解 説 1

ソーシャル・ネットワークを取り巻く技術常識

― プログラムできるSNS ―

ソーシャル・ネットワーキング・サービス（以下、SNS）と聞くと、コミュニティー、ブログ、友達、写真共有、つぶやきといった、個人的な情報交換活動を支援するツールと理解している人が多いと思います。ほとんどのソーシャル・ネットワークにおいてはユーザー・インターフェース（以下、UI）も重要な要素ではありますが、それだけでは今日のようにソーシャル・ネットワークが世界中に広まることはなかったでしょう。その裏側には、大量の「連携機能」が隠されており、そのおかげで情報がスムーズに伝達し、世界中に情報の輪が出来上がっているのです。また、SNS はもはや個人活動のみならず経済活動の多くの場面で使われるようになったことから、業務システムとの連携も求められてきています。本記事では、過去から蓄積されてきた Web の統合技術や、各種 SNS が持つさまざまな連携・統合機能に焦点を当て、技術の現状について解説します。

❶ 連携の必要性

SNSにはプログラミング基盤、すなわち「プラットフォーム」としての位置付けがあり、Web 2.0 時代には「プログラマブル Web（プログラムできる Web）」と呼ばれていました。今でもその血は受け継がれています。ソフトウェア開発者にとってのプログラミング・プラットフォームが、OS から始まり、ミドルウェア、Web、SNS と広まりを見せていることを意味します。

SNS は、ゲームなどのアプリケーションをプラグイン型で開発できるようにとの目的で周辺ベンダーによりプラットフォーム化が進められ、さらに今日では、マーケティングやプロダクト・セリングにも有効であるためデジタル・マーケットの基盤となっているともいえます。コンシューマー製品を取り扱う企業からすれば、デジタルの世界でビジネスを行う「仮想社会」の 1 つでもあり、業務用システムを SNS に融合させることは、成功の鍵ともなります。

また、仕事の道具として SNS を利用する「ソーシャル・ビジネス」においては、仕事上の相互関係や書き込みデータなどが、関連付けされたデータ「Linked Data」として保存されていくことに意義があります。IBM Connections を使った社員同士の意見交換や IBM Rational Team Concert（以下、RTC）の Jazz サーバーに保存される開発の議論はテキストマイニングやデータ・マイニングするための格好のデータであり、ビジネスの現場で何が起きているかを調べることができる大切な記録です。

次章からは過去を振り返りながら Web 技術や各種 SNS が持つ機能を解説します。SNS にまつわる技術の進化の過程を時系列に追い掛けることが、プログラミング・プラットフォームとしての SNS を理解するに役立つはずです。

❷ 旧来からある技術

SNS の基礎技術は Web 技術です。HTML はブラウザーという「表示装置」で人に見せるために作られたものであるため、プログラムによる機械処理には向いていません。そのため、データとして扱うことができるようにする試みの 1 つとして定着した技術が「XML」です。「旅行情報用マークアップ」や「住所情報用マークアップ」といったものを自作できるようになったことが、Web による情報連携を一層普及させたことは言うまでもありません。

「XML はシステムの中で使うもの」と理解している人を多く見かけますが、もともとのコンセプトはインターネット・コンテンツを自動処理できるようにすることでした。このため、今でもインターネットには XML を基礎としたフォーマットが多数存在しています。

❸ Web2.0 技術

ブログや Wiki といった今日の SNS の基礎となったサイトが立ち上がり始めた「Web2.0 時代」と呼ばれるころに発生した技術があります。SNS 技術の多くがこれら Web2.0 技術を基礎にしており、SNS に対して開発を行う際の必須知識です。以下にそのそれぞれのキーとなる技術を解説していきます。

フォーラムやコメント

技術というよりは、アプリケーション・サービスです。1つの話題を「トピック」「スレッド」「板」などと呼び、これに対して複数の人が順次「コメント」を書き込めるサイトが2005年ころから急激に増加しました。より以前の「パソコン通信」時代にあったフォーラムとよく似た形状のものです。

トラックバック

誰かの書き込み「A」に対してコメント「B」を書いたとき、旧来A→Bのリンクを張るのが難しかったことから、ブログ・ソフトウェアなどに搭載された機能で、コメントと同様に、別のサイトに書いた記事へのリンクを追加できる機能です。

