グリッドとSOAとの意外な関係

丸山不二夫 （稚内北星学園大学） maruyama@wakhok.ac.jp
中田秀基 （産業技術総合研究所） hide-nakada@aist.go.jp

連載開始にあたって

Web 上での情報システム構築法の新たな潮流として，Web サービスという言葉を聞く機会が増えています．Web サービスの基本部分は，HTTP をはじめとした比較的単純な要素技術の上に構築されています．Web サービスは，こうした基本的な技術の土台の上に，多くの技術を取り入れ，それらの技術を標準化しながらその能力を拡大しています．**図 -1** に，代表的な Web サービス標準技術を示しました．

もちろん，図 -1 で，すべてであるわけではありません．W3C[1] や OASIS[2] で標準化されているものに限っても，Web サービス関連の標準技術は多数存在しています．もちろん，こうした技術は現実の必要の中から生まれ，先行した技術を必要に応じて利用して，相互に関連しあっているのですが，残念ながら，直接に仕様書を読むだけでは，そうした背景を十分に知ることはできません．Web サービスの全体像を捉えることは，なかなか容易ではありません．

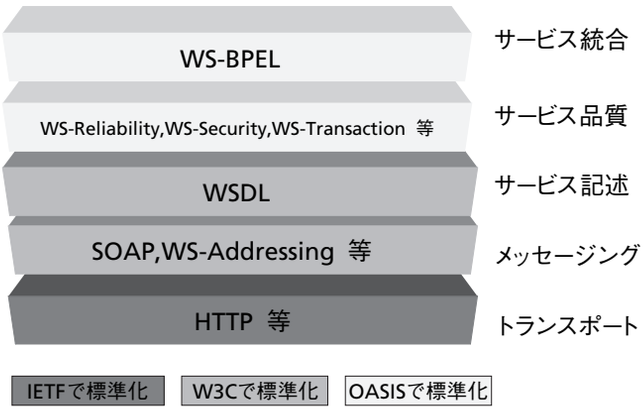

図 -1　代表的な Web サービス標準化技術

この連載のアプローチ

この連載は，Web サービス関連の標準技術を，Web サービスを知らない人にも分かりやすく紹介することを目的としています．そのために，次のようなアプローチをとろうと考えています．まず，Web サービスの近年の発達に大きな影響を与えた，グリッド技術と SOA（Service Oriented Architecture：サービス指向アーキテクチャ）という考え方の 2 つに注目して，Web サービスの標準技術を紹介していきたいと思います．その意味では，先の図 -1 には出ていないのですが，グリッドの世界で重要な役割を果たしている Web サービス標準技術を，重点的に紹介したいと思っています．

Web サービスを積極的に活用しようとしているグリッド技術に注目することで，多様な Web サービス関連技術間の関係が具体的に見えてくるのではと思います．また，非常にアクティブな変化が続いている SOA 周辺のさまざまなトピックを紹介することで，いろいろな切り口から Web サービスに迫ることができればと考えています．

はじめに

◉ Web サービスと Web アプリケーション

前置きはこれくらいにして，いよいよ，Web サービスの説明に入りたいと思います．ただ，Web サービスがどのような技術なのかを紹介する前に，ひとつ断っておきたいことがあります．筆者は，この「Web サービス」というネーミングが，誤解を招きやすいものだと感じています．似たような言葉に，「Web アプリケーション」という言葉があります．「Web サービス」と「Web アプリケーション」とは，はっきり異なったものなのですが，その違いを説明できる人は，意外と少ないかもしれません．

「Web サービス」という言葉を初めて聞いた人が，それを，「あの Web を利用したサービスだろう」と考えるのは，ありそうなことです．でも，たいていの場合，それは誤解なのです．といいますのは，「あの Web」と言うとき，たいていの人がイメージしているのは，誰か

