SQL Server 2005
Visual Studio 2005
スキルアップセミナー

【Advanced トラック:5】
Visual Studio 2005による Webアプリケーション開発【実践編】

フリーライター
山田祥寛（http://www.wings.msn.to/）

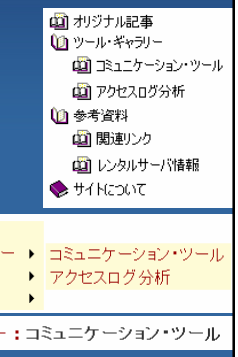
Microsoft

Agenda

- サーバ・コントロール
- キャッシング機能の強化
- クライアントサイド・スクリプト連携
- .NET Framework 2.0の新機能
- 関連リソース・関連セッション

サーバ・コントロール
Navigation API

- ナビゲーション・コントロール
  - 最小限のコードでさまざまなビューのナビゲーション機能を実現
    - TreeView　サイト構造をツリー形式で表示
    - Menu　メニュー形式のサイトリンク
    - SiteMapPath　パンくずリスト
- Navigation API
  - TreeView／Menuの動的生成
    - TreeView、TreeNodeCollection、TreeNode など
    - Menu、MenuItemCollection、MenuItem など
  - web.sitemap（XMLによるサイト定義ファイル）からコンテンツ間の前後、親子関係を動的に取得
    - SiteMap、SiteMapNode など



サーバ・コントロール
Navigation API（解説1）

- web.sitemapから前後ページへのリンクを生成

用意するXML定義ファイル（web.sitemap）

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="home.aspx" title="ホーム">
    <siteMapNode url="sample1-1.aspx" title="第1章1節" />
    <siteMapNode url="sample1-2.aspx" title="第1章2節" />
    <siteMapNode url="sample2-1.aspx" title="第2章1節" />
    <siteMapNode url="sample2-2.aspx" title="第2章2節" />
    <siteMapNode url="sample2-3.aspx" title="第2章3節" />
    <siteMapNode url="sample2-4.aspx" title="第2章4節" />
  </siteMapNode>
</siteMap>
```

サーバ・コントロール
Navigation API（解説2）

web.sitemapから前後ページへのリンクを生成（「.aspx」ファイル）

```
<asp:HyperLink id="Prev" runat="server" Text="前へ" />
<asp:Literal id="Sep1" runat="server" Text=" | " />
<asp:HyperLink id="Prt" runat="server" Text="上位へ" />
<asp:Literal id="Sep2" runat="server" Text=" | " />
<asp:HyperLink id="Nxt" runat="server" Text="後へ" />
```

web.sitemapから前後ページへのリンクを生成（「.aspx.vb」ファイル）

```
If Not SiteMap.CurrentNode.PreviousSibling Is Nothing Then
    Prev.NavigateUrl = SiteMap.CurrentNode.PreviousSibling.Url
Else
    Prev.Visible = False : Sep2.Visible = False
End If
If Not SiteMap.CurrentNode.NextSibling Is Nothing Then
    Nxt.NavigateUrl = SiteMap.CurrentNode.NextSibling.Url
Else
    Nxt.Visible = False : Sep1.Visible = False
End If
Prt.NavigateUrl = SiteMap.CurrentNode.ParentNode.Url
```

兄ノードの有無で前ページへのリンク表示を判定

弟ノードの有無で次ページへのリンク表示を判定

サーバ・コントロール
Navigation API（解説3）

web.sitemapからインデックスページを生成（「.aspx.vb」ファイル）

