特集

Web2.0の現在と展望

2. Web2.0の情報アーキテクチャ

■川崎 有亮　(株)リクルート

Web2.0の持つ概念やそれを実現する技術は，ある日突然生まれたものではない．その理解には，1990年代後半から脈々と積み重ねられた技術要素を背景として発展してきた現在のインターネットの状況を的確に把握しておく必要がある．本稿では情報アーキテクチャとしての側面から，Web2.0において重要となる主なデータフォーマットやAPI仕様などの具体的な技術や，Web2.0的サービスの普及に伴って注目されているマッシュアップについて解説する．

Web2.0を支える技術トピック

2005年の流行語大賞トップテンにランクインした「ブログ」という言葉は，単なるブームにとどまらずWebから社会全体に定着してきた感もある．現在のインターネットを表すキーワードである「Web2.0」も，まずはマーケティング向けの最新バズワードの代表格として見られがちだが，Tim O'Reillyによる論文の中では「Web2.0的」と呼べる企業姿勢の定義に加えて，実際のサービス・製品を実現するための具体的な技術が紹介されているので，そこから実体を捉えておく必要があるだろう．

「SOA（Service Oriented Architecture）」や「セマンティックWeb」といったキーワードも個々の技術や構想ではWeb2.0と重なる部分が多いが，今のところWeb2.0のような大きなムーブメントとなるまでに至っていない．そもそもターゲットの違いもあるが，しかし実際にはWeb2.0もその実装としては，SOAやセマンティックWebに関連して研究・実現されてきた技術の上に成り立っている側面が大きい．

■ブログとRSS/Atomの普及

Timは，ブログの興隆がWeb2.0時代に最も重要な役割を果たしたと指摘している．ブログは，企業や組織が作成したページを閲覧するパブリッシング型から，利用者自らが情報を発信していく参加型に，Webの形態を大きく変えようとしている．

Webサイト上に日々の出来事を書き綴ったWeb日記形態のWebサイトは従来から多く存在していたが，それらとブログの違いはRSSフィードにあるといえる．RSSは，サイト更新情報をはじめとするメタ情報をXMLにより配信するためのフォーマットである．

そのRSSはその仕様策定の経緯により複数のバージョンが並存している．RSSバージョン2.0はReally Simple Syndicationの略であり，日本国内で採用の多いRSSバージョン1.0はRDF Site Summaryの略とされている．

図-1に示すのは，Six Apartのブログ管理ソフトウェアMovable Typeが生成するRSS 2.0フォーマットの更新情報サンプルデータである．

XML文書中の<item>要素1つが，それぞれブログ

```
<rss version="2.0">
  <channel>
    <title>ブログサイト名</title>
    <link>http://blog.example.com/</link>
    <description>ブログサイトの説明文.</description>
    <language>ja</language>
    <copyright>Copyright 2006</copyright>
    <lastBuildDate>Wed, 15 Nov 2006 00:00:00 +0900</lastBuildDate>
    <generator>http://www.sixapart.com/movabletype/</generator>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <item>
      <title>ブログ記事名</title>
      <description>ブログ記事の概要文.</description>
      <link>http://blog.example.com/archives/post.html</link>
      <guid>http://blog.example.com/archives/post.html</guid>
      <pubDate>Wed, 15 Nov 2006 00:00:00 +0900</pubDate>
    </item>
  </channel>
</rss>
```

図-1　RSS 2.0 フォーマット例

の各記事(1 ページ)に対応する．RSS の用途はブログに限らないため，サイト全体の構成・最新情報を表すために必要な情報を 1 ファイルに集約してダイジェストとして配信できる．そのため，1 サイトにつき RSS を 1 ファイルずつ作成するのが基本となるが，1 サイトから目的別に複数の RSS を配信する場合もあれば，逆に複数のサイトの情報を 1 つの RSS に集約する利用方法もある．

RSS バージョン 1.0・バージョン 2.0 に加えて，2005 年 12 月には Atom バージョン 1.0 仕様も確定した(RFC4287)．Atom は，サイトのメタ情報や全文配信の機能が強化されただけでなく，AtomPP（Atom Publishing Protocol）としてコンテンツの編集を行うための仕様も含んでいる．このほかにも RSS 0.91 や Atom 0.3 といったバージョンも存在したが，最近は採用サイトは減少しつつある．

