リッチクライアントから見る Webアプリケーション開発の将来

HTMLクライアントに代表されるシン・クライアントの限界がささやかれる昨今、リッチ・クライアントへの期待が高まっている。Macromedia Flash（Flex）、ClickOnce、JWS、Curl、PDFフォーム、VSTO、InfoPathなど主要なリッチクライアント技術を紹介すると共に、リッチクライアント普及にあたっての問題点、課題を紹介する。

山田祥寛 (YAMADA, Yoshihiro)
CQW15204@nifty.com
http://www.wings.msn.to/

リッチ・クライアントの定義（1）

- **1990年代の「リッチ・クライアント（Rich Client）」**
  - 登場当初（1990年代半ば）
    - ✓ ダム端末やネットワーク・コンピュータなどのシン（Thin）・クライアントに対応する言葉
    - ✓ クライアント/サーバ（C/S）・システムやWindowsアプリケーションなど、PCの豊富なクライアント・リソースを利用したシステム（=ファット・クライアント）
    - ✓ デスクトップPCの環境そのものを指す場合も
  - Webコンテンツにおけるプラグイン技術
    - ✓ 動画や音楽、3Dなどのリッチ・コンテンツを処理するソフトウェア
    - ✓ Flash PlayerやShockWave、Real Playerなど

→ 歴史的に形態や利用局面は異なり、その意味は時として「多義的」。要は、UIの操作性、表現力、機能性に富んでいる（Richな）クライアント・マシン、または、ソフトウェアの総称

「リッチ・クライアント」の定義（2）

- 現在よく言われるところの「リッチ・クライアント」
  - ≠ファット・クライアント
  - 標準のWebブラウザに較べて、高いUI（操作性・機能性）を実現する
  - サーバとの通信、連携機能を備えている
  - クライアント・アプリケーションの配布、バージョン管理を簡素化するための なんらかの機能を備えている
  - クライアント内で一連の完結した処理、制御を実現可能である

  → 本セッションで紹介する「リッチ・クライアント」

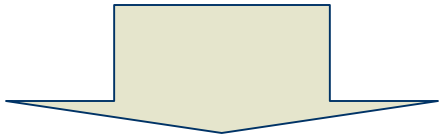
リッチ・クライアントの必要性（1）

- ファット・クライアント（C/Sアプリ）とシン・クライアント（Webアプリ）

	メリット	デメリット
ファット・クライアント	・クライアントの資源を利用した高度な処理が可能 ・ドラッグ＆ドロップなど、直感的なUIを利用可能	・プラットフォームに依存 ・処理パフォーマンスが、クライアントの性能に依存 ・クライアントごとにインストール、環境設定等が必要（配布が困難）
シン・クライアント	・（基本的に）プラットフォームに非依存 ・軽量なので、クライアントの性能に依存しにくい ・ソフトのインストール、環境設定が不要（配布が容易）	・貧弱なUI（操作性、表現力×） ・ブラウザの種類、バージョンによって、表示レイアウトが異なる ・サーバサイドに処理負荷が集中 ・サーバ・ラウンドトリップが常に発生するため、レスポンスが悪い ・データの再利用、アプリ連携が困難 ・作業は常にオンラインで ・紙帳票とのレイアウトの不整合

リッチ・クライアントの必要性（2）

- **周辺環境からの要請**
  - シン・クライアントによる業務システムの増加
    → シン・クライアントの問題が顕在化
  - 基幹業務システム（ホスト・C/Sシステム）のリプレイス時期が迫っている
  - ブロードバンド環境の整備



- **「良いとこ取りの技術」=リッチ・クライアント**
  - クライアントの資源を利用した高度な処理、表現が可能
  - アプリケーションの配布、バージョン管理が容易

  → ファット・クライアントとシン・クライアントの「良いところ取り」の技術

リッチ・クライアントの分類（1）

- **ファット・クライアント（C/Sアプリ）拡張型**
  - 従来のファット・クライアントに自動配布、バージョン管理の機能を追加したもの
  - スマートクライアント（ノータッチデプロイメント、ClickOnce）、Java Web Start等
- **ブラウザ・プラグイン型**
  - プラグインや拡張ランタイムによって、ブラウザの機能そのものを拡張
  - Macromedia Flash、Curl、Biz/Browser、Nexaweb、Visual Frame、JFrame Serverなど
- **スタンドアロン・アプリケーション拡張型**
  - 従来、スタンドアロン・アプリケーションとして利用されてきたものに、サーバ通信機能を追加したもの
  - Microsoft Office（InfoPath／VSTO）、Acrobatなど

リッチ・クライアントの分類（2）

- その他
  - リッチ・クライアント対応型のアプリケーション・フレームワーク
  - クライアント機能を含む、総合フレームワーク製品
  - JSF（JavaServer Faces）、SKreen Mill

✓ かならずしも綺麗に分類できるものではない
✓ベンダ、ソリューション、製品によって、リッチ・クライアントの概念は様々
✓ ただし、その多くは標準的な技術（または、それに準ずる技術）を前提に

リッチ・クライアントの導入効果（1）
～Webクライアントの課題から～

- **入力生産性の向上**
  - 操作が一画面内で完結（アプリケーションの動作が軽快）
  - 入力項目が多くても、すっきりとしたUIを実現
  - ファンクションキーやショートカットキーなどの割り当てが可能
  - オフラインでの作業が可能（ネットワーク負荷の軽減）
  - 異なるプラットフォームでも動作、表示を統一しやすい
  - 利用者の教育、サポートが容易
- **パフォーマンスの改善**
  - ページリフレッシュに際しても、差分データのみの送信でOK
  - サーバ・ラウンドトリップの発生を抑えることで、画面レスポンスを改善
  - ネットワーク負荷の軽減
  - サーバに対する処理負荷の軽減

