簡単になったWebアプリケーションの開発

普及誌編集委員会　編

1. はじめに

2003年度の経営情報学会全国大会の発表申し込みおよび予稿論文提出はすべてインターネットを介して行われました。このようにインターネット上で動作するアプリケーションを総称して**Webアプリケーション**と呼びます。現在、多くのアプリケーションがWebアプリケーションとして開発されています。

もし、15年前に同様のシステムを作成するとしたら、どのようになっていたでしょうか。おそらく、専用線を用意し、多額のソフトウェア開発費用がかかることでしょう。しかし、インターネットとフリーソフトウェアが発達した現在、このようなWebアプリケーションを開発するコストは信じられないくらいに低下しています。たとえば、前述した経営情報学会全国大会で利用した発表申し込み・予稿論文提出のシステムの基本部分の開発期間はわずか20時間程度です。

本稿では、これからのソフトウェア開発の主流となると思われるWebアプリケーションの仕組みについてソフトウェア開発の知識がない方でも分かるように解説したいと思います。

2. Webアプリケーションとは

まず、Webアプリケーションについて定義しておきます。大雑把に言えば、WebアプリケーションとはWeb(World Wide Web)、すなわちインターネット上で動作するアプリケーションのことを指します。それでは、電車の時刻表を掲載したホームページはWebアプリケーションでしょうか。一般的に言えば、単に電車の時刻表を掲載しただけのホームページはWebアプリケーションとは呼びません。しかし、出発する駅と到着する駅を指定して、その駅の間を走る電車の時刻表を表示するシステムはWebアプリケーションであると言えます。つまり、ある条件を指定したり、指示を与えるとそれに対応したホームページが表示されるシステムをWebアプリケーションと呼びます。Web上での予約システム、列車案内システム、ショッピング、ネットオークションなど、実に多彩なソフトウェアがWebアプリケーションとして構築されています。

Webアプリケーションのもう1つの特徴は、その多くがデータベースを利用していることです。もちろん、データベースを利用することがWebアプリケーションとしての必須の条件ではありませんが、ほとんどのWebアプリケーションはデータベースを利用しています。

3. Webアプリケーションの実行環境

Webアプリケーションを実行するには、それをサービスするサーバーが必要になります。ここで言うサーバーには、ハードウェアとソフトウェアの両方が含まれます。

ハードウェアは、大規模なサービスを行う場合にはかなり高性能のコンピュータが必要になりますが、数千人程度の登録ユーザを対象としたシステムで、かつ同時にアクセスするユーザが数百人程度の場合には、ネットワークの回線に問題がなければ10万円台のパソコンでも十分に機能します。小規模(数十人から100人

程度)の場合には、ラップトップパソコンでも十分に間に合います。

ソフトウェアは代表的なものとして、(1) Apache、(2) Internet Information ServerとActive Server Pagesなどがあります。

Apache[1]は、インターネットのホームページをサービスするサーバーソフトウェアで現在圧倒的なシェアを誇っています。Apacheの歴史は、まさにホームページを記述するためのHTML(ホームページ記述言語)と密接な関係があります。1989年にスイスの欧州粒子物理学研究所 (CERN) がホームページの閲覧機能を開発しましたが、一方アメリカでもNCSA (National Center for Supercomputing Applications)がMosaicと呼ばれるホームページ閲覧機能を開発しました。Apacheは、このNCSAが開発したサーバーにパッチ(プログラムのミスを修正する継接ぎプログラム)を当てていたグループが独立して作成したシステムです。名前のApacheは、A pache(パッチを当てる)ということとアメリカの先住民族のアパッチをもじってつけたそうです。実は、ApacheだけではWebアプリケーションは構築できません。一般にサーブレットエンジンと呼ばれるソフトウェアが必要になります。代表的なソフトウェアにTomcatと呼ばれるものがあります。Tomcat[2]は、The Jakarta ProjectによってJavaベースのWebアプリケーション用のサーバーソフトウェアです。特記すべきことは、これらのサーバーソフトウェアがフリーソフトウェアとして提供されているという点です。このことによって、非常に低コストでサーバー環境を構築することが出来るわけです。

一方、Active Server Pages (ASP) はMicrosoft社のWebサーバーであるInternet Information Server 3.0 以上に組み込まれているソフトウェアです。多くの方にとって馴染みのあるWindows環境での設定が可能になっています。

4. ホームページの制約

ホームページは、HTML(Hyper Text Markup Language)という特殊な言語によって記述されています。HTMLはタグと呼ばれる命令を文章中に埋め込むことにより表示方法を指示します。

Internet Explorerのようなブラウザは、URLに指定されたWebサーバーを探し出し、そのWebサーバーに対してHTMLファイルのダウンロードを要求します。Webサーバーはブラウザからのリクエストに応じて、対応するHTMLファイルをブラウザに送ります。そして、ブラウザは送られてきたHTMLファイルをHTMLのタグの指示に従って表示します。