RSS・Atom はどちらも XML を利用しているが，語彙・フォーマットが異なるので完全な相互変換は難しく，またすでに多数のサイトでそれぞれ普及が進んでしまったため，フィードの受信側（クライアント）で各フォーマットに対応する必要が生じている．今後も完全な仕様統一は難しいと考えられている．

さて，ブログにおける RSS フィードの主たる導入目的としては，ブログの読者がいちいちサイトに訪れなくても更新情報を受け取れるメリットが挙げられる．RSS リーダー（RSS 受信・閲覧用の専用ソフト）を利用することで更新情報が通知されるため，更新された場合にのみアクセスすれば済むようになった．

また，Web サイトを記述する HTML は CSS が導入されていても，どうしても文書の構造と体裁の分離を完全には実現できないため，計算機を用いて HTML 文書内の意味を解釈して完全に自動処理することは難しい．しかし，RSS フィードではサイト全体と記事ごとのメタ情報が各々分離して取得できるため，自動的な解釈も限定された語彙ながら比較的，容易に実現することができる．

メタ情報の蓄積・加工・伝播が容易になったことを受けて，RSS フィードを軸としたサイト間を超えたコンテンツの流通や，それに伴うサービスの連携が活発となってきた．

■検索 API を提供する Web サービス

Web サイトの検索サービスでは，通常は各サイトのフォーム欄にキーワードを手入力して，その検索結果が列挙された HTML ページ上のリンクから目的の Web サイトを探す手順になるが，Google や Yahoo! は，人間でなくアプリケーションプログラムからも同様に検索サービスを利用できる API を相次いで公開している．

Google は SOAP/WSDL ベースの検索 API[☆1] を，Yahoo! は REST ベースの検索 API[☆2] をそれぞれ提供する．検索クエリおよび結果を返すフォーマットは両社で異なるが，どちらも XML 形式で検索結果が得られる．ただし，人手による検索利用とは異なり，プログラムからは同時に多数の検索クエリを生成できてしまうため，アプリケーションごとに 1 日に検索できるクエリ数が

☆1　http://code.google.com/apis/soapsearch/
☆2　http://developer.yahoo.com/search/

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:typens="urn:GoogleSearch"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <typens:doGoogleSearch>
      <key xsi:type="xsd:string">XXXX</key>
      <q xsi:type="xsd:string">Web 2.0</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">true</filter>
      <restrict xsi:type="xsd:string">countryJP</restrict>
      <safeSearch xsi:type="xsd:boolean">true</safeSearch>
      <lr xsi:type="xsd:string">lang_ja</lr>
      <ie xsi:type="xsd:string">utf8</ie>
      <oe xsi:type="xsd:string">utf8</oe>
    </typens:doGoogleSearch>
  </soap:Body>
</soap:Envelope>
```

図 -2　Google SOAP Search API 向けの SOAP リクエスト例

制限される運用となっている．

図 -2 に示すのは，Google の提供する Google SOAP Search API 向けの検索を行う SOAP リクエストのサンプルコードである．

WSDL（Web Services Description Language）は，Web サービスの具体的なインタフェースを定義するための XML 形式の言語である．WSDL により，異なるソフトウェア間でデータオブジェクトの形式を明確にした上で，相互に通信が行える．また，SOAP（Simple Object Access Protocol）はサーバ間でリモートプロシージャコールを実現するプロトコルである．ここでは，SOAP over HTTP として WSDL による定義に則った検索クエリと検索結果の通信に利用している．

サービス指向アーキテクチャ SOA を実現するための技術の 1 つとして注目され，またその実用化の中で厳しい実証評価も繰り返されてきた SOAP/WSDL であるが，オープンな Web サービスにおける使い勝手としては REST の評価が高まっている．

REST（Representational State Transfer）は，従来からの Web の概念を延長したアーキテクチャスタイルで，その歴史は HTTP/1.1 の策定に遡る．ステートレスな設計が重要であり，すべてのリソースにそれぞれ識別子（URI）を持たせる．HTTP プロトコルの持つ GET/POST/PUT/DELETE の各メソッドを活用して，リソースに対する CRUD（Create/Read/Update/Delete）操作を提供する（表 -1）．

Create	リソースの作成	POSTメソッド
Read	リソースの取得	GETメソッド
Update	リソースの更新	PUTメソッド
Delete	リソースの削除	DELETEメソッド

表 -1　REST の CRUD と HTTP メソッドの対応

ブログに特化した検索サービスの Technorati も，REST ベースの検索 API[☆3] を提供している．検索対象となる Web サイトをブログに特化し，わずか 10 分前に投稿されたブログ記事も検索結果に登場させるなど，RSS によるメタ情報配信を最大限に活かした例といえよう．

図 -3 と図 -4 に示すのは，Technorati の提供する Technorati API 向けの検索を行う REST クエリと，検索結果のレスポンス XML のサンプルである．Technorati は，この独自の tapi 形式に加えて RSS 2.0 形式での検索結果出力にも対応している．これにより，RSS リーダでブログ検索結果を随時受信して新着記事を閲覧できる．また他サイトの更新情報と同様に検索結果をプログラム処理することが容易になった．

ほかにも，Amazon の提供する Web サービス[☆4] で

☆3　http://technorati.com/developers/api/
☆4　http://developer.amazonwebservices.com/