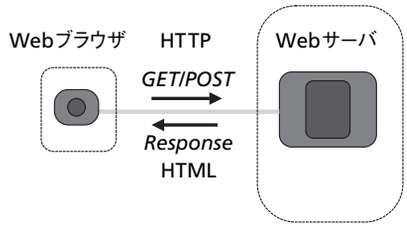

図 -2　Web のしくみ

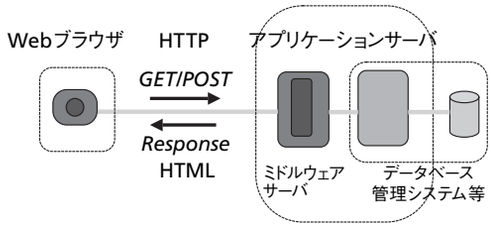

図 -3　Web アプリケーションのしくみ

が Web のブラウザを使ってインターネットを見ているところのイメージだと思います．そもそも，Web のブラウザの存在しないインターネットの世界をイメージするのは，とても難しいことです．

◉ Web アプリケーションの発想

〈Web ブラウザを汎用クライアントとして利用する〉

通常のサーバ･クライアント･モデルでは，アプリケーションごとにサーバとクライアントを開発し, かつ, サーバとクライアント間のプロトコルを設計しなければなりません．現在最も一般的に普及しているサーバ･クライアントシステムである Web の場合も，サーバとしての Web サーバと，クライアントとしての Web ブラウザと，両者をつなぐプロトコルとしての HTTP が，基本的な構成要素となります（**図 -2**）．

エンタープライズ向けのシステムである Java EE を始めとする，Web アプリケーションの基本的な発想は，アプリケーションごとに異なるクライアントを開発するのをやめて，「汎用クライアント」として Web のブラウザを利用しようとする試みであると考えることができます．プロトコルはもちろん HTTP を使います．このように，クライアントとして，Web のブラウザの機能を利用したアプリケーションを「Web アプリケーション」と呼びます．最近話題の Web2.0 も，Web のブラウザを徹底的に「プラットホーム」として利用しつくそうとするものですので，基本的には，Web アプリケーションだということになります．

このアプローチでは，データベースの管理システムなどサーバ側で行われるべき処理と「汎用クライアント」としての Web ブラウザの「中間」に，サーバ側での処理の結果をクライアントである Web ブラウザ向けに変換するサーバが介在することになります．これを，ミドルウェア・サーバといいます．「中間」にあるサーバという意味だと考えていいと思います（**図 -3**）．

◉ Web サービスの発想

〈Web サーバを汎用サーバとして利用する〉

ところで，Web サービスでは，Web アプリケーションのように Web のブラウザが大きな役割を果たすことは，ほとんどありません．それでは，どうして「Web サービス」などという名前がついたのでしょうか？それは，Web サービスが，Web の「クライアント」である Web ブラウザではなく，Web の「サーバ」である Web サーバの機能を利用したシステムだからです．

Web サービスの発想は，アプリケーションごとに，新たにサーバを作ることをやめて，1 つの汎用のサーバで済まそうというものです．Web サービスは，この 1 つの汎用のサーバに，おそらく世界中で最もたくさん存在している Web のサーバを選んだのです．Web サービスは，サーバとサーバ・クライアント間のプロトコルについても，アプリケーションごとに独自のプロトコルを作ることをやめて，1 つの共通なプロトコルで済ませようとします. Web サービスは,「1 つの共通なプロトコル」として，最も基本的な Web のプロトコルである HTTP を選んだのです．

◉ Web から Web サービスへ

〈Web サーバと HTTP の再利用〉

Web というのは，単純化していえば，次のようなシステムです．

「Web サーバは，HTTP というプロトコルを使って，主に HTML という形式で書かれている人間が閲覧するためのドキュメントを，リクエストに応じて，クライアントとしての Web ブラウザに送りだす」（図 -2）

