



# IBM WebSphere Application Server Feature Pack for Web 2.0 の概要

2007年12月19日

日本アイ・ビー・エム株式会社  
ソフトウェア事業、  
WebSphereテクニカル・セールス

# 目次

- Feature Packとは
- Web 2.0 と SOA
- Ajaxの狙いとWAS Feature Pack for Web 2.0 のメリット
- Feature Pack for Web 2.0 ハイライト
- Feature Pack for Web 2.0 機能詳細
- サマリー

# Feature Packとは

- WAS V6.1に特定のテーマに沿った最新機能を追加
  - ▶ 拡張機能をアドオン
    - 最新の仕様をWAS上で使うことができますようになります
  - ▶ Bug Fixではない
    - Fixは今まで通りFixPackという形で提供されます
- 現在公開されている、または公開予定のFeature Pack
  - ▶ WAS 6.1 Feature Pack for Web Services **公開中!**
    - <http://www.ibm.com/jp/domino01/mkt/websphere.nsf/doc/001637B1>
  - ▶ WAS 6.1 Feature Pack for EJB3.0 **公開中!**
    - <http://www.ibm.com/jp/domino01/mkt/websphere.nsf/doc/000AD260>
  - ▶ WAS Feature Pack for Web 2.0 **今回の資料の対象**
    - <http://www.ibm.com/jp/domino01/mkt/websphere.nsf/doc/0008DBBF>
  - ▶ WAS 6.1 Feature Pack for SOA **ベータ・プログラム**
    - <https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/soawas61/>



全てWebからダウンロードして利用可能。今すぐ試してみることができます！

# なぜリリースアップではなく、Feature Packなのか？

- 新しいテクノロジーに追従する必要のないシステムにとって、頻繁に行われるバージョンアップやリリースアップは、混乱のもとである。
- 新しいテクノロジーを検証したり、早期適用を行いたいというお客様の要望もある。
  - ▶ JAX-WS、JAX-B、WS-RM、WS-SC、MTOM、SCA/SDOなどなど