```
http://api.technorati.com/search?format=xml&language=ja&key=XXXX&query=Web+2.0
```

図 -3　Technorati API 向けの REST リクエスト例

```
<?xml version="1.0" encoding="utf-8"?>
<!-- generator="Technorati API version 1.0 /search" -->
<!DOCTYPE tapi PUBLIC "-//Technorati, Inc.//DTD TAPI 0.02//EN"
          "http://api.technorati.com/dtd/tapi-002.xml">
<tapi version="1.0">
  <document>
    <result>
      <query>Web 2.0</query>
      <querycount>22758</querycount>
      <rankingstart>0</rankingstart>
    </result>
    <item>
      <weblog>
        <name>ブログサイト名</name>
        <url>http://blog.example.com/</url>
        <rssurl>http://blog.example.com/index.xml</rssurl>
        <atomurl></atomurl>
        <inboundblogs>0</inboundblogs>
        <inboundlinks>0</inboundlinks>
        <lastupdate>2006-11-15 09:00:00 GMT</lastupdate>
      </weblog>
      <title>ブログ記事名</title>
      <excerpt>ブログ記事の概要文. </excerpt>
      <created>2006-11-15 09:00:00 GMT</created>
      <permalink>http://blog.example.com/archives/post.html</permalink>
    </item>
  </document>
</tapi>
```

図 -4　Technorati API 向けの XML レスポンス例

は，SOAP/WSDL と REST の両方式が提供されている．Amazon にとっては，書籍用の ISBN コードに加えて自社サイトで扱うすべての商品に付与した ASIN コードの提供こそが重要であり，プロトコルは適材適所で使うべきという考え方だろう．SOAP/WSDL と REST の相互比較や優位性についてはここでは論じないが，RSS・Atom のフォーマット不統一と同様に，代替可能な部分を持った実装技術が複数並列して，仕様の確定・統一を待たずに具体的なサービスが続々登場してくる状況は，技術的な側面における Web2.0 らしさの 1 つの表れであるといえよう．

Ajax の浸透と発展

2004 年に登場した Google Maps[☆5] と Gmail[☆6] を先頭に，Web アプリケーションは新たな時代に突入した．Web2.0 的なアプリケーションに求められる［Rich User Experiences］を最大化するための技術要素として，Ajax（Asynchronous JavaScript + XML）が重要となっている．この Ajax という言葉は，2005 年 2 月に Jesse James Garrett によるエッセイ [1] の中で命名された．Garrett は，Ajax を以下の技術の組合せであると定義している．

- XHTML および CSS を利用した標準に基づく表現
- DOM を利用した動的表示とインタラクション
- XML および XSLT を利用したデータ交換や操作
- XMLHttpRequest を利用した非同期データ取得
- それらすべてを結びつける JavaScript

これらの個々の技術は Ajax という言葉が誕生する以前から存在したものであり，現存するほとんどすべての

☆5 http://maps.google.com/
☆6 http://mail.google.com/mail/