リッチ・クライアントの導入効果（2）
～Webクライアントの課題から～

- **開発生産性、セキュリティの向上**
  - サブミットボタンの2度押しや途中ページへの直接アクセスなど、Webアプリケーション特有の問題を解決
  - 本来のビジネスロジックとは無関係な、冗長なコーディングが不要に
- **エンドユーザへの訴求効果**
  - 利用者に対するサポート、導入教育が容易
  - これまでに実現できなかったグラフィカルな表現力
  - 紙帳票と画面レイアウトとの整合
  - 新たなビジネスの創造、既存サービスの改善

→ Webの課題を克服するのみならず、Webを通じた新たな可能性の実現

Macromedia Flash

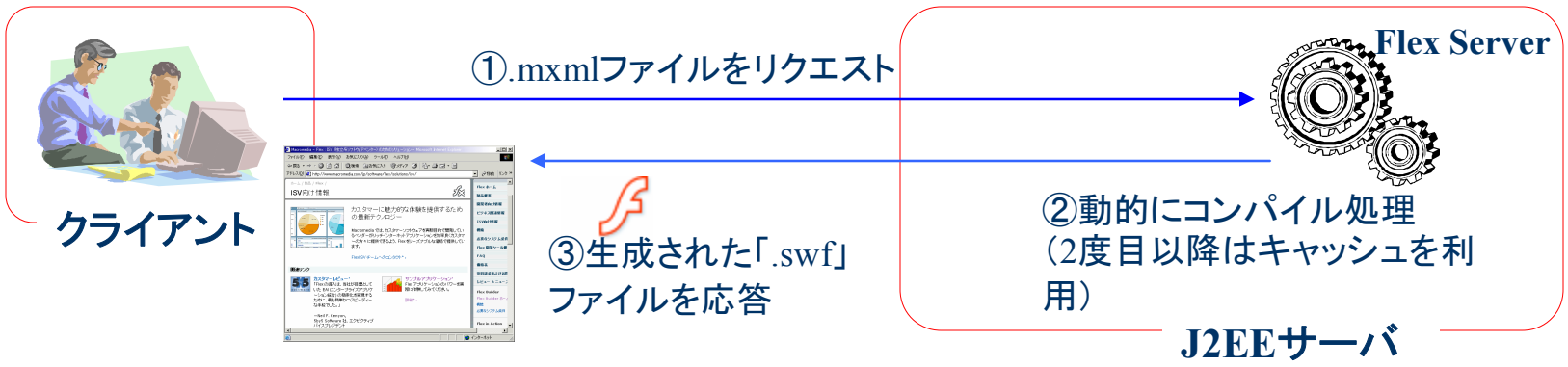
http://www.macromedia.com/jp/software/flash/

- 古くて新しい プラグイン技術の老舗
  - 高い表現力、アニメーションなどを実現
  - 開発言語はECMA準拠のスクリプト言語Action Script
  - ユーザも見慣れたユーザ・インターフェイス
  - 定評ある開発環境Macromedia Flash MX ／ ColdFusion MX
  - 必要なクライアント・アプリケーションは「Flash Player」のみ
    → ほとんどあらゆる環境にインストールされており、（事実上）環境設定は不要
  - Flash Remotingによるサーバサイド連携（=AMF：Action Message Format）
    → Java、.NET Framework ColdFusionなど主要なアーキテクチャに対応
    → AMFPHPやFLAPなどの利用でPHP、Perl環境でも利用可能