最新Webサービスを  
早期適用したいお客様



V6.1の現状機能を  
維持したいお客様



SCA/SDOを検証したい  
お客様



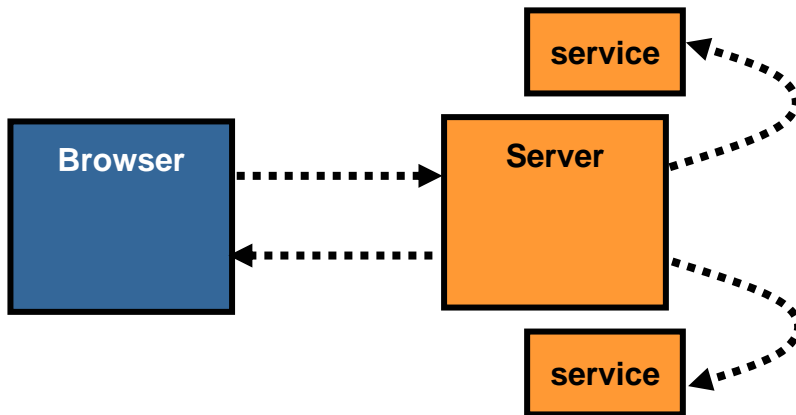
Webサービス  
Feature Pack

SOA Feature Pack

WebSphere V6.1

# WebアプリケーションとSOA

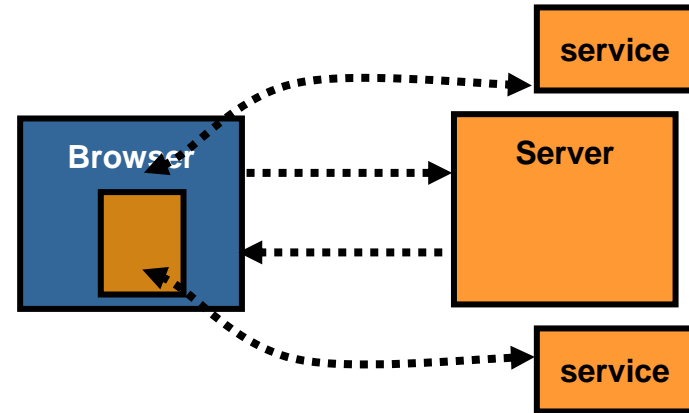
従来のWebアプリケーションのパターンは、サーバー側でのサービスアクセスを推進



既存のWebサービス標準 (WS\_\*, WSDL, SOAP) では、通常このモデルを採用

様々な通信プロトコルを利用した多様な言語からのアクセスに焦点

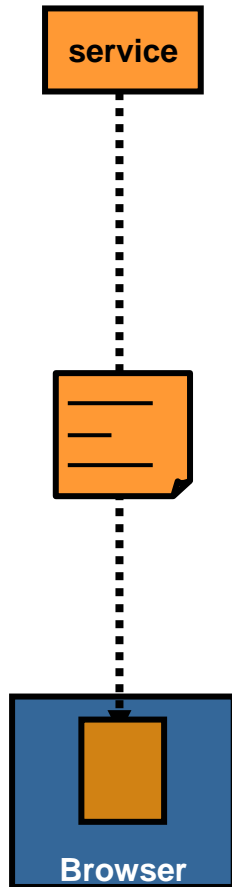
Web 2.0 アプリケーションパターン は、クライアント側(ブラウザ)でのサービスアクセスを推進



RESTやJSONを活用した新たなWebサービスのアプローチは、通常このモデルを採用

ブラウザからAjaxを利用して、ひとつの言語 (JavaScript) で、ひとつのプロトコル(HTTP)からのアクセスに集約

# Web 2.0 スタイルのサービス – キーコンセプト



## REST

- ▶ **RE**presentational **S**tate **T**ransfer
- ▶ サービスやリソースにアクセスするために、HTTPのセマンティクスに依存したサーバーサイドのアーキテクチャスタイル
- ▶ Ajaxでブラウザから簡単に呼び出し可能

## JSON

- ▶ **J**ava**S**cript **O**bject **N**otation
- ▶ ブラウザーとサービスの間で情報交換をするためのデータ・フォーマット
- ▶ JavaScriptクライアントから直接そのまま利用できる

## Ajax

- ▶ **A**synchronous **J**ava**S**cript **A**nd **X**ML
- ▶ 非常にインタラクティブで、わかりやすいWebページを提供することができるブラウザベースのテクノロジー
- ▶ クライアントから直接サービス呼び出しを可能にする

## Web 2.0 スタイルのサービス – 解こうとする問題点は？

- ブラウザーからSOAP Webサービスへのアクセスは複雑
  - ▶ ブラウザーは、SOAP Webサービスのフォーマットとプロトコルを、ブラウザーと親和性の高いJavaScriptやHTMLへ翻訳する必要がある。
  - ▶ ブラウザー・サイドのロジックが複雑になると、ブラウザーの互換性問題から生じる個別テストとデバッグが増えてしまう。(Microsoft IE, Firefox, Safari, Opera, など)

Web 2.0 スタイルのサービスは、ブラウザー・アクセスの簡素化に焦点

- HTTP 中心のサーバーパターン (REST)
- JavaScript と親和性の高いデータ・フォーマット (JSON)
- ブラウザーによる最小のオーバーヘッドでのサービス呼び出し(Ajax)

## その他のWeb 2.0アプリケーションの課題

- Ajaxを使えば、リッチなユーザー・エクスペリエンスが可能になりますが、アプリケーション開発者に新たな課題をもたらします。
  - ▶ ブラウザーとサーバーでのイベント処理
    - 従来型のWebアプリケーションは、サーバー中心。一方、Ajaxアプリケーションは、クライアント・サーバー同居モデル
    - イベントは、クライアントで作成されサーバーに送信されるものもあれば、サーバーで作成されクライアントに戻ってくるものもある
    - Ajaxアプリケーションは、クライアント・サーバー間のイベントのフローを処理するのに、通常Pub Subエンジンを利用する
  - ▶ Ajaxのセキュリティにまつわる懸念
    - JavaScriptコードの急増は、多くのセキュリティ問題をもたらします。特に顕著なのは、クロスサイトスクリプティング攻撃です。
    - プログラマーがアプリケーションを構築する上で、正規に複数のクロスサイト機能を使いたい場合、ブラウザーのセキュリティ制限が問題を引き起こします。