Web サービスのベースになっているのは，これにきわめてよく似た，次のようなシステムです．

「Web サーバは，HTTP というプロトコルを使って，XML という形式で書かれた機械可読な（コンピュータが処理することを前提とした）メッセージ

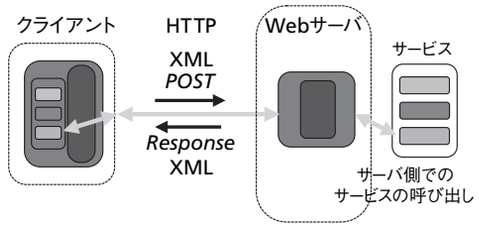

図 -4　Web サービスのしくみ

を，リクエストに応じて，クライアントに送りだす」（**図 -4**）

HTML が XML に変わったくらいで，ほとんど同じ仕掛けであることが分かると思います．「リクエストに応じて」と書きましたが，肝心なことは，Web サービスでは，Web サーバのすぐ後ろに，リクエストに応じてサーバ側のサービスを呼び出す仕掛けが用意されていることです．

◉ Web サービスの基本的な仕組みと SOAP

こうして，サーバ・クライアント・モデルの一変種として，次のような Web サーバを汎用サーバとして，サーバ側のサービスを呼び出そうとする枠組みが登場します（**図 -5**）．

1. サービスは，すべて Web サーバ上に「配置（deploy）」される．
2. クライアントは，Web サーバ上のサービスを，「ネットワーク上の手続き呼び出し（RPC：Remote Procedure Call）」として呼び出す．
3. クライアントは，XML を使って手続き名（Operation Name）と引数たちをコード化したメッセージを作成して，サーバに送り込む．サーバは，同じように XML を使って，返り値をメッセージとして返す．
4. この時，サーバとクライアント間のプロトコルは，HTTP を使う．
5. 手続きの引数・返り値には，基本的な型だけでなく，単純ではあれオブジェクトをとることができる．

Web サービスで最も基本的な標準技術である SOAP（Simple Object Access Protocol）は，先のリストの 5 のようなネットワーク上の手続き呼び出しの枠組みを規定しています．SOAP は，そのほかにも，交換される XML メッセージの形式や，型を持つデータの XML でのコード化の方法を定義しています．

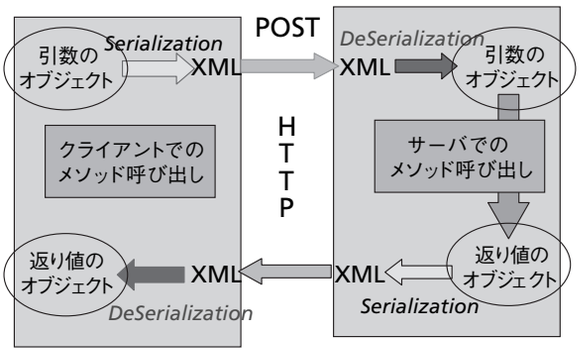

図 -5　Web サービス／ SOAP-RPC

SOA という考え方

SOA（サービス指向アーキテクチャ）というのは，本連載のテーマにもなっている大事な考え方です．SOA とは，すこし単純化していえば，ソフトウェア部品の組合せとして構築していたソフトウェアのシステムを，さまざまなサービスの組合せ（composite）として構成しようという考え方です．SOA システムの設計者は，システムを，独立した複数のサービスの協働（collaboration）として構成します．このレベルでは，設計者は，サービスをどのように実装するのかという関心とは独立に，システムを構想することができます．複数のサービスから構成されたものも，また一つのサービスであることに注意してください．こうして，小さな規模のサービスから，再帰的に組織化することによって，大きな規模のサービスを構成することが可能となります．

実装ではなくサービスに注目するという，こうしたアプローチは，同一のサービスが提供されるのなら，そのサービスの実装を取り替えてもシステムの機能は変わらないということですので，柔軟なシステム構成が可能となります．システムの設計者の立場を離れて，システムの利用者の立場で考えてみると，SOA の考え方は，むしろ自然なものです．私たちは，システムに対して，もっぱら，あるサービスの提供を期待しているのであって，そのサービスがどのように実現されているのかには関心がないはずです．これは，後述のように，実は，グリッドの考え方と基本的には同じものです．