Macromedia Flex

http://www.macromedia.com/jp/software/flex/

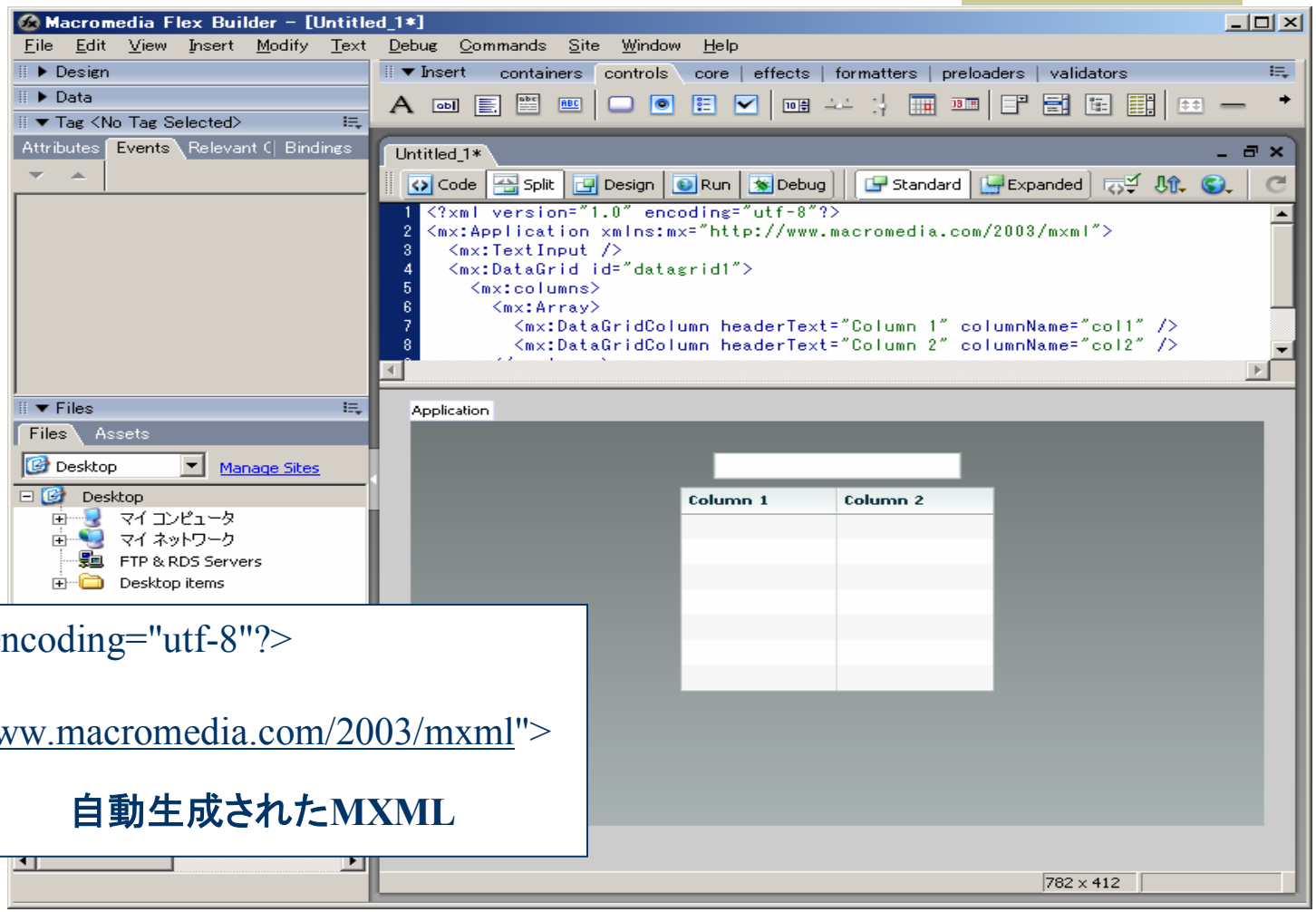
- **Flashを動的に生成するプレゼンテーションサーバ**
  - MXMLでFlashレイアウトをマークアップ言語で開発可能に
  - J2EE、.NET環境で利用可能（.NETは近日予定）
  - サーバサイドのビジネスロジックとの連携が容易（Flexクラスライブラリ）
    → データ・バインディング、イベント駆動型モデル
  - DreamWeaverをベースとした開発ツールFlex Builder
    → デザイナもプログラマも利用可能なツール
    → 将来的には、Eclipse プラグインのリリースも予定



