カジュアルプログラミングの奨め

会津大学短期大学部
名誉教授
高田　容士夫

カジュアルプログラミングの奨め

高田　容士夫

平成 24 年 1 月 10 日受付

【要旨】

カジュアルプログラミングというプログラミング手法が可能となったことが示される。そのための道具としての Script 言語、表現の手段としてのブラウザ、ブラウザの新しい規格としての HTML5 の役割が示され、いくつかの例が紹介される。

An Introduction to a Casual Use of Programming

【Resume】

It is shown that a casual use of programming technique is available now, using scripting language Ruby and Web browser_scripting language JavaScript and HTML5　tecnique. Several simple examples of text data transformation and some simple web applications are shown．

１．はじめに

近年のコンピュータをめぐる状況の変化は著しい。それは、コンピュータの処理速度の向上によってもたらされるコンピュータ本体の機能の向上に加え、「ユビキタスコンピューティング」とか「どこでもコンピュータ」という言葉に象徴されているコンピュータの遍在化に伴うコンピュータの普遍化は年々進みつつある。コンピュータが現代社会の中で不可欠な構成要素、基盤となっていることを疑う人はいないであろう。このことはそれぞれの人の毎日の暮らしの中でも嫌というほど感じさせられる。携帯電話は言うに及ばず、あらゆる家電製品、家庭内電気装置の中にコンピュータが組み込まれ、われわれは意識するにしろ意識しないにせよそれを通してしか器具を操作できないといっても言い過ぎではないまでになってきている。中でもインターネットは、日常生活の中で必要な様々な情報を得る手段として不可欠なものとなってきている。また、このインターネットへの接続するための道具もパーソナルコンピュータから携帯電話、タブレット型 PC とその種類も豊富となり、それらの上で動くソフトウエアも多種多様なものとなり、Web を使って得ることの出来る情報の種類・量も年々増え続けている。

コンピュータと私たちとの関係が日常化するに伴い取り扱うデジタル情報の量も増大し、情報の種類も多種類なものとなってきている。したがって、これらの情報を利用するときその情報は自分の目的に最も適した内容、形式のものである必要があり、そのためには得られたデータを加工する必要にせまられることが少なくない。このようなデータの加工は、もちろん既成のソフトウエアを通して行うことができることも少なくない。しかし、データを加工する目的は各人各様であり必ずしも既成のソフトウエアの中では十分に達成することができないことも間々存在する。たとえば、近年 Web 上で得られる文字データがブラウザでは同じように見えても PDF 形式、HTML 形式、XML 形式など一様でなく、一つのデータの中から必要な部分だけを抽出し、自分の目的に合った形式に変換する必要があることも少なくない。扱うデータの量が膨大なものであるときには、カットアンドペイストのような素朴な方法では実用的ではない。こんなとき、コンピュータの利用方法の一つとして、自家製のプログラミングを使うという手段がある。

プログラミングのために不可欠なプログラミング言語をめぐる状況も近年大きく変化しつつある。実行環境を選ばないプログラミング可能という特性を持つ Java の普及とともに Script 言語と呼ばれるインタープリター型言語の目覚ましい発展がある。これは、コンピュータの処理速度の向上に伴いインタープリター言語による処理が充分実用になるようになったという事情によるところが大きい。このような Script 言語として有名なものとして Perl、PHP、Python、Ruby などがあげられる。

また、インターネット上のデータ表示ソフトとしてのブラウザをめぐる状況も HTML5 対応が次第に進行し、AJAX というWeb Application 開発の新しい手法が広く受け入れられて行く中で、そこで用いられるブラウザ上の Script 言語としての JavaScript とその舞台としての DOM（Document Object Model）の設定は、ブラウザを決まったデータ表示のための道具という位置づけからデータ加工のための環境へと変化させていったということが出来る。このことは、プログラミング環境としてブラウザがこれまで以上の存在へと成長しつつあるとも言え、プログラミング環境の更なる充実・発展の証しであるとも言えよう。

ところで、『プログラミング』というと学生時代に情報処理教育のなかで経験したことを思い出し、「今更プログラミングなど」と考える人も少なくないであろう。あるいは、「プログラミングは、専門家がするもので自分とは関係のないものだ」と考える人も少なくないかもしれない。しかし、先に述べたようなプログラミングをめぐる環境の変化が起きている状況の下では、専門家による汎用性のあるプログラミングとは別に非専門家による日常業務のためのプログラミングが新しいプログラミング作法として可能となり、重要になりつつあると考えられる。もちろんこのようなプログラミング作法・環境は、表計算ソフトや、データベースソフトにおけるマクロプログラミングとしてこれまでも実行され、利用されてきたものであるが、これらよりほんの少しだけ汎用性のあるプログ