しかし、HTMLファイルで作成した場合の最大の欠点は常に同じ情報しか表示できないという点です。もちろん、電車の時刻表などのように固定的な情報を発信する場合には便利なのですが、個人個人のデータを個別にカスタマイズして表示することができません。

そこで、CGI(Common Gateway Interface)という方法が考え出されました。CGIとは、ブラウザからデータを受け取り、それをサーバ側のプログラムで加工し、新たにHTMLファイルを生成し、それをクライアント側に送ることによって、個人別のデータを個別に表示することができるようにする仕組みのことです。つまり、動的にHTMLファイルを生成してクライアントに送り返すのです。そうすると、クライアントでは各ユーザに対して個別にカスタマイズされたホームページを見ることができるというわけです。ただし、ユーザはそのことにはまったく気づかず、通常のアプリケーションを使っているように感じるだけです。

ところで、CGIは、1つのリクエスト(処理要求)ごとに1つのプロセスを割り当てます。プロセスとは、プログラムの動作の単位ですが、プロセスを起動するにはかなりのコストがかかります。Webアプリケーションは、同時に多くのユーザからリクエストがくる可能性があります。その点では、CGIではかなり効率が悪くなります。

そこで、CGIに変わる方法として、Javaのサーブレットや前述のActive Server Pages (ASP)などが提案され実用化されました。

5. Javaのサーブレット

Javaは、1995年にサンマイクロシステムズによって発表されたプログラミング言語ですが、発表当初はアプレットと呼ばれるホームページの中で動作するプログラムに注目が集まっていました。しかし、現在はWebアプリケーション記述言語として多くのユーザによって利用されています。JavaでWebアプリケーションを開発する場合には、サーブレットと呼ばれる機能を用います。ただし、サーブレットを起動するには、サーバー側に前述のTomcat(サーブレットエンジン)を動作させていることが前提条件となります。サーブレットは、ブラウザからのリクエストに対して1つのスレッドを起動します。**スレッド**は、複数のスレッド間でメモリなどを共有でき、またプロセスに比べてより高速に起動できます。その点でCGIよりもかなり効率が良くなります。

この点を除けば、サーブレットの基本的な考え方はCGIと同じです。ユーザからのリクエストに応じてHTMLファイルを動的に生成し、それをユーザのブラウザに送り返します。すると、ユーザのブラウザの画面には、個別にカスタマイズされた情報が表示されます。

さて、サーブレットでよく紹介される例がショッピングカートを用いた例です。Web上で買い物をする際に、自分が選択した商品をショッピングカートの中に入れて最後にクレジットカードなどで支払うというWebアプリケーションは多いのですが、実はこれをWebアプリケーションとして実現するには面倒なことが多いのです。というのは、ブラウザを使ってWebアプリケーションを利用しているユーザは、Webアプリケーションと1対1で会話しているように感じますが、Webアプリケーション側から見ると同時に多くのユーザを相手にしているわけです。ここで大きな問題は、Webアプリケーションはそのままではページをまたいで記憶を保持できないということです。たとえば、あるユーザが図1に示すように次へ進むボタンを押して次のページを表示したとします。実はサーバーは、この時点でこのリクエストが誰によるものなのか知ることができないのです。つまり、ブラウザからサーバーに対するリクエストは、1つ1つが独立していてユーザのアクセスの連続性を確認することができないのです。これは、インターネットのhttpプロトコルの仕様によるもので、ブラウザは一つ一つのリクエストを独立したものと見なしているからです。おそらくインターネットの初期にはこのような使われ方をするとは思わなかったのでしょう。

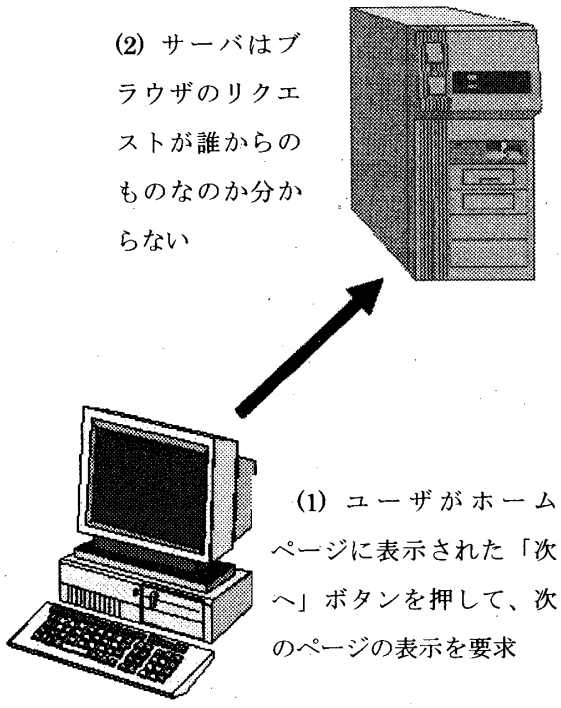

図1 ブラウザからのリクエストは誰か分からない

そこで、同一のユーザの連続するページのアクセスを1つの塊として見なす必要がでてきました。それを**セッション管理**と呼びます。セッション管理を行うにはさまざまな方法があるのですが、最も一般的に行われている方法がクッキーを使う方法です。**クッキー**とは少量のデータをクライアントのコンピュータに保存す

