最新サーバサイド技術動向 入門の入門

代表的なサーバサイド技術であるJava EE、PHP、ASP.NETを軸に、昨今、取り上げられることの多いキーワードを紹介。
アプリケーションフレームワーク、Ajax、Web APIなど、乱立する最新技術についての基本知識を整理する。

山田祥寛 (YAMADA, Yoshihiro)
yoshihiro@wings.msn.to
http://www.wings.msn.to/

WINGS More Progress, More Future

新たなアプリケーション開発の潮流 Web 2.0（1）

そもそもWeb 2.0とは…？

- 技術や仕様のバージョンではない　Cf. Tim O'Reilly氏「What is Web 2.0」
  → Web活用のための次世代のコンセプト／方向性（技術／ユーザ／マーケティング）
- Web 2.0と比較して、従来型のコンセプトを…
  Web 1.0：静的なHTML。更新頻度も稀
  Web 1.5：CMS等を利用した動的なページ。サイト内で完結したサービスを提供
- プラットフォームとしてのWeb
  → アプリケーションでの利用を目的としたデータ配布（シンジケーション）
  自分の使いやすい形にデータを再加工（リミックス）
  → Ex. Web API／RSS
- ユーザの参加によってデータの価値が増す
  → Ex. SNS／Amazon Review／WikiPedia／はてなソーシャルブックマーク等
- 永遠のベータ版
  → 日常的なフィードバックとバージョンアップ。リリースが終わり、ではない

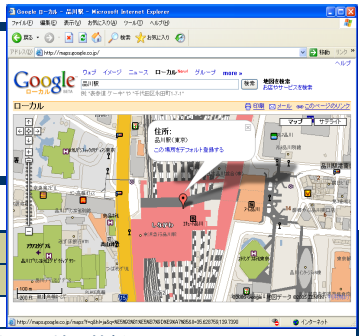
新たなアプリケーション開発の潮流 Web 2.0（2）

- **Web2.0アプリケーションの特徴**
  - 短サイクルでの開発／バージョンアップが必要（数週間、場合によっては数時間単位）
  - 軽量プログラミングモデルが求められる
    → PHP、RubyなどのLight Weight言語の普及
    → EoD（Ease of Development：開発の容易性）がますます重要な要素に
    → Visual Studio 2005との高い親和性と生産性が特徴の「.NET」技術が伸張
  - 「WebアプリといえばJava EE」という状況に変化の兆し
  - Webサービス（Amazon Webサービスなど）の積極的な活用
    → ビジネス価値の本質を担う部分への注力
  - リッチなインターフェイスの追求とAjax技術の浸透（いかに魅せるか、参加を促す仕組み）
    → 互換性などの問題から敬遠されてきたJavaScriptが復権の兆し
- **各技術の特徴を理解し、適切な技術を取捨選択できる知識が求められる**
  - ますます広範囲になってきたWebアプリ開発の世界
    → 単一の技術だけで賄うことは困難
  - 本セッションで扱う内容… Ajax、Web API、アプリケーションフレームワーク

Theme 1）リッチなUIを標準技術だけで実現 Ajax（Asynchronous JavaScript And Xml）

- **Ajax（Asynchronous JavaScript And Xml）**
  - JavaScriptでサーバとの非同期（Asynchronous）な通信を行い、その処理結果に基づいて、ページ上の必要なコンテンツだけを動的に更新する手法

Ajax技術の導入例／効果



Ajax技術の導入例

サイト名	URL
概要	
Google Maps	http://maps.google.co.jp/
マウスによって表示位置を移動したり、拡大／縮小率を変更できる地図情報サービス	
Windows Live	http://www.live.com/
RSS/Ajaxをフル活用したパーソナライズ可能なポータルページ（ニュースや天気、検索機能などを提供）	
Amazon.com Diamond Search	http://www.amazon.com/gp/search/finder/?productGroupID=loose%5fdiamonds
ダイヤモンドの商品検索システム。形状やカラット数、価格などの条件に応じて商品を動的に表示	

Ajax技術の導入効果

- ページ内の必要な箇所のみをリフレッシュ（チラツキの改善）
  - → サーバ処理の間もクライアント側では操作を継続できる（作業を中断されない）
  - → クライアント／サーバ間のトラフィック量を抑制
- 新たな機能の可能性（メールの自動保存のような定期的なサーバとの通信処理など）
- 標準的なJavaScript（DHTML）やXMLの知識のみで開発が可能

Ajax導入における問題点

Ajax技術の問題点

- クロスブラウザ問題（ブラウザ間でのJavaScriptの挙動の違い）

```
if(window.XMLHttpRequest){    // Internet Explorer以外のブラウザ
  req = new XMLHttpRequest();
}else if(window.ActiveXObject) { // Internet Explorer
  try{
    req = new ActiveXObject("Msxml2.XMLHTTP"); // 6.0
  }catch(e){
    try{
      req = new ActiveXObject("Microsoft.XMLHTTP"); // 5.x以前
    }catch(e){
      return null;
    }}}
```