サーバー側のサポート、例えば、イベント処理やプロキシ・サポートは、Ajaxアプリケーションの課題の解決に有効です。

# WAS Feature Pack for Web 2.0 を使用するメリット

- **Feature Packが提供する機能**
  - ▶ SOAとWeb2.0の接続
  - ▶ インタラクティブなAjaxアプリケーションが開発可能
  - ▶ 外部サービスのマッシュアップ
  - ▶ 開発コストとリードタイムの削減
  - ▶ 企業に必要となる、Ajaxの標準ライブラリーを提供

内部および外部にあるサービスやデータに接続可能な、リッチなユーザー・エクスペリエンスを提供します。

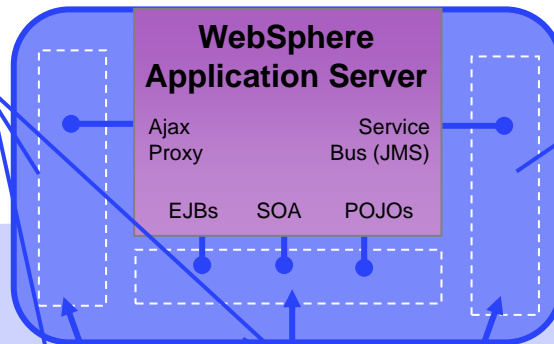
# WAS Feature Pack for Web 2.0 ハイライト

## Web 2.0 から SOA への接続

AjaxクライアントからSOAサービスや他のJEEアセットへ接続が可能になります。Webフィードを使って企業データをお客様やパートナー様へ拡張できます。

## AJAX メッセージング

Ajaxクライアントは、株価やインスタントメッセージングのようなリアルタイムに更新されるデータに接続できます。



## 外部 Web Services

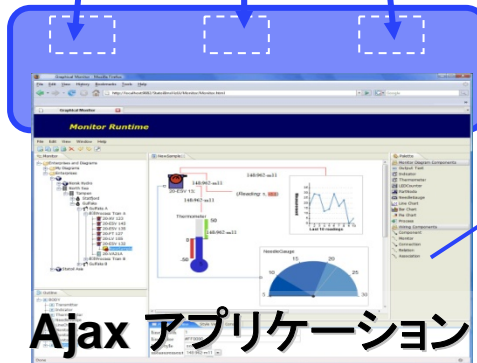


## イベントドリブンデータ

IBM \$125.25 +\$2.50... MSFT \$43.75 -\$1.50 ...