◉ 疎結合と位置透過性

SOA では，サービスは，基本的にはネットワークを通じて提供されます．サービスを提供する SOA システムの構成要素は，ネットワーク上に配備されます．通常

のシステムでは，システムの構成要素は，1つのマシンのメモリの内部で密に結合しているのですが，SOAのシステムの構成要素は，ネットワークを介して疎な結合をしています．

このことは，物理的にはネットワークのどのような場所にサービスが存在していても，論理的には，同一のシステムを構成できることを意味しています．こうした性質をサービスが「位置透過性」を持つと呼ぶことがあります．逆の見方をすれば，グローバルなネットワーク上に存在するさまざまなリソースを結合して何かをしようとするならば，SOA的なアプローチは，きわめて有力なものになります．

SOAとWebサービス

それでは，SOAとWebサービスの関係は，どのようなものでしょう？

すでにお気付きだと思いますが，ネットワーク上でサービスを提供するSOAの基本的な構成要素の，最も有力な候補の一つがWebサービスなのです．

先にSOAの特徴の一つとして，どんな場所からでもサービスにアクセスできるという位置透過性をあげましたが，現実の世界のネットワークでは，セキュリティの関係で，それぞれのサイトがサービスへの自由なアクセスを許すということは通常はあり得ません．こうしたネットワークの現状が，SOAの中でのWebサービスの選択に大きな意味を持ちました．なぜなら，頑丈なファイアウォールで守られたサイトも，多くの場合，80番のHTTPのポートは開いていることが多いからです．Webサービスは，ファイアウォールを通過させやすいという特徴を持っているのです．

このように，SOAを実現するための重要な構成要素としてWebサービスの役割が期待されています．しかし，Webサービスによって実現されるのは，位置透過的にサービスを配置するところまでであり，これだけではSOAを実現することはできません．SOAを実現するためには，サービス相互を連携して機能させるための実行時サポート技術が必要になります．その実現のための有力な技術として最近注目されているのがグリッド技術です．SOAとグリッドは，もともと独立した技術分野でしたが，技術が実践的なものに進展するにつれて相互の関係が次第に深くなりつつあります．本稿の後半では，このグリッド技術に焦点をあて，グリッドとSOAの意外な関係と，これらの技術におけるWebサービスの役割について説明しようと思います．

グリッドとSOAの意外な関係

グリッドとは，噛み砕いていうと，「別々の所有者が管理しているたくさんの計算機やディスクを一時的に借りてきて，あたかも自分のものであるかのように使う技術」です．グリッドの用語を使って正確にいうと「複数の管理主体に属する資源にまたがって，仮想組織（Virtual Organization: VO）を動的に構成し，効率的な計算を実現する技術」となります．

グリッド技術の要素技術を考えてみましょう．グリッド技術は，複数の管理主体にまたがって通信を安全に行い，複数のセキュリティドメイン間で連携を行うセキュリティ技術，各サイト内の資源を有効に活用するための資源管理技術，各サイトの資源情報を共有する情報技術などの要素技術から構成されているといえるでしょう．このグリッド技術はSOAとどうかかわっているのでしょうか？

SOAは，ソフトウェア構築技法という側面でとらえれば，関数やループ構造による構造化，オブジェクト指向化，コンポーネント化と進んできた，「ソフトウェア構成の進歩過程の新しい段階」だといえます．ここで重要なことは，これまでの進歩の背後には，コンパイラ技術や実行時サポート技術の進歩があったことです．こう考えると，新たなソフトウェア構築技法であるSOAを実現するためには，新たな実行時サポート技術が必要だということが分かるでしょう．それはどのような技術でしょうか？