参考）Flex Builder 開発イメージ

http://www.macromedia.com/jp/software/flex/flexbuilder/

- **Flex Builder のメイン画面**



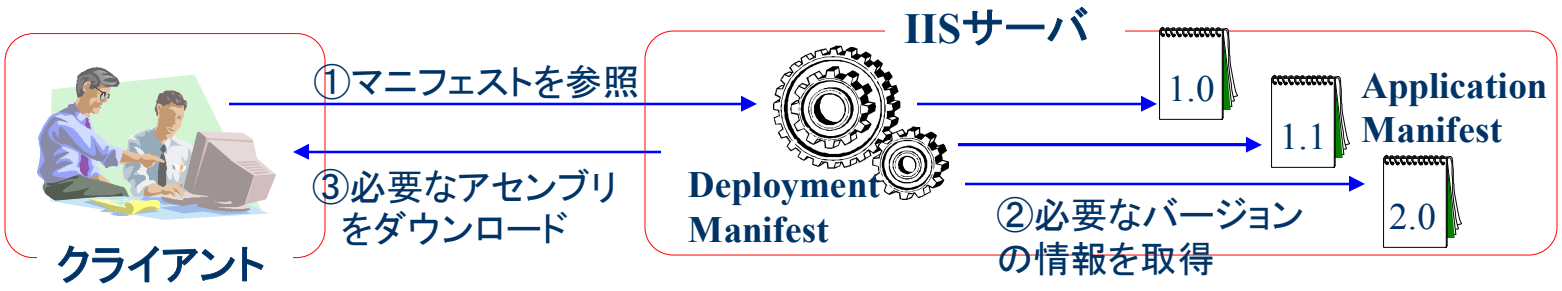

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
  xmlns:mx="http://www.macromedia.com/2003/mxml">
  <mx:TextInput />
</mx:Application>
```

自動生成されたMXML

ClickOnce（.NET Framework 2.0）

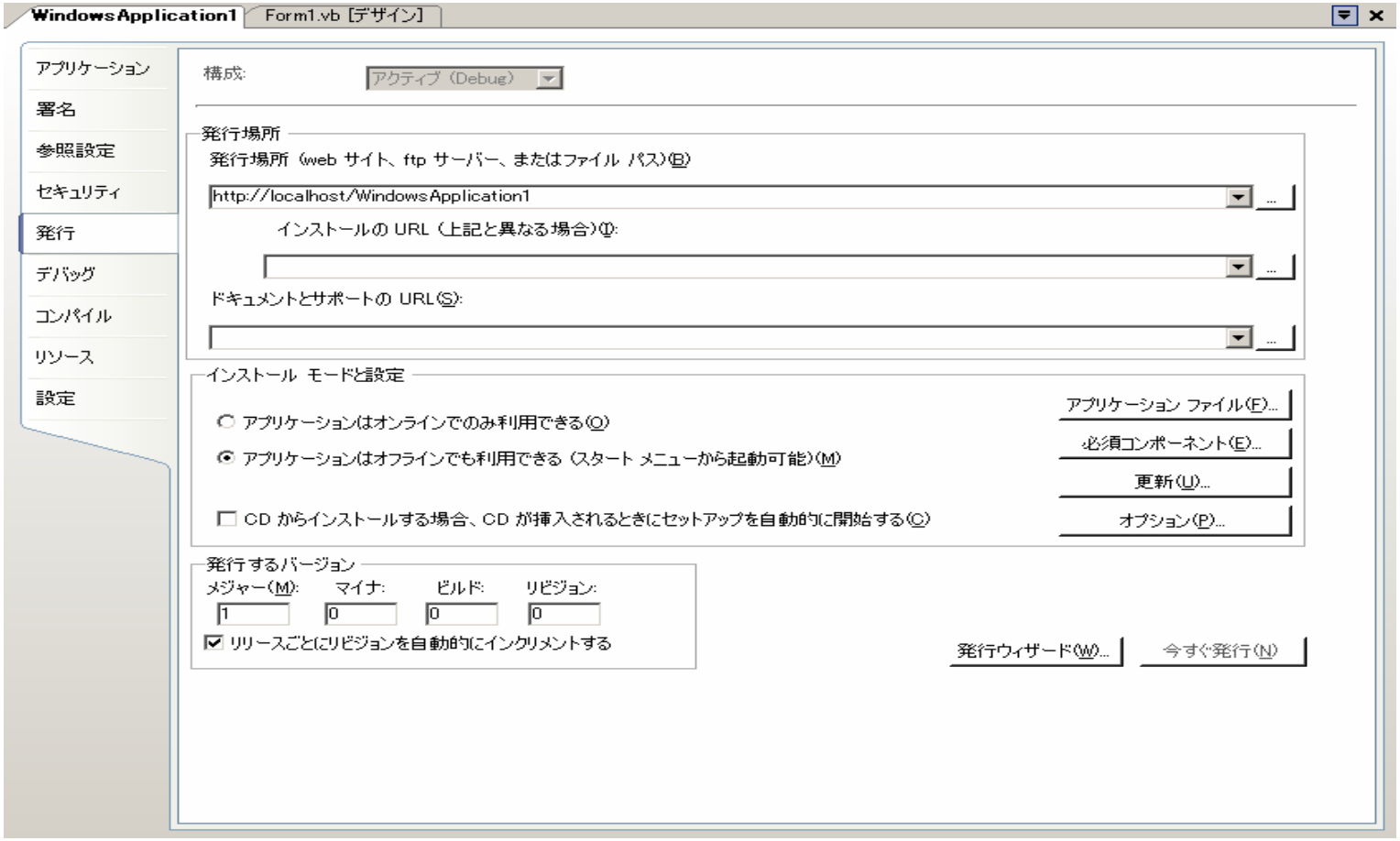
http://www.microsoft.com/japan/msdn/net/winforms/clickonce.asp

- .NETアプリケーションを簡単に配置するしくみ
  - スマート・クライアントの進化版
    - オフライン・サポート（2回目以降のアクセスはスタートメニューから可）
    - Windowsシェル統合（プログラムメニューへの追加等）
    - アプリケーションの自動更新＆必要に応じたロールバック
    - 配置、更新を制御する専用のAPI（System.Deployment名前空間）を提供
  - クライアントに.NET Framework 2.0の環境が必要
  - セキュリティ面も.NET Frameworkが管理
    - SandBoxによるセキュアな実行（ファイルやネットワークアクセスの制限）
    - マニフェストへの署名（更新には発行者キーが必要）
  - マニフェスト・ファイルで使用するアセンブリや配置すべきバージョンを記述



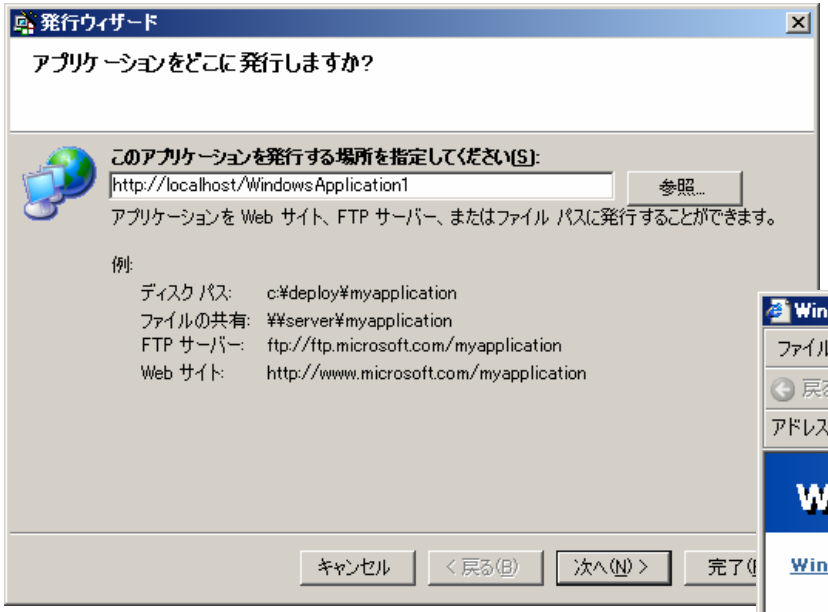
参考）Visual Studio 2005によるClickOnceアプリ発行のイメージ（1）

- ClickOnceアプリケーション発行の設定

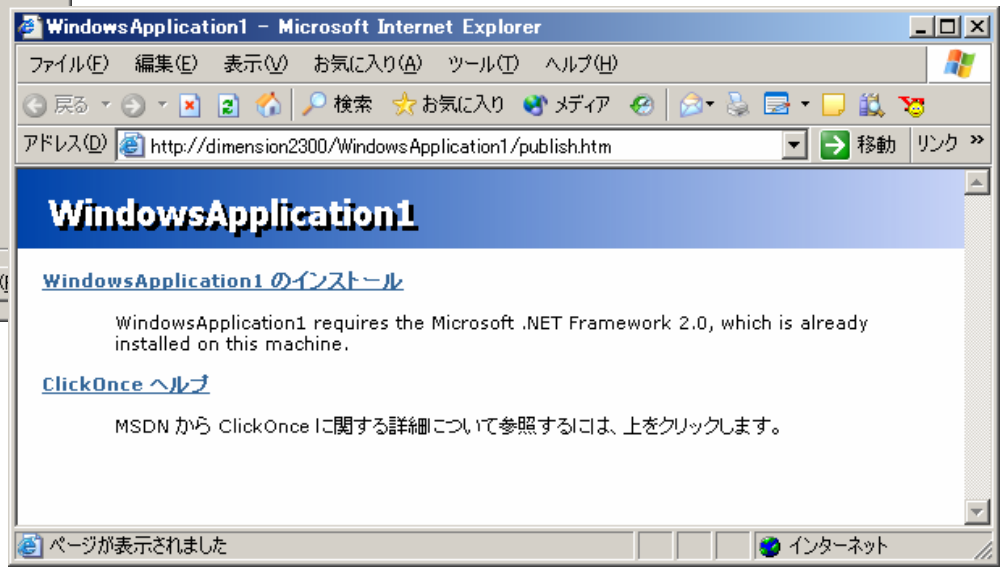


参考）Visual Studio 2005によるClickOnceアプリ発行のイメージ（2）