```
var http;
if ( typeof(window.XMLHttpRequest) != 'undefined' ) {
    http = new XMLHttpRequest();
} else {
    http = new ActiveXObject('Microsoft.XMLHTTP');
}
var callback = function () {
    if ( http.readyState != 4 ) return;
    var stat = http.status;
    var elem = document.getElementById('news_here');
    if ( ! stat || ( 200 <= stat && stat < 300 ) ) {
        elem.innerHTML = http.responseText;
    }
};
http.onreadystatechange = callback;
http.open('GET', 'news.txt', true);
http.send('');
```

図 -5　XHR を利用したスクリプト例

Web ブラウザで利用できるなど，進歩のスピードが速い Web の世界の中ではすでに枯れた技術の部類に入る．Ajax はそれらの技術を組み合わせることでリッチな表現を可能にし，従来の Web では実現不能であると諦められていた直感的なインタフェースを実現できる点が重要である．

JavaScript は 1995 年に登場してから不遇の 10 年を過ごしてきたといわれるが，Google Maps をはじめとする Ajax アプリケーションの普及に伴って，Web サービスの中で最も利用者側に近い環境で動作するプログラミング言語としての JavaScript が改めて熱く注目されている．

■XMLHttpRequest の再発見

Ajax の主要素の 1 つである XMLHttpRequest は，JavaScript で Web ブラウザ～サーバ間の非同期通信の機能を提供するクラスである．もともと Microsoft 社の Web ブラウザ Internet Explorer において ActiveX 経由で利用可能となった XMLHTTP の機能を原型に，Mozilla が XMLHttpRequest として採用したのが始まりである．続けて Opera や Safari といったその他の主な Web ブラウザにも普及することでデファクトスタンダードとなった経緯がある．Internet Explorer バージョン 7 では従来からの XMLHTTP に加え，XMLHttpRequest も逆輸入されるかたちで採用された．なお，開発者たちはスペルの長い XMLHttpRequest を XHR と省略して表記することも多い．

Ajax アプリケーションの定義には複数の解釈があり，狭義の Ajax としては XHR を用いた XML データの非同期通信が必須とされる．しかし実際には，JavaScript で非同期通信を実現する手法は XHR 以外にも存在する．XHR を用いた場合も，同期通信や XML 形式以外のファイルの通信も可能である．サーバ内部の実装手法やクライアント～サーバ間の通信方式よりも，Web サービスの提供する機能や付加価値，良好なユーザエクスペリエンスの実現の方がより重要であると考えられるため，現在は XHR による XML データの非同期通信を伴わない場合も広義の Ajax アプリケーションに含まれると考えられている．

図 -5 は，XHR でサーバ上のテキストファイルを受信してページを更新するサンプルコードである．XHR 登場の経緯により，現在主流である Internet Explorer バージョン 6 で動作させるために，ブラウザごとの判別を行って互換性を確保する必要がある．Microsoft 社の推奨する手順では，バージョン 6 以外の Internet Explorer での性能や互換性を向上させるために，さらに複雑な判別処理を必要としている．

■JavaScript ライブラリの充実

JavaScript の重要度が再認識され，多数の Ajax アプリケーションが開発されるに従い，開発環境を整える必要が増してきた．クライアント PC の CPU 性能の向上を受けて JavaScript の処理性能は実用上問題のないレベルに到達しているが，JavaScript アプリケーションの開発を困難にしているのは，JavaScript のテスト・デバッグ環境が近代的な他言語と比較してかなり劣っている現状にその一因がある．ヨーロッパ電子計算機工業会（ECMA）によって標準化が進められ，ECMAScript として言語仕様も改めて定義されたが，XHR も含めたブラウザ間の非互換性は現在もまだ残っている．

そのような混乱した状況を緩和・改善すべく，多数の JavaScript ライブラリが登場している．単一の機能を提供する小さなスクリプトから，ユニットテストを含めた多くの機能を提供するツールキット，サーバ側の他言語と連携するフレームワークまで，ほぼ毎日のように新しいライブラリのリリースが続いている．

それらの JavaScript ライブラリの中でも代表格といえるのが，Sam Stephenson のリリースする prototype.js である．JavaScript におけるオブジェクト指向プログラミングを助けるメソッドや，各ブラウザでの互換性を確保したクロスブラウザ対応 Ajax クラスなどが提供されている．図 -5 と同等のコードも，prototype.js のクラスを活用すると図 -6 のように単純化される．

また，prototype.js のように個人がリリースする草

