解説 2

迅速に開発・テストを立ち上げる

ー 開発、テスト、プロジェクト管理をクラウドで準備 ー

ソフトウェア開発を迅速に立ち上げること、つまりインストールや設定作業を省略し、ソフトウェア開発を迅速に立ち上げることが可能な基盤は、クラウド・コンピューティング（以下、クラウド）で準備することが可能です。

ダウンサイジングによってインフラ・コストが極小化したことにより、システム基盤を用途ごとに調達、設置、構築し、個別運用するというスタイルは、1980 年代から広く好まれ、これまで長く使われてきた手法です。しかしそのことが、用途ごとに異なったアーキテクチャーのサーバーの乱立を招き、データの移行も難しいため塩漬けとなり、結果としてインフラ・コストの総量を爆発的に増加させています。

この運用形態に一石を投じるインフラ提供形態がクラウド・コンピューティングです。インターネットでのパブリック・サービスに始まり、自治体が行うクラウド・センターや、企業グループでのクラウド・センター、企業内でのプライベート・クラウドなど提供母体や範囲も広がりを見せつつあります。

本記事では、旧来の個別調達型の開発と比較してクラウドの上でソフトウェア開発を行うことがどのように違うのか、特に「迅速に立ち上げる」という視点から、事例を交えつつその利点や注意点について解説します。

Article 2

Quick Start-up for Software Development and Testing

- Preparing Software Development, Testing, and Project Management Using Cloud Computing -

Quick start-up for software development: An infrastructure that enables a quick start-up for software development without any need for software installation and configuration can be prepared using cloud computing.

Traditionally, the preferred style of software development has been to minimize costs through downsizing so as to enable the system infrastructure to be used in the parts procurement, installation, assembly, and operation stages of each software development project. However, the use of multiple different architecture servers operating in parallel leads to migration difficulties as a result of the increased total infrastructure cost.

Cloud computing can make a significant difference to such an operational model. Although it started out as a public service available on internet, the use of cloud computing environments has spread to the extent that governmental bodies and business groups are now using cloud centers and individual companies are now using private clouds.

This article will look at how software development using a cloud environment differs from software development using the traditional practice of procurement for individual projects with particular emphasis given to explaining the benefits and risks of cloud computing from the point of view of achieving a quick start-up.

❶ 開発・テスト環境を迅速に立ち上げる

クラウドは IT インフラのレンタカーのようなものであると考えます。使おうと思ったらすぐ使え、短期間でも使った分だけのコストで済み、使い終わったらすぐに返せるという、共同利用のためのリサイクル可能な仕組みです。

起動し続けるシステムの運用において高効率化を実現するのはむしろ仮想化の効能であり、クラウドの本質ではありません。クラウドの効果が最も現れているのは、システムを立ち上げる瞬間とリサイクルにあり、ソフトウェア開発にとって大きな変化をもたらすものといえます。

1.1 PaaS 上で開発作業をする

ソフトウェア開発にとって必要なのは運用基盤ではなくソフトウェアの実行基盤です。ソフトウェアの実行基盤を提供するクラウドは「PaaS（Platform as a Service）」と呼ばれていますが、パブリック・クラウドで提供される PaaS として有名なものに、SalesForce 社の Force.com、Google 社の Google App Engine（以下、GAE）、Microsoft® 社の Windows Azure™ などがあります。Force.com は画面レイアウト+ Apex 言語によるプログラミングで、ブラウザー上のビルダーや Eclipse プラ

グインでの開発が可能であり、GAE は Python あるいは Java™ アプリケーションを、SDK を使って PC 上で開発し、GAE サイトにアップロードして運用します。また Windows Azure は、基本は .NET アプリケーションを Azure プラットフォームへアップロードして運用します。Force.com と GAE はパブリック・サービスのみであり、Windows Azure は製品としてライセンス購入することで、プライベートで運用することもパブリック・サービスを利用することもできるハイブリッド型として、既存の固定基盤とも互換性があります。以上の内容を**表 1** にまとめました。