Web フィード

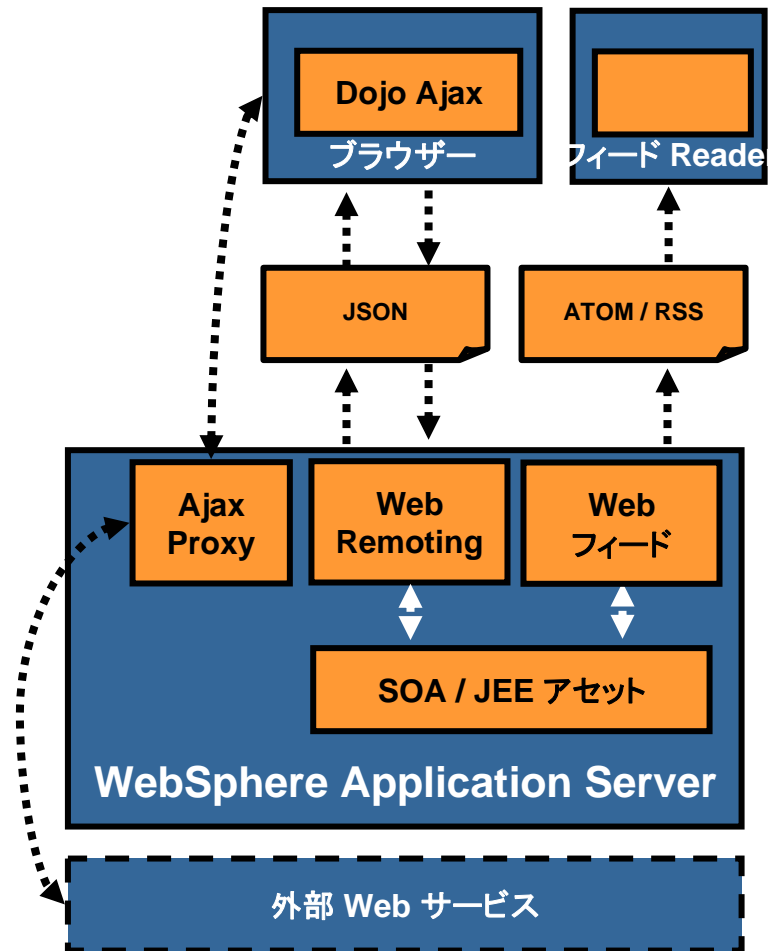


## Ajax 開発ツールキット

DojoというオープンソースJavaScript実行環境をベースにしたWebSphere Application Server向けの最高クラスのAjax開発ツールキット

## Web 2.0からSOAへの接続 – 概要

- **Web 2.0 から SOA への接続** – Ajaxクライアントから外部サービス、内部SOAサービス、その他のJEEアセットへの接続を可能にします。
- Webフィードを利用して、お客様やパートナー様に企業データを配信します。



## Web 2.0 から SOA への接続 – 機能リスト (1 of 2)

### ■ Web Remoting

- ▶ Java EEアセット(EJB, PoJo, Web Services Proxy)のメソッドを公開するための、軽量なWebのエンドポイントを提供
- ▶ Ajaxアプリケーションから、JSON, XMLフォーマットを使用して、簡単に呼び出し可能
- ▶ メソッドのHTTP GET/POSTマッピングをサポート
- ▶ オリジナルのアセット(Java Object, EJB, Web Services)を書き直す必要なし。簡単な設定オプションのみ。

### ■ JSON4J ライブラリー

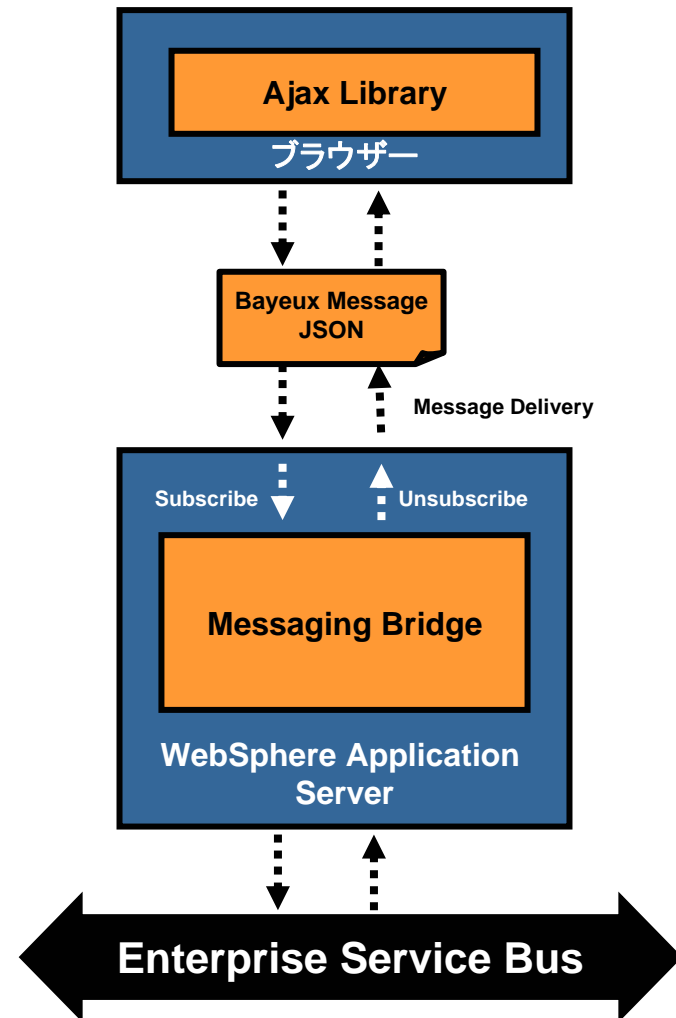
- ▶ Java環境で、JSON(JavaScript™ Object Notation)を取り扱うための実装
- ▶ JSONとは、軽量のデータ交換フォーマットとして、Ajax アプリケーションで徐々に人気が出てきています。(JSONについてはこちらをご参照ください。 <http://www.json.org>)