- JavaScript対応のデファクトスタンダードな統合開発環境は不在（開発生産性に疑問）
- XML DOMによるデータ処理はコードが冗長になりがち
- セキュリティ上の課題（よりクライアントサイドに裁量が委ねられている）

→ そこでAjax対応ライブラリ

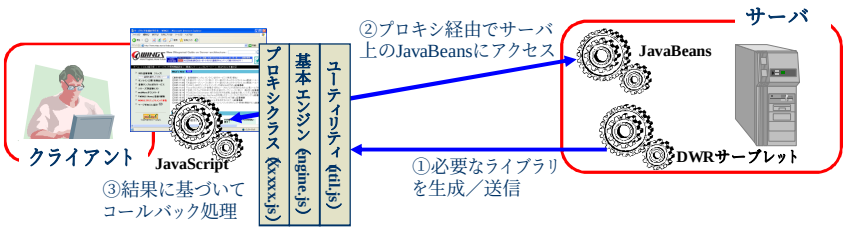
Ajax 対応ライブラリの分類

- **機能特化ライブラリ型**
  - オートコンプリートやタブペインなどの特定機能をAjaxで実現するためのライブラリ
  - Ex. Yahoo! UI Library、AjaxTags、AjaxFaces、ASP.NET AJAX Extensionsなど
- **汎用ライブラリ型**
  - Ajax開発を支援する、より汎用的なライブラリ／フレームワークを提供（Ajaxを主目的においたライブラリ）
  - XMLHttpRequestオブジェクト（ブラウザ依存）のラッピングやデータ交換の支援、テンプレートエンジン、更新時のエフェクト効果提供など
  - Ex. Ajax.NET、AJASON、PEAR::HTML_Ajax、DWR、JKL.Hina+JKL.ParseXML、Prototype.jsなど
- **汎用フレームワーク型**
  - Ajax開発に対応した汎用フレームワーク（Ajaxは一機能にすぎない）
  - Ex. Ruby On Rails、JSF（JavaServer Faces）、ASP.NET（次期バージョン）

DWR（Direct Web Remoting）

http://getahead.ltd.uk/dwr/ - Java環境 -

- **サーバ側のJavaBeansをクライアントサイドから呼び出すための簡易RPC**
  - サーブレット環境さえあれば動作可能（特定のフレームワークに依存しない）
  - DWRサーブレットがクライアント／サーバ通信に必要な処理を制御



  - バッチ処理、エラーハンドル、構造データの変換機能などを提供
  - サーバメソッドの単体テスト機能（ブラウザ上からサーバメソッドの動作を確認）
  - HTML操作のための諸ユーティリティを提供
    → 選択ボックスやテーブルの編集、通信メッセージのカスタマイズ等

例：DWRによる”Hello, World”アプリケーション

名前: 山田祥寛 送信
こんにちは、山田祥寛さん!

- DWR動作に必要なコード
  - JavaBeans（サーバメソッド）
  - DWR設定ファイル
  - クライアントページ（「.html」ファイル）

クライアントサイドでの呼び出し

```
// サーバメソッドの呼び出し
function submit_onclick (){
 Hello.showMessage($('name').value
   ,setResult);
}
// コールバック関数
function setResult(result) {
 DWRUtil.setValue("labelResult",result);
}
```

サーバサイドで動作するJavaBeans

```
package to.msn.wings.ajax.dwr;

public class Hello {
   public String showMessage(String name){
    return "こんにちは、" + encode(name)
              + "さん!";
   }
}
```

DWR設定ファイル（JavaBeansの登録）

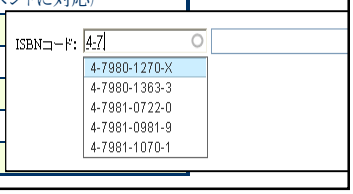
```
<dwr>
  <allow>
    <create creator="new" javascript="Hello"
       class="to.msn.wings.ajax.Hello" />
  </allow>
</dwr>
```

AjaxTags タグライブラリ

http://ajaxtags.sourceforge.net/ - Java環境 -

- 定型的なAjax機能を限りなくコーディングレスで実現
  - JSP（JavaServer Pages）ページ上で利用可能なAjax対応タグライブラリ集

タグ名	概要
Autocomplete	ユーザの入力値に応じてマッチした値リストをポップアップ（Google Suggestライクなオートコンプリート機能）
Callout	任意のHTML要素をクリック／マウスポイントしたタイミングでバルーンメッセージを表示
DisplayTag	ソート／ページングが可能なグリッド表を生成
Area/Anchor	ページ内のコンテンツを部分的に更新
HTML Content	ページ内のコンテンツを部分的に更新（任意のイベントに対応）
Portlet	ポートレット風のコンテンツを生成
Select	階層式のドロップダウンメニューを生成
TabPanel	タブパネル式のページを生成
Toggle	On／Offの状態を管理可能なトグルボタン
UpdateField	サーバサイドから取得した情報をフォームに反映