表1. PaaSの比較

	言語	形態	互換
Force.com	Apex	パブリック	×
GAE	Python/Java	パブリック	×
Azure	.NET	ハイブリッド	○

PaaS の特長は、「実行基盤を待機させておく」ことにあります。アプリケーション開発者は、開発基盤の構築を待つことなく、開発したソフトウェアを即時コーディングし、テストし、運用を開始することを可能にします。それは「即時開発作業を開始できる」という点で、ソフトウェア開発シナリオに大きく影響します。

一般的に、何かを作ろう、何かを試そうと思い立っても「開発基盤」「テスト基盤」「運用基盤」の調達問題を解決しない限り、開発を始めることはできません。しかし、PaaS を利用することで、クラウド効果によるプロジェクト単位でのインフラ・コストが極小化し、開発・テストを即時始めることができます。

1.2「作ってみる」という取り組み

IBM では職位に隔たりなくパテントの申請を推奨しています。また Google 社では業務の 20% の時間をイノベーションに当てることになっています。こうした取り組みは、新しい技術やコンセプトをビジネスに活用する 1 つのアプローチといえ、「永遠にβ」といった、本番システムに小さな改変を加え続けることも、新技術を取り入れる別のアプローチであり、こういった取り組みにも、クラウドはとても便利な基盤となります。

クラウドを活用した開発プロジェクトでは単位コストがとても小さくなり、まとまった予算を組まなくてもソフトウェア開発が可能なため「稟議（りんぎ）」という承認プロセスを省略することも可能な場合もあるでしょう。

事例記事にある東京工科大学のクラウド「Lcloud」は、IBM WebSphere® sMash の開発基盤 AppBuilder をクラウド提供し、プログラミング授業に活用した PaaS の例です（本誌 24 ページ以下：インタビュー②参照）。また、秋に開催予定の iSUC（アイザック）［1］という集中研修プログラムの受付システムも、WebSphere sMash で開発されており Amazon のクラウド環境で稼働させます。こうした環境ではアプリケーションをいきなり開発し、少しずつ構築し、少しずつ改変を加え続けることを容認するというアジャイル的な発想が可能です。しかも、それを行いやすいように開発基盤そのものがクラウド上に提供されていることも重要なポイントとなります。

❷ クラウドによるプログラミング環境

PC・ワークステーションがソフトウェア開発環境の主流となって以来、プログラミング環境といえば、Eclipse や Microsoft Visual Studio® に代表されるようなクライアント・ソフトウェアとして提供されるのが通常で、すべての開発者の開発用マシンに導入して使われるものでした。一方クラウドで運用されるサービスのユーザー・インターフェースは、Web ブラウザーでアクセスする Web アプリケーションとして提供されるのが一般的です。クライアント環境には通常の Web ブラウザーさえあれば、追加のソフトウェアをインストールすることなしに、すぐにクラウド・サービスを利用することができます。しかし、このクラウドのパラダイムを利用して、迅速にプログラミング環境を構築するというアプローチにおいて、通常、長時間の集中した作業が要求されるプログラミングという開発タスクを Web アプリケーションでどこまで効率よく行えるかは議論の余地があることも事実です。

そこで次節では、IBM WebSphere sMash AppBuilder を例にとって、クラウドにより提供されるプログラミング環境のユース・ケースを眺めてみましょう。

2.1 WebSphere sMash AppBuilder

Groovy や PHP といったスクリプト言語をサポートした軽量なプログラミング・プラットフォームである WebSphere sMash は、コマンドライン・インターフェースや Eclipse プラグインとして提供される開発ツールのほかに、

AppBuilderと呼ばれる、ブラウザー・ベースの開発環境を提供しています。AppBuilder自身もWebSphere sMash のプラットフォーム上で稼働するWebアプリケーションとして実装されており、通常のWebSphere sMashアプリケーション同様、コマンドライン・インターフェースなどで起動した後は、WebSphere sMashが稼働しているマシン以外からも、AppBuilderが提供するプログラミング環境に Webブラウザーでアクセスすることができます。つまり、プログラミング環境をクラウドとして提供することを可能にします。