```
new Ajax.Updater('news_here','news.txt',{method: 'GET'});
```

図 -6　prototype.js を利用したスクリプト例

ライブラリ	サイト URL	ライセンス	作者
Direct Web Remoting	http://getahead.ltd.uk/dwr/	Apache License 2.0	Getahead
dojo	http://dojotoolkit.org/	AFL / LGPL (dual)	Dojo Foundation
Google Web Toolkit	http://code.google.com/webtoolkit/	Apache License 2.0	Google Inc.
jQuery	http://jquery.com/	MIT / GPL (dual)	John Resig
MochiKit	http://mochikit.com/	MIT / AFL (dual)	Mochi Media, LLC
prototype.js	http://prototype.conio.net/	MIT License	Sam Stephenson
Rico	http://openrico.org/	Apache License 2.0	Richard Cowin
script.aculo.us	http://script.aculo.us/	MIT License	Thomas Fuchs
Spry framework for Ajax	http://labs.adobe.com/technologies/spry/	BSD License	Adobe Systems Inc.
Yahoo! UI Library	http://developer.yahoo.com/yui/	BSD License	Yahoo! Inc.
XML.ObjTree	http://www.kawa.net/	Artistic License	川崎　有亮

表 -2　主な JavaScript ライブラリ

の根的なライブラリだけでなく，Google・Yahoo!・Adobe といった大手企業が提供するライブラリも増えてきた．中でも注目されている主要な JavaScript ライブラリを表 -2 に示す．

■XML を補完する JSON フォーマット

データ交換のフォーマットとして XML の持つ役割を考えると，クライアント～サーバ間の通信に限らず，サーバ～サーバ間においても XML の標準性はきわめて有効であり，今後も重要な位置を占め続けるであろう．

しかし Web サービスの用途によっては，XML ほどの厳密性がそれほど必要とされない分野も増えてきた．クライアント側は基本的に JavaScript で稼働するため，クライアント～サーバ間のデータ形式としては，JavaScript ネイティブの記述形式に近い JSON（JavaScript Object Notation）が脚光を浴びつつある．2006 年 7 月には，JSON は RFC 4627 として仕様が改めて明確にされた [2]．

図 -7 は先に登場した図 -1 の RSS フィード内容を，そのまま JSON 形式に変換したものである．属性や CDATA，文字エンコーディングの指定など XML の持つすべての機能を JSON で再現できるわけではないが，JavaScript を利用した Web アプリケーション実装において必要となるデータはすべて JSON 形式で表現することが可能である．

JSON は JavaScript が言語としてネイティブで理解できる形式であるため，Web ブラウザにおける処理が最も軽量になる点が大きなメリットとなる．クライアント側だけでなく，サーバ側でも冗長な XML コードを生成する必要がなくなる．ここでも通信フォーマット仕様自体よりも，データの内容自身とそれによって Web サイト上で実現できることが重要とされている．

万年β版ソフトウェアとテスト

パソコン向けソフトウェアの販売では，従来の FD/CD-ROM といったメディア販売の形態から，最近は箱パッケージを介在させないダウンロード販売も多くなってきた．ブロードバンド回線の普及に伴い，ソフトウェアのネットワークダウンロードにかかる時間・コストが低下している背景がある．パソコン購入時にプリインストールされるソフトウェアも，初回起動時にインターネット経由で最新版をダウンロードし，自身を自動更新するものも多い．さらに，メディア販売・ダウンロード販売に加えて，ソフトウェア自体をネットワーク経由のサービスとして利用するシーンが増えている．

ソフトウェアが Web サービスとして提供されることで，サービス提供者（ソフトウェア開発者）は不具合の対策や機能強化など，随時バージョンアップを実施することが可能になった．従来と比べて飛躍的にソフトウェア更新頻度を高められるだけでなく，場合によっては新機能の一部利用者への先行限定提供や，前バージョンへの差し戻しもサービス提供者側で自在にコントロールできるようになった．

従来のソフトウェア製品のような企画～開発～販売～保守のフェーズを経ずに，新たなサービスが次々に提供されることになる．中には完成度よりもリリース速度を優先することが必要な場合もある．従来ならば限られた