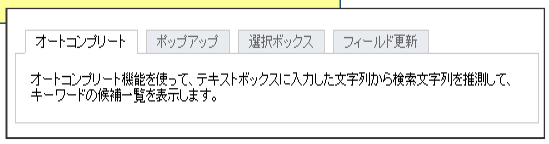


例：AjaxTagsによるタブパネルの生成

- **AjaxTags動作に必要なコード**
  - タブパネルを制御する「.jsp」ファイル
  - 個々のタブコンテンツを表す「.jsp」ファイル

タブパネルを定義

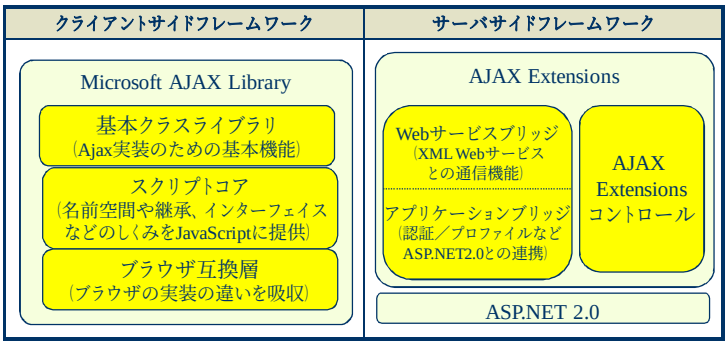
```
<ajax:tabPanel panelStyleId="tabPanel“ …>
   <ajax:tab caption="オートコンプリート" baseUrl="p1.jsp"
      defaultTab="true" />
   <ajax:tab caption="ポップアップ" baseUrl="p2.jsp" />
   <ajax:tab caption="選択ボックス" baseUrl="p3.jsp" />
   <ajax:tab caption="フィールド更新" baseUrl="p4.jsp" />
</ajax:tabPanel>
```



ASP.NET AJAX（開発コード“Atlas” Framework（1））

http://ajax.asp.net/ - ASP.NET環境 -

- **ASP.NET 2.0ベースで動作する高機能なAjax対応フレームワーク**
  - Microsoft ASP.NET 2.0 AJAX Extensions（サーバサイドフレームワーク）、Microsoft AJAX Library（クライアントフレームワーク）から構成

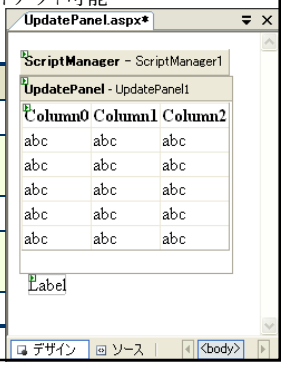


  - クライアント機能だけを用いることでPHP／Java EEなどの環境での動作も可能

ASP.NET AJAX（開発コード“Atlas” Framework（2））

http://ajax.asp.net/ - ASP.NET環境 -

- **統合開発環境Visual Studio 2005とも高度に連携可能**
  - 「ASP.NET AJAX-Enabled Web Site」テンプレートを提供
    → 必要なアプリケーション構成の設定をデフォルトで提供
  - AJAX Extensionsコントロールをフォームデザイナ上でレイアウト可能
  - 次期ASP.NETには、標準で統合の予定（?）



コントロール名	概要
ScriptManager	Ajax動作に必要なJavaScriptのコードを管理
ScriptManagerProxy	Ajax動作に必要なJavaScriptのコードを管理（マスタページ対応）
UpdatePanel	部分的に更新可能な領域を定義
UpdateProgress	非同期通信中に状態 メッセージ／画像を表示
Timer	一定時間おきに特定の処理を実行

▲ASP.NET AJAXで利用可能なサーバコントロール

ASP.NET AJAX（開発コード“Atlas” Framework（3））

http://ajax.asp.net/ajaxtoolkit/ - ASP.NET環境 -

- **ASP.NET AJAX Control Toolkit**
  - ASP.NET AJAXを基盤に構築されたサーバコントロール集
    → 2006年11月現在で約30のAjax対応サーバコントロールを提供
  - サイト上で動作デモ環境も提供

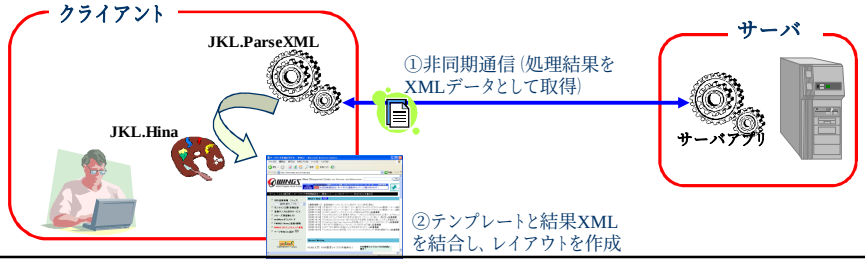
コントロール名	概要
Accordion	開閉可能なマルチペインを生成
Animation	ページ内要素にアニメーション効果を定義
CascadingDropDown	階層式のドロップダウンメニューを生成
DragPanel	ドラッグ可能なパネル領域を定義
DynamicPopulate	XML Webサービス経由で取得した結果をページに反映
HoverMenu	マウスホバー時にポップアップ可能なメニューを生成
ModalPopup	モーダルなポップアップウィンドウを生成
Slider	数値入力のためのリッチなスライダーを生成