ラミング作法が現在のプログラミング環境の下ではありうるのではと考え、ここでは、これを『カジュアルプログラミング』と名付けることとした。以下の小節では、『テキストデータの処理プログラミング』、『個人使用のためのWeb Application』のいくつかの試みを紹介することとしたい。

しかしながら、このような目的のプログラミングの環境としてブラウザや Script 言語、ブラウザの現状に問題点がないわけではない。それは、これらの環境の特徴とも言えることであるが、これらの技術、環境の不安定さと言っても良いほどの仕様の変化、進歩の速さである。しかし、だからこそ安定性が強く求められる商用プログラムと違って、かなり気易くメンテナンスができる個人レベルのカジュアルユースのためのプログラミング環境としては適しているとも言えるのではないだろうか。

２．テキストデータのスクリプト言語による処理

私たちは様々な情報をインターネットを通して得ることができるようになった。しかし、この情報にはこれまでの紙の上の情報とは異なる性質がいくつかある。

１．データへのアクセスの手軽さ
２．データ加工の容易さ
３．情報の存在の不安定さ
４．情報の存在形式の多様さ

などがある。１．２．はインターネット情報の優位性と考えられているが、意外に知られていないのが３．４．の特徴である。インターネット上の情報は、

Simple text
XML data
HTML data
PDF data

などさまざまな形式で提供されている。もちろん情報提供者の立場ではその形式を採用する理由があるのであろうが、利用者の立場では必ずしもその形式が必要な情報を得るのに最も良い形式であるとは言えないことがままある。すなわち、元のデータを自分の必要なデータへと変換したり、必要なデータだけを抽出したりすることが必要となる。このようなとき、Script 言語はその威力を発揮する。Script 言語はそれぞれ特徴があり、歴史があるが、どの言語を使うかは、使い手の好み、趣味に合わせて選択するという以外特に基準となるものは存在しないと言ってよいであろう。ここでは、私が今一番使い易いとと感じている Ruby を使ったいくつかの例を紹介する。

テキストファイルの編集は、最も基本的なものであるが、これは、ほとんどの場合ワープロか表計算ソフトを使えば解決できることも少なくない。しかし、XML ファイルの解析や HTML ファイルの解析をし、必要なデータを取り出すためには、何らかのプログラミング行為が必要となる。例えば、ブラウザ上で

に示されるようなデータがブラウザ上で表示されたときこの中から歌の作者と歌本文だけを取り出したいとする。ソースファイルを見てみたときそれは

```
<html>
<head>
<title>和歌</title>
</head>
<body>
<h1>千載集</h1>
<table border="1">
<tr>
<table width="650" border="1">
<tr>
<td>00001</td><td>
<div align="left">
<a name="i009-001-001">
　　[詞書] 春たちける日よみ侍りける<br>
</a>
</div>
<div class="author" align="right">俊頼 源俊頼朝臣 (074)
```

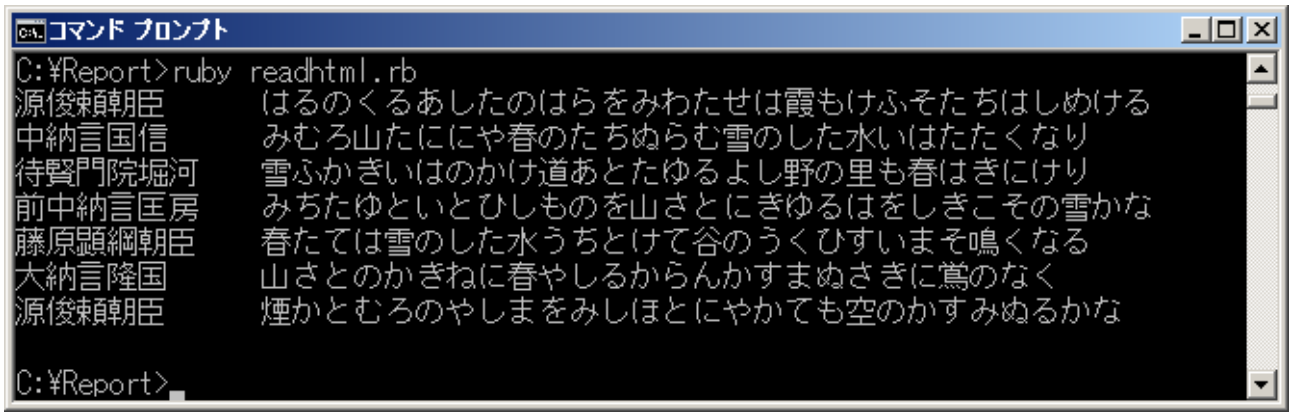
```
          <br>
<br>
</div>
<div  class="poem" align="left">はるのくるあしたのはらをみわたせは霞もけふそたちはしめける<br>
<br>はるのくる－あしたのはらを－みわたせは－かすみもけふそ－たちはしめける<br>
<br>
</div>
<div align="right">
            異同資料句番号:00001</div>
</td>
</tr>
</table>
<br>
</table>
・・・…・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
</body>
</html>
```

