セキュアで使いやすいWebアプリケーションを作るには

細かいところに気を配り、正しくアプリケーションを構成する

大澤 文孝
OSAWA, Fumitakaa

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ❑ Other:

Level

★ ★ ☆ ☆ ☆

Samples

はじめに

ASP.NETではサーバーコントロールを使うことで、Windowsアプリケーションとほぼ同じ方法でWebアプリケーションを構築できるようになりました。

しかしサーバーコントロールに頼り切ると、いくつかのWebブラウザで想定したように機能しないなどの問題が起こることがあります。また、Webアプリケーションでは、不正な入力データを排除しないと、セキュリティホールにもなりえます。

そこで本稿では、Webアプリケーションを作るときに気をつけたい細かな配慮について説明します。

どのようなWebブラウザでも動作させるための配慮

構築したWebアプリケーションは、Webブラウザの環境にできるだけ依存せずに動くようにしたいものです。

そこでまずは、Webブラウザの環境に依存しないようにする工夫から説明します。

グリッドレイアウトは使わない

Visual Studio .NETでは、Webフォームのレイアウトのデフォルト設定が「グリッドレイアウトモード」になっています。

グリッドレイアウトモードでは、CSSを使って、サーバーコントロールの描画位置を絶対位置として定めます。そのため、ドット単位で正確にサーバーコントロールを配置できます（図1）。

しかしInternet Explorer以外のWebブラウザでは、表示位置がずれるだけでなく、サーバーコントロール同士が重なる可能性もあります。

たとえばTextBoxコントロールの幅の単位はWebブラウザのフォント幅に依存します。そのためWebブラウザによっては、図2のように、TextBoxコントロールと他のコントロールが重なってしまうことがあります。

そこでASP.NETでは、グリッドレイアウトモードではなく、「フローレイアウトモード」にして作業するようにし

図1：グリッドレイアウトモードでの編集

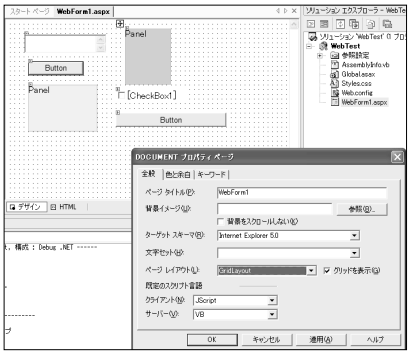
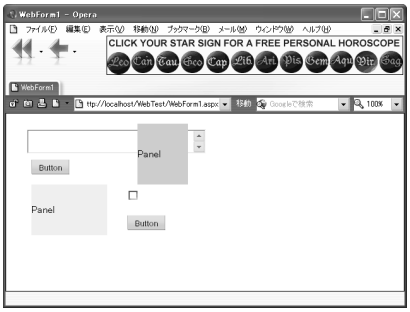


図2：図1をOperaで参照したところ



リスト1：AutoPostBack機能を使ったDropDownListコントロール

```
<%@ Page Language="vb" AutoEventWireup="false"
 Codebehind="WebForm1.aspx.vb" Inherits="WebTest.WebForm1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <body>
 <form id="Form1" method="post" runat="server">
  <asp:DropDownList id="DropDownList1" runat="server" AutoPostBack="True">
   <asp:ListItem Value="1">リンゴ</asp:ListItem>
   <asp:ListItem Value="2">バナナ</asp:ListItem>
   <asp:ListItem Value="3">ミカン</asp:ListItem>
  </asp:DropDownList>
  <P>
   <asp:Label id="Label1" runat="server">Label</asp:Label>
  </P>
 </form>
 </body>
</HTML>
```

①

リスト2：SelectedIndexChangedイベントの処理

```
Private Sub DropDownList1_SelectedIndexChanged( _
 ByVal sender As Object, ByVal e As System.EventArgs) _
  Handles DropDownList1.SelectedIndexChanged
   Label1.Text = DropDownList1.SelectedItem.Text & "が選択されました"
End Sub
```

ます。

フローレイアウトモードでは、サーバーコントロールの位置が登場順に成り行きで決まるので、正確な位置の指定はできません。

しかし逆に言えば、描画位置がWebブラウザ上のサーバーコントロール（HTML要素）の実際の幅だけズレるため、サーバーコントロール同士が重なってしまうという事態を避けられます。

AutoPostBack機能に頼らない

CheckBoxコントロールやDropDownListコントロールなど、ユーザー入力を受け付けるコントロールにおいて、AutoPostBack属性をTrueに設定すると、そのコントロールの状態が変化したときに、SUBMITボタンを押したのと同じく、サーバー側へ処理を移す「ポストバック状態」となります。

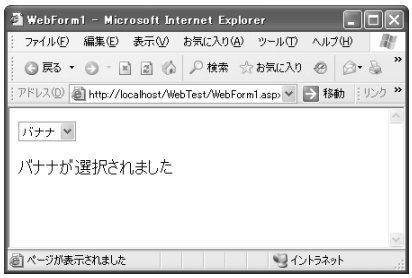
たとえば、リスト1のWebフォームを考えます。

リスト1では、①の部分でAutoPostBackプロパティをTrueに設定したDropDownListコントロールを用意しています。

このときリスト2のようにDropDownListのSelectedIndexChangedイベントを実装すると、選択肢をクリックして変更したときに、新たに選んだ項目がLabelコントロールに表示されるようになります（図3）。

AutoPostBack機能は、ユーザーがSUBMITボタンを押すことなくサーバー側で処理ができるため、ユーザーインターフェイスとして優れた面をもっていると言えます。

図3：リスト1およびリスト2の実行結果



しかしAutoPostBack機能は、JavaScriptによって実装されているので、クライアント側のWebブラウザがJavaScriptをサポートしていないと機能しません。

実際、クライアントに送信されるコードを見ると、リスト3のようになっています。

リスト3では、①の部分にあるように、onchangeというJavaScriptのイベントを利用して、__doPostBack関数を呼び