るための仕組みであり、Webサーバーはその少量のデータをクライアントコンピュータに保存し、不特定多数のリクエストの中からユーザを特定しています。クッキーは、名前と値のペアからなりますので、最初にアクセスしてきたユーザのクライアントコンピュータに対して、たとえば名前がWeb-app-01で値が01というクッキーを送付します。このブラウザが次回にアクセスしてきたときは、リクエストと同時にそのクライアントが保存しているクッキーを送り返してくるので、Web-app-01という名前のクッキーがあるかどうか探し、なければ「新規アクセス」あれば「2度目以降のアクセス」であると判断します。値をユーザごとに変えることでWebサーバーはユーザを特定することができます。

このようにすることで、Webアプリケーションは多くのユーザからのリクエストの中から特定ユーザを選び出すことが可能となるのです。同一ユーザの一連のリクエストを**セッション**と呼びます。実は、プログラムの中でクッキーを使ってセッションを管理することもできるのですが、それは面倒なので、セッションを管理する機能が多くの場合備わっていて、手軽にセッション管理ができるようになっています。

たとえば、ユーザ名とパスワードでログインしてショッピングを楽しむというWebアプリケーションの場合、ログイン認証が終了した時点でユーザが特定できるので、セッションを開始します。セッションを開始するということは、新しいクッキーを作成してそれをブラウザに送り込み、次回以降のリクエストからそのユーザを特定できるようにするということです。

6. データベース

Webアプリケーションでは、データベースが大きな役割を演じます。近年、PostgreSQLといったフリーでありながら本格的な利用ができるデータベースやMySQLのようにフリーで使うこともでき、かつ商用サポートも受けられるデータベースなどさまざまなものが利用できるようになっています。PostgreSQLは、カリフォルニア大学バークレー校で研究されていたシステムから派生し、その後、地道に研究開発が続けられたリレーショナルデータベースシステムです。現在では、多くのユーザによって使われています。

Javaをはじめ、最近の多くのプログラミング言語は、データベースを利用するための機能が備わっていて、手軽にデータベースと接続してデータを出し入れできるようになっています。データベースを操作するためにはSQLと呼ばれる記述言語を用いますが、多くの場合、そのSQLをそのままプログラムの中に記述することができます。簡単な例を示して説明します。リレーショナルデータベースは、１つのデータベースの中に複数のテーブルと呼ばれるデータの集合を保持しています。たとえば、学生の情報を保持している「学生情報」というテーブルがあるとします。テーブルは複数のカラムから構成されます。たとえば、「学生情報」というテーブルに、次のカラムが用意されているとします。カラムというのはテーブルの中の個々のデータであり、データベースのアクセスのひとつの単位になります。

「学生情報」テーブル

学籍番号	氏名	得点
Serial (一連番号)	Text (文字列)	Integer (整数型)

図2 データベースのテーブルの例

さて、次のSQL文は学生情報テーブルに格納されたデータの中から「得点が80点以上の学籍番号を抜き出しなさい」という指示になります。

```
SELECT 学生番号 FROM 学生情報 WHERE 得点 >= 80;
```

たとえば、次のSQL文は、同じようにして80点以上のデータを取り出すわけですが、その際に「得点の大きい順に並び替えて」から取り出します。

SELECT * FROM 学籍番号 WHERE 得点 >= 80 ORDER BY 得点;

最後に「ORDER BY 得点」と記述するだけでデータが得点順に並び変わります。また、このSQL文をそのままJavaなどのプログラム言語の中に容易に記述できるようになっていますので、SQLで記述できる処理は実に簡単にプログラムの中に組み入れることが可能となります。また、あるSELECT文によって抽出した結果を別のSELECT文の検索対象とすることができるなど、ほとんどの処理がSQL文で片付いてしまいます。このことにより、プログラミング作業がとても楽になります。

7. JSP(JavaServer Pages)

Webアプリケーションの場合、見栄えもまた重要です。しかし、サーブレットは基本的にJavaのプログラムなので、あまり見栄えのよいHTMLファイルを生成するのには適していません。というのは、サーブレットの仕事は、HTMLファイルを生成する出力命令を記述することなので、デザインに凝ったHTMLを手軽に記述するということができないからです。

