B-045

外部データと連携して自動更新する要素を含むWeb画面の実現
Development of WWW pages containing visual elements which update automatically according to data

大崎 雅代† 寺岡 照彦†
Masayo Osaki Teruhiko Teraoka

1. はじめに

インタネットの普及とともに、様々な産業分野においてWebを利用したシステムが増加している。Webシステムの中には、防災情報システムや道路交通情報システムのように、河川の水位や雨量、道路の渋滞情報など、刻々と変化する情報を提供するシステムが存在する。それらのWeb画面には、水位を表す文字列や雨量の推移を表すグラフ、交通量を表す道路地図が含まれ、Webブラウザの更新ボタンを押すなどの操作を行うことなしに、自動的にWeb画面を更新させたいという要求がある。水位などのデータはWebサーバ側で保持されているため、Webブラウザ上に表示された内容を更新するためには、Webサーバから最新情報を取得し、画面更新する手続きが必要となる。また、このようなWeb画面を簡単に作成したい、デザイン性を高めたい、という要求がある。

本稿では、Web画面全体を再読み込みすることなく、Web画面の中でデータと連携した部分のみを表示更新する部分更新手法と、それを利用して自動更新を行うWeb画面の実現について述べる。

2. Web画面の自動更新

インタネットを利用したWebシステムでは、その通信プロトコルがHTTPに制限される場合が多い。HTTPを利用した通信でサーバに存在するデータの変更を知るには、サーバプッシュとクライアントプルと呼ばれる２つの方式がある[1]。

(A) サーバプッシュ方式
サーバ側からクライアントに対してデータを送信する方式。データ変更のタイミングでクライアントに通知可能だが、クライアントとサーバが接続し続ける必要がある、サーバ側で特殊な実装が必要、対応Webブラウザが限定される、などの課題がある。

(B) クライアントプル方式
クライアント側からサーバに対してデータを要求する方式。データの変更の有無に係らず、定期的にサーバに対して問い合わせする必要があるが、Web画面の再読み込みを行うのみのため、実現が容易。

また、Web画面をWebサーバの保持するデータに応じて自動更新するには、以下の方法がある。

(i) Web画面の再読み込み
HTMLのMetaタグのRefresh属性やスクリプトを利用して、定期的にWebブラウザからWeb画面を再読み込みする方法。実現が容易で、市販オーサリングツールを用いて簡単にWeb画面を作成できる。

(ii) 埋め込み型コンテンツの利用
通信処理や表示更新処理を記述した、Webブラウザ上で実行可能なAppletやActive-Xなどの専用モジュールを作成する方法。自由度が高い反面、モジュール作成の手間がかかる、汎用性が低い、Appletの場合は起動・実行に時間がかかる、などの課題がある。

ここでは、実現性の容易さ、市販オーサリングツールとの親和性、対応Webブラウザを考慮して、通信には(B)のクライアントプル方式、Web画面の自動更新には(i)のWeb画面の再読み込みを採用することとした。

3. Web画面の部分更新手法

3.1 Web画面の部分更新手法

Web画面を更新する単純な方法は、ユーザが閲覧中のWeb画面自体を再読み込みすることである。しかしこの方法には以下の問題点がある。

- クライアントとサーバ間の通信量が多い
- データとの連携度を高めるために再読み込みの頻度をあげると、負荷が高くなる
- 閲覧中のWeb画面を再読み込みすると、スクロールなどのユーザの操作が取り消される
- 画面全体を再描画するため、ちらつきが発生する

これらの問題を解決するため、HTMLのフレームとスクリプト機能を利用したWeb画面の部分更新手法を提案する。本手法は、ユーザに見えているWeb画面の再読み込みを行わず、表示領域を持たないWeb画面の再読み込みを行い、スクリプトを利用して表示中のWeb画面の一部分を書き換えるものである。図１に概略を示す。

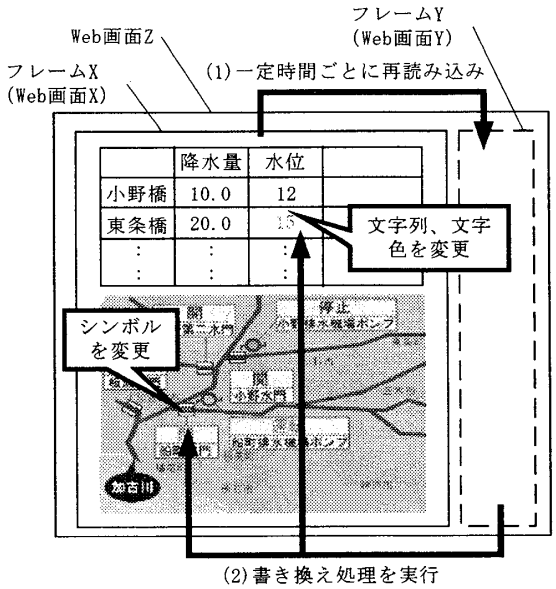

図１：Web画面の自動更新の仕組み

† 三菱電機株式会社 先端技術総合研究所,
Advanced Technology R&D Center, Mitsubishi Electric Corp.