```
{
  "rss": {
    "@version": "2.0",
    "channel": {
      "title": "ブログサイト名",
      "link": "http://blog.example.com/",
      "description": "ブログサイトの説明文.",
      "language": "ja",
      "copyright": "Copyright 2006",
      "lastBuildDate": "Wed, 15 Nov 2006 00:00:00 +0900",
      "generator": "http://www.sixapart.com/movabletype/",
      "docs": "http://blogs.law.harvard.edu/tech/rss",
      "item": {
        "title": "ブログ記事名",
        "description": "ブログ記事の概要文.",
        "link": "http://blog.example.com/archives/post.html",
        "guid": "http://blog.example.com/archives/post.html",
        "pubDate": "Wed, 15 Nov 2006 00:00:00 +0900"
      }
    }
  }
}
```

図 -7　JSON フォーマット例

一部のユーザのみで動作テストを行ったβ版ソフトウェアでも，そのまま一般公開するサイトが多い．広くユーザに使ってもらいながら，サービスの改良を続けることになる．万全なサービスを提供することよりも，いち早くサービスを提供し随時アップデートすることが尊重される．ソフトウェア検査に対する概念が大きく変化している．

「β版」というフレーズも，テストの段階を表すための言葉ではなく，単にサービスの鮮度を表すために利用される場合も多い．ソフトウェアの改良が常に続くので，いっこうにβ版から抜け出さないままサービスが提供され続ける．Google の Web メールサービス Gmail は，初公開から 2 年経った現在もβ版のままである．2006 年 9 月現在，570 万人の利用者を抱える日本最大のソーシャル・ネットワーキングサイト mixi（ミクシィ）[☆7] にも，いまだβ版の表記が残る（図 -8）．

http://mail.google.com/

http://mixi.jp/

図 -8　Gmail・mixi のロゴ画像

■軽量プログラミングモデルと疎結合

RSS や REST ベースのシステムを導入する際のキーワードとして，疎結合（loosely coupled）が挙げられる．従来型の企業システムに見られたような密結合した複数システム間の調整は避け，各システムが緩やかに連携する運用を実現する．API によりデータ操作のインタフェースを確定することで，API の内側・外側の具体的なシステム実装方式は互いに制限されなくなったほか，各システムごとに動的に改良を加えることも可能である．

提供される API を最大限に活用することで，アプリケーション開発の自由度が高まった．API 仕様に沿う限り，システム環境やプログラムの開発言語は問われない．サービスの性格によって，信頼性やスケーラビリティの確保を目的として Java 言語が選択される場合もあるが，Perl/Python/PHP/Ruby といったスクリプト言語を採用した Web サービスも非常に多い．このような軽量プログラミング言語を用いた迅速な開発が求められている．