となっている。この中から歌の作者と歌の本文データだけを取り出したいとする。これを Ruby で処理させるには、

```
require 'hpricot'
doc=open('poem.html')
AUTHORS=Array.new
POEMS=Array.new
html=Hpricot(doc)
d=html.search("//div[@class='author']")
d.each{|x|
if x.inner_text!="" then
AUTHORS<< x.inner_text.split(' ')[-2]
end
}
d2=html.search("//div[@class='poem']")
d2.each{|x|
POEMS<< x.inner_html.split("<br />")[0]
}
L=AUTHORS.length
(0..L-1).each{|k|
puts AUTHORS[k]+"¥t"+POEMS[k]
```

}

というスクリプト readhtml.rb を書いて、Windows のプロンプト画面で実行してやると、



という形でデータを受け取ることができる。もちろんファイルに書き込むこともできる。これがこのように簡単にできるためには元の HTML ファイルがデータの入れてあるタグの中に class=”author”とか class=”poem”といったデータの内容を示す class contents property を指定してあったためでこれがないような HTML ファイルの場合には、かなりてこずることを覚悟しなければならない。

　これに対して、XML ファイルの場合は、このようなデータの意味づけを最初から意図して作成されているので極めて簡単にデータを抽出することができる。例えば、

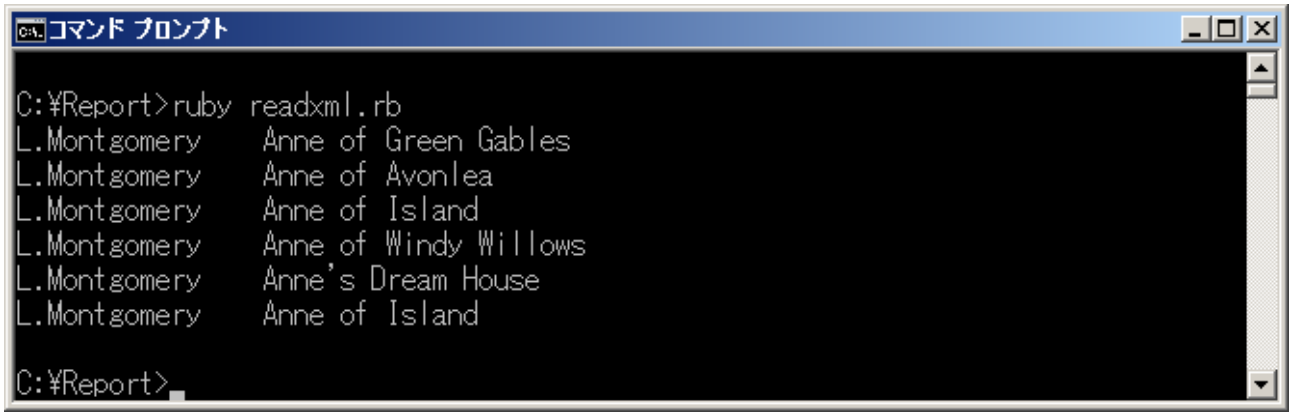
```
<root>
<book>
  <title>Anne of Green Gables</title>
  <author>L.Montgomery</author>
  <price>1000</price>
</book>
<book>
  <title>Anne of Avonlea</title>
  <author>L.Montgomery</author>
  <price>1200</price>
</book>
<book>
  <title>Anne of Island</title>
  <author>L.Montgomery</author>
  <price>1000</price>
</book>
<book>
  <title>Anne of Windy Willows</title>
  <author>L.Montgomery</author>
  <price>950</price>
```

```
</book>
<book>
  <title>Anne's Dream House</title>
  <author>L.Montgomery</author>
  <price>1000</price>
</book>
<book>
  <title>Anne of Island</title>
  <author>L.Montgomery</author>
  <price>1050</price>
</book>
</root>
```

というような書籍データのXMLファイルbooks.xmlがあったとき、ここから書籍名と作者名だけを抽出するには

```
    require "rexml/document"
    require "kconv"
    include REXML

    titles=Array.new()
    authors=Array.new()
     doc = Document.new File.new("books.xml")

    doc.elements.each("*/book/title") do |e|
     titles<<e.text
    end
    doc.elements.each("*/book/author") do |e|
     authors<<e.text
    end
 i=0
while(i<titles.length) do
    puts authors[i]+"¥t"+titles[i]
    i=i+1
end
```