全表示領域を占めるフレーム X と、表示領域を持たないないフレーム Y から構成されるフレームセットからなる Web 画面 Z を用意する。フレーム X にはユーザの閲覧内容を全て含む Web 画面 X、フレーム Y にはスクリプトを実行する Web 画面 Y が指定されている。

実行時には、ユーザは Web サーバに Web 画面 Z を要求する。Web 画面 X のスクリプトで実行されるタイマにより、一定時間ごとに Web 画面 Y が再読み込みされる（図１の(1)）。Web 画面 Y はロードされると Web 画面 X の一部分を書き換えるスクリプトを実行する（図１の(2)）。これにより、ユーザが閲覧中の Web 画面は、画面全体を再読み込みすることなく自動更新される。スクリプトのみが記述された Web 画面 Y を再読み込みするため、通信量を少なくし、再読み込みの頻度をあげた場合においても負荷を抑えることができる。また、更新の際のユーザ操作の中断や画面のちらつきをなくすことができる。

3.2 Web 画面の作成

自動更新する Web 画面の作成方法を述べる。

(1) ベースとなる Web 画面の作成

市販オーサリングツールを用いて Web 画面のレイアウトを HTML 形式、図的情報を SVG[2]形式で作成する。このとき、データと連携したい要素には、識別子として id 属性に一意の値を指定しておく。

例１：HTML の文字要素の場合

```
<var id="v1"> *** </var>
```

例２：SVG の図形要素の場合

```
<rect id="rect1" x="10" y="10" width="100"
 height="100" stroke="black" fill="white" />
```

(2) データ連携ロジックの作成

(1)で識別子を指定した要素とデータとの連携ロジックを記述する。以下にロジックの記述イメージを示す。

例１：文字要素 v1 の値にデータ data1 の値を設定

```
${v1}.value = ${data1};
```

例２：データ data1 の値が 50 より大きければ文字要素 v1 の文字色を赤、そうでなければ緑に設定

```
if ( ${data1} >= 50 ) then
   ${v1}.style = "color:red;";
else
   ${v1}.style = "color:green;";
```

(3) 実行用 Web 画面の生成

(1)(2)で作成した Web 画面とデータ連携ロジックを元に、実行用 Web 画面を生成する。生成される Web 画面 Z、Web 画面 Y の例をそれぞれ図２、図３に示す。図３は(1)(2)の HTML の例において、データ data1 の値が 80 の場合に生成されるスクリプトである。

```
<frameset rows="*" cols="100%,*">
   <frame name="frameX" src="x.jsp">
   <frame name="frameY" src="y.jsp">
</frameset>
```

図２：Web 画面 Z の例

```
<head>
<script language="JavaScript"><!--
function set() {
  parent.frameX.setData('v1', '80');
  parent.frameX.setAttribute('v1', 'style', 'color:red;');
}
// --></script>
</head>
<body onload="set()"/>
```

フレーム X の Web 画面 X の文字要素 v1 の値を 80、文字色を赤に設定するスクリプト

図３：Web 画面 Y の例

3.3 適用例

図４はビルの入退室管理システムに適用した画面例である。Web サーバと接続した監視サーバが保持するデータに応じて、各部屋のドアの開閉状態や在室者の有無が変更される。本 Web 画面の図的情報は Microsoft Visio2003®で作成、SVG 形式で保存した。作成した SVG 内の図形を更新要素として、ドアや在室者の表示パターンをデータ連携ロジックに記述した。

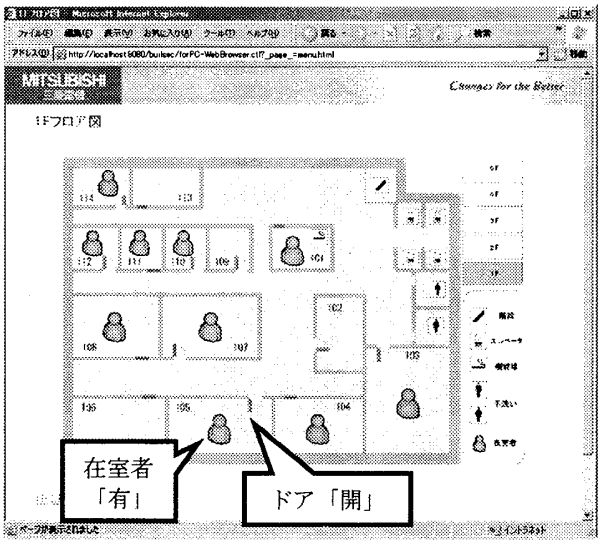

図４：ビルの入退室管理システムの画面例

4. おわりに

本稿では、Web 画面の中でデータと連携した部分のみを表示更新する部分更新手法と、それを利用して自動更新を行う Web 画面の実現について述べた。表示領域を持たない Web 画面を再読み込みするため、通信効率の向上、および、更新の際のユーザ操作の中断や画面のちらつきをなくすことができた。また、ベースとなる Web 画面の作成に市販オーサリングツールを利用できるため、自動更新する Web 画面を簡単に作成できるようになった。

今後は、実システムへの適用を行うとともに、データ連携ロジックの作成支援環境を開発していく予定である。

参考文献

[1] Chuck Musciano, Bill Kennedy: HTML&XHTML 第５版, オライリー・ジャパン (2003)

[2] W3C Scalable Vector Graphics: http://www.w3.org/Graphics/SVG/