プログラミング環境特論

# Ajaxによる Webプログラミング入門

佐藤



## 参考

- 「Ajaxを勉強しよう」
  http://www.openspc2.org/JavaScript/Ajax/Ajax_study/index.html
- Life is beautiful ：Ajaxの本質、「非同期メッセージ型ウエブ・アプリケーション」のススメ
  http://satoshi.blogs.com/life/2005/06/ajax.html



## Ajaxとは

- Ajaxは「Asynchronous JavaScript + XML」
- 非同期通信を利用したJavaScriptとXML、つまりJavaScriptを利用して非同期通信を行いXMLデータでサーバーとのやりとりを行うもの
- RIA (Rich Internet Application：リッチ インターネット アプリケーション)
  - Googleが公開しているGoogle MapsやGmailなどAjaxを利用したサービス



## これまでの流れ

- DHTMLが登場する以前
  - 静的なHTMLページ、つまり画面上に文字や画像がはりついたままでスクリプトなどで自由に動かすことができない
- HTMLデータを自由に操作でき、ページ上のデータをオブジェクトとして扱う、これがDHTML
  - HTMLデータを単なるテキストや画像ではなく、情報を持ったオブジェクトとする=DOM(Document Object Model )
  - JavaScriptでスタイルシートにアクセスすることで行う
  - IE4で導入。バグが多かった。仕様が統一されていない。（これは以前として問題？！）



## 類似の技術

- DHML
- Flash
- Java Applet
  - Appletの中しか扱えない
  - DHML/Ajaxでは、HTML全体を扱う
  - サーバーとの通信（非同期）



## サンプルプログラム

- test1: 簡単な例
- test2: HMLの表示
- test3：XMLの表示
- test4：画像(gif)の表示
- test5: HTMLの書き換え
- test6: HTMLの書き換え
- map-test: google mapの最初のサンプル
- map-test1: ズーム、情報ウインドウ
- map-test1: イベント、マーカー

## サーバーとの非同期通信

- 通信用オブジェクトを作成する
  `httpObj = new XMLHttpRequest()`
  - IEでは、`ActiveXObject("Microsoft.XMLHTTP");`
- 取得したいデータ、送信したいデータをサーバーに送る
  `httpObj.open("GET","data.txt",true);`
  `httpObj.send(null);`
- サーバーからの応答に応じて処理を行う
  `httpObj.onload = displayData;`
  ロードされたときに、呼ばれる関数をセットする。



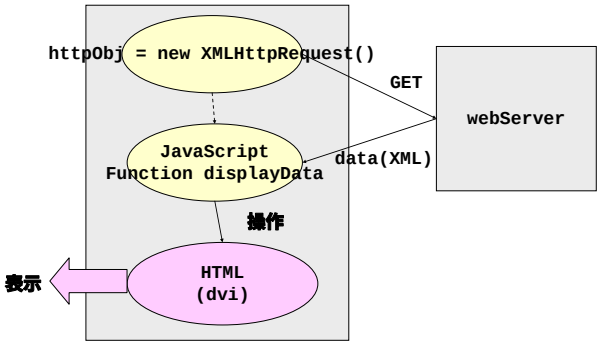
## JavaScriptのセット

- ボタンなどのGUIにJavaScriptの関数をセットし、actionに応じて、起動

```
<form name="myform">
<input type="button" value="read"
  onClick="loadTextFile()"><br>
  ….
</form>
```



## 非同期通信のモデル






## divを使ったHTMLの埋め込み：test2.html

- HTMLソース内にタグをいれておく
  `<div id="resultData"> </div>`
- `document.getElementById(tagId);`タグを探して、それを書き換え
- 従来に比べて、idでＨＴＭＬの要素をアクセスできる

```
function displayData(){
if((httpObj.readyState == 4) &&
   (httpObj.status == 200)) {
   $("resultData").innerHTML = httpObj.responseText;
} else {
   $("resultData").innerHTML = "<b> Loading ... </b>";
}
```



## Mylib.js

```
<script type="text/javascript" src="mylib.js"></script>

// library
function createXMLHttpRequest(cbFunc)
{
 var XMLhttpObj = null;
 XMLhttpObj = new XMLHttpRequest();
 XMLhttpObj.onreadystatechange=cbFunc;
 return XMLhttpObj;
}

function $(tagId)
{
  return document.getElementById(tagId);
}
```



## XMLの表示：test3.html

- responseXMLで、XMLデータを取得する
- それを、DOMとして処理することにより表示

```
xmlData = httpObj.responseXML;
 userListTags = xmlData.getElementsByTagName("user");
 numberListTags = xmlData.getElementsByTagName("number");
 usernameListTags = xmlData.getElementsByTagName("username");
 userLen = userListTags.length;
 resultText = "";
 for(i=0; i<userLen; i++){
 num = numberListTags[i].childNodes[0].nodeValue;
 uname = usernameListTags[i].childNodes[0].nodeValue;
 resultText = resultText + num + " : " + uname + "<br>";
 }
 document.getElementById("resultData").innerHTML = resultText;
```