以下に、AppBuilderを利用したWebアプリケーション開発の流れを紹介します。

① アプリケーションの作成

一般のプログラミング環境同様、AppBuilderでの最初のタスクは新規アプリケーション（プロジェクトに相当）の作成です。Webブラウザーでアクセスした初期画面のメニューからダイアログを開いて、アプリケーションの名前やパスなどの基本情報を入力します。入力された情報に基づいてAppBuilderが稼働しているサーバー上に空のアプリケーションの構成が作成されます。

② データ・モデルの定義

WebSphere sMashでは、データ・アクセスAPIが提供されており、RDBを含むさまざまなデータへのアクセスを Groovyなどのスクリプト言語で実装することができますが、クラウド・ベースのプログラミング環境で威力を発揮するのが、ZRM（Zero Resource Model）と呼ばれるデータ・モデルを実装するフレームワークです。簡単なデータ・フィールドの記述をJSON（JavaScript Object Notation）形式で記述するだけで、データベース・テーブルとそれにアクセスするサービスまでが定義でき、アプリケーションで利用可能となります。

③ 画面のデザイン

クライアント側のユーザー・インターフェースの実装には、ビジュアルな画面デザイン用のエディターが利用可能です。画面を構成する部品（ウィジェット）はパレットからドラッグ・アンド・ドロップで挿入し、ダイアログで部品のプロパティーを編集して、画面をデザインします。部品は主に高機能JavaScriptライブラリーであるDojo Toolkitのウィジェットとして提供されていますので、タブ、グリッド（スプレッド・シート）、カレンダー付きテキスト・ボックスなどの高機能な部品を組み合わせて、リッチな画面を作成することができます（**図1**）。

④ ビジネス・ロジックの実装

サーバー側のサービスにビジネス・ロジックを追加したい場合には、Groovyなどのスクリプト言語のソース・コードを編集する作業を行います。ソース・コードのエディターは、カット、コピー、ペースト、アンドゥなどの基本的な編集に加えて、構文強調の機能も提供しています。

また、データ・フィードを組み合わせて、加工する処理はフロー・エディターでグラフィカルに定義することも可能です。フィードのソースや変換処理を画面上に配置し、それらを接続して、一連の処理を定義することにより、データのマッシュアップを実現することができます（**図2**）。