- **ClickOnceアプリケーションの発行**
  - 専用のウィザードを用意

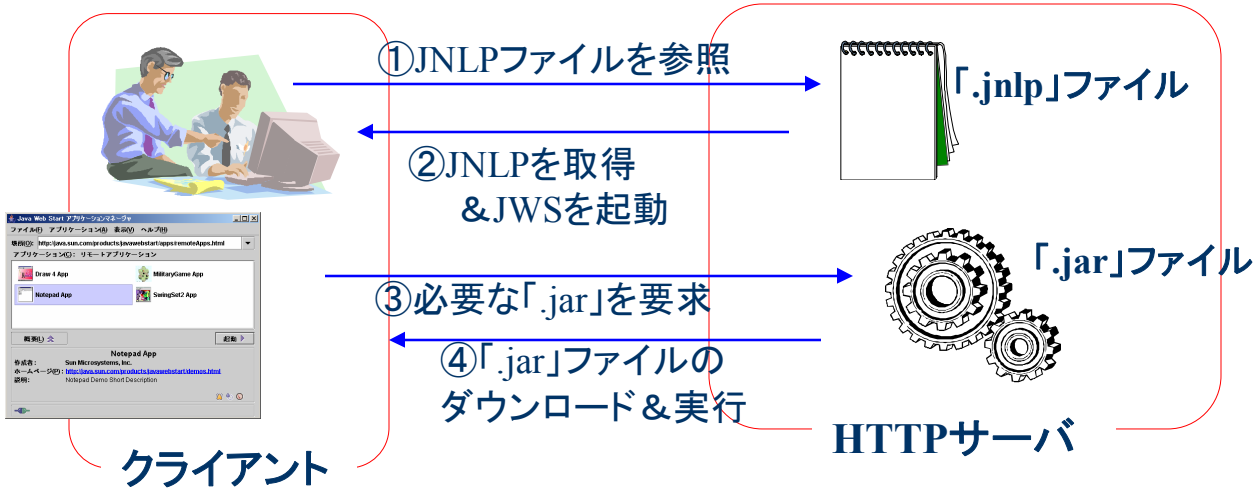


- **ClickOnceアプリへのアクセス**
  - ダウンロード用画面を自動生成



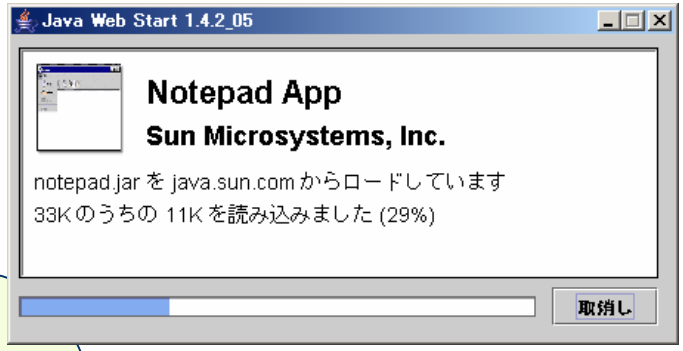
Java Web Start
http://java.sun.com/products/javawebstart/

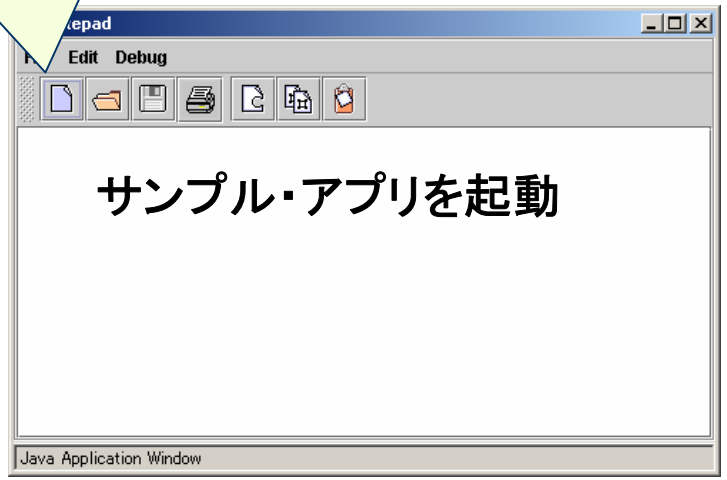
- **Javaアプリケーション（Swing、AWT、SWT等）を簡単に配置するしくみ**
  - JavaアプリをHTTPでダウンロード、クライアント側で実行（バージョン管理はJWS）
  - JNPLファイルで使用する「.jar」ファイルや必要なバージョンを制御
    → ClickOnceによく似たしくみ
  - 2回目以降のアクセスでは更新の有無のみを確認
    → 登録されたJWSアプリはアプリケーション・マネージャから起動可能



参考）Java Web Start Application Manager とサンプル・アプリケーション

- JWSアプリケーション・マネージャ

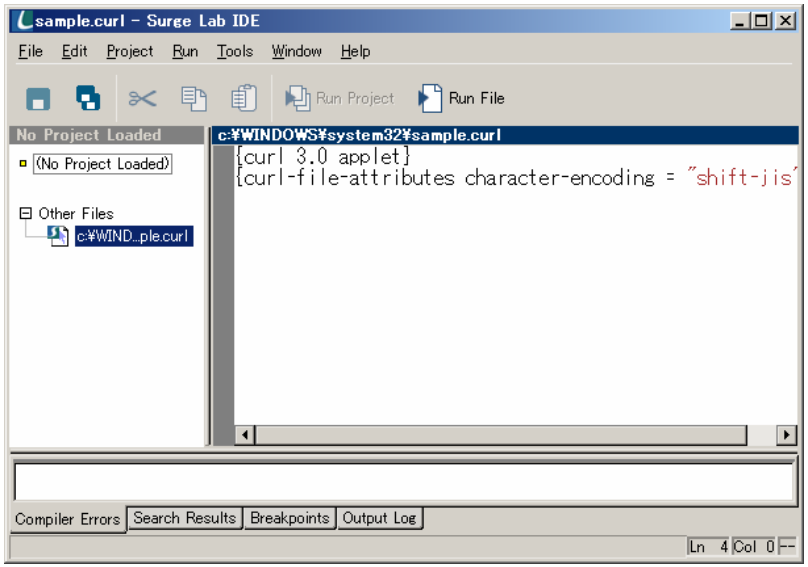


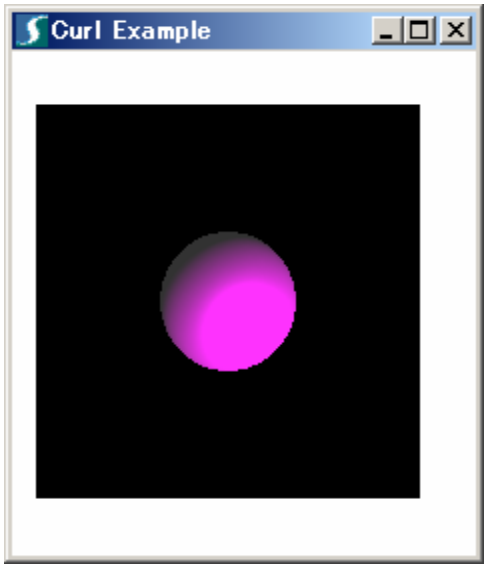


Curl

http://www.curl.com/

- マークアップ、スクリプト、オブジェクト指向言語を統一された環境で提供
  - マサチューセッツ工科大学で開発された言語
  - グラフィックやCGシミュレーションなどの用途に適する
  - CSS的なレイアウトデザインからDHTML的なインタラクティヴな動作、マルチメディア表現までを単一言語で実現（初学者からエキスパートまで）
  - ブラウザ機能をSurgeプラグインで拡張（インタプリタで実行）

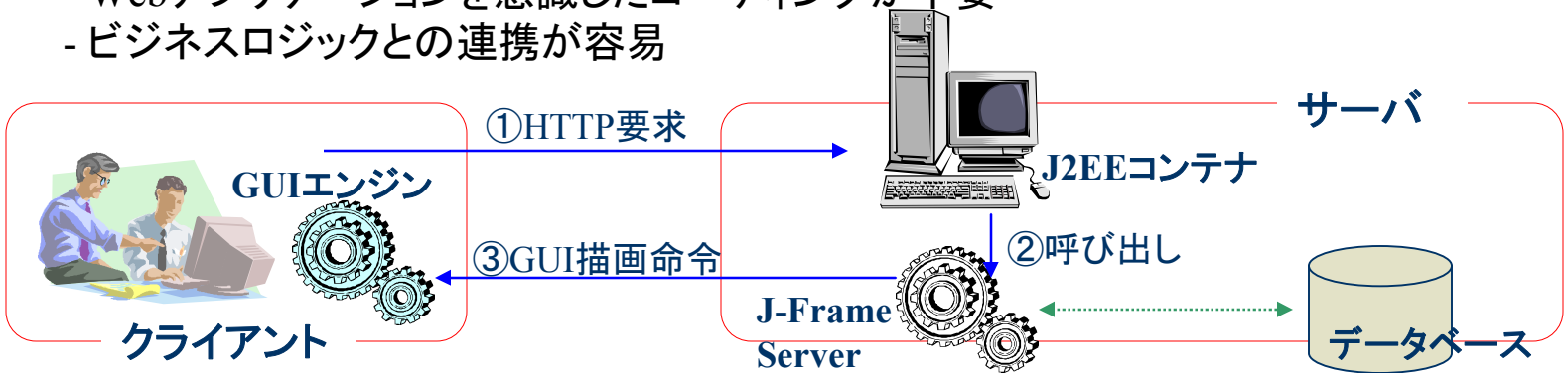




（左）Surge Lab IDE…ソース編集からデバッグまで対応
（右）Curlアプレットの実行画面

Biz/Browser **http://www.axissoft.co.jp/biz/**
