形態は BaaS（Backend as a Service）[4] と呼ばれ、まだ発展途上の分野ではありますが Parse や Appcelerator Cloud Services など複数のサービス提供者が登場しつつあります。バックエンド・システムの運用には公開 Web アプリケーションを維持・運用するのと同等の負担がかかるため、BaaS のような仕組みがモバイル・アプリケーションの提供者にもたらすメリットは大きいと考えられます。

バックエンド・システムとの連携に関してもう 1 つ考慮すべき点は、モバイル・アプリケーション開発者にとっての開発容易性です。バックエンド・システムとの連携手法として一般的なのは REST 形式の API ですが、HTTP 通信のハンドリングが煩雑であったり、セキュリティーの確保が開発者任せになってしまうなどの問題をはらんでいます。そこで、低レベルの HTTP 通信やセキュリティー確保の仕組みを隠ぺいした、バックエンド・システム連携のための SDK を提供する手法が効果的と考えられます。SDK を提供することで、開発者はライブラリーの機能を呼び出すだけで透過的にバックエンド・システムと連携するアプリケーションを実装することが可能になります。

Worklight は、コンポーネントとして Worklight Server と Adapter を提供することで、バックエンド・システム連携に対する統合的なソリューションを実現しています。システムの全体構成は**図 2** のようになります。

Worklight Server はモバイル・アプリケーション単体では実現できないさまざまな付加機能を提供するためのサーバー・ランタイムです。Java EE アプリケーション（war）として実装されており、IBM WebSphere Application Server および Tomcat 上で稼働します。Worklight アプリケーションには Worklight Server と暗黙的に HTTP(S) で通信を行う機能が組み込まれており、開発者が特に意識することなく Worklight Server と連携できるようになっていますので、見方を変えると、Worklight アプリケーションはバックエンド・システム連携の SDK を包含しているといってもいいでしょう。

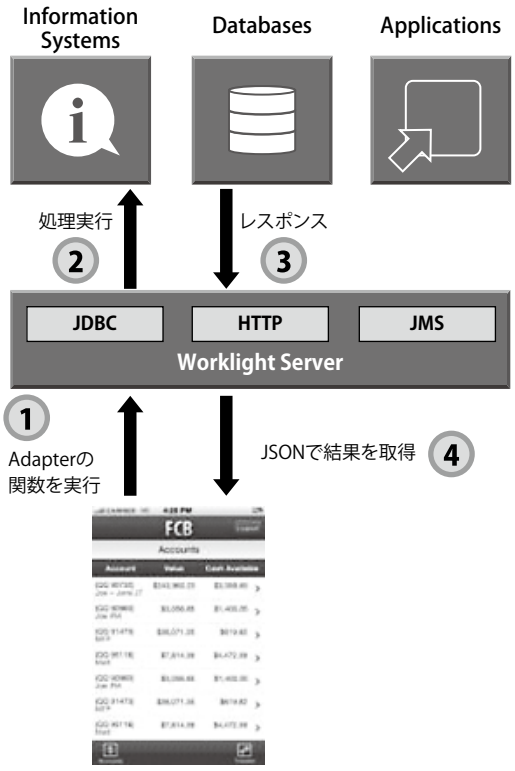

図 2. Worklight Server と Adapter によるバックエンド連携

Adapter は Worklight Server 上で稼働するコンポーネントで、以下のタイプのバックエンド・システムと連携することができます。

- HTTP（REST/SOAP）
- RDB（JDBC）
- JMS
- WebSphere Cast Iron [5]

単純な連携であれば Workilght Server から HTTP や JDBC で直接接続してもよいでしょう。ただし、連携先が SaaS や ERP などの場合、特殊な接続プロ