パーマネントURL

Web上の記事がURLで正しくポイントされることを意味します。一見当たり前のように聞こえますが、実現できていないサイトも見られます。例えばFlashなどを使った動的コンテンツ型のニュース・サイトなどでは、ニュースを閲覧したときのURLに後日アクセスすると、まったく違うニュースが表示されるという状態がいまだに存在します。こういった状況は、リンクによる参照を妨げるため、近年のWebコンテンツではご法度となっています。

フィード

元来、ニュースのデータをWebブラウザーに配布するための形式でしたが、Web 2.0時代には増え過ぎてしまったニュースをクライアント側で自動処理して取捨選択するための重要な技術となりました。

フィードには大きく分けて「RSS」と「ATOM」という二種類が存在します。さらにRSSは0.9、1.0、2.0という異なったバージョンに、ATOMは、「ATOM Syndication Format」という本来のフィードとしての役割のものと、「ATOM Publishing Protocol」という書き込み用の技術とに分かれています。これらを利用すれば、今日の多くのSNSサイトやソーシャル・ソフトウェアにおいて、記事の読み出しや自動投稿などをプログラミングすることができます。

フィードに対応したSNSのトピック・データを収集するだけであれば、さまざまな製品に組み込まれている「フィード・リーダー」や「フィード・クローラー」が使えますが、ATOMに対応したサイトに対する書き込みやデータ収集を自動化するには、プログラミングを行う必要があります。

RESTful Webサービス

「RESTful」は、「ROA（Resource Oriented Architecture）」に基づく分散データのアクセス方式のことですが、しばしばシステム連携用APIの開示にも使われます。

システム間連携にはリモート・ジョブ、RPC、メッセージング、分散オブジェクトなどのスタイルがあり、非常に多くの技術が以前から乱立してきました。

「プログラム間を連携する」というコンセプトのものとRESTfulが基本にしているROAはいくらか違うもので、ROAは「リソース」という概念を基に、HTTPメソッド（PUT、GET、POST、DELETEなど）を使ってCRUD（Create、Read、Update、Delete）を実現するというアーキテクチャーです。実際にはリソースの読み書きだけでなく、機能実装にも活用されています。

Asynchronous JavaScript and XML（以下、Ajax）

ブラウザー上のJavaScriptプログラムをクライアント実行環境として使い、サーバーと通信させることによってクライアント・サーバー型のUIを実現する技術です。JavaScriptに通信用オブジェクトが標準化されブラウザーに搭載されるようになったおかげで今では一般的になりました。

Ajax登場のおかげでSaaS（Software as a Service）型アプリケーションが急速に発展し、クラウド・コンピューティング時代の到来に一役買ったのは間違いないでしょう。また、Ajax技術を活用してWebページに埋め込むことができるUIコンポーネント技術を「ウィジェット」と呼びます。

JavaScript Object Notation（以下、JSON）

JavaScriptのオブジェクトを記述する方法で構造化データを表記（Notation）する方式です。Webサービスのデータ記述にはXMLが長く重宝されてきましたが、「タグがバイト数を増やしてしまう」などの欠点が指摘されることもありました。JSONは、｛、｝、［、］、などを使って構造を表記するためにバイト数が無駄に増えることを抑制でき、XMLより軽量です。また、今日ではクライアント・プログラムの記述にJavaScriptを利用することが多いので、親和性が高いという特長もあります。サーバー用のミドルウェアにもJSONのライブラリーの活用が一般的となってきており、Javaなどからもデータを生成したり読み取ったりすることが容易になってきています。ただし、XMLと比