J-Frame Server **http://pf.toshiba-sol.co.jp/prod/f_serv/termserv/**

- **200社以上への導入実績を誇る国産リッチクライアント・ツール Biz/Browser**
  - 独自のCRS（Chain Reflection Script）ファイルで画面描画指示のみを転送し、高速画面表示を可能に（IE上のプラグイン、スタンドアローン双方で動作）
  - 専用のGUI/スクリプト開発環境 Biz/Designerを提供
- **J2EEサーバと連動するGUIサーバ J-Frame Server**
  - クライアント側はサーバからのGUI描画命令のみを受け取り、画面を表示
  - GUIアプリケーションをサーバサイドで一極管理
    - セキュリティ向上
    - クライアント環境による違いが少ない
    - Webアプリケーションを意識したコーディングが不要
    - ビジネスロジックとの連携が容易



Acrobat

http://www.adobe.co.jp/products/acrobat/

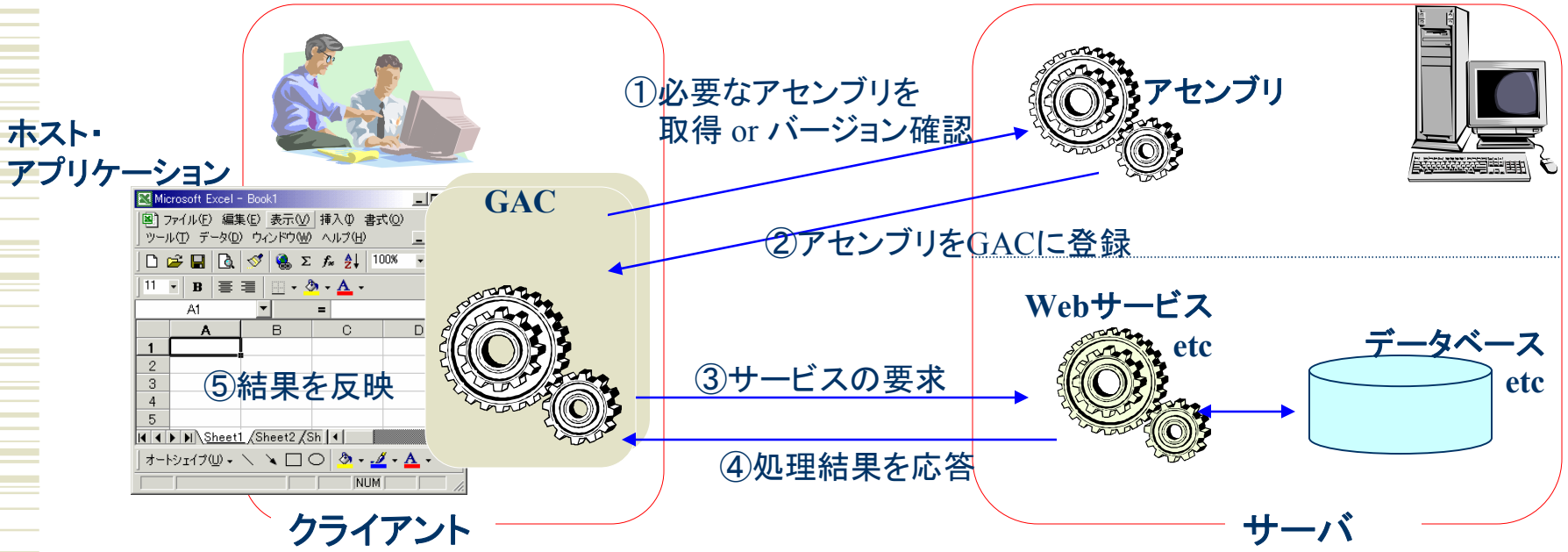
- **PDFとXMLの融合でドキュメントとビジネスプロセスを融合する**
  - ハイブリッドな情報管理プラットフォーム
    - システムのフロントエンド
    - 情報コンテナ（XMLデータ、マルチメディアコンテンツ）
    - アウトプット／アーカイブフォーマット（電子ドキュメントの原本化）
      → 定型・非定型、人とコンピュータ、文書とプロセスの融合
  - 各種Adobeサーバ製品との緊密な連携
    - Form Server = PDFフォームの動的生成
    - Central Pro Output Server = バックエンドとの連携による帳票出力
    - Document Server for Reader Extensions = PDFへの権限付与を管理
    - Graphics Server = 画像情報を元に動的にカタログなどのデザインを出力
  - ユニバーサル・クライアント
    → ユビキタスな利用が可能な「リッチクライアント」

VSTO（Visual Studio Tools for Office）
http://www.microsoft.com/japan/msdn/vstudio/office/