## 画像の表示：test4.html

- <img src="…" />をつくって、セット
- webBrowserが画像を取得、表示してくれる

- 他の例
  - ExcelのＸＭＬ出力を表示させるなど



## HTMLの書き換え:test5.html

- HMTLの属性を書き換えることができる。
  - 背景色の変更

```
function setBGColor(){
 document.body.bgColor = "#ffffef";
}
```

- タグに加えることも可能

```
function addImage(){
 imgObj = document.createElement("img");
 imgObj.setAttribute("src","mark1.gif");
document.getElementById("subContent").appendChild(imgObj)
  ;
}
```

- スタイルシートccsも書き換えることができる



## HMTLの生成、書き換え：test6.html

- 画像を生成
- 一部のattributeを変える
  - Imgタグのsrc属性

```
function changeContent(){
 document.getElementById("subContent").
    childNodes[1].setAttribute("src","mark1.gif");
}
```



## GoogleMaps

- まず、登録して、キーをもらってこなくてはならない




## ズーム、情報ウインドウ:map-test1.html

- ズームのセットの仕方はいろいろある

```
function newPoint(x,y,z){
   map.centerAndZoom(new GPoint(x,y),z);
}
```

- 情報ウインドウの表示

```
function dispInfo()
{
 map.openInfoWindowHtml(map.getCenterLatLng(),"this
  is Mt. Fuji");
}
```



## マウス、マーカー：map-test2.html

- マウスのイベントの設定
  - GEvent.addListener(map, "click", getMapXY);

- イベントハンドラー

```
function getMapXY()
{
var LatLngObj = map.getCenterLatLng();
document.getElementById("mapX").innerHTML =
   "<b>"+LatLngObj.x+"</b>";
document.getElementById("mapY").innerHTML =
   "<b>"+LatLngObj.y+"</b>";
}
```

- 表示領域

```
<ul>
<li> Longitude : <div id="mapX" /> </li>
<li> Latitude : <div id="mapY" />  </li>
</ul>
```

## マウス、マーカー：map-test2.html

◆ マーカの表示

```
function setMarker()
{
      var LatLngObj = map.getCenterLatLng();
      marker = new GMarker(LatLngObj);
      map.addOverlay(marker);
}
```

◆ 設定

```
<form>
<input type="button" value="set marker" onClick="setMarker()"
  />
</form>
```



## Ajaxの本質、「非同期メッセージ型ウェブ・アプリケーション」のススメ ～Satoshi Nakajimaさんの Blogから～

◆ Googleなどが進めている第二世代のウェブ・アプリケーションのアーキテクチャーの本質は、XHTMLやXMLやJavascriptにあるのではない。その本質は、

（1）アプリケーションの明示的なインストールが必要ない。
（2）サーバーとの通信を非同期に実行することにより、通信遅延によりＵＩをブロックしない。
（3）サーバーとのやり取りは、ＲＰＣではなく、メッセージで行う。
（4）データ・バインディングはサーバー側ではなく、クライアント側で行う。
（5）ＵＩにインテリジェンスがあり、ある程度はサーバーに戻らずにユーザーとやり取りをする。



◆ （1）アプリケーションの明示的なインストールが必要ない。
  – これはある意味、これからのweb中心のアプリでは本質的
  – Web OSなどのながれ

◆ （2）サーバーとの通信を非同期に実行することにより、通信遅延によるＵＩのブロッキングを避ける。
  – 第一世代のウェブ・アプリケーションでは、ハイパーリンクをクリックするたびにユーザーを待たせるため
  – 通信を非同期にして、ネットワーク遅延をユーザーからできるかぎり隠し、ストレスの少ないユーザー・インターフェイスを提供する点が第二世代のウェブ・アプリケーション。



◆ （3）サーバーとのやり取りは、ＲＰＣではなく、メッセージで行う。
  – ＲＰＣはシームレスにプログラムをする環境を提供する。
  – 「プログラマーにシームレスにプログラムを組まれては困るのである」⇒通信を意識しろということ？
  – これには少し疑問？

◆ （4）データ・バインディングはサーバー側ではなく、クライアント側で行う。
  – 第一世代のウェブ・アプリケーションは、データとビューのバインディングをサーバー側で行っていたため、新しいデータが必要になるたびに、UI（HTMLページ）全てを再度サーバーに取りに行き、再構築
  – 第二世代のウェブ・アプリケーションは、まずビューをクライアントに取り込み、次にデータをメッセージの形でサーバーから取ってくる仕組みであるため、データだけを変更する場合の使い勝手は格段に向上する。
  – これはただしい！



◆ （5）ＵＩにインテリジェンスがあり、ある程度はサーバーに戻らずにユーザーとやり取りをする。
  – javaScriptのツールの豊富さ



## まとめ

◆ Webがインタフェースになることによって、プログラミングの概念が変わりつつある
  – プログラミングとはなにか？
    ▪ Wiki, … web 2.0
  – それによって、「プログラミング環境」も変わる

◆ 概念を捉えることが大切
  – オブジェクト指向, 設計技術
  – Reflection, introspection → plug-in, … eclipse
  – Ajax…