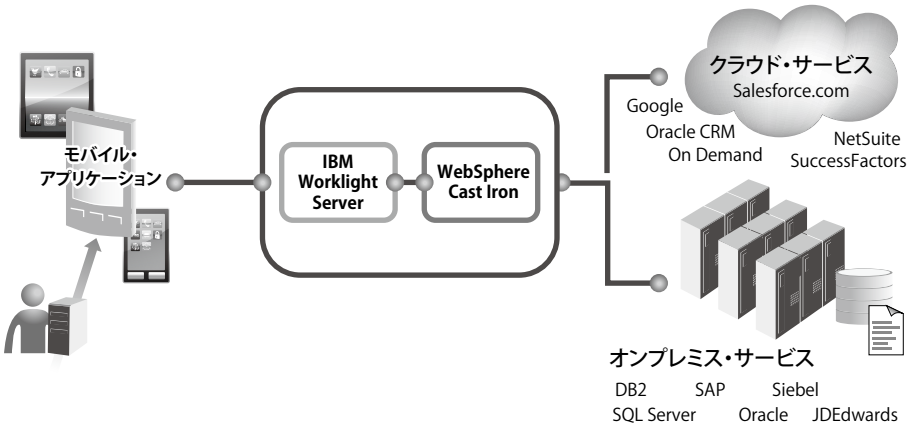

図 3. Cast Iron Adapter を利用したバックエンド連携時のシステム構成例

トコルや複雑なデータ変換が必要になる場合が多いため、インテグレーション・アプライアンスである IBM WebSphere Cast Iron の併用を検討した方がよいでしょう。Worklight の提供する Cast Iron Adapter を利用することで、モバイル・アプリケーションから Cast Iron とシームレスに連携することが可能です。この方法により、複雑なデータ変換などの作り込みを極力排除して、バックエンド・システムと見通しよく連携することが可能になります。Cast Iron Adapter を利用したバックエンド連携時のシステム構成は**図 3** のようになります。

ここまではモバイル・アプリケーションが起点となってバックエンド・システムにアクセスするパターンを取り上げてきましたが、逆にバックエンド・システムからモバイル・アプリケーションに対して何らかのメッセージを送信したい場合もあります。例えば、緊急通報メッセージを送る場合や、ワークフローの承認依頼を送る場合などです。この場合、モバイル・アプリケーションのプッシュ通知を利用することで、特定のユーザーおよび特定のアプリケーションに対してピンポイントにメッセージを送り、さらにアプリケーションの起動までシームレスに連携することができます。

ただし、プッシュ通知機能はモバイル・デバイスごとに仕組みが異なって（iOSではApple Push Notification［以下、APN］、AndroidではGoogle Cloud Messaging［以下、GCM］）おり、それぞれ個別に対応する必要があります。Worklightはこの差異を抽象化して共通化する「統合プッシュ通知機能（Unified Push Architecture）」を提供しており、プッシュ通知に対してもクロスプラットフォーム対応が可能になっています。統合プッシュ通知機能を利用してiOSおよびAndroidにプッシュ通知を送る場合の処理の流れは**図4**のようになります。

1. ユーザーがアプリケーション上で、各ベンダーのプッシュ・サービスを利用するための認証処理を行う。
2. ユーザーがプッシュ通知を受け入れたことを証明するデバイス・トークンが発行される。
3. Worklight アプリケーションは発行されたデバイス・トークンを Worklight Server に送信する。
4. Worklight Server はユーザーおよびアプリケーションとデバイス・トークンの関連付けを行い、その結果を保持する。