というスクリプトファイルreadxml.rbを作ってやり、先と同様にプロンプト画面でこれを実行すると、

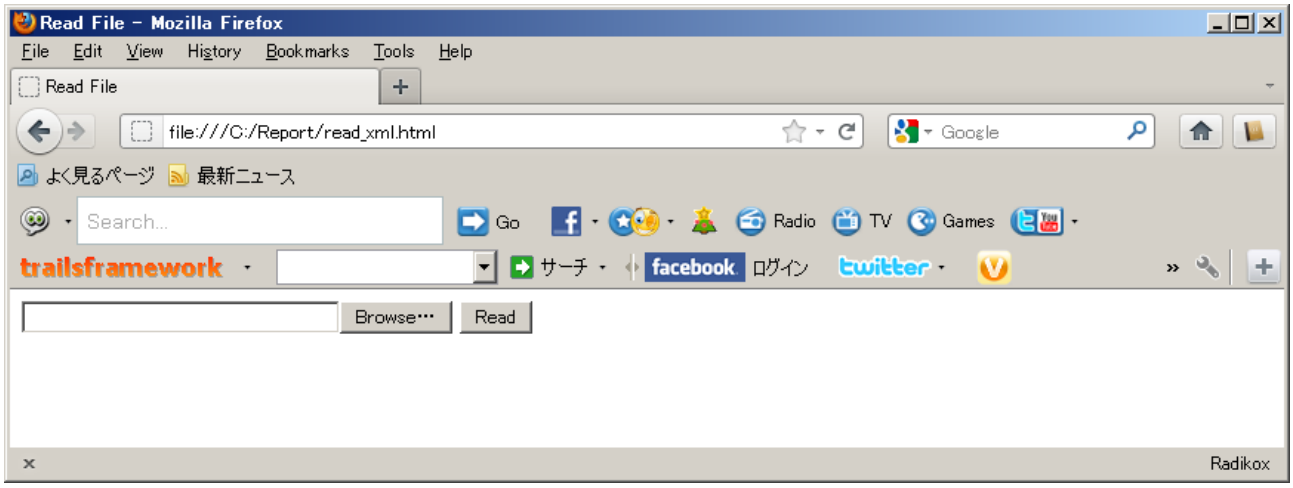


のように同様の結果を得ることができる。このようなXMLファイルの解析は、ブラウザ上で、JavaScriptを使っても実行できる。実際、

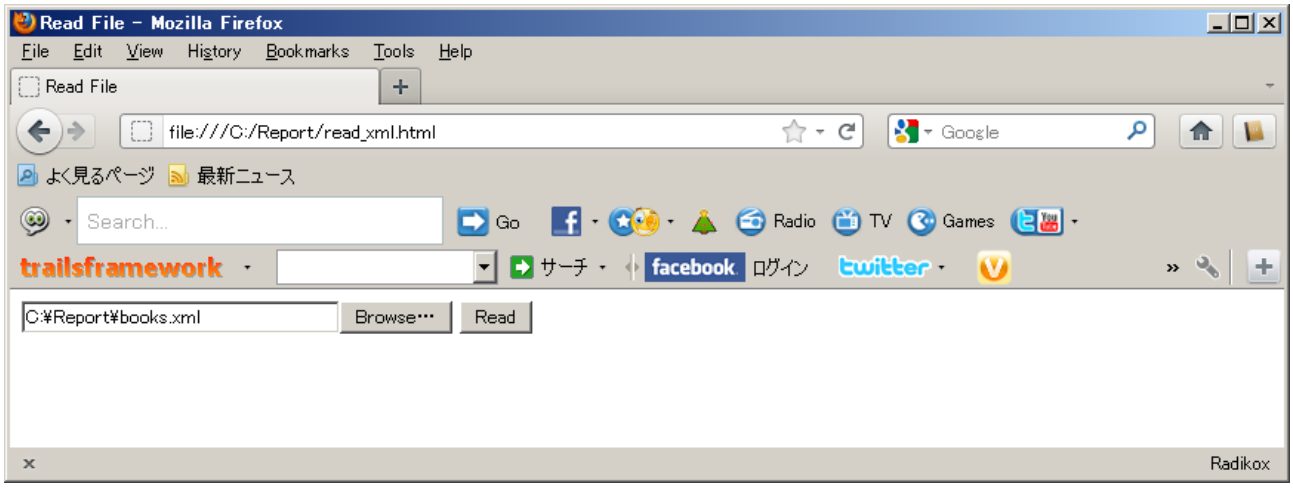
```
<html>
<head>
<title>
Read File
</title>
<script>
function read(){
        var D=document.getElementById('text');
        var F=document.getElementById('file').files[0];
        var reader=new FileReader();
        reader.readAsText(F,'Shift_JIS');
        reader.onload =function(){
        TEXT=reader.result;

      TEXT1=TEXT.split('<book>');
        var data1=[];
        var data2=[];
        var X="<table><tr><th>Title</th><th>Author</th></tr>";
      for(i=1;i<TEXT1.length;i++){
            var n1=TEXT1[i].indexOf('<title>');
            var n2=TEXT1[i].indexOf('</title>');
            var m1=TEXT1[i].indexOf('<author>');
            var m2=TEXT1[i].indexOf('</author>');
              data1[i-1]=TEXT1[i].slice(n1+7,n2);
              data2[i-1]=TEXT1[i].slice(m1+8,m2);
              X=X+"<tr><td>"+data1[i-1]+"</td><td>"+data2[i-1]+"</td></tr>";
```

```
        }
        X+"</table>";
        D.innerHTML=X;
         }
}
</script>
</head><body>
<input id="file" type="file" size=40>
<input value="Read" onclick="read()" type="button">
<div id="text"></div>
</body>
</html>
```