```
Dim node As SiteMapNode = SiteMap.CurrentNode.ChildNodes(0)
Do While True
    Dim link As New HyperLink()
    link.NavigateUrl = node.Url
    link.Text = node.Title
    Page.Controls.Add(link)
    Page.Controls.Add(New LiteralControl("<br />"))
    node = node.NextSibling
    If node Is Nothing Then Exit Do
Loop
```

カレントノードの情報をハイパーリンクとして出力

次ノードがなくなったらループを脱出

サーバ・コントロール
Navigation API（解説4）

- DBから動的にTreeView、Menuを生成しよう

用意するテーブル（sitemapテーブル）

フィールド名	データ型	概要
id	INT	コンテンツID（主キー）
title	VARCHAR(100)	タイトル
url	VARCHAR(255)	URL
parent	INT	親コンテンツID （親がない場合には“0”）

サーバ・コントロール
Navigation API（解説4）

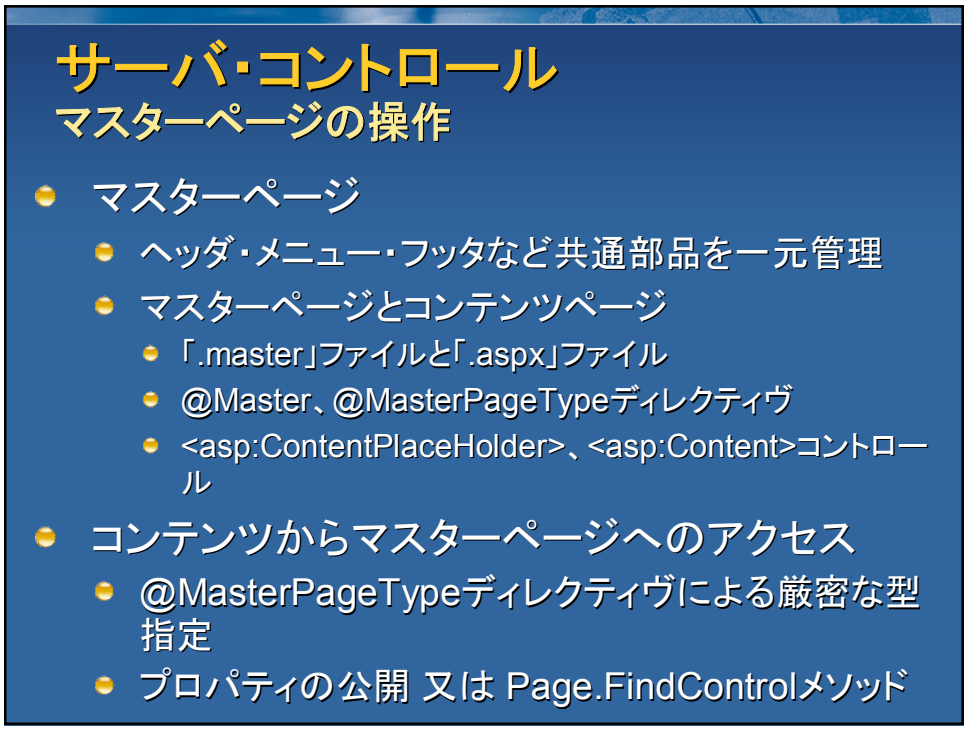
データベースからTreeViewを生成

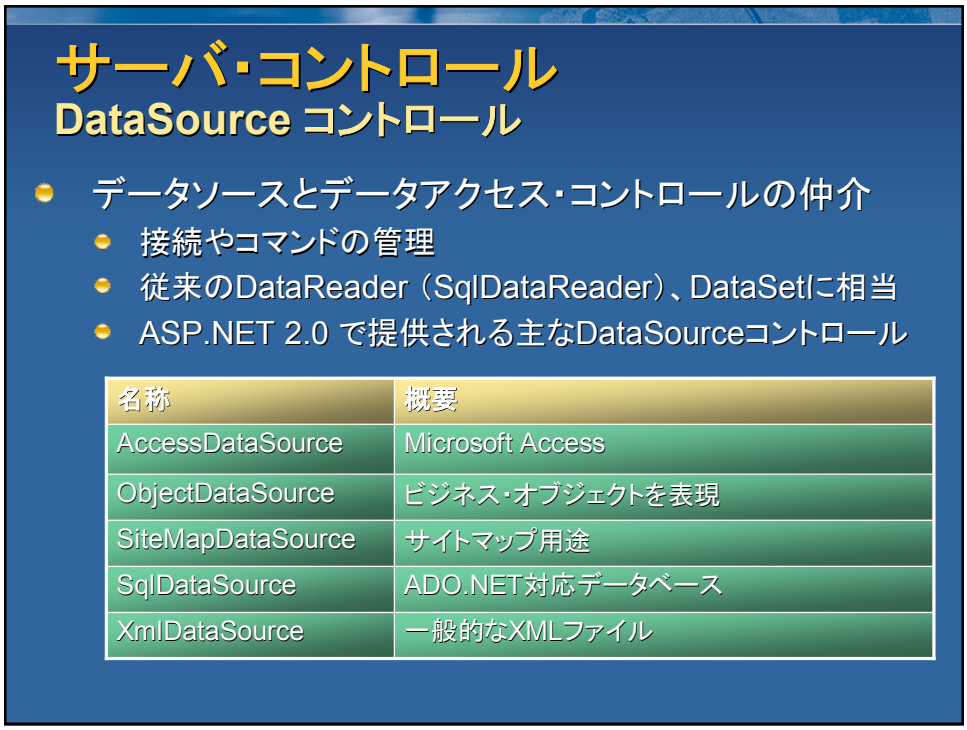
```
Sub Page_Load(sender As Object, e As EventArgs)
    SetNewNode(0, tree.Nodes)
End Sub
Sub SetNewNode(parent As Integer, nodes As TreeNodeCollection)
    Dim objDb As New SqlConnection _
            (ConfigurationManager.ConnectionStrings("db").ToString())
    Dim objCom As New SqlCommand("SELECT id,title,url" & _
            " FROM sitemap WHERE parent=@parent", objDb)
    objCom.Parameters.AddWithValue("@parent", parent)
    objDb.Open()
    Dim objDr As SqlDataReader = objCom.ExecuteReader()
    Do While objDr.Read()
        Dim node As New TreeNode()
        node.NavigateUrl = objDr.GetString(2)
        node.Text = objDr.GetString(1)
        SetNewNode(objDr.GetInt32(0), node.ChildNodes)
        nodes.Add(node)
    Loop
    objDb.Close()
End Sub
```

SetNewNode（親ID,追加先のノード群）

取得したメニュー情報を元にTreeNodeを生成

再帰的な呼び出しによる子ノードの生成





サーバ・コントロール
XmlDataSource コントロール

- XML文書に対するデータバインディング
  - 階層構造のデータ表現に適したデータソース
  - XSLTによるデータ変換やXPath式によるフィルタリング
  - ノード単位のマッピングが可能

サンプルで利用するXMLファイル（books.xml）