べてスキーマなどの拡張性や表現力に乏しく、またスキーマ定義概念がないためにプログラムの自動生成や自動読み取りなどへの対応に弱いのも事実です。

現在では、JSON 対応コードを Java で実装するための標準仕様「JSR-353」が議論されています。

❹ クライアント・モデル

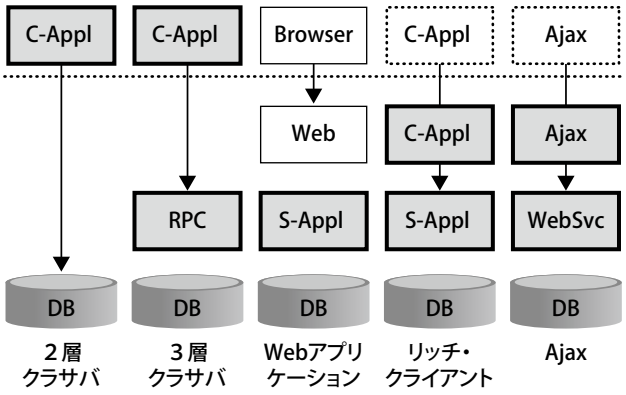
ソーシャル・ネットワークの特徴的な技術は、Web UI に見られるような Ajax、ウィジェット、そしてスマートフォンやタブレットへの対応でしょう。そして、これらに共通の特徴はクライアント実装です。本章ではそのクライアント実装技術について解説します。

リッチ・クライアントの延長

e-business 時代のアプリケーションは「サーバーでページを生成してブラウザーに送信する」という紙芝居型でした。この場合、Web ブラウザーは単なる「表示装置」であり、メインフレーム時代のコンピューター・ターミナルの表示装置サブシステムに近い存在だったといえなくもありません。

しかし今日の Web 技術では、ブラウザーで実行されるプログラム（パソコンの CPU とメモリーを利用する）や、スマートフォンやタブレットの上で実行されるプログラムが台頭してきています。クライアント・プログラミングの時代となってきているのです。

クライアント・プログラミングの流行はこれまでにも 2 回ほどありました。1 回目は 1990 年代後半の「クライアント・サーバー（以下、クラサバ）」時代であり、もう 1 回は 2005 年前後の「リッチ・クライアント」時代です。**図 1** はそれらのモデルを概観したもので、クラサバでは Windows や OS/2 などの専用アプリケーションをパソコンに配布し、Web アプリケーションではサーバーにのみアプリケーションをデプロイ（配置）していました。リッチ・クライアントはクラサバへの回帰であり、配布をネット経由で行うようになったことが特徴的です。



C-Appl : クライアント・アプリケーション
S-Appl : サーバー・アプリケーション
WebSvc : Webサービス

※太線枠はデプロイ位置、破線枠は配布後実行位置を示す。

図1. 旧来のクライアント実装モデル

今日の UI のトレンドは、「Dojo ツールキット」「jQuery」といった JavaScript ライブラリーや「HTML5」を活用した Web ブラウザー・ベースの Ajax 系アプリケーションおよびウィジェット、スマートフォンやタブレットなどへの配布型アプリケーションです。スマートフォンの専用アプリケーションは「ストア」や「マーケット」と呼ばれる配布基盤でアプリケーションをダウンロードする仕組みを使うため、リッチ・クライアントと同様にデプロイメントのコストが低い点が特長です。今や低いコストでリッチな UI を提供するのが当たり前の時代となりました。

非同期・コールバック関数型実装

Ajax、スマートフォン、タブレットのアプリケーションにおいて、サーバー呼び出しの実装方法に共通するのが「非同期型」あるいは「コールバック関数型」と呼ばれるアーキテクチャーを使う点です。旧来のクラサバでは同期型（Remote Procedure Call：以下、RPC）が一般的でしたので、この点も旧来のクライアント実装と異なる点です。

例えば、マルチプラットフォームのスマートフォン向けアプリケーションの展開基盤である IBM Worklight に搭載されているクライアント API では、リスト1のようなコードでサーバー機能を呼び出します。

この例では、上部の invokeProcedure 部分でプロシー

リスト1. Worklight API の例

```
WL.Client.invokeProcedure(
  invocationData, {
    onSuccess: procOK,
    onFailure: procFAIL
  }
);

function procOK(response){
  // 正常時の継続処理
}

function procFAIL(response){
  // 異常時の継続処理
}
```