## Web 2.0 から SOA 接続 – 機能リスト (2 of 2)

- **Ajax Proxy** – 複数サイトに跨ったサービスへブラウザからアクセス
  - ▶ ブラウザーがWeb2.0らしく、複数サイトにあるサービスへアクセスできるようにするための軽量プロキシー
  - ▶ プロキシーは、Java EEアプリケーションに組み込むことも、スタンドアロンで個別にたてることも可能
  - ▶ プロキシーへのアクセス制御には、Java EEアプリケーション・レベルのセキュリティーを利用
  - ▶ ホワイトリスト・ポリシーのサポート。クッキー、MIMEタイプ、HTTPヘッダー、HTTPメソッド (GET, POST, PUT etc)といったリクエストのクライテリアに応じたフィルタリングが可能。
  
- **Web Feeds**
  - ▶ ATOM と RSS ライブラリー。Java EEリソースをWeb2.0スタイルの“データフィード”として公開するもの。フィードとは、データが更新されると、変更をクライアントにプッシュするモデルです。

## Ajaxメッセージング – 概要

- **Ajaxメッセージング** - Ajaxクライアントを株価やインスタント・メッセージング(チャット)のようなリアルタイムに更新されるデータに接続します。
- ブラウザーをサービス統合バス(SIBus)のTopicに接続し、サーバー側のイベントをブラウザーに配信します。

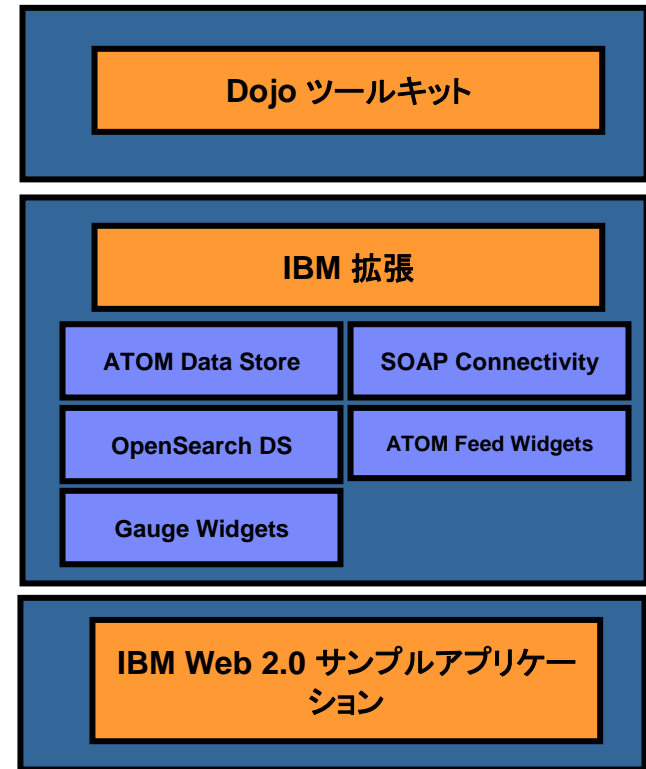


## Ajax メッセージング – 機能リスト

- Publish/subscribeメッセージングの実装により、ブラウザをWebSphere Application Serverのサービス統合バスに接続し、サーバー・サイドのイベントをブラウザにPushすることができるようになります。
  
- Ajaxメッセージングは、“Comet”サーバーモデルを実装
  - ▶ 擬似的な長時間接続
  - ▶ トピックベースのpublish/subscribeメカニズム
  - ▶ Java EE アプリケーションは、Ajaxクライアントとの双方向メッセージへのアクセスに、標準のJMS APIを利用。新しくServlet APIを追加する必要はなし。
  - ▶ クライアントとサーバーの通信には、Bayeux プロトコルを採用
  