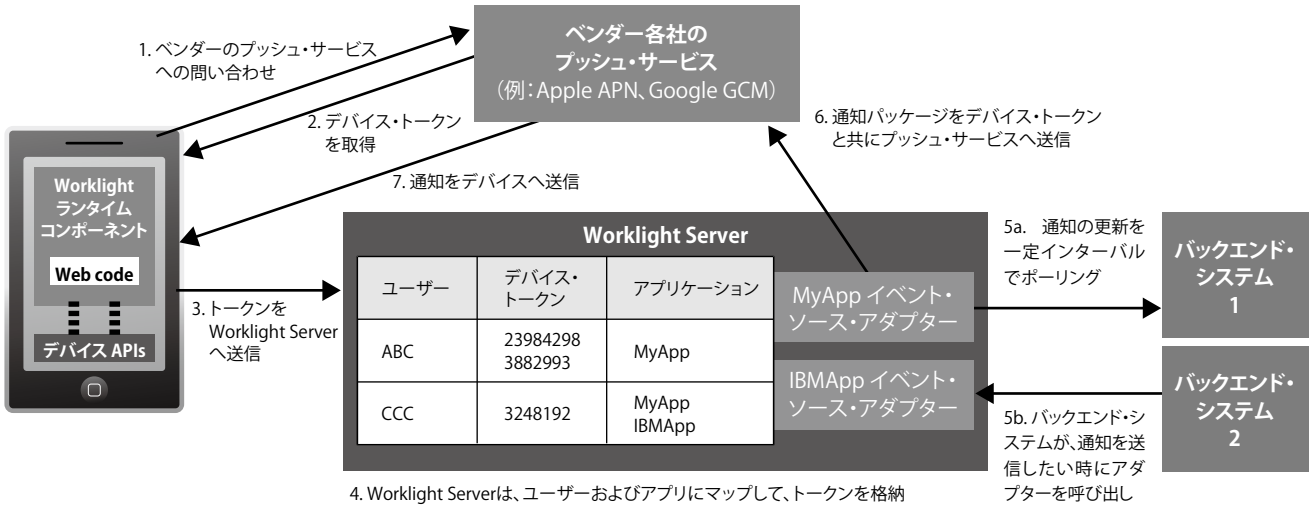

図4. 統合プッシュ通知機能を利用した処理の流れ

5. Worklight Server は Adapter を利用してバックエンド・システムからプッシュ通知のトリガー（ポーリングによるデータ変更確認、ないしはバックエンド・システムからの通知）を受け取る。
6. Worklight Server は通知メッセージとデバイス・トークンをベンダーのプッシュ・サービスに送信する。
7. ベンダーのプッシュ・サービス（APN/GCM）から、ユーザーのデバイスに対して通知メッセージが送信される。

❹ 認証およびシングル・サインオン

モバイル・アプリケーション（ネイティブおよびハイブリッド）においてユーザー認証を行う場合、アプリケーションとリモートの認証システムが連携して認証情報（クレデンシャル）の受け渡しおよび妥当性検証を行う必要があります。モバイル・アプリケーションでは特に Facebook や Twitter による認証など、サービス提供者独自の認証機構ではなく第三者認証を利用するケースも多くなります。また、エンタープライズ・システムにおいては、LDAP や IBM Tivoli Access Manager などの既存のシングル・サインオン（SSO）システムとの連携が必要となる場合も多いでしょう。これらの一連の処理をセキュリティー脆弱（ぜいじゃく）性などのリスクを排除しつつ正確に行う仕組みを構築することはそれなりに難易度の高い作業であり、アプリケーション開発者に実装を委ねるのではなく、アプリケーションの基盤として提供すべき機能であると考えられます。

Worklight はこのような多様なニーズに対応するための認証フレームワークを提供しています。例えば、以下のようなパターンの認証に対応可能です。

●フォーム入力による認証
●HTTP ヘッダーや Cookie による認証
●デバイス ID やセキュリティー・トークンなどを併用した複数要素認証
●多段階認証
●バックエンド・システムごとに異なる認証レルム（後述）を割り当て

Worklight の認証フレームワークの動作は**図 5** のようになります。

1. Worklight Serverから認証要求が送信されると、Worklightアプリケーションの認証フレームワーク（Challenge Handler）が起動され、認証に必要なクレデンシャルを収集する。
   ※ 認証要求が発生するタイミングは、アプリケーション起動時や保護された Adapter 呼び出し時など柔軟に設定可能。
2. クレデンシャルは認証フレームワークの機能により自動的に Worklight Server に送信される。
3. Worklight Server 上で指定された Login Module が実行され、クレデンシャルの検証が行われる。検証の結果（認証状態や認証システムから受領した認証トークンなど）は Worklight Server 上でセッションに関連付けられて保管されるため、認証状態はセッション・タイムアウトになるまで有効となる。

Worklight の認証フレームワークは「レルム」と呼ばれる抽象的な保護領域を定義することができるようになっており、レルムごとに異なる Login Module を対応付けることが可能となっています。例えば、あるアプリケーションから複数のバックエンド・システムに接続するような場合、バックエンド・システムごとに異なる認証メカニズムを割り当てることも可能です。

認証データの処理を行うLogin ModuleはJAAS（Java Authentication and Authorization Service）の仕様に基づいて実装されたものが利用できますので、Worklightの標準機能でサポートされていない認証システムに