ジャー・アダプターという機能を使ってサーバー機能を呼び出し、返事が戻ってきたら下部に定義された procOK または procFAIL のいずれかが呼び出されます。呼び出されるべき関数をパラメーター値として渡すのです。これによって旧来の RPC のようにリモート呼び出しが終了するまでアプリケーションを待ち状態（ブロック）にすることがなくなり、画面のフリーズや、別の処理がまったくできないといったユーザーをイライラさせてしまう出来事が減少しました。Ajax の XMLHttpRequest も同じ考え方で、これはサーバーが遠く最長で地球の裏側にある可能性のある、インターネット、クラウド時代の新しい常識なのです。

統合基盤としてのクライアント

Ajax やスマートフォン・アプリケーションは、旧来のものと比較すると統合基盤としての役割が強くなっている点に特徴があります（**図 2**）。これは、スマートフォン用の OS（Apple 社の iOS やオープンソースの Android）のアプリケーション・フレームワークに「アプリケーション連携」と呼ばれる機能が最初から備わっており、単独で動かすよりほかのアプリケーションと連携することを前提に考えられているからです。Ajax やウィジェットも同様で、ほかのアプリケーションと連携する機能は一般的となってきています。

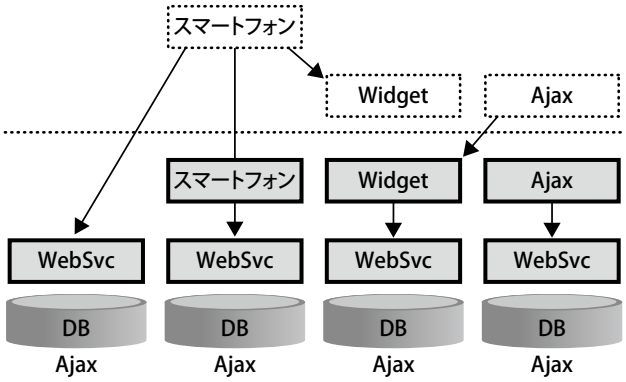


図2. クライアントによる統合

❺ ソーシャル API 群

アプリケーション基盤としての SNS

冒頭にソーシャル・ネットワークはアプリケーション基盤（プラットフォーム）であると述べましたが、実際に Facebook のタイムライン、リンク、写真、動画などの多くの機能はプラグインされたものであり、本体機能ではありません。Google+、MySpace や mixi などもアプリケーションを実行するためのウィジェット統合、認証、ストレージ、ソーシャル・グラフなどの機能を提供し、アプリケーションを追加・統合できるようにしています。スマートフォンやタブレットにアプリケーションを提供しやすくすることも重要な要件なのです。

これは、サイトをプラットフォームとして提供することでサイト上の「住民」を増やし、収益を上げるベスト・プラクティスです。この考え方は、企業システムにも応用できます。実際、IBM Connections には似た機能があり、アプリケーションの開発ツールとして IBM Social Business Toolkit を発表しています［1］。

ソーシャル・グラフ

ソーシャル・ネットワークは旧来の電話のような「ピア・トゥ・ピアの通信装置」ではなく、間違いなく「ソーシャルな通信装置」としての位置付けが強く、「複数の人がかかわり合いを持つ」ことが重要な意味を持ちます。インターネットを介し、大量のサーバー群が世界中のユーザーの活動を支える形でクラウド上に存在することが、ソーシャル・ネットワークには重要なファクターです。

このソーシャル・ネットワークにおいて重要な意味を持つ「ユーザーとユーザーの関係」を表すデータを「ソーシャル・グラフ」と呼びます（**図 3**）。ソーシャル・グラフ API にも標準がありますが、ソーシャル Web サイトの機能はすべてのサイトで共通ではありません。それぞれが特徴を持つことで人気を得ている部分があり、完全に同一にはできないからです。

もっとも有力なソーシャル・グラフを定義しているオープン標準は「OpenSocial」です。Google 社や MySpace 社などが中心となって始めたコンソーシアムで、日本でも mixi、GREE、gumi といったサイトが OpenSocial に対応しています。また、IBM 製品でも IBM WebSphere Portal、IBM Mashup Center、RTC などが OpenSocial のガジェット（Google Gadget）の統合に対応しています。