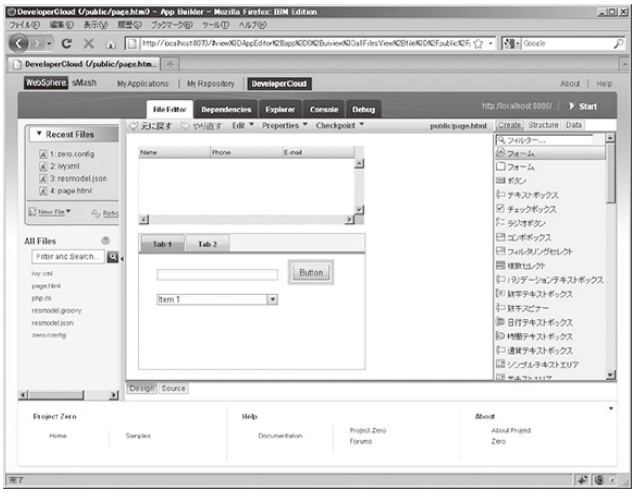

図1. 画面のデザイン

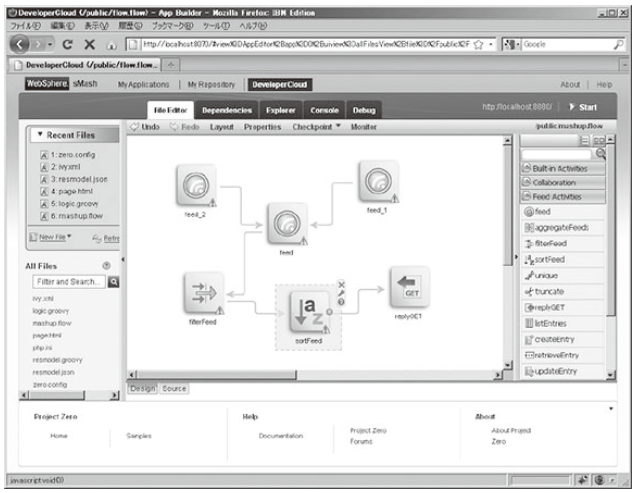

図2. フローの編集

⑤テスト実行とデバッグ

作成したアプリケーションは、配置する手間をかけずに、クリック１つでその場でテスト実行可能です。またGroovy のコードにブレーク・ポイントを設定して、その場でデバッグすることもできます。この俊敏性がクラウド・ベースの開発環境のアドバンテージです。

2.2 プログラミング・ツールのタイプ

AppBuilder での作業にも見受けられるように、テストとデバッグを除けば、クラウド・ベースのプログラミング環境で主な編集作業を遂行するためのツールは以下のタイプに分類できます。

- ソース・コード編集ツール
- 画面定義編集ツール
- ダイアグラム編集ツール

これらはまさに、既存のプログラミング環境でも代表的なツールの形態です。

まず、最も古くから行われているソース・コードの編集ツールがブラウザー上で提供される場合には、そのパフォーマンスや動作の安定性が大切です。熟練したプログラマーがキーボードにより、ストレスなく、ソース・コードの編集作業ができるためには、ブラウザーをはじめとしたクライアント側の処理速度向上に加えて、サーバー側とのインタラクションを最適化するような工夫が求められます。また、ソース・コード編集を助けるための、構文強調、入力補助、リアルタイムのエラー・問題検出などの機能の充実により、作業効率を上げていくことも重要です。

ビジュアルな画面定義に関しては、ターゲット・アプリケーションが Web アプリケーションという前提であれば、ブラウザー・ベースのツール活用が利点の１つとなります。ブラウザー上で画面を再現させながら、編集することができるため、実際の見栄えに非常に近いデザインが可能になります。HTML 要素の配置だけではなく、CSS（Cascading Style Sheet）による細かなスタイルの指定を分かりやすいインターフェースで提供できれば、これまでの Web アプリケーションの課題であった、デザイナーとプログラマーの協業という点でも、同じツール・プラットフォーム上で作業することが現実のものとなります。

ダイアグラム編集などのモデリング系のツールも、最近のブラウザーの性能や Dojo Toolkit などベクター・グラフィックスをサポートするライブラリーにより、クラウド・ベースのツールとしてブラウザー上で問題なく提供できるようになってきています。一昔前の Web アプリケーションでは考えられなかった、グラフィック・オブジェクトのドラッグや柔軟なコネクションの描画機能を備えたツールもブラウザー上で提供されています。モデリングはもともと上流工程で活用される手法であり、ターゲット・ユーザーという点でも、クラウド・ベースでツールが提供されれば、上流工程の担当者が、開発ツールの導入という煩わしい作業なしで、開発チームに参加できるようになります。

2.3 今後の課題

クラウドによるプログラミング環境の提供というアプローチの最大の魅力は、迅速に開発環境を構築できるという点にありますが、今後、広く普及するには課題も幾つかあります。その代表的なものが、オフライン・サポートです。デスクトップ PC やワークステーション上で行われる、通常の開発拠点での開発に際しては、ネットワークの常時接続が保障され、オフラインでの作業の必要性はそれほど感じないかもしれません。ただし、ノート PC などを利用したモバイル環境でも開発作業の一部を行うような場合や、サーバーの負荷によるレスポンスの低下が懸念されるような場合には、オフラインでも作業を継続できることが求められます。HTML 5 やブラウザー・プラグインを利用した Web アプリケーション用のクライアント側のデータ・ストレージや Web アプリケーション・コードのキャッシュなどの最新技術によって、オフライン・サポートがシームレスに提供されることは、プログラミング環境だけにとどまらず、クラウド・ベースのサービスがさらに広く利用されるためにも不可欠な要素であると考えます。