- **Microsoft Officeの機能を拡張するための.NET アセンブリ**
  - 使い慣れたOfficeインターフェイスをそのままに サーバ連携を実現
  - VB.NET、C#などの.NET標準言語を開発に利用可能
    → Visual Studio .NETで一元的な開発が可能
    → バックエンドとフロントエンドとのビジネスロジック共通化
  - 必要なアセンブリを自動的にGAC（Global Assembly Cache）にダウンロード
    → 2回目以降のアクセスではアップデート時のみアセンブリを自動取得
    （オフラインでの作業も可能）
  - CAS（Code Access Security）によるコード実行の制御
    → ファイルシステムやネットワークへのアクセス、印刷などの制御
  - .NET Framework、プライマリ相互運用アセンブリ（PIA：Primary Interop Assembly）などをクライアント側にあらかじめインストールの必要あり

参考）VSTO実行のしくみ

- 必要な.NETアセンブリを動的に取得／実行



InfoPath

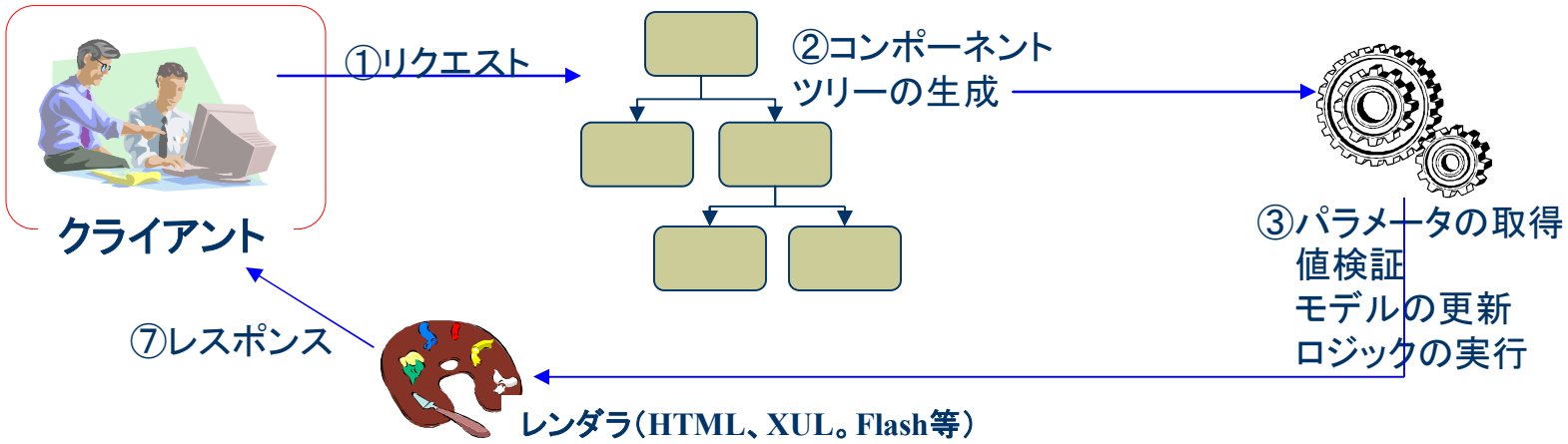
http://www.microsoft.com/japan/office/infopath/prodinfo/

- **XMLプラットフォームでデータの再利用性を最大限に追求**
  - フォームの作成／入力機能、データの入出力機能を提供
  - フォームが使用するスキーマ、スタイルシート、その他リソース（画像ファイル等）、入力検証のためのロジックを一元的に管理
  - 入力検証のようなフォーム生成に際する定型的な処理をノンコーディングで実現
  - XML Webサービスを介することでサーバサイドとの連携も容易に可能
  - クライアント側にInfoPathが必要

JSF（JavaServer Faces）

http://www.jcp.org/en/jsr/detail?id=127

- **J2EE標準のアプリケーション・フレームワーク**
  - イベントドリブン型のアプリケーション開発（VBライク）
  - 画面の構築はUIコンポーネントで行う
    → 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