```
<?xml version="1.0" encoding="utf-8" ?>
<book name="独習ASP.NET">
  <chapter number="1" name="イントロダクション">
    <section number="1" name="ASP.NETとは何か？" />
    <section number="2" name="ASP.NETプログラミングの基本設定" />
    <section number="3" name="仮想ディレクトリの設定方法" />
  </chapter>
  ...中略...
</book>
```

サーバ・コントロール
XmlDataSource コントロール（解説）

TreeViewコントロールへのバインディング

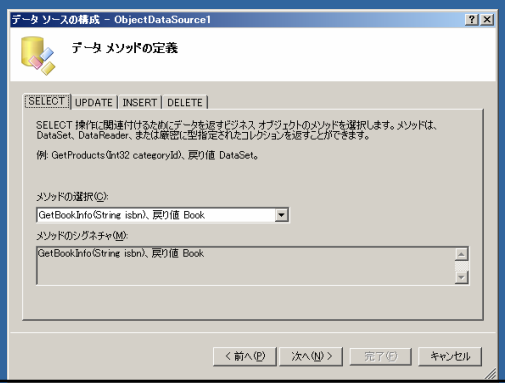
```
<asp:TreeView ID="TreeView1" runat="server" DataSourceID="XmlDataSource1">
  <DataBindings>
    <asp:TreeNodeBinding DataMember="book" TextField="name" />
    <asp:TreeNodeBinding DataMember="chapter" TextField="name"
      NavigateUrlField="number" />
    <asp:TreeNodeBinding DataMember="section" TextField="name"
      NavigateUrlField="number" />
  </DataBindings>
</asp:TreeView>
```

NevigateUrlの動的な生成