▲ASP.NET AJAX Control Toolkitで利用可能なサーバコントロール

JKL.Hina＋JKL.ParseXMLライブラリ

http://www.kawa.net/ - JavaScript環境 -

- **JavaScriptベースの簡易なAjax対応ライブラリ**
  - 川崎有亮氏が開発（日本語ドキュメントも充実）
  - JavaScriptベースのライブラリでサーバサイド環境に依存しない
  - 非同期通信からXMLデータの解析、テンプレート処理まで対応
    - XMLHttpRequestオブジェクトによる非同期通信を隠蔽
    - サーバから送信されたXMLデータにも連想配列の形式でアクセス可能
    - レイアウト定義とロジックとを分離
      → JavaScriptテンプレートとしてAjaxPagesやJavaScriptTemplaesなども



その他のAjax対応ライブラリ／ツール

- **JavaScriptベースのツールも増えつつある**
  - ライブラリ導入を使っても、やはりJavaScriptを完全にブラックボックス化するのは困難
  - ロギング（クライアント側の動作を確認）
    Log4JavaScript（Log4Jの移植版：http://log4js.sourceforge.net/ ）
  - テスティングフレームワーク
    総合テスト（シミュレート）：Selenium（ http://www.openqa.org/selenium/ ）
    サーバ側サービス：Jakarta HTTPClient（http://jakarta.apache.org/commons/httpclient/）
    クライアント側ページ：JsUnit（Junitの移植版：http://www.jsunit.net/ ）
    ※DWRやASP.NETのように標準で単体テスト機能を持つライブラリも
  - デバッガ
    Venkman JavaScript Debugger
    （FireFoxの機能拡張／導入が容易（ブレイクポイントや変数チェックなど対応）
    （ http://www.mozilla.org/projects/venkman/ ）
    Internet Explorer Developer Toolbar（動的に生成されたDOMを確認）
    （ http://www.microsoft.com/downloads/details.aspx?familyid=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en ）

現実的なAjax技術活用のために

- **Ajaxは今後もますます普及の見込み**
  - 汎用型フレームワークが次々とAjaxに対応
    → アプリケーションの一機能としてAjaxを活用する環境が整いつつある
  - 「Ajax対応」と謳うこと自体が、いずれ無意味に
- **Ajaxは入力／操作支援の目的で部分的に採用するのが効果的**
  - Ajaxはアプリケーション開発の一手法にすぎない
  - 本当にAjaxが必要なのか？
    → Google Mapsのような例は、特別なケースであると考えるべき
  - 開発生産性／パフォーマンスの理由から、Ajax技術でアプリケーション全体を構築するのは非効率
  - やりたいことはなにか？オートコンプリートやツリーメニューの動的読み込み？

参考資料

- **Ajax技術学習のための書籍／記事**
  - 「サーバサイドAjax入門
    – Java／PHP／ASP.NET連携でAjaxプログラミングを極める！ –」
    （翔泳社：ISBN 4-7981-1206-2）

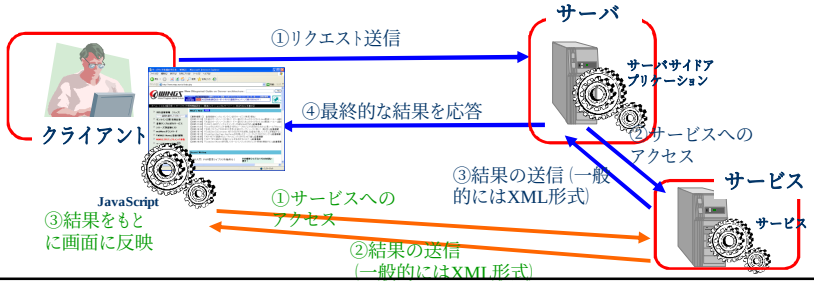
  - 連載：マイクロソフトのAjax対応フレームワーク「Atlas」入門
    http://itpro.nikkeibp.co.jp/members/bn/mokuji.jsp?OFFSET=0&MAXCNT=20&TOP_ID=251329
  - 連載：今からでも遅くないAjax 基本のキ
    http://itpro.nikkeibp.co.jp/article/COLUMN/20060529/239205/