Ajax アプリケーションにおいては，サーバとの通信が HTTP プロトコルであるため内容も把握しやすく，また JavaScript で記述されたクライアント側ソース

☆7　http://mixi.jp/

コードすべてを閲覧して，利用者が独自の改良を加える（ハックする）ことも比較的容易だ．

しかし，実装が優先されたサービスが提供されていくなかで，その弊害も見受けられるようになってきた．国際規格のように厳密な調査・検査を経ずに実装が進むため，前述した RSS フィードのバリエーションが乱立したような事態も発生する．詳細な仕様の明文化が足りない曖昧な状態となれば，必要な機能が不足した場合に独自の解釈で機能を追加される．それらの実装が繰り返された後では，相互運用性の確保も徐々に難しくなるだろう．

ブログ間を相互にリンクで接続することで，ブログの発展に大きな役割を果たしたトラックバック（trackback）機能も，認証機能が整備されていれば spam の問題も避けられただろうという意見もある．とはいえ，後日になって仕様を後悔するよりも，システムの実装を進めて新しいサービスを利用できる点が優先されている．

マッシュアップと今後のアーキテクチャ

先に挙げた Google・Yahoo!・Amazon といったネットの「あちら側」にある企業が，検索サービスや地図データといった自社コンテンツを無償で API 提供するのは，なぜであろうか？　自社サイトへのトラフィック（アクセス数）の増加とそれに伴う広告収入も大きな目的として考えられるが，それに限らない．API の公開により，API を利用したサービスを利用者に提供するサードパーティの登場が期待できる．鈴木[3]は，API 提供者が対応しきれない多くの最終利用者へのリーチをサードパーティが拡げる点から，Web2.0 のビジネスモデルを指摘した．

もともと API 提供も考えて構築したサービスでない既存の Web サイトから，新たに API 提供する際には以下のような懸念点が考えられる．

- データの利用目的・露出場所の把握が困難である
- 競合他社もデータの比較・傾向分析に活用できる
- API 提供にかかるコストはそのまま売上に直結しない
- 自社サービスと，サードパーティのサービスの競合
- API 経由のコンテンツ再配布の著作権管理の明確化

このようなリスクを乗り越えて，国内・海外からも企業が自社データを Web サービスとして API 経由で提供する事例が増えてきた．

■マッシュアップによるニッチサービスの提供

複数の Web サービス等を組み合わせて，新たなサービスとして提供する形態をマッシュアップ（mashup）と呼ぶ．マッシュアップとはもともと音楽関係の用語で，複数の曲（音源）を 1 つの曲として合成する手法を転じて，Web サイトについても用いられるようになった．他社の既存サービスを利用して，その上に自社サービスを構築し利用者に提供する．

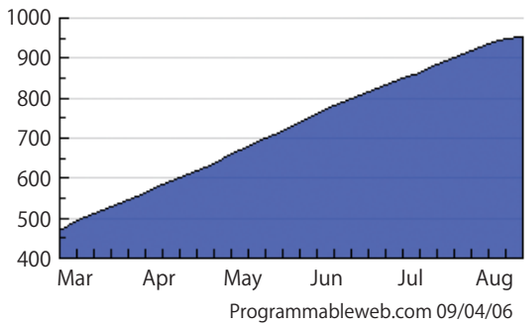

図 -9　最近 6 カ月のマッシュアップサイト数

Google の地図サービス Google Maps では，Google 自身による公式の API 提供開始前からリバースエンジニアリングによって，地図データと他データのマッシュアップを実現するハックが見られた．世界のマッシュアップ事例を掲載する ProgrammableWeb[☆8] には現在，1 日あたり平均 2.7 サイトが新規に登録されている．世界中のマッシュアップサイトの総数も，2006 年 9 月には 1,000 件を超えた（図 -9）．