```
Sub TreeView1_TreeNodeDataBound(sender As Object, e As TreeNodeEventArgs)
  Select Case e.Node.Depth
    Case 1
      e.Node.NavigateUrl = "chapter.aspx?id=" & e.Node.NavigateUrl
    Case 2
      e.Node.NavigateUrl = "section.aspx?id=" & e.Node.NavigateUrl
  End Select
End Sub
```

サーバ・コントロール
ObjectDataSource コントロール

- データベースへの読み書き処理をカスタマイズ
  - 標準的な参照／更新ならば SqlDataSource など
  - n 階層アプリケーションを構築する場合、ビジネス・オブジェクトの呼び出しに利用



サーバ・コントロール
ObjectDataSource コントロール（解説1）

DetailViewコントロールとの関連づけ

```
<asp:DetailsView ID="DV" runat="server"
  DataSourceID="ODS" AutoGenerateRows="False">
  ...中略...
</asp:DetailsView>
<asp:ObjectDataSource ID="ODS" runat="server"
  SelectMethod="GetBookInfo"
  TypeName="Book"
  UpdateMethod="UpdateBookInfo">
  <SelectParameters>
    <asp:ControlParameter ControlID="DropDown"
      Name="isbn" PropertyName="SelectedValue" Type="String" />
  </SelectParameters>
  <UpdateParameters>
      <asp:Parameter Name="isbn" Type="String" />
      <asp:Parameter Name="title" Type="String" />
      <asp:Parameter Name="publish" Type="String" />
      <asp:Parameter Name="price" Type="Int32" />
  </UpdateParameters>
</asp:ObjectDataSource>
```

サーバ・コントロール
ObjectDataSource コントロール（解説2）

用意するビジネス・オブジェクト（Book.vb）

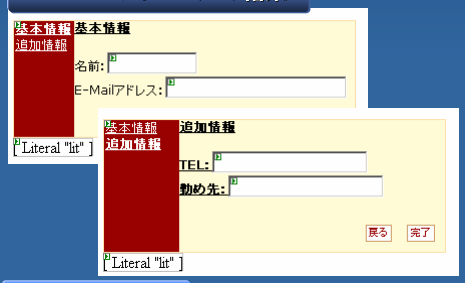
```
Shared Function GetBookInfo(ByVal isbn As String) As Book
        Dim objBook As New Book()
        Dim objDb As New SqlConnection( _
              ConfigurationManager.ConnectionStrings("db").ToString())
        Dim objCom As New SqlCommand( _
"SELECT isbn,title,publish,price FROM books WHERE isbn=@isbn", objDb)
        objCom.Parameters.AddWithValue("@isbn", isbn)
        objDb.Open()
        Dim objDr As SqlDataReader = objCom.ExecuteReader()
        If objDr.Read() Then
           objBook.Isbn = objDr.GetString(0)
           objBook.Title = objDr.GetString(1)
           objBook.Publish = objDr.GetString(2)
           objBook.Price = objDr.GetInt32(3)
        End If
        Return objBook
End Function
Shared Function UpdateBookInfo(ByVal isbn As String, ByVal title As String,
ByVal publish As String, ByVal price As Int32) As Integer
```

サーバ・コントロール
Wizard コントロール

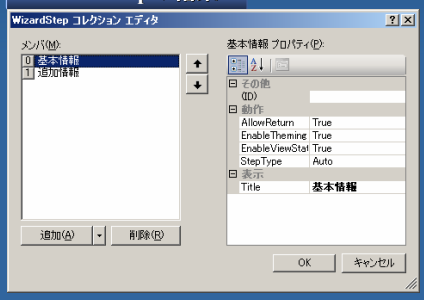
- ASP.NET 1.xにおけるウィザード構築の問題
  - 複数画面にまたがる入力情報の維持
  - エンド・ユーザによる途中ページへのアクセス防止
    - global.asaxによるアクセスの制御を自前で用意
- Wizardコントロール
  - 複数のWizardStepで順番を持つ多段階フォームを定義可
  - ステップ内の入力内容はビューステイトで維持
  - 各イベントハンドラでウィザード内の処理を規定可能
    - ActiveStepChanged、CancelButtonClick、FinishButtonClick、NextButtonClick、PreviousButtonClick、SideBarButtonClickなど

サーバ・コントロール
Wizard コントロール（解説）

Wizardコントロールの編集



WizardStepの編集



[完了]時の処理

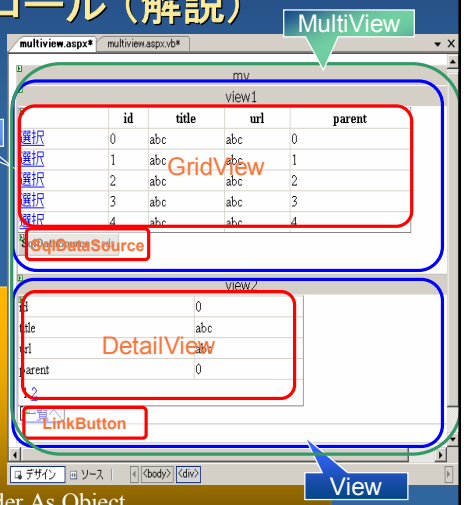
```
Sub wiz_FinishButtonClick(sender As Object, e As WizardNavigationEventArgs)
    Dim sb As New StringBuilder()
    sb.Append("名前:" & txtNam.Text & "<br />")
    sb.Append("E-Mail:" & txtEmail.Text & "<br />")
    sb.Append("TEL:" & txtTel.Text & "<br />")
    sb.Append("勤務先:" & txtCompany.Text & "<br />")
    lit.Text = sb.ToString()
End Sub
```

サーバ・コントロール
MultiView／View コントロール

- ページ内で複数のビューを管理
  - 関連する表示／入力項目を一元的に管理
    - 一覧/詳細画面を一画面で管理
    - 入力項目の多い業務画面を複数ビューに分割することで、入力生産性を向上
- MultiView／View コントロール
  - 配下に複数の View コントロールを定義可能
    - Wizard コントロールのように順番は規定しない
    - ActiveViewChangedイベント
  - 各 View の入力内容はビューステイトで維持

サーバ・コントロール
MultiView／Viewコントロール（解説）

- 一覧/詳細の切り替え
  - ActiveViewIndex プロパティ



MultiViewによる一覧/詳細の切替