Theme 2）これからはあるものを使う時代
本格化する Web API 活用

- **従来のWebアプリケーション開発といえば…**
  - サーバサイド技術（ASP.NETやJava EE、PHP）やクライアントサイド技術によって一から構築
    （長所）自由度／柔軟性のあるアプリケーションを構築できる
    （短所）サイト規模によっては膨大な開発工数と時間を要する
  - ライブラリや出来合いのアプリケーション（掲示板やCMS、Blogなど）のような既存のソフトウェアを導入して構築
    （長所）定型的な機能を一から作成する必要がない（工数の削減）
    （多くの場合）レイアウトのカスタマイズが容易に可能である
    （短所）自サーバ上にインストールや環境構築が必要
    データは基本的にサイト単位で管理
- **Web APIを利用することで…**
  - 他のサイトで提供しているサービスを利用したアプリケーションを構築
    → Ex. たとえばグルメサイトでGoogle Maps上に指定地域の食事処が表示されるなど
  - では、既存のライブラリやアプリケーションとなにが違うのか？

What is Web API？
～従来のAPI／アプリとなにが違うのか？～

- **What is Web API？**
  - 従来のWebアプリケーションは人がアクセスするもの
    → Web APIはコンピュータ（アプリケーション）がアクセスするもの
  - 従来のAPIはOSの機能にアクセスするためのもの
    → Web APIはWeb上のサービスにアクセスするためのもの（Webがプラットフォーム）
  - 従来のライブラリが提供するのは、データを操作するための「道具」のみ
    → Web APIが 提供するのは「道具＋（ネット上の膨大な）**データ**」



Web APIの分類 - SOAP (1) -

- **SOAP型Webサービスのポイントは「拡張性」にあり**
  - クラシカルなWebサービスの実現手段
  - 下位プロトコルに依存しない拡張性（HTTP、FTP、SMTPなどで利用可）と高機能性
  - 各プラットフォームでSOAP利用のためのライブラリも充実
    → WSDL（サービスの仕様書）からクラスを自動生成
    アプリケーションからはローカルAPIにアクセスする要領でサービスを利用

開発環境	開発ライブラリ
.NET Framework	標準の「.asmx」ファイル
Java	Apache Axis（http://ws.apache.org/axis/）
PHP	SOAP関数（PHP5以降） NuSoap（http://sourceforge.net/projects/nusoap/）

  - プロキシクラスを介する分、パフォーマンスは低い場合も
  - Ex. Google Web APIs、Amazon Webサービスなど

Web APIの分類- SOAP (2) -

- **SOAPメッセージの構造**

```
POST /WebSite2/WebService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    ...ヘッダ情報...
  </saop:Header>
  <soap:Body>
    <HelloWorld xmlns="http://www.wings.msn.to/">
      <str>string</str>
    </HelloWorld>
  </soap:Body>
</soap:Envelope>
```

プロトコル バインディングヘッダ
（下位プロトコルに依存するヘッダ情報）

SOAP Envelope
（SOAPメッセージ全体）

SOAP ヘッダ（任意）
（認証情報など付随データ）

SOAP 本体
（サービス名やパラメータ情報）

Web APIの分類 - REST -

- **REST（REpresentational State Transfer）型Webサービスは「手軽さ」がウリ**
  - SOAPの高機能性は必要ない、もっと手軽に！（マルチプロトコルは不要）
  - クエリ情報経由でパラメータを送信、XMLで結果を取得（プロキシクラスは不要）
    シンプルでパフォーマンスにも優れる

    http://api.search.yahoo.co.jp/WebSearchService/V1/webSearch?
    appid=xxxxx&query=山田祥寛
  - 一般的なXMLパーサさえあれば、REST APIは利用できる
    （専用のRESTライブラリなどは不要）
  - （厳密な議論はあるが…）SOAPを利用しないHTTPのみを使ってXML文書を送信するしくみ → **現時点では**、SOAPよりも人気があるアーキテクチャスタイル
  - あくまでシンプルな用途が前提（高度な操作には未対応）
    → トランザクションやセキュリティ制御などは不可
    Ex. SOAPでは、WS-SecurityやWS-AtomicTransaction、
    WS-ReliableMessagingなどWS-*に対応
  - Ex. Amazon Webサービス、Yahoo! Japan Webサービスなど

Web APIの分類 - REST（2）-

- **RESTサービスの例**

  http://api.search.yahoo.co.jp/WebSearchService/V1/webSearch?
  appid=xxxxx&query=山田祥寛

```
<?xml version="1.0" encoding="UTF-8" ?>
<ResultSet …  totalResultsReturned="10" firstResultPosition="1">
 <Result>
   <Title>サーバサイド技術の学び舎 - WINGS</Title>
   <Summary>...</Summary>
   <Url>http://www.wings.msn.to/</Url>
   <ClickUrl>http://wrs.search.yahoo.co.jp/...</ClickUrl>
...中略...
</ResultSet>
```

Web APIの分類 - その他 -

- **JavaScriptライブラリ型**
  - JavaScript上で利用可能なライブラリ
  - クライアントサイド技術だけを理解していれば、すぐに使える手軽さが特徴
  - Ex. Google Maps API、Google Homepage APIなど
- **RSS（Rich Site Summary）フィード型**
  - サイトの更新／新着情報を配信するための標準XML形式（特定ベンダに非依存）
  - サイトが配信しているRSSフィードは、専用のRSSアグリゲータ（リーダ）、または RSS対応ライブラリ、XMLパーサなどを利用することで自由に利用できる
  - スパムメールの蔓延がRSSフィード普及を後押し
  - 代表的なニュースサイトやブログツールなどもRSS配信に対応
  - Windows Vista、Internet Explorer 7.0でRSSに標準対応
  - バージョンによる仕様差が大きい
    → Atom（コンテンツの配信や保存／編集に対応）を模索