前述のとおり，SOAではネットワークによって疎に結合したコンポーネントサービスが位置透過的に配備されます．これを実現するためには，コンポーネントサービス間の安全な通信のためのセキュリティ技術，コンポーネントを動的に計算機上で配備実行する資源管理技術，コンポーネント間で情報を共有する情報技術が必要になります．

これらの要素技術は，グリッド技術の提供する要素技術とほとんど同じです．このため，SOAを実現する実行時サポート技術として，グリッド技術が期待されているのです．

当初のグリッドは科学技術計算を指向しており，SOAのメインフィールドである，ビジネス系のシステムとはあまり縁がありませんでした．ですからビジネス用途中心のSOAを支える技術としてグリッド技術が取り上げられることを意外に思われる方もおられるでしょう．しかし，すでに見たとおり，純粋に技術的な観点からは，両者の要請にはほとんど差がないのです．

図 -6　SOA とグリッドと Web サービス

グリッドと Web サービス

さて，SOA は Web サービスで構築されること，SOA はグリッド技術で支えられることを見てきました．ではグリッドと Web サービスはどのような関係なのでしょうか．実は，現在ではグリッドも Web サービスで構築されているのです．三者の関係を図示してみると，**図 -6** のようになります．Web サービスがグリッドを支え，そのグリッドと Web サービスによって SOA が支えられているのです．

以下でグリッドと Web サービスの関係を詳しく見てみましょう．

複数の管理主体が参加することを前提とするグリッドにおいて，モジュール間の相互運用性の確保は最重要課題の一つです．相互運用性を実現するためには，モジュール間のプロトコルを明確なかたちで標準化しなければなりません．グリッドのプロトコルのベースとなるフレームワークには下記のようにいくつかの要件があります．これらすべてを満たすものとして Web サービスが用いられているのです．

- **インタフェースを明示的に記述できる機構を持つこと**
  グリッドでは，定義されたインタフェースを複数の実装者が独自に実装することを想定しなければなりません．このためには，誰が読んでも誤解のないかたちでインタフェースを記述できる方法が必要です．Web サービスをフレームワークとして用いることで，強力な記述力を持つ WSDL（Web Service Description Language）を用いてインタフェースを記述することができます．
- **インタフェースを拡張性に富むかたちで定義できること**
  現在のように，計算処理の対象が高速に変遷する環境下では，1 つのインタフェース定義をずっと使い続けることは現実的ではありません．特にグリッドのように動的に変化する環境においては，インタフェースそのものを変化させる必要があります．このためには，インタフェースを定義する際に拡張性を考慮する必要があります．インタフェース記述に Web サービスの WSDL を用いることで，拡張性を持つ定義を行うことが比較的容易になります．
- **ツール類が提供されていること**
  定義されたインタフェースを実際のソフトウェアとして実装する際，ゼロから実装するのは非常に大変です．実装を支援するツール群が，さまざまな言語に対して整備されている必要があります．Web サービスに対しては，Java や C#，C++，Perl など，多くの主要な言語でツールが提供されています．特に，Java や C# では，ソースコードにわずかな変更を加えるだけで，通常のオブジェクトを Web サービスとして使用できるようにする，高度なツールサポートが提供されています．Web サービスをグリッドのベースとして使用することで，これらのツールをそのまま利用することができます．

GGF と Web サービス

グリッドの世界で Web サービスが用いられるようになったのは，2001 年に OGSA（Open Grid Service Architecture）が提唱されてからです [3]．OGSA は，Web サービスをベースにコンポーネントプログラミングに必要な機能を追加し，その上にグリッドの構築に必要とされる個々の機能を「グリッドサービス」として実装しようという試みです．当初の OGSA は，Web サービスと似て非なる OGSI（Open Grid Service Infrastructure）という機構を用いることを前提としていましたが，2003 年に方向を修正し，Web サービスそのものを基盤とする WSRF（Web Services Resource Framework：後述）を使用するようになりました．