オフライン・サポートが、クラウド・ベースのプログラミング環境をこれまでのクライアント・ソフトウェアによる環境に近づけようという課題であるのに対して、クラウドの利点を生かす方向でのチャレンジは、共同編集機能です。すでにクラウド・ベースで提供されるツールの中には、サーバー上のデータに対する複数のユーザーの同時編集を実現しているものもあります。今後さらにグローバル化の進展が予想される開発の現場において、クラウドであるからこそ実現できる新たな価値として、リモートの技術者同士のより円滑な共同作業を実現するプラットフォームの提供が期待されます。

❸ 開発者負担軽減のためのクラウド

開発者が新たに何かを作ろうと思ったとき、意外と余計なコストとなりがちなのが「環境設定」です。何かのAPIを利用しようと思ったとき、そのAPIを利用するための設定を行うだけで、何日も要してしまうことがあります。しかし、その環境準備は開発の本質ではなく無駄なコストと考えることができ、そうした無駄な作業の負担を軽減するためにもクラウドは役立ちます。

3.1 設定済み環境を即時提供する

Amazon EC2でアプリケーションを開発する際、OSやミドルウェアがすでにインストール済みのイメージ（AMI：Amazon Machine Image）を選択し、起動すればすぐに開発が始められることは広く知られています。IBMソフトウェアもAMIで提供されているものがあり、IBMのクラウド「Development and Test Cloud」でも同様のことが可能です（**図3**）[2]。

Pay Pal Xといったサード・ベンダーが提供するAPI開発キットも提供されており[3]、インターネット決済システムの開発をクラウド上で試し、評価し、実施することもできます。またRational® 製品も含め、さらに幅広いソフトウェアが提供されています。

IBM社内にはTAP（Technology Adoption Program）と呼ばれるパイロット運用クラウドもあります。これは、ソフトウェア開発者がソフトウェアのパイロット運用を自ら行わずクラウドに任せることができる仕組みで、ソフトウェア開発者の負担を大幅に軽減します。

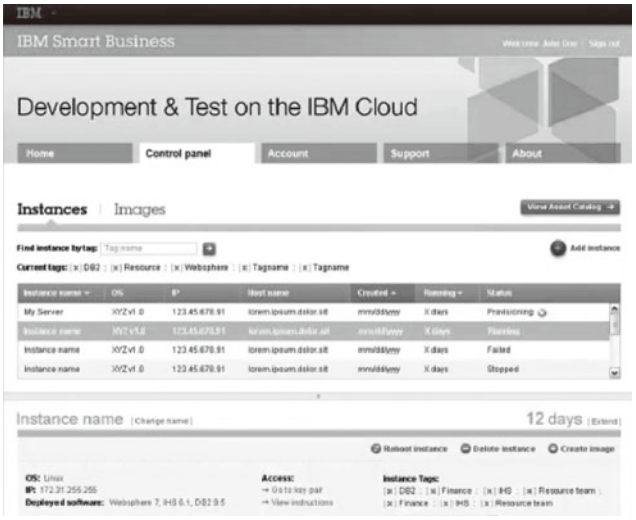

図3. Development and Test Cloud

こうした環境の提供は、開発者の作業負担を減らすだけではなく、学習時間や作業工程の短縮化により全体のコストを削減する効果があることにも注目すべきです。

3.2 プロジェクト管理環境を即時提供する

ソフトウェアを迅速に開発する別の側面として「プロジェクトを迅速に立ち上げる」ということも重要なポイントとなります。ソフトウェア開発プロジェクトを開始するには、チームを編成し、共同作業するための場所としてファイル共有、バージョン管理、バグ管理、タスク管理、進ちょく管理といった旧来からあるものに加え、最近ではフォーラム、Wiki、ブログといったソーシャル機能も利用されます。