参考）Web APIの挙動（比較）

参考）利用可能な主なWeb API

- アプリケーション開発に役立つWeb API

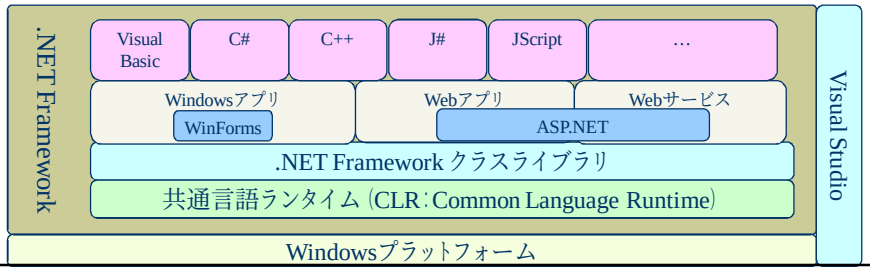
サービス名	URL	主な検索対象
Amazon Webサービス	http://www.amazon.co.jp/exec/obidos/tg/feature/-/451209/	Amazon商品情報
Yahoo! Webサービス	http://developer.yahoo.co.jp/	Webページ／画像／オークション情報
はてな Webサービス	http://www.hatena.ne.jp/info/webservices	ダイアリー／アンテナ／ブックマーク
Google Web API	http://code.google.com/	Webページ／Google Mapの操作等
辞書Webサービス	http://www.btonic.com/ws/	国語／英和／和英辞書
RSSナビ REST API	http://www.rssnavi.jp/rsetapi.html	複数RSSフィードの串刺し検索
Simple API	http://simpleapi.net/	指定サイトのサムネイル画像を生成
Weather Hacks	http://weather.livedoor.com/weather_hacks	直近の天気予報情報
Geocoding API	http://www.geocoding.jp/api/	住所→緯度／経度情報
hon.jp Webサービス	http://hon.jp/doc/about_rest.html	電子書籍情報

Theme 3）アプリケーション開発には欠かせない アプリケーションフレームワーク

- アプリケーションフレームワークとは？
  - アプリケーション設計のために考案された枠組み／ルール／思想
  - 設計を実装するために利用可能なテンプレート／ライブラリ／ツール／ドキュメント等
- 今やアプリケーションフレームワークの導入は当たり前
  - ユーザ要件の高度化／ミッションクリティカルなシステムへの適用
  - ビジネスニーズの流動化と多様化
  - 多様なフロントエンド（HTML、リッチクライアント、携帯端末 等）
- アプリケーションフレームワークの導入効果
  - 定型的な記述をフレームワークにゆだねられる
    → 設計ノウハウの再利用、品質の均質化（人依存からの脱却）
  - アプリケーション固有のビジネスロジックに専念できる
  - ビジネスロジックの再利用が容易に
  - 脆弱性（SQLインジェクションやXSS）の防止、標準準拠のための諸機能も

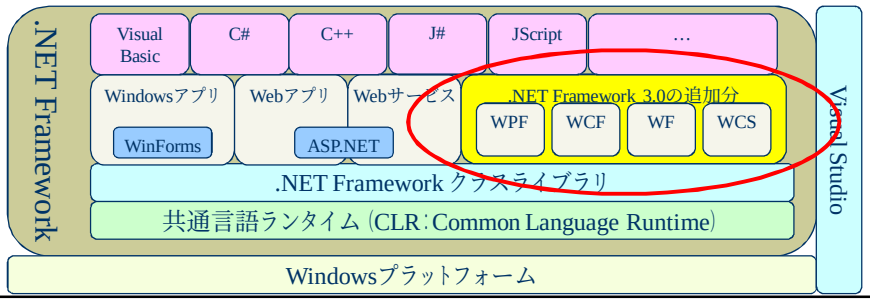
.NET Framework 3.0（1）

- **.NET標準のオールインワンな開発プラットフォーム**
  - メモリ管理／セキュリティの制御は.NET Frameworkが管理
  - すべてのアプリケーションはCLR上で動作 → 言語非依存（得意な言語を選択可能）
  - 文字列操作のような基本的な操作からデータベース操作（ADO.NET）まで
  - ツリービュー、グリッド表、ログインページ、Webパーツのような高度なコントロール群
  - 無償IDE（Visual Studio Express Edition）で導入も手軽