- クライアント側のBayeux プロトコルのサポートは、Dojoツールキットが提供
  
- セキュリティ – 認可されないドメインへのクロスサイトアクセスは制限

## Ajax 開発ツールキット – 概要

- **Ajax 開発ツールキット** – オープンソースのDojoツールキットをベースにした、WebSphere Application Server向けの最高クラスのAjax開発ツールキット ([www.dojotoolkit.org](http://www.dojotoolkit.org))
- IBMは、Dojo Foundationのアクティブ・コントリビューターです。
- WAS Feature Pack for Web 2.0 は、Rational Application Developer 7, Eclipse 3.2, および Eclipse 3.3と互換性があります。



## Ajax 開発ツールキット – 機能リスト

- Dojo Toolkit v1.0のサポートつき配布
- 追加のIBM拡張 – 異なるプロトコル、データ形式、接続に対する作業を軽減
  - ▶ ATOM (APP) Datastore
    - ATOM APP 1.0と互換性のあるサービスの呼び出しを簡単にします。また、ATOMフィードをデータソースとして使用できます。Ajaxアプリケーション中でウィジェットにバインドされます。
  - ▶ SOAP connectivity
    - 公開されているSOAPベースのWebサービスをAjaxアプリケーションから簡単に呼び出せるようにします。
  - ▶ ATOM feed widgets
    - ATOM (APP) Datastoreを使用したサンプル・フィード・ウィジェット
  - ▶ Gauge widgets
    - アナログゲージとバーゲージ・ウィジェット。数値のモニターを表示するのに便利です。
    - リアルタイム更新をサポート (例えば、Webメッセージング・サービスを利用)
- IBM サンプル・アプリケーション
  - ▶ Quote Streamer
  - ▶ Ajax Enabled Plants by WebSphere

# WAS Feature Pack for Web 2.0 – サマリー

- Feature Pack for Web 2.0が提供する機能
  - ▶ **Web 2.0 から SOAへの接続**
    - Ajaxクライアントおよびマッシュアップ・アプリケーションから外部Webサービス、内部SOAサービス、および既存のJava EEアセットへ接続できるようになります。Webフィードを通じて、企業データを、お客様やパートナー様にまで配信できます。
  - ▶ **Ajax メッセージング**
    - Ajaxクライアントから、株価やインスタント・メッセージなどのリアルタイムに更新されるデータに接続できるようになります。
  - ▶ **Ajax 開発ツールキット**
    - Dojo ([dojotoolkit.org](http://dojotoolkit.org))ベースにIBM拡張機能を付加した、WAS用のAjax開発ツールキットを提供します。
- サポートするプラットフォーム
  - ▶ WAS V6.0.2 / V6.1
  - ▶ WAS Community Edition V2.0

# 補足資料

## AJAXとは？

- AJAXは、**A**synchronous **J**avaScript **A**nd **X**ML の短縮形
- 動的で、使いやすいWebページを作成するための技術
- WebクライアントをSOA (Service Oriented Architecture) の中でどんなサーバー (JEE, PHP, ASP.Net, Ruby on Rails, etc) にも接続できるようにする技術
- AJAXは、既存のテクノロジーと標準を活用 : JavaScript と XML
- AJAXは、Webアプリケーションの反応性とパフォーマンスに大きな改善をもたらす。例 Yahoo! Mail, Google Map, live.comなどその他多数ですでに利用されています。
- AJAX は、過剰に宣伝されているわけではありません。インタラクティブなアプリケーションを構築する上で大変有益であり、すでに現実のものとなっています。

# RESTとは？

- RESTは、„**R**epresentational **S**tate **T**ransfer“ の短縮形
- World Wide Web上のアーキテクチャー・モデル
- RESTの原則
  - ▶ リソース中心のアプローチ
  - ▶ すべての適切なリソースは、URIを通じてアドレスできる
  - ▶ HTTP – GET, POST, PUT, DELETEを使ったアクセスに統一
  - ▶ コンテンツネゴシエーションで、同じURIから異なる表現を検索できる
- REST スタイル・サービス
  - ▶ Webブラウザ、その他のクライアント、もしくはサーバーで稼動するコードから、簡単にアクセスできる
  - ▶ 同じリソースに対して複数の表現を提供する
- 参照:  
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