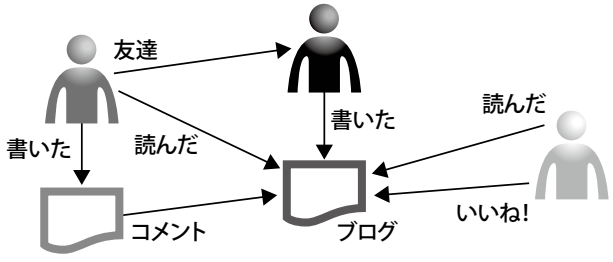


図3. ソーシャル・グラフの例

OpenSocialの柱は、画面表示に対応した「Gadget」と、ソーシャル・グラフ・データを操作するための「Graph API」です。OpenSocial はオープン技術であるため、「Apache Shindig」[2]というオープンソースのコンテナ実装もあります。

一方、対抗する別の大きな勢力はFacebookでしょう。「Facebook Graph API」[3]と呼ばれ、JavaScript向けなどにライブラリー・パッケージが用意されています。Facebookではifameをベースとした「Facebookアプリケーション」で画面にUIを埋め込み、「Facebook Graph API」でグラフ・データを操作します。グラフ・データは「FQL」というSQLに似たクエリー言語で詳細に検索することもできます。

Open Graph Protocol（OGP）

ソーシャル・グラフにおいてインターネット・サイトなどを引用する際に参考になるURL情報などを、Webページに<META>タグなどで埋め込むオープン技術です[4]。これによって、リンクを掲載した記事に自動的にサムネイル（小さなビットマップ）や引用記事のアブストラクトなどが挿入され見やすくなったり、一般ページに「いいね!」ボタンを配置してFacebookの「いいね!」にカウントさせたりするトラックバックと同様のことができるようなっています。

❻ ソーシャル・ネットワークを調べる

ソーシャル・ネットワーク分析（Social Network Analysis：以下、SNA）

ソーシャル・グラフの可視化などにより、人々の関係を調べることを指します。前述のグラフAPIでデータを収集すれば「誰と誰が友達か」「この文章は誰が読んだか」「誰が『いいね!』を押したか」「コメントはどれか」といったデータを入手できますが、それらは、文字通りグラフ化やネットワーク図にて可視化することで活用が可能となり、さまざまな情報として利用できるようになります。IBM東京基礎研究所のTENA（Text Nextwork Analysis）[5]はSNAの一種です。

メッセージ取得

ソーシャル・サイトにはOpenSocialやFacebookほどの複雑さを持たない簡便なものもあり、簡単な故に人気を博しているものもあります。書き込み量が膨大であるため、独自のAPIにてメッセージを検索したり取得したりできるようになっています。

短いメッセージを発信できるTwitterや中国の類似サイトWeiboは、多くの人々の公開された意見を収集できるよい場所であり、テキスト分析をすることでマーケットの状況が分かることをPROVISION 70号48ページ以下解説②でも紹介しました[6]。

「Twitter API」は、Twitterサイトのつぶやきを取得したり自動的につぶやいたりするためのAPIです。HTTPベースのRPC型APIであり、XMLやJSONを使います。Weiboにも同様のAPIが存在します。Javaでコーディングするための Twitter4J といったライブラリー[7]もあります。

TwitterAPIには無料APIと有料APIがあります。無料APIでは「1時間の呼び出しは350回まで」「1回のクエリーでは150行まで」などといった制約があり、工夫をしても全体の10%程度のデータしか取得できません。それでも1日当たり数十万件のデータを取得可能です。有料APIにはランクがあり、契約金額が高額になればなるほど多くのデータを取得でき、100%のデータを取得するには非常に高額のAPI契約を結ぶ必要があります。

Twitter、Weiboは共にオープンソースでJavaのAPIライブラリーが存在します。REST系のAPIを直接呼び出すのは非効率ですので、企業のシステム統合時にはそれらのAPIを活用する方がいいでしょう。

OAuth

ソーシャル・ネットワークを語る上で重要となる、サイト間での権限委譲のオープン技術です。smart.fm の真武氏の記事[8]によれば以下の通りとされています。

- あらかじめ信頼関係を構築したサービス間で
- ユーザーの同意の下に
- セキュアにユーザーの権限を受け渡しする
- セキュアにユーザーの権限を受け渡しする

（「ゼロから学ぶOAuth」より引用）

例えば、この技術はサインアップの伝播（でんぱ）に利用されていることがあります。あるサイトAへのサインアップが終わっており、そのユーザーIDでほかのサイトBへのログインを試みると、一部のデータがAからBへ引き継がれることがあります。これは、Bのサイトのプログラムが Aのサイトへアクセスし、ユーザー・データを参照した（アクセス