.NET Framework 3.0 （2）

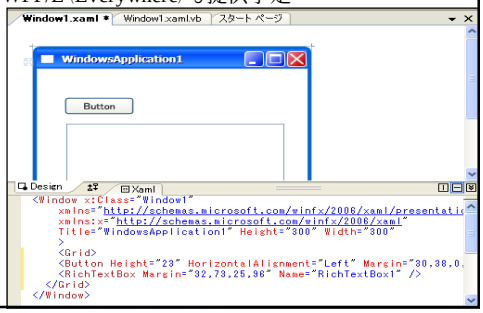
- **.NET Framework 3.0＝.NET Framework 2.0 ＋ Wxx基盤**
  - Windows XP SP 2／Windows Server 2003 SP1／（Windows Vista）対応
  - .NET Framework 3.0再配布パッケージ（ランゲージパックも同梱）
  - Visual Studio 2005 extensions for .NET Framework 3.0 (WF対応)
  - Visual Studio 2005 extensions for .NET Framework 3.0 (WCF & WPF), November 2006 CTP（今後、更新の予定なし → 正式版は次期Orcasまで保留）



.NET Framework 3.0

- プレゼンテーション技術Windows Presentation Foundation -

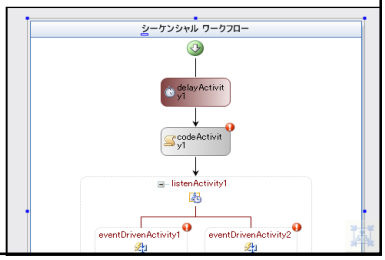
- **リッチなユーザインターフェイスを共通したモデルで実現可能**
  - 2D／3Dグラフィック、静止画像／動画、ドキュメント等のコンテンツをリッチに表現
    → Direct3D、GPU (Graphic Processing Unit) によるレンダリング
  - Windows/Webアプリに関わらず、共通したプログラミングモデル
    - セルフホスト型アプリ (フル機能を利用可能)
    - Webブラウザホスト型アプリ (セキュリティサンドボックスによる一部機能の制限)
    → 非Windows環境での動作にはWPF/E (Everywhere) も提供予定
  - マークアップ言語XAML
    (eXtensible Application Markup Language)
    - デザイナ向けツール
    Expression Interactive Designer
    を提供予定
    → Windows／Webアプリ双方で
    プログラマとデザイナとの分業が
    可能に



.NET Framework 3.0

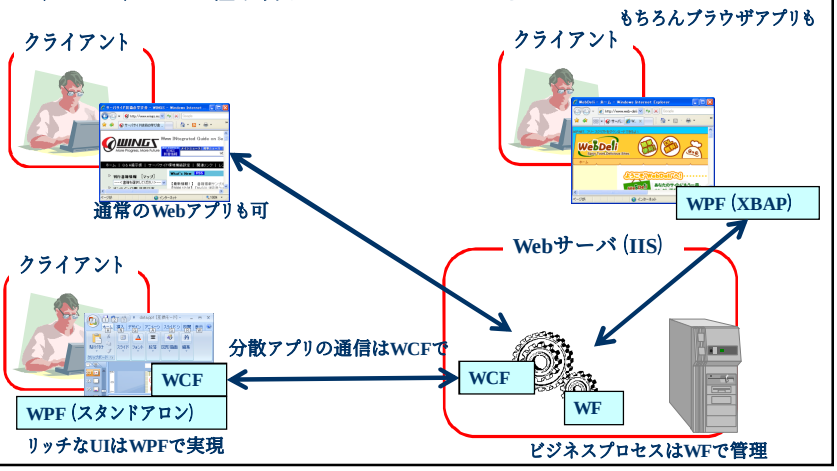
- Windows Communication Foundation & Windows Workflow Foundation -

- **WCF：SOA (Service Oriented Architecture) のためのコミュニケーション基盤**
  - 従来：ASP.NET Webサービス、.NET Remoting、Enterprise Services、WSE (Web Services Enhancements) など、数ある分散アプリケーション技術でプログラミングモデルはすべて異なる → 目的に応じて使い分けが必要 (学習コスト、変更時の影響大)
  - WCFが共通のプログラミングモデルを提供 (移行工数を削減)
    → プロトコルバインディングやエンコーディング方法、セキュリティ設定などの
    変更が発生した場合にも設定ファイルでの適用が可能
- **WF：システムフローとビジネスロジックを分離する標準ワークフロー技術**
  - ビジネスプロセスとビジネスロジックの分離
    → 保守／拡張性の改善
  - Office、SharePoint、BizTalk Serverなどでも
    共通的なワークフロー基盤として採用予定
  - シーケンシャルワークフロー
    分岐/繰り返しによる構造的なフロー制御
    ステートマシンワークフロー
    イベント駆動的なワークフロー