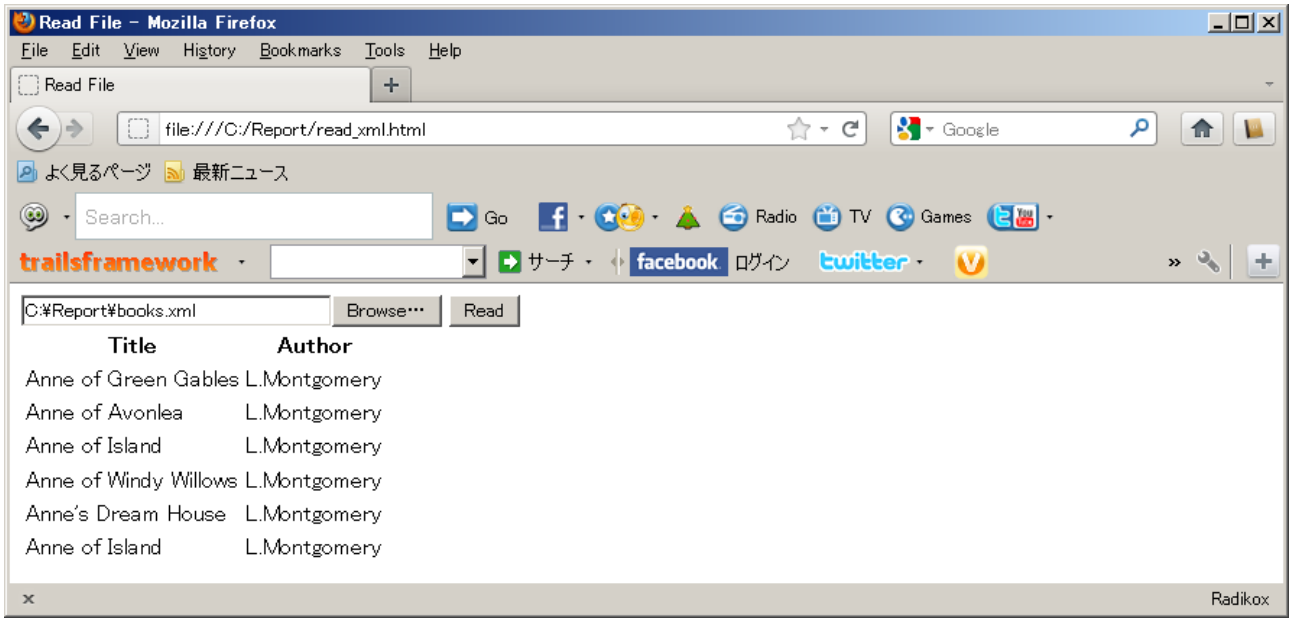
というHTMLファイルを作ってこれを表示すると、



といった画面が表示されるので、Browseボタンを押して、処理したいファイルを選択すると、



という画面になるのでReadボタンをクリックすると、



となってRubyで行ったことをそのまま表示できる。この意味でもWeb上のデータがXMLファイル形式で与えられるとデータの利用が非常に簡単になることが分かる。

3. Web Application

ブラウザをアプリケーションのプラットフォームとして利用する試みは、すでにいろいろ試みられており商用アプリケーションも数多く存在しているが、ここでは、カジュアルプログラミングの名前にふさわしい手軽で簡便なものを紹介したい。

3-1 Interactive Presentation Tool

プレゼンテーションツールにはすでにさまざまなものがあり、いまさらわざわざ個人用のツールを作る意味はないとも考えられるが、プレゼンテーションツールを実際使って見ると多少物足りなく感じることがある。それは、完全に出来上がった原稿をある決まった順序で表示したい時には問題はないが、話す内容、順序を聴衆とのやりとりの中で変更したり、膨らませたりしたいと思う時、つまり聴衆との情報のやりとり、情報の追加を適宜行いたい時である。このような時、聴衆との相互作用をそのまま反映出来るプレゼンテーションツールという意味で、Interactive Presentation Tool とも呼べるツールの必要性を感じることはないだろうか。ブラウザをプレゼンテーションの道具として利用するという考え方は多くの人がすでに実行しているところではあるが、HTML5 で追加されたいくつかの新しい機能を使うとこれまで以上に多機能なプレゼンテーション画面を作ることができる。例えば、campus タグを使ってグラフをその場で入力したデータをもとに作成したり、textEditable プロパティを使って表示データを追加したり、draggable Attribute を使って drag&drop 機能を簡単に表示画面に付加することが出来る、などなど。