OGSA はグリッド関連の標準化団体である GGF[☆1]（Global Grid Forum）[5] の旗艦アーキテクチャとして採用され，策定が進められています．OGSA 以外でも，GGF で行われている標準化の多くが Web サービスを基盤としています．

図 -7 に，代表的な Web サービス関連技術標準化団体の OASIS および W3C と，GGF との関係を示します．W3C は，WSDL や SOAP などの Web サービスの基礎的な技術を策定し，OASIS はこれらの基礎技術の上に，ジャンルを特定しない汎用の応用技術を定義しています．

☆1 2006 年 6 月，GGF はビジネスに特化したグリッド標準化団体の EGA (Enterprise Grid Alliance) と合併し，OGF（Open Grid Forum）となりました．

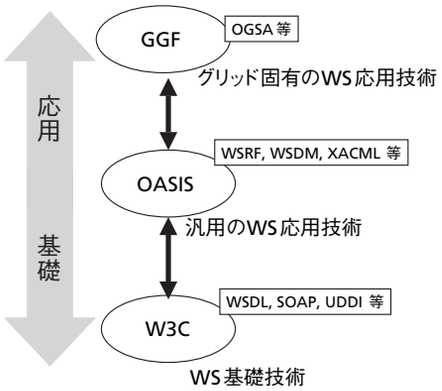

図 -7　GGF と OASIS と W3C の関係

図 -8　OGSA と Globus ツールキットと Web サービスの関係

これに対して GGF は，OASIS, W3C が策定した技術を用いて，グリッド固有の Web サービス応用技術を策定しています．WSRF は，当初 GGF で提案されましたが，標準化は OASIS で行われています．これは WSRF がグリッド固有の技術ではなく，汎用の技術であることの現れです．

WSRF に関しては次回以降の連載で詳しく述べられますが，一言で説明すると Web サービスの世界にリソースというかたちでサービス状態を持ち込んだものです．通常の Web サービスは状態を持ちませんから，通常のコンポーネントの発想を Web サービスに適用することには無理がありました．WSRF はサービスとリソースを対にすることで，サービスに擬似的な状態を与え，コンポーネントとサービスを直接対応付けることを可能にしています．

Globus Toolkit と Web サービス

Globus Toolkit[4] は，米国アルゴンヌ国立研究所と南カリフォルニア大学が主導する Globus プロジェクトで作成されたグリッド環境を構築するためのソフトウェア群です．Globus プロジェクトを主導している Ian Foster は OGSA 提唱者の 1 人でもあります．このことからも分かるように，Globus プロジェクトも Web サービスを重視しており，Globus Toolkit の Version 3 以降は Web サービスをベースに構築されています．Version 3 は前述の OGSI を基盤にしていましたが，Version 4[6] は WSRF を基盤として構築されています．

Globus Toolkit はリモートジョブ起動，情報サービス，データ転送などの基本的なグリッドを構成するための基本的な機能を提供します．また，Globus ツールキットの一部を，WSRF を用いたサービスの構築ツールキットとして使用することもできます．OGSA と Globus ツールキット，WSRF の関係を**図 -8** に示します．

グリッドミドルウェアの動向

グリッドのミドルウェアは続々と Web サービスに移行しつつあります．

ヨーロッパのグリッドミドルウェアの一つである UNICORE は，スーパーコンピュータセンターを接続し，複数のセンタにまたがるワークフロージョブ実行を実現するシステムです．従来は Java ネイティブのオブジェクトエンコーディングをベースとしたプロトコルを採用していました．しかし，UNICORE プロジェクトの後継となる UniGrids プロジェクトで開発された UNICORE/GS では，WSRF に準拠したプロトコルを採用しています．

ウィスコンシン大学の Condor プロジェクトでは，内部プロトコルは従来のバイナリプロトコルを使用していますが，ユーザとのインタラクションの部分では，Web サービスに基づくプロトコルをサポートしています．