```
Protected Sub
  GridView1_SelectedIndexChanged( _
  sender As Object, _
  ByVal e As System.EventArgs) _
  Handles GridView1.SelectedIndexChanged
    mv.ActiveViewIndex = 1
End Sub

Protected Sub LinkButton1_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles LinkButton1.Click
    mv.ActiveViewIndex = 0
End Sub
```

サーバ・コントロール
検証コントロールの改善（1）

- ValidationGroup属性によるグループ定義
  - 複数のサブミットボタンがあるページで、検証コントロールをグループ化
  - ASP.NET 1.xではCausesValidation属性
    - ただし、検証コントロールを無効にするのみ
    - 条件付き検証を行うには、サーバサイドの検証が必要

CausesValidation属性

```
<asp:Button ID="btn" runat="server"
        CausesValidation="False" Text="削除" />
```

ValidationGroup属性

```
<asp:RequiredFieldValidator ID="Req" runat="server" ...
  ValidationGroup="ins" />
<asp:Button ID="btnSbm" runat="server" Text="登録" ...
  ValidationGroup="ins" />
```

サーバ・コントロール
検証コントロールの改善（2）

- SetFocusOnError属性
  - エラー時に、該当要素にフォーカスを自動セット
  - ASP.NET 1.xでは、クライアントサイド・スクリプトでスクリプト宣言ブロックの記述が必要

SetFocusOnError属性を利用した例