そこで、HTMLファイルの中にJavaのプログラムを埋め込むという方法が開発されました。これがJSP(JavaServer Pages)です。

図3のプログラムは、午前中にアクセスすると「現在は午前です」と表示し、午後にアクセスすると「現在は午後です」と表示するホームページを記載したJSPのプログラムです。

このプログラムを見ると、ほとんどHTML記述と変わらないことがお分かりいただけると思います。<% と %>で囲まれた部分にJavaのプログラムを記述し、<%= と %> で囲まれた部分の計算結果がその部分に埋め込まれます。このようにJSPではHTML記述がベースなのでWeb上のデザインに重きを置いたページを作成できるのです。

```
<%@ page contentType="text/html;
charset=Shift_JIS"
import="java.util.*" %>
<head>
<title>時間帯(午前/午後)のチェック
</title>
</head>

<%
Calendar cal
          = Calendar.getInstance();
String status
          =
(cal.get(Calendar.AM_PM)==Calendar.AM)
                    ?"午前":"午後";
%>

<body bgcolor="Ivory">

時間帯をチェックします。
現在は<%= status %>です。

<br><br>
</body>
</html>
```

図3 JSPプログラムの例

8. 実例:大会発表登録・論文受取システムの概要

それでは、今回、青学大会と函館大会で用いた登録・論文受け取りシステムの概要について説明します。このシステムは、JavaのサーブレットとJSPを用いて作成されており、データベースにはPostgreSQLを用いました。サーバーはLinuxでApache、Tomcatを動作させています。

このシステムは、管理者用の部分も含めると8個のHTMLファイル、3個のサーブレット、33個のJSPから構成されます。1つ1つのファイルは200行から300行程度の小さなものになっています。

このようなシステムの開発では、データベースの構成とユーザ画面のレイアウト設計が終われば、あとは単純なプログラミングで実際の開発は終わってしまうので比較的短時間で開発が終わります。多少、時間がかかる部分がユーザの誤操作に関する設計と実装の部分です。しかし、この点はWebアプリケーションに限った話ではありません。

青学大会で初めてこのシステムを使用したわけですが、やはり最初ということもあって使い方の問い合わ

せやアップロードできないという問い合わせが数十件ありました。中には、アップロードできない状況を詳細にレポートしてくれた人などがいてシステムの改善に役立てることができました。しかし、函館大会では、問い合わせの件数が10分の1以下に激変しました。これは、おそらくこのようなシステムに対する会員の方の習熟度が上がったからであると考えられます。

このシステムを使って、青学大会で約120論文、函館大会で約110論文を受け取りました。たいした金額にはなりませんが、郵送代が不要になっただけではなく、電子媒体で受け取ることにより、今回の函館大会で作成して頂いたような大会予稿集のCD-ROMの作成が可能になったわけです。また、大会の受付の登録状況がリアルタイムで大会委員に通知されるため、大会委員が参加者を増やすための戦略強化に繋がると考えられます。

なお、このようなシステムではセキュリティが重要です。通信は、SSL(Secure Socket Layer)を用いて暗号通信を行っています。また、函館大会からはユーザがアップロードした予稿ファイルを確認のためにダウンロードできるようにいたしましたが、その際、他の人のファイルをダウンロードできないようにいたしました。もちろん、大会当日になればCD-ROMが配布されるため、大会参加者はそのファイルを入手できるわけですが、大会前に執筆者以外の方が他人の予稿ファイルを入手できるのはやはりまずいわけです。

9. 終わりに

本稿では、JavaによるWebアプリケーション作成の概要と大会受付システムという事例を説明してきました。もちろん、ここで紹介した内容はWebアプリケーション構築の基本的な部分を説明しただけであり、本格的な大規模なシステムの構築とはかなり異なる部分があることをお断りしておきます。

しかし、いずれにせよ、インターネットという通信のインフラが整備され、またサーバープログラムを書くためのソフトウェアも著しく進歩したことにより、Webアプリケーションの構築が非常に簡単になったことは事実です。ところが、そうは言っても、サーバーを運用管理したり、Javaのプログラムを開発したりするということは、コンピュータの専門家でない方にとってはまだまだ敷居が高いということもまた事実です。しかし、これからますますソフトウェア技術が発達し、情報インフラが整備されれば、このようなシステム開発が飛躍的に単純化され、いわゆるエンドユーザコンピューティングが現実のものとなる可能性が高くなるものと期待できるのではないかと考えています。

参考文献

[1] Apacheについて　http://www.apache.jp/

[2] Tomcatについて

http://jakarta.apache.org/tomcat/

（普及誌編集委員会 編集長 内田　智史）