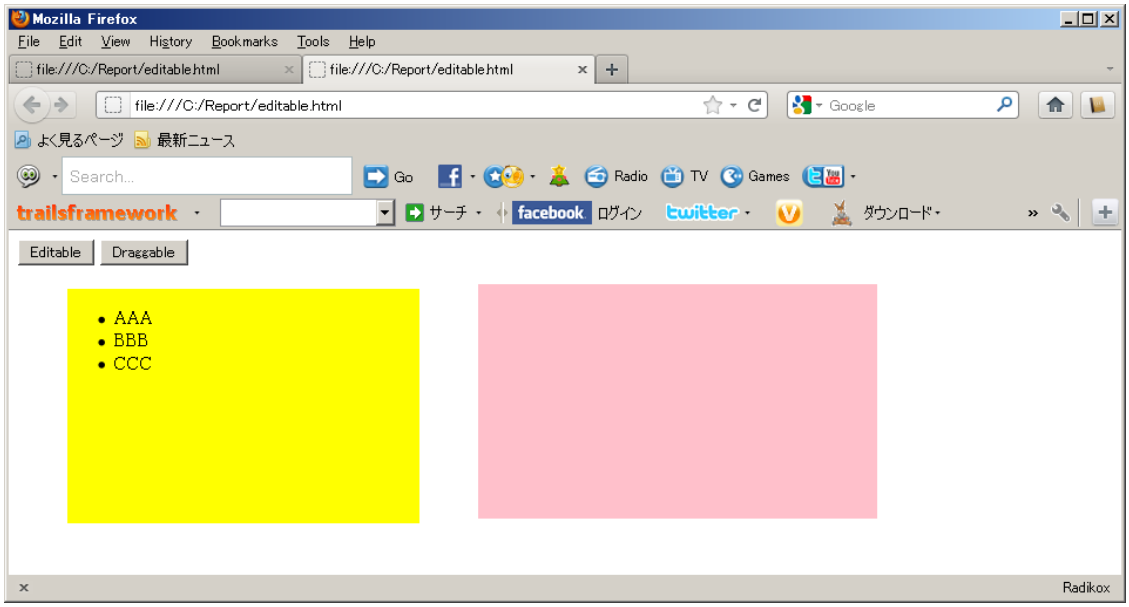
ここでは、あらかじめ用意した項目を順序付けを行う場合、さらに新しい項目を追加し、これらも含めて順序付け項目を作成する場合を考える。以下のようなHTMLファイル（editable.html）を作る。

```
<html>
<head>
<style>
#dragArea{background-color:yellow;position:absolute;top:50px;left:50px;width:300px;height:200px}
#dropArea{background-color:pink;position:absolute;top:30px;left:400px;width:300px;height:200px}

</style>
<script>
function init(){
    var items=document.querySelectorAll("#dragArea li");
    for(var i=0;i<items.length;i++){
        items[i].draggable='true';
        items[i].id=i;
        items[i].addEventListener('dragstart',function(e){
            e.dataTransfer.setData('text',e.target.id);
            },true);
     }

    var drop=document.getElementById('dropArea');
    drop.addEventListener('drop',function(e){
    e.preventDefault();
    var ITEM=e.dataTransfer.getData('text');
    var text1=items[ITEM].firstChild.nodeValue;
    var liTag=document.createElement('li');
    var aText=document.createTextNode(text1);
    liTag.appendChild(aText);
        if(items[ITEM].parentNode){
        items[ITEM].parentNode.removeChild(items[ITEM]);
        }
        else{
            return;
            }
        drop.appendChild(liTag);

    },true);

    drop.addEventListener('dragenter',function(e){
        e.preventDefault();
```