サン・マイクロシステムズとリクルートは，2006 年 6 月からマッシュアップサイト開発のコンテスト Sun×RECRUIT Mash up Award[☆9] を開催した．コンテストに合わせて，リクルートの持つ不動産・宿・アルバイト・中古車といった情報コンテンツを API 経由で提供した．約 2 カ月の開催期間に 50 件を超える Web サイト（マッシュアップ作品）の応募があり，ネットの「こちら側」にある企業から「あちら側」にいる開発者たちに対する新たなアプローチとして注目された．

入賞作の 1 つである「家すぐ MAP」[☆10] は，リクルートの住宅情報と Google Maps の地図データに加えて，国土地理院の土地利用データを組み合わせることで，物件のある地域が 30 年前は田畑だったのか工業地帯だったのかも地図上で確認することができる（図 -10）．このような API 提供側だけでは対応しきれない用途に対しても，マッシュアップによりサービスを構築できる．

☆8 http://www.programmableweb.com/
☆9 http://jp.sun.com/mashupaward/
☆10 http://convivial-web.com/PHMaps/

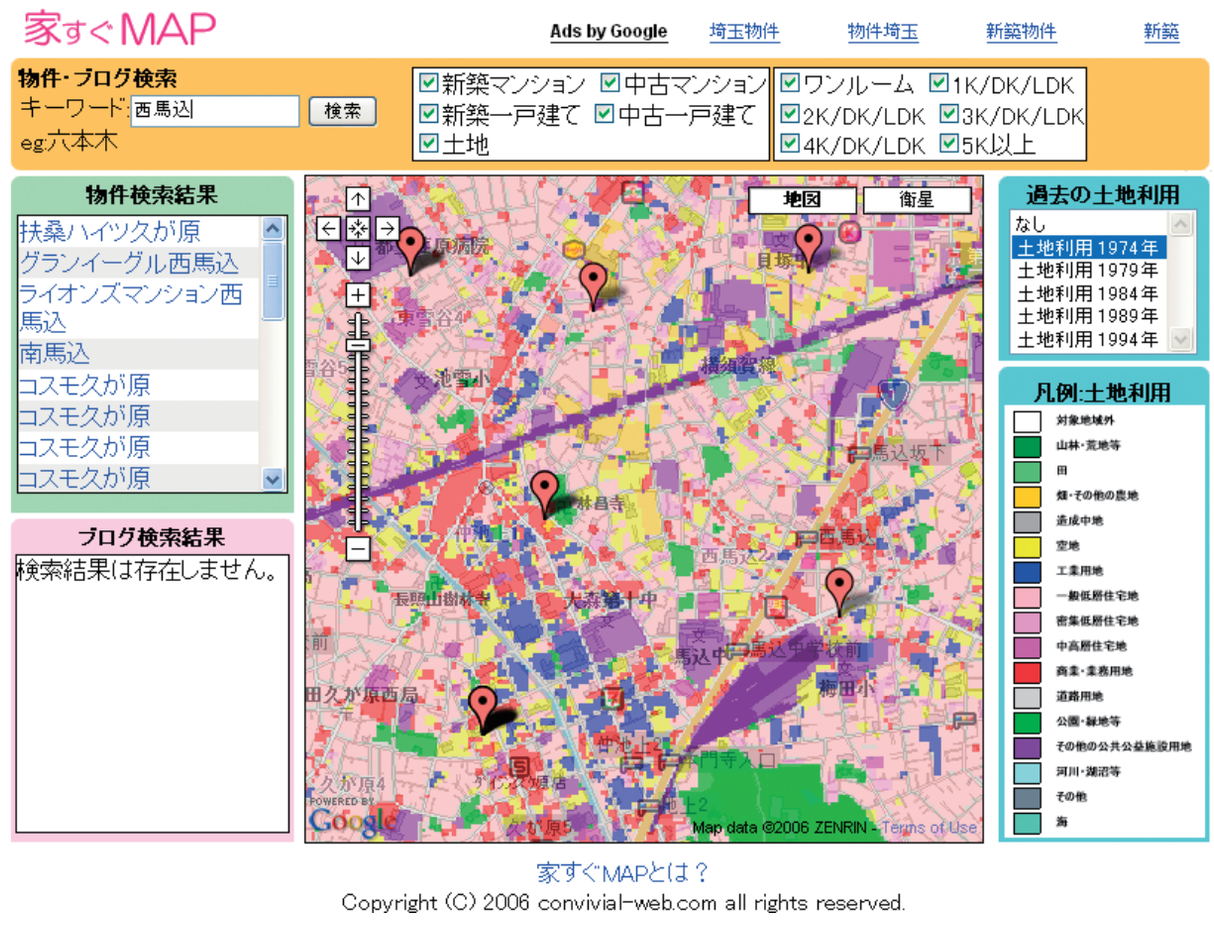

図-10　マッシュアップサイト例（家すぐ MAP）

■今後の Web 発展を担うアーキテクチャ

本稿では，Web2.0 を支える具体的な技術として RSS や Atom，SOAP/WSDL および REST を利用した Web API，また Ajax を中心とした JavaScript 技術の発展を解説した．また，軽量プログラミングモデルに伴う開発手法やテストの意味の変化と，マッシュアップ技術の拡大に触れた．

ネットの「あちら側」の企業では，スピーディで低コストなアプリケーション開発・運用が常識化している．さまざまな基礎技術を取り入れ，また新たに積み重ねることで現在の Web2.0 が成り立ってきたように，そこで得られた知見や枠組みをネットの「こちら側」に取り入れることもできるだろう．それらは対岸の話でも机上の空論でもない．世界中で何億人という利用者が実際に使っている生の技術・サービスの中から，キラリと光るものを見つけ出すアンテナを常に張っておきたい．

参考文献

1）Garrett, J. J. : Ajax : A New Approach to Web Applications (2005). http://www.adaptivepath.com/publications/essays/archives/000385.php
2）The Application/json Media Type for JavaScript Object Notation (JSON) (2006). http://www.ietf.org/rfc/rfc4627.txt
3）鈴木雄介：Web2.0 から学ぶ SOA の本質，SOA フォーラム 2006 (2006). http://www.arclamp.jp/blog/archives/000845.html

（平成 18 年 10 月 2 日受付）

川崎　有亮
u-suke@kawa.net

1977 年東京生まれ．千葉大学大学院修士課程修了．（株）かっぺ取締役副社長を経て，現在は（株）リクルートに所属．著書「Ajax／実装のための基礎テクニック」（技術評論社，2006 年，共著）．Web コミュニケーションを活動テーマとする．
http://www.kawa.net/