```
<asp:RequiredFieldValidator ID="Req" runat="server" ...
  SetFocusOnError="true" />
```

キャッシング機能の強化
データベース・キャッシング

- テーブル変更時にキャッシュデータをリフレッシュ
  - SQL Server 7.0、2000、2005に対応
  - 従来は、キャッシュキー、ファイルの変更にしか依存性を持たせることができなかった
- データベース・キャッシングの利用方法
  - Polling-based Invalidation（for SQL Server 7.0/2000）
    - 該当テーブルの有効化（asp_regsql コマンド）
    - 構成ファイル（web.config）に対象データベースの登録
    - @OutputCacheディレクティヴ（SqlDependency属性）
      or <asp:SqlDataSource>（SqlCacheDependency属性）
  - Notification-based Invalidation（for SQL Server 2005）
    - asp_regsqlコマンドの実行やweb.configの設定は不要
    - @OutputCacheディレクティヴ（SqlDependency属性）
      or <asp:SqlDataSource>（SqlCacheDependency属性）

キャッシング機能の強化
Polling-based Invalidation（解説1）

データベース・キャッシングの有効化（コマンド・プロンプト）

```
> aspnet_regsql.exe -S ServerName -U UserName -P Password -ed -d
DatabaseName -et -t TableName
```

データベース・キャッシングの設定（web.config）

```
<connectionStrings>
  <add name="ConnectionName" connectionString="ConnectionString"
    providerName="System.Data.SqlClient" />
</connectionStrings>
  …中略…
<caching>
  <sqlCacheDependency enabled="true" pollTime="MilliSecond">
    <databases>
      <add name="CacheName" connectionStringName="ConnectionName" />
    </databases>
  </sqlCacheDependency>
</caching>
```

キャッシング機能の強化
Polling-based Invalidation（解説2）

データベース・キャッシングの利用（@OutputCache）

```
<%@ OutputCache Duration="120" VaryByParam="none"
          SqlDependency="CacheName:TableName" %>
```

or

データベース・キャッシングの利用（データソース・コントロール）

```
<asp:SqlDataSource EnableCaching="true"
          CacheDuration="Infinite"
          SqlCacheDependency=" CacheName:TableName " ... />
```

キャッシング機能の強化
Notification-based Invalidation（解説）

データベース・キャッシングの利用（@OutputCache）

```
<%@ OutputCache Duration="120" VaryByParam="none"
        SqlDependency="CommandNotification" %>
```

or

データベース・キャッシングの利用（データソース・コントロール）

```
<asp:SqlDataSource EnableCaching="true"
        CacheDuration="Infinite"
        SqlCacheDependency=" CommandNotification " ... />
```

キャッシング機能の強化
Substitutionコントロール＆WriteSubstituion

- 従来のキャッシュ技術の問題点
  - 大部分が静的なページ＆一部動的なコンテンツ
    - 動的なコンテンツをユーザ・コントロール化
      → フラグメント・キャッシュを利用
    - ページ全体のキャッシュを無効化
- 部分的にキャッシュ処理を無効化する方法
  - Substitutionコントロール
    - 常に動的に処理されるメソッドの呼び出し
  - Response.WriteSubstitutionメソッド
    - 常に動的に出力されるWriteメソッド

キャッシング機能の強化
Substitution コントロール（解説）

Substitutionコントロール

```
<%@ OutputCache VaryByParam="none" Duration="120" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
  lbl.Text = DateTime.Now.ToString()
End Sub
Shared Function GetCurrent(ByVal context As HttpContext) As String
  Return DateTime.Now.ToString()
End Function
</script>
現在時刻（キャッシュ有効）:
<asp:Label ID="lbl" Runat="server" />
現在時刻（キャッシュ無効）:
<asp:Substitution ID="sbt" Runat="server"
          MethodName="GetCurrent" />
```

キャッシング機能の強化
Response.WriteSubstituion メソッド（解説）

Response.WriteSubstitutionメソッド