国内の大きなグリッドプロジェクトとしては NAREGI（National Research Grid Initiative）とビジネスグリッドプロジェクト（2005 年度で終了）がありますが，両者とも GGF での活動を重視し，Web サービスを多用しています．NAREGI ミドルウェアは，内部コンポーネント間のインタフェースにいたるまで，全面的に Web サービスに基づいて実装されています．ビジネスグリッドは Web サービスを用いたコンポーネントの管理機構である WSDM の策定に大きく貢献しました．WSDM は SOA にも大きくかかわってくるはずです．

Web サービスの問題点

Web サービスは上述のようにグリッド技術の基盤として受け入れられつつありますが，問題点がないわけではありません．グリッドを Web サービスで構築する際の最大の問題点は，Web サービスのオーバヘッドです．Web サービスで使用される XML ベースのプロトコルは，通常のデータプロトコルと比較すると，非常に複雑で多階層になります．このためレイテンシ，スループット両方の側面で実行速度が低下します．

レイテンシで問題になるのは，データを XML に変換する際のコストです．スループットの問題も，XML フォーマットに由来するものです．Web サービスではすべてのデータを可読文字から構成される XML で表現しようとするため，データ量が増えてしまうのです．

このため，現在のグリッドでは，大量のデータを通信する際には，Web サービスではない別のプロトコルが使用されています．Web サービスで通信の下準備を行い，実際のデータのやりとりには他のバイナリ転送プロトコルを用いるのが一般的です．

グリッドと Web サービスの今後

GGF で OGSA が提唱されてからすでに 5 年が過ぎ，ようやく OGSA サービス群の一部の仕様が策定されはじめました．これらは当然 WSRF を使用して規定されています．ツール環境としては，Globus ツールキットが成熟した WSRF 基盤を提供しています．速度面の問題点も，オーバヘッドの解析が進み，対処が行われた結果，ほぼ問題ない程度まで改善されました．

これまでは，グリッドの世界では，提案はされていましたがあまり利用されていなかった Web サービスですが，今後は一挙に普及期に入ることが予想されます．それにともなってグリッドの技術を利用した SOA もさらに普及していくことになるでしょう．

次号の予告

冒頭でも触れましたが，この連載では，グリッドで利用されている技術に少し，軸足をおいて標準技術にアプローチしたいと考えています．次回は，グリッドでの Web サービス利用の基本技術である WSRF と，それを支える WS-Addressing を紹介しようと思っています．

参考文献

1）World Wide Web Consortium, http://www.w3.org/
2）Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/home/index.php
3）Foster, I., Kesselman, C., Nick, M. J. and Tuecke, S. : The Physiology of the Grid, in Grid Computing : Making the Global Infrastructure a Reality, John Wiley & Sons Ltd (2003).
4）http://www.globus.org/
5）http://www.gridforum.org/
6）Foster, F. : Globus Toolkit Version 4 : Software for Service-Oriented Systems, in IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp.2-13 (2005).

（平成 18 年 8 月 1 日受付）

column

EcoシステムとしてのWeb

なぜ，Web のサーバも Web のクライアントも，その後のシステムの設計思想にこれほど大きな影響力を発揮しているのでしょうか？ それには，技術的というよりは，歴史的な理由があると思います．1990 年代の半ば，我々の目の前で進行したのは，本来は，インターネットの中のアプリケーションの一つにすぎない Web のサービスが，インターネットの中で卓越した地位を確立し，インターネットそのものの代名詞となって急速に拡大するという，歴史的な現象だったのです．

こうして無数の Web ブラウザと多数の Web サーバが存在し，通信プロトコルとしては HTTP が最も一般的である世界が生まれました．この世界は，新しいネットワーク技術に対して，先在する Eco システム＝環境として機能します．少し強い言い方をすれば，新しい技術は，まずこの環境に適応しないと，うまくは生き残れないのです．筆者は，AJAX や Web2.0 系の技術は，基本的には，こうした Web/HTTP の Eco システムとしての強い拘束力のなかから生まれたものだと考えています．