.NET Framework 3.0の適用事例

- WPF／WCF／WFの組み合わせでこんなことができる

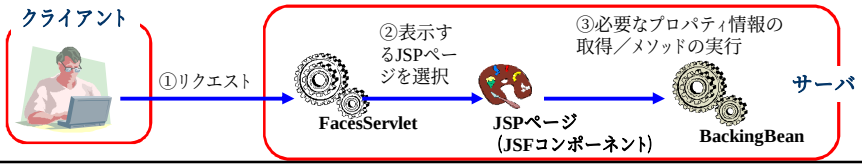


Java EE（Enterprise Edition）5.0（1）

- EoD (Ease of Development) を追求したJava EEの新バージョン
  - 従来の「難しい」Java EE開発への解決策として
    Hibernate、Springなど、多くのオープンソース製品が登場
    → Java EE 5はこれらオープンソースにおける成果を標準としてまとめたバージョン
  - CMP Entity Beanが担当していたO/Rマッピング機能をJPA (Java Persistent API) として独立
    → Hibernateに近いモデルを導入 (創始者Gavin King氏も策定に参加)
    EJBから独立したことでJava SEでもO/Rマッピング機能を利用可能
  - JAX-WS 2.0 (従来のJAX-RPC 1.1) の導入でWebサービス開発が簡素化
    → 標準的なデータバインディングAPIとしてJAXB 2.0
- 普通のJavaクラスで記述できるようになったEJB
  - 従来のEJB：Home／Componentインターフェイスなど多くの規約 (開発の難しさ)
    EJBコンテナ依存 (単体テストの難しさ) → EJBによる開発は敬遠
  - EJB 3.0では…
    ・POJO (Plain Old Java Object) ＋アノテーション (@Session/@Stateless等) による開発
    ・シンプル、かつ、単体動作にEJBコンテナが不要 (単体テストも容易)

Java EE（Enterprise Edition）5.0（2）

- **J2EE標準に組み込まれたF/W「JSF（JavaServer Faces）」**
  - Visual Basicライクなイベントドリブン型のアプリケーション開発
  - 画面の構築はUIコンポーネントで行う
    → ブラウザごとの挙動の違いやJavaScriptの実装などを隠蔽
    直感的なユーザインターフェイスの構築が可能
  - レンダラを切り替えることでHTML以外の出力も容易（リッチクライアント対応）
  - Sun Java Studio Creator、JDeveloper、JBuilderなどの開発ツールも充実しつつある
    → 画面デザイン、遷移、コンポーネント設定などをGUIベースで開発可能
  - ただし、現時点で採用実績がまだ多くなく、Strutsとの住み分けもこれからの課題
  - 参考：JDeveloperで学ぶJSF入門（http://www.thinkit.co.jp/category/dev_bn.html#44）



Ruby On Rails

http://www.rubyonrails.org/

- **多くのフレームワークに影響を与えたRubyベースのフレームワーク**
  - Ruby製のオープンソースフレームワーク（David Heinemeier Hansson氏中心に開発）
  - Convention over Configuration（設定よりも規約）
    → 規約（モデルは単数形、テーブル名は複数形など）に従うことでコード量を削減
  - DRY（Don't Repeat Youeself）：変更時の影響箇所を最低限に
  - コードジェネレータ：アプリケーションの骨格からテストコードまでを自動生成
    → scaffold（足場）：データベースのCRUD操作のためのコード生成も可能
  - フルスタック：開発サーバからテスティングフレームワーク、Ajax／Webサービス対応など、アプリケーション構築に必要な機能をオールインワンで提供
    → 組み合わせによる検証負荷を軽減
  - After Rails製品も多く登場
    → MonoRails（.NET）、CakePHP（PHP）、Trails（Java）、Catalyst（Perl）

PHP（PHP：Hypertext Preprocessor）

- **軽量言語の代表PHPの世界でもフレームワーク活用に弾み**
  - PHPで利用可能なフレームワーク製品は以下の通り

フレームワーク	概要
Symfony（http://www.symfony-project.com/）	RoRの影響を色濃く受けた製品
Zend Framework（http://framework.zend.com/）	Zend社から提供されているF/W
Ethna (http://ethna.jp/)	GREEでも利用されている代表的な国産F/W製品
Maple（http://kunit.jp/maple/）	国産F/Wの代表
Mojavi（http://www.mojavi.org/）	PHP F/Wの草分け的な存在

  - 現時点ではデファクトスタンダードは不在の状況
    → 本家ZendのZ/Fに期待
    国内ではEthna／Mapleが強勢？
    Mojaviの流れを汲み、RoRの影響を色濃く受けたsymfonyにも期待
  - 参考：symfony入門（1）：symfonyで始めるPHPフレームワーク
    http://codezine.jp/a/article/aid/704.aspx

まとめ

- **Web2.0時代のアプリケーション開発**
  - ますます広範囲になってきたWebアプリ開発の世界
  - 単一の技術だけですべてをサポートすることは困難
- **それぞれの技術を「相対的」に概観できる視点が重要**
  - 複数技術の共通点を理解すること
  - 複数技術の相違点を理解すること
  - 適材適所で最適な技術を使い分けられる能力を！

ご静聴ありがとうございました

- 「サーバサイド技術の学び舎 - WINGS」
  - http://www.wings.msn.to/
  - サーバサイド技術に関する記事や書籍、関連サイトなど最新の情報を日々紹介
  - RSSフィードも配信中

サーバサイド技術の学び舎 - WINGS