の許可が出た）ためです。

また、ソーシャル・アプリケーションが SNS 内のグラフにアクセスする場合にも利用されます。OAuth に対応したサイトでアプリケーションを作成し、SNS のアプリケーションとして登録してあると、相互認証によってアプリケーション・サイトのプログラムは SNS 上のグラフにアクセスできるようになります。これによって、SNS 上のユーザー同士の関係、つぶやきや日記、カレンダー、「いいね!」などの情報にアクセスする権利を得ることができるのです。

❼ まとめ

冒頭よりここまで、Web 技術、Web2.0 技術、SNS のプラットフォーム技術などを順番に説明してきました。SNS はこうした技術の集大成であり、活用するにはとても幅広い知識が必要となることが分かるはずです。ただし、プログラミング・モデルや思想が変化していることも重要で、それを理解した上で使い分ける必要があります（**図 4**）。

分散プログラミングを経験した人であれば多くの方がお気付きと思いますが、ソーシャル・ネットワークを取り巻く技術のほとんどは Web 技術に基づいており、非常に緩い結合（粗結合）技術を採用しています。これはそもそも、違う運用母体がインターネットで結合されることを前提にしているからです。API ひとつをとっても、DB ドライバーやローカル結合の Java ライブラリーのようなものではなく、XML や JSON を使った REST などの Web サービスといった疎結合技術を利用したものが主流になってきています。

伝統的なものと比較し、ソーシャル・ネットワークにおける UI 技術や結合技術の思想の違いを整理すると次のようになります。

1. 型定義（IDL）を利用しない
2. 分散トランザクション制御をしない BASE 指向
3. プラットフォーム非依存（XML、JSON など）
4. クライアントを経由したシステム連携
5. 結合方式の公開
6. 多対多結合
7. 一部無償、一部有償（フリーミアムとも呼ぶ）
8. アプリケーション・プラットフォーム
9. クラウド的運用（リソース共有、自販機型）

図4. 新しいアプリケーション基盤

ソーシャル・ネットワークには旧来の Web（1.0）や Web 2.0 から引き継がれたさまざまな技術に加え、アプリケーション・プラットフォームとしてのさまざまな機能や、ほかのサイトとの連携機能などが大量に含まれていることに触れてきました。そしてそれらの根底には「ユーザー同士のつながり」や「サイト同士のつながり」を意識した技術改革が継続的に進められていることに注目していただきたいと思います。この視点が職場環境を「働く人のつながり」で改革するための一助となれば幸いです。

［参考文献］

［1］ IBM: IBM Social Business Toolkit, http://www.ibm.com/cloud-computing/social/jp/ja/toolkit/ (2011).
［2］ Apache: Apache Shindig, http://shindig.apache.org/ (2007).
［3］ Facebook: Facebook Graph API, https://developers.facebook.com/docs/reference/api/ (2012).
［4］ OGP, http://ogp.me/ (2010).
［5］ IBM: テキスト・ネットワーク分析 -TENA- , http://www.research.ibm.com/trl/projects/tena/
［6］ 米持 幸寿：PROVISION 70 号「震災時ソーシャル・ネットワークの効果と脅威」, pp48-53（2011）.
［7］ Yusuke Yamamoto: Twitter4J, http://twitter4j.org/ja/index.html (2007).
［8］ 真武信和：「ゼロから学ぶ OAuth」, http://gihyo.jp/dev/feature/01/oauth/0001（2009）.

日本アイ・ビー・エム株式会社
スマーター・シティー事業
クラウド・マイスター

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・ソフトウェア障害対応技術員、ジョブ･ワークフロー･エンジン開発、Web アプリケーション開発など経て先進技術専門のソフトウェア事業エバンジェリストとして 10 年ほど活動。現在はクラウド、ビッグデータ処理、テキスト分析、ソーシャル･ネットワーク、スマートフォン･アプリケーション開発などの新規事業開発の技術リーダー。著書多数。