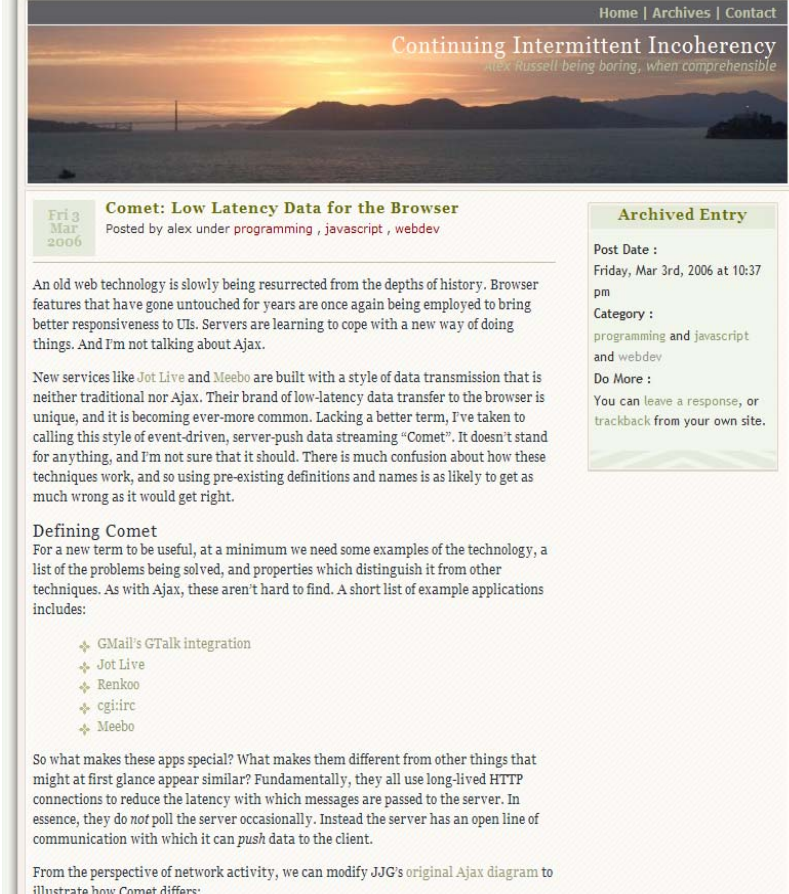
## JSONとは？

- RESTfulサービスのクライアントのほとんどは、JavaScriptで書かれている
- これを踏まえて、JSON(JavaScript Object Notation)は、JavaScript オブジェクトの迅速な交換を可能とし、しかも、シンプルで人間が読める形式である
- JSONは、名前-値のペア、および値の順序リストのコレクションから作成される

```
{  
  "customer" : {  
    "name" : "Jane Doe",  
    "company" : "Acme Enterprises"  
  }  
}
```

# Comet

- “Comet: Low Latency Data for the Browser”の中でAlex Russellによって提唱された、イベント駆動、サーバープッシュ型のブラウザへのデータ・ストリーミング
- Cometは、従来のAJAXポーリング・アプローチよりもブラウザに対してタイムリーにイベントを通知することができる
- ブラウザー・プラグインや専用のクライアントを必要としない
- イベント駆動I/Oを使って、サーバーのユーザー数を大きくスケールすることができる
- SOAに自然にフィットする、クライアント、サーバーの両方におけるイベント駆動アーキテクチャー
- AJAXという名前がアプローチを議論するのに便利だったように、同様にCometという名前と呼ばれる
- 蛇足：（自明だが、彗星を表す）Cometとはまったく関係はない



Home | Archives | Contact

## Continuing Intermittent Incoherency

*Alex Russell being boring, when comprehensible*

Friday, Mar 3rd, 2006

### Comet: Low Latency Data for the Browser

Posted by alex under programming , javascript , webdev

**Archived Entry**

Post Date : Friday, Mar 3rd, 2006 at 10:37 pm

Category : programming and javascript and webdev

Do More : You can leave a response, or trackback from your own site.