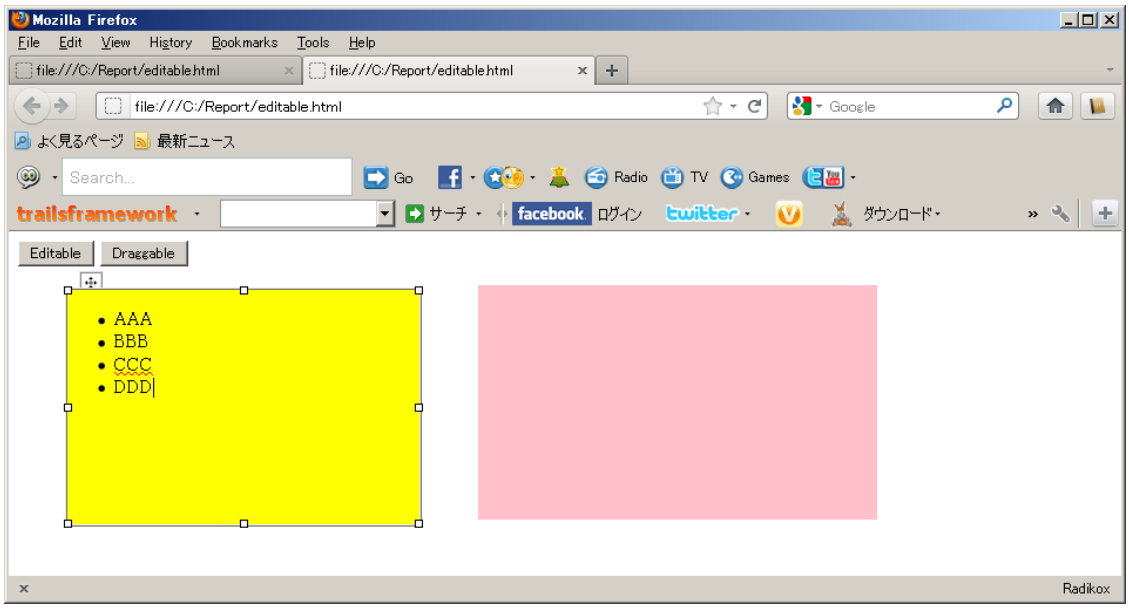
```
        },true);
      drop.addEventListener('dragleave',function(e){
          e.preventDefault();
          },true);
       drop.addEventListener('dragover',function(e){
       e.preventDefault();
       },true);
}
function editable(){
   var DOMAIN=document.getElementById('dragArea');
   DOMAIN.contentEditable='true';
}
function Draggable(){
   var DOMAIN=document.getElementById('dragArea');
   DOMAIN.contentEditable='false';
 var items=document.querySelectorAll("#dragArea li");
 for(var i=0;i<items.length;i++){
        items[i].draggable='true';
        items[i].id=i;
        items[i].addEventListener('dragstart',function(e){
            e.dataTransfer.setData('text',e.target.id);
            },true);
     }
    init();
}
</script>
</head>
<body>

<input type='button' onClick="editable()" value='Editable' />
<input type='button' onClick="Draggable()" value='Draggable' />
<div>
<ol id='dropArea'>
</ol>
</div>
<div  id='dragArea'>
<ul>
<li>AAA</li>
```

```
<li>BBB</li>
<li>CCC</li>
</ul>
</div>
</body>
</html>
```

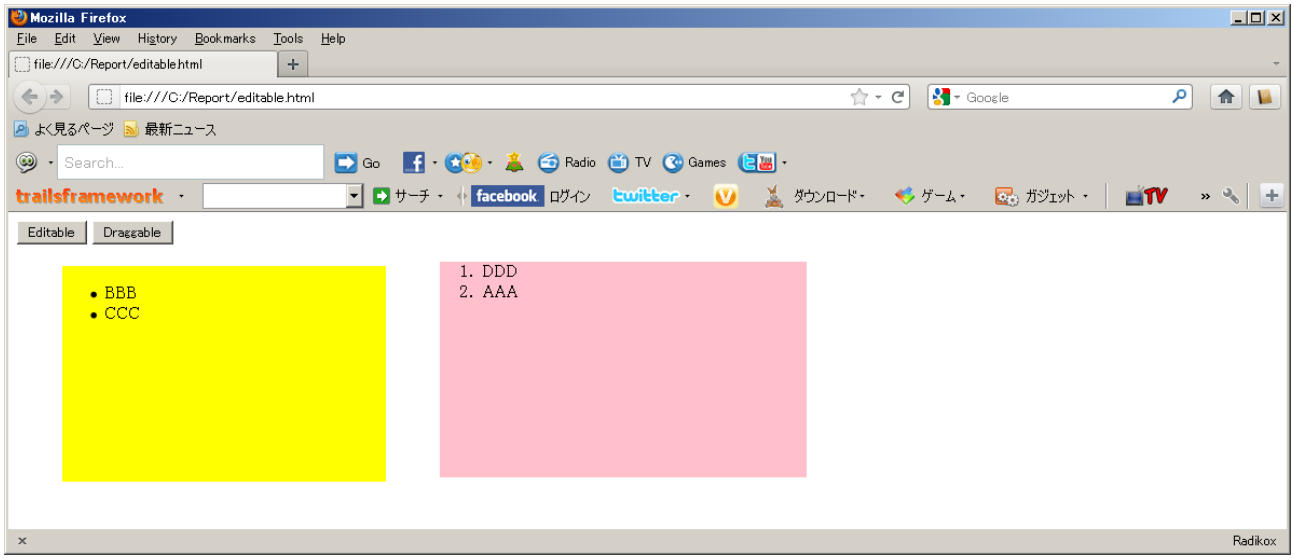
これをブラウザ上に表示すると、次のものとなる。



この左側の黄色の部分にある項目に新たな項目を追加したい、あるいは削除・訂正が行い時には、画面上部にある Editable ボタンをクリックすると、黄色い部分が『編集可能』となり、次の画面へと変化するので新しい項目を追加する。