  - Sun Java Studio Creatorなどの開発ツールも充実
  - UIとビジネスロジックとを明確に分離



リッチ・クライアント利用に伴う問題点（1）

- **開発生産性の観点**
  - 標準化された開発ツール、手法が確立されていない
  - 工期・費用が見積もりにくい
  - スキルのある人材を集めにくい
  - リッチクライアント導入の実績・経験が浅い
- **パフォーマンスの観点**
  - 委ねる処理によっては、クライアントマシンに高い性能を要求
  - プログラムのダウンロードサイズが大きくなった場合、ネットワーク負荷も大

リッチ・クライアント利用に伴う問題点（2）

- **システム保守・運用の観点**
  - ビジネスロジックの混在による保守性の低下
  - リッチ・クライアントを動作するための基礎アプリの設定が必要
  - 一部のアーキテクチャでは、クライアントライセンスが発生
- **システム連携の観点**
  - 異なるリッチ・クライアント間の連携が困難（セッションの維持、データ授受etc）
  - バックエンド・システムとの標準化された連携手段が不在

リッチ・クライアントは、いわゆる「銀の弾丸」ではない！

まとめ：リッチ・クライアント技術は過渡期（1）

- **標準的な開発手法の確立**
  - リッチクライアント対応アプリケーション・フレームワークの充実
    → .NET Framework、JWS、JSFなど
  - リッチクライアント同士の連携モデルを確立
    （Ex. JSFや.NET Framework？）
  - Macromedia FlexやVisual Studio（Orcas）、Eclipse etc
- **開発手法、アプローチの標準化**
  - 開発担当者の役割分担
    → 特にデザイナとプログラマの協業は益々重要
  - エンドユーザの視点から見たユーザビリティ、アクセサビリティの追求
  - ユニバーサル・デザイン

まとめ：リッチ・クライアント技術は過渡期（2）

- **適材適所、最適なクライアントを取捨選択**
  - 複数のクライアント技術並存は当たり前
  - ×リッチ・クライアントとはどうあるべきかの理念論ではなく
    → Webのメリットを活かすクライアント技術の模索を
  - シン・クライアントの課題を克服 ＋ 新たなビジネスに向けた付加価値を