An old web technology is slowly being resurrected from the depths of history. Browser features that have gone untouched for years are once again being employed to bring better responsiveness to UIs. Servers are learning to cope with a new way of doing things. And I'm not talking about Ajax.

New services like *Jot Live* and *Meebo* are built with a style of data transmission that is neither traditional nor Ajax. Their brand of low-latency data transfer to the browser is unique, and it is becoming ever-more common. Lacking a better term, I've taken to calling this style of event-driven, server-push data streaming "Comet". It doesn't stand for anything, and I'm not sure that it should. There is much confusion about how these techniques work, and so using pre-existing definitions and names is as likely to get as much wrong as it would get right.

**Defining Comet**

For a new term to be useful, at a minimum we need some examples of the technology, a list of the problems being solved, and properties which distinguish it from other techniques. As with Ajax, these aren't hard to find. A short list of example applications includes:

- ✦ GMail's GTalk integration
- ✦ Jot Live
- ✦ Renkoo
- ✦ cgi:irc
- ✦ Meebo

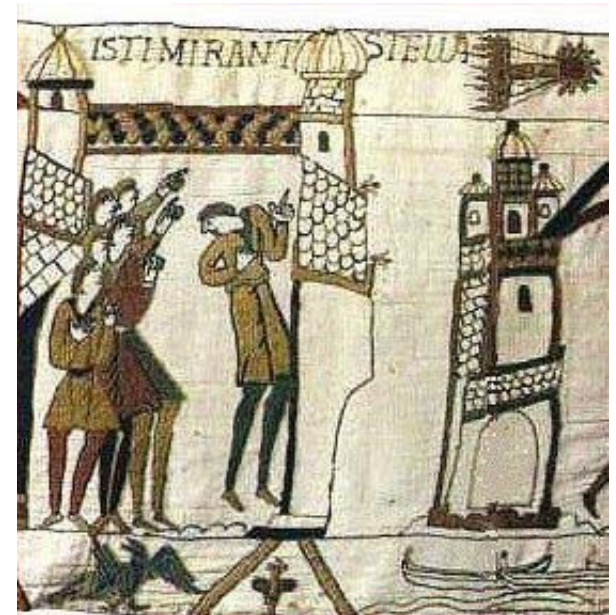
So what makes these apps special? What makes them different from other things that might at first glance appear similar? Fundamentally, they all use long-lived HTTP connections to reduce the latency with which messages are passed to the server. In essence, they do *not* poll the server occasionally. Instead the server has an open line of communication with which it can *push* data to the client.

From the perspective of network activity, we can modify JJG's original Ajax diagram to illustrate how Comet differs:

<http://alex.dojotoolkit.org/?p=545>

# Bayeux プロトコル

- Bayeuxは、JSONベースのプロトコルで、クライアントはイベントをSubscribeでき、サーバーは、従来のAJAXベースのポーリングよりもよりタイムリーにイベントを伝達できる
- ゴール
  - ▶ イベント通達を早くする
  - ▶ シンプルであること
  - ▶ プロトコルを拡張可能にする



*Extract from the Bayeux tapestry showing the arrival of Halley's comet*

# Dojo Toolkit v1.0



- リッチなユーザーインターフェイスのAjaxアプリケーションを開発するためのJavaScriptツールキットのひとつです。
- 主な特徴
  - ▶ ほとんどの主要ブラウザで稼働します。
  - ▶ 軽量かつ高機能
  - ▶ 機能
    - Dojo コア
      - ▶ ユーティリティ
      - ▶ イベント処理システム
      - ▶ Ajaxサポート
      - ▶ ドラッグ&ドロップ
      - ▶ 言語ユーティリティ、ローカライゼーション・サポート
      - ▶ データ・アクセス
    - Dojo ウィジェット
      - ▶ アクセシビリティ
      - ▶ 高品質で自然なデフォルトテーマ (置換可能)
      - ▶ 拡張可能なレイアウト、フォーム
      - ▶ データ・バインド・ウィジェット
  - 多くのコミュニティが拡張モジュールを提供