ここで、上部右側の Draggable ボタンをクリックすると、各項目がドラッグ可能となり、右側のピンクの領域へドラッグすると、



となり、順序付けられた項目群を表示できるということになる。

ここで使った『ContentEditable』というプロパティは、これまでの HTML 文書のあり方とは大きく異なるものであり、色々な応用が可能となるものであると考えられる。

3-2 Local Storage の利用

これまで、Web Application を組み立てる時、多くは Web Server を通して行われ、ブラウザで入力したデータは、CGI を通して処理し、データは Server に保存するという形態のものが主流であった。しかし、個人が利用する Web Server が必ずしも使いたい CGI が使えなかったり、そもそも CGI そのものが許可されていない場合も少なくない。今進められている HTML5 では、Local Storage という機能をブラウザの標準機能として認定されようとしている。これは、ブラウザ上で入力されたデータをクライアントコンピュータ内に保存するという機能

で、これまでもクッキーとして利用されてきた機能を強化したものとなっている。HTML5 の当初の案には local database の構想もあったがそれは廃案されたが、それの代用品とは言えないにしても、ブラウザを表示しているコンピュータに一定の量のデータを保存出来るという仕組みは、Web Application を作成し利用する立場からすると利用価値の有る仕組みである。以下のようなHTML ファイル local_Storage.html を作成する。

```
<html>
<head>
<title>
Local Storage
</title>
<script>
    function store(x,y){
        localStorage.setItem(x,y);
        alert("Saved");
    }
    function show(k){
        var data=localStorage.getItem(k);
    }
    function del_all(){
      localStorage.clear();
    }
    function show_all(){
        var n=localStorage.length;
         data="";
        for(i=n-1;i>=0;i=i-1){
            var key=localStorage.key(i);
            data=data+key+"¥t"+localStorage.getItem(key)+"¥n";
            }
          alert(data);
}
</script>
</head>
<body>
Key:<input type="text" id="key" value=""><br />
Value:<input type="text" id="value" value=""><br />
<input type="button" value="Save"
onClick="store(document.getElementById('key').value,document.getElementById('value').value)"><br />
<input type="button"  value="ShowAll" onClick="show_all()"><br />
<input type="button"  value="Delete All" onClick="del_all()"><br />
```

</body>
</html>

これをブラウザで表示すると

ここで、いくつかのデータを入力した後で、ShowAll ボタンを押すと

とすべてのデータが表示される。

４．おわりに

以上述べてきたように、現在は、普通の人の普通の用途のための普通のプログラミング（カジュアルプログラミング）という行為がそれほどの障壁もなく、とりたてた道具立て、技術の必要もなく行うことの出来る時代・環境となってきている。これを利用しないというのはいささか勿体ない話ではないだろうか。ここで紹介した以外にも Web Application 関連ではスクリプト言語それぞれに対応したフレームワークが存在し、これらを用いると、Web Application そのものをほとんどなんの苦労もなく作り上げることが出来るまでになっている。例えば、Ruby on Rails という Web Application Framework があるが、これを使うと、Web Application で使いたいデータの構造を決定すれば、後は Rails に任せてやれば、Application の雛形が瞬時に出来上がってしまう。これは、カジュアルプログラミングを志す者にとってはありがたい話である。ただし、問題点として、これまで紹介してきたこと全体を通して言えることではあるが、Script 言語、フレームワーク、HTML 基準及びそれのブラウザ対応などの仕様の変化が著しく速いということがある。これはこの分野の開発の速さの表れであるが、利用者にとっては良くもあり不便でもある。一年前に開発したプログラムが全く役に立たないということは稀にしても、か

なりの手直しを必要とすることもままあるということになる。これは、カジュアルプログラミングの手軽さとうらはらの関係にあるとも言えるものであるとも言えよう。

最後にWEB上で公開されているデータは、最近PDF形式で与えられることが多いが、データを利用する者の立場に立つとこれは必ずしも歓迎できる事態では無い。勿論、データを提供する立場からすれば、データを勝手に改竄されないようにするための手段として、ロック付きのPDFファイル形式は望ましいものであることは十分に理解出来ることではある。しかし、例えば、食品栄養分析表のようにそのデータを使ってコンピュータ処理を行うことを目的とする場合にはPDF形式のデータは、歓迎出来るものであるとは言えない。また、多くのデータがそうであるようなHTML形式で与えられる場合においても、HTML形式がブラウザ上でどのように表示されるかということだけに重点がおかれ、データの意味内容を中心にデータを編集したい時にはその扱いに苦労することが多い。HTML形式のデータであっても、先の例で示したように、tagのプロパティをそのデータの意味内容を示すよう定義しておく必要があると考えられる。