こうした環境をソフトウェアの開発ごとに準備していたのでは当然作業を迅速に始めることはできません。そうしたものを迅速に準備するためにもクラウドは役に立っています。

オープンソースの世界では国境をまたいでチームを編成し、プロジェクトを迅速に立ち上げています。そこには「見積もり」や「発注」という概念がないため、すぐにでも開発を始められる素養が備わっており、そのような考え方を支えるためのクラウドがすでに存在しています。例えば、SourceForge.netはオープンソース・ソフトウェアを開発するプロジェクトを実施するための必要な基盤を提供しています[4]。IBM社内にも2000年から「IIOSB」、2005年から「Community Source」という、IBM社員が使うためのプロジェクト管理基盤がクラウドとして提供されてきました[5]。これらの環境を利用することで、「こういうプログラムを書いてみよう」ということが決まれば数時間後にはプロジェクト単位でバージョン管理基盤を含んだグループが生成され、すぐに開発を始めることができます。これまで多くのソフトウェアやアセットの開発が、同基盤で

図4. My developerWorks

管理・運用されています。

Web 2.0 時代になりソーシャル・ネットワークのコミュニティー機能をプロジェクト管理に利用するケースが増えてきています。例えば Google に関連する技術開発の多くが Google Group というコミュニティー機能で運用されており、IBM も My developerWorks というソーシャル・ネットワーク機能の展開が、社内外の技術者のグループ活動を支援しています（**図 4**）。

企業内でプロジェクト管理を行うクラウドに相当するものを起動しようと思えば、前記事（本誌 32 ページ以下：解説①参照）で解説した IBM Rational Team Concert™ は同様のコンセプトで利用が可能であり、企業内で起動しなくても IBM Cloud の上で利用することも可能です。プロジェクト・メンバーが自ら登録し開始できる環境がクラウド上にあれば、プロジェクトを立ち上げようと思えばすぐに環境は整い、自主的な開発プロジェクトを開始することができます。

❹ 全社的な取り組みを

しかし、こうした開発クラウド基盤は開発プロジェクトごとに準備するものではなく、あらかじめ用意されていて開発時に利用することで威力を発揮するものです。

こうした環境がもたらすものは、ソフトウェア開発を地道なものからイノベーティブなものへと変え、最新の IT 技術やコンセプトをビジネスに取り込むための、以下のような重要な活動を支えることができます。

- 余剰労力を利用する（余暇での開発）
- 試験的な開発（失敗の損失を最小化する）
- 修正を繰り返す（新陳代謝の良いシステム）

すべての開発プロジェクトがこのスタイルになるわけではありませんが、部分的に取り入れることで、新しいビジネス・モデルへの効果的な投資を実現する手法の 1 つとなることを期待します。

［参考文献］

[1] iSUC, http://www.uken.or.jp/isuc/isuc21/
[2] IBM development and test cloud, (2010). http://www.ibm.com/cloud/enterprise/beta/dashboard
[3] Pay Pal X API, https://www.x.com/, (2010).
[4] SourceForge.net, http://sourceforge.net/, (2010).
[5] Pat Gibney, IBM's New Community Source Development Strategy, (2005). http://www.developer.com/tech/article.php/3551806/QA-with-Pat-Gibney-on-IBMs-New-Community-Source-Development-Strategy.htm

日本アイ・ビー・エム株式会社
ソフトウェア事業
クライアント・テクニカル・プロフェッショナルズ
クラウド・エバンジェリスト

米持 幸寿 **Yukihisa Yonemochi**

［プロフィール］
1987 年、日本 IBM 入社。メインフレーム OS、システム・ソフトウェア障害対応技術員、解析ソフトウェア、自動運用ワークフローエンジンなどの開発を経て 2000 年よりソフトウェア・テクノロジー・エバンジェリスト。

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
シニア・テクニカル・スタッフ・メンバー

野口 雅人 **Masato Noguchi**

［プロフィール］
1990 年、日本 IBM 入社。ホームページ・ビルダー、Rational Application Developer などの製品開発に従事。現在は Web 2.0/RIA 関連のソフトウェア開発を担当。