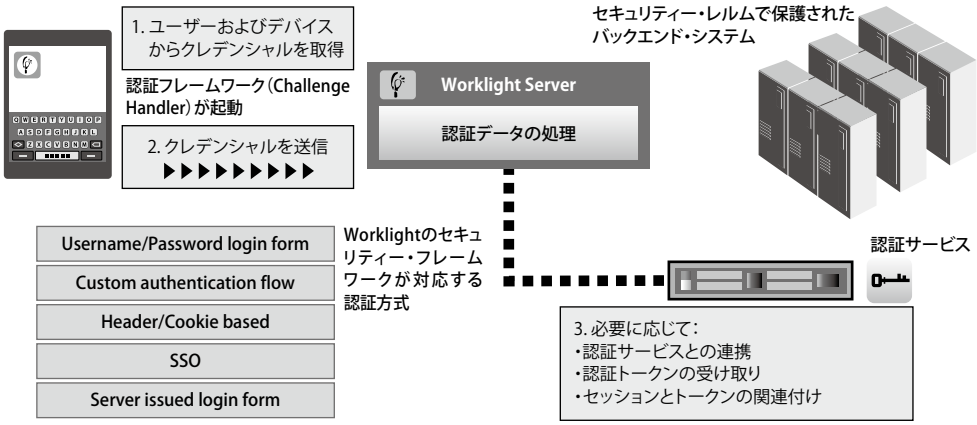

図 5. Worklight 認証フレームワークの動作

対しても対応可能です［6］。

❺ アクティビティー・ロギング（行動追跡）

アプリケーションの利用状況、すなわちアプリケーション内でユーザーが何をしたかを追跡したいというニーズは多くあります。例えば、機能別の利用頻度を分析してアプリケーション改善のためのフィードバックを得たり、投資対効果を計測したりするなどです。

モバイル Web では、通常の Web と同様にアクセス・ログを解析することで利用状況を追跡したり統計処理したりできますが、モバイル・アプリケーションの場合は行動追跡のための仕組みも用意する必要があります。このような仕組みはアプリケーションによらず汎用的なものであるため、個別に実装するのではなく、アプリケーション基盤の機能として共通化するべきでしょう。

アクティビティー	説明
Init	アプリケーション初期化
Login	ログイン
Query	Adapter 呼び出し
Logout	ログアウト
SetUserPrefs	ユーザー・プリファレンス格納
LogActivity	API 呼び出しによる任意のアクティビティー・ログ記録

Worklight はアクティビティー・ロギングを標準機能として提供しており、この機能を有効化することで以下のログを取得することができるようになっています（デフォルトでは無効化されています）。

LogActivity を利用すれば、アプリケーション上でログ出力の API を実行することで、任意のタイミングでログを記録することができます。例えば、ユーザーがどのような操作を行ったか、どの画面に移動したか、などを記録することが可能です。アクティビティー・ログは RDBMS の特定のテーブルに記録されるようになっており、テーブルのスキーマは公開されていますので、任意の BI ツールや Excel などにエクスポートして分析することが可能です。

❻ まとめ

本記事では、エンタープライズ・システムにおいてモバイル・デバイスを活用していくに当たり、カスタム・アプリケーションの構築、および運用に関して想定される課題や考慮点、また IBM Worklight が提供するソリューションについてご紹介しました。

モバイル・デバイスの急速な普及を受け、今後はエンタープライズ・システムにおけるモバイル・アプリケーションの重要性がより高まっていくと考えられます。しかし、モバイル・アプリケーションを本格的に利用していくためには、多様なプラットフォームへの対応やバックエンド・システムとの連携、セキュリティーの確保などさまざまな課題に対処していかなければなりません。

IBM Worklight は、このような課題に対して「モバイル・アプリケーション基盤」を提供することで、総合的に対処するためのソリューションと位置付けられます。今後モバイル・アプリケーションを活用していくに当たって、IBM Worklight がお役に立てれば幸いです。

［参考文献］

［1］ Can I use...Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers , http://caniuse.com/

［2］ IBM Worklight : 製品概要, http://www.ibm.com/software/jp/websphere/mobile-solutions/worklight/

［3］ Apache Cordova, http://incubator.apache.org/cordova/

［4］ Publickey: モバイル向けの新クラウド、BaaS（Backend as a Service）とは何か。「Parse」が正式サービス開始, http://www.publickey1.jp/blog/12/baasbackend_as_a_serviceparse.html

［5］ IBM WebSphere Cast Iron, http://www.ibm.com/software/jp/websphere/castiron/

［6］ IBM DeveloperWorks: Getting started with IBM Worklight, https://www.ibm.com/developerworks/mobile/worklight/getting-started/index.html

日本アイ・ビー・エム株式会社
ソフトウェア事業 WebSphere 事業部
クライアント・テクニカル・プロフェッショナル

須江 信洋 **Nobuhiro Sue**

［プロフィール］

エンタープライズ Java 関連のプロジェクトやミドルウェアのプリセールスを経験した後、2007 年より日本 IBM で WebSphere ソフトウェアのプリセールスを担当。WebSphere Application Server を中心として Web2.0 や JVM 言語（Groovy）などにかかわり、2011 年ごろからモバイル Web や Worklight などのモバイル・ソリューションも担当している。