```
<%@ OutputCache VaryByParam="none" Duration="120" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
  lbl.Text = DateTime.Now.ToString()
End Sub
Shared Function GetCurrent(ByVal context As HttpContext) As String
  Return DateTime.Now.ToString()
End Function
</script>
現在時刻（キャッシュ有効）:
<asp:Label ID="lbl" Runat="server" />
現在時刻（キャッシュ無効）:
<% Response.WriteSubstitution(
                New HttpResponseSubstitutionCallback(GetCurrent)) %>
```

キャッシング機能の強化
ディスク・キャッシング

- DiskCachable 属性（@OutputCache）
  - キャッシュ情報をディスク上に保存
  - アプリケーションの再起動時にもキャッシュを維持

ディスク・キャッシングの利用（@OutputCache）

```
<%@ OutputCache Duration="120" VaryByParam="none"
          DiskCachable="true" %>
```

ディスク・キャッシングの設定（web.config）

キャッシュ上限を2MBに制限

```
<caching>
   <outputCache>
      <diskCache enabled="true" maxSizePerApp="2" />
   </outputCache>
</caching>
```

キャッシング機能の強化
Cache Configuration

- アプリケーション共通のキャッシュ設定を管理
  - 「.aspx」ファイルでの記述を簡素化
  - メモリ使用上限など、より細かな制御が可能に
- 構成ファイルで利用可能な主要要素
  - <cache>要素：利用メモリ量の制限など
  - <outputCache>要素：ページ/フラグメント・キャッシュの有効/無効。ディスク・キャッシュの制限値など
  - <outputCacheSettings>要素：アプリケーション共通で利用可能なキャッシュ設定
  - <sqlCacheDependency>要素：データベース・キャッシング（Polling-based Invalidation）の定義

キャッシング機能の強化
Cache Configuration（解説）

コンフィギュレーションの引用（@OutputCache）

```
<%@ OutputCache CacheProfile="MyCache" %>
```

キャッシングの定義（web.config）

```
<outputCacheSettings>
  <outputCacheProfiles>
    <add name="MyCache" duration="60" diskCacheable="true"
      location="Client" varyByParam="*" />
  </outputCacheProfiles>
</outputCacheSettings>
<cache disableExpiration="True"
          percentagePhysicalMemoryUsedLimit="50" />
```

クライアントサイド・スクリプト連携
クリックイベントの処理

- OnClientClickプロパティ
  - ボタン・クリック時のクライアントサイド・スクリプトを定義
  - ASP.NET 1.x ではスクリプト宣言ブロックでの記述が必要

ASP.NET1.xにおけるクライアントサイド・スクリプトの定義

```
<script runat="Server">
btn.Attributes.Add("onclick","return confirm('本当によろしいですか？');")
</script>
<asp:Button id="btn" runat="Server" ... />
```

ASP.NET2.0におけるクライアントサイド・スクリプトの定義

```
<asp:Button id="btn" runat="Server" ...
          OnClientClick="return confirm('本当によろしいですか？');" />
```

クライアントサイド・スクリプト連携
フォーカスの制御（1）

- ASP.NET 1.xにおけるフォーカス設定
  - Page.RegisterStartupScript メソッドによるスクリプト宣言ブロックからの登録が必要

ASP.NET 1.xにおけるフォーカス設定

```
Dim scriptString As String = _
      "<script language='JavaScript'>" _
      + "document.getElementById('" _
      + TextBox.ClientID + "').focus(); <" + "/script>"
Page.RegisterStartupScript("Startup", scriptString)
```

クライアントサイド・スクリプト連携
フォーカスの制御（2）

- Focus API & DefaultButton
  - フォーム要素へのフォーカスを設定するFocus／SetFocusメソッド
  - フォーム上のデフォルト・ボタンを定義するDefaultButtonプロパティ
  - エラー時のフォーカス先を指定するSetFocusOnErrorプロパティ（検証コントロール）

Focus API & DeafultButton の例

```
TextBox.Focus();
Page.SetFocus(TextBox);

<form runat="Server" DefaultButton="Button" DefaultFocus="TextBox">
```

クライアントサイド・スクリプト連携
クライアント・コールバック

- 従来の処理の問題点
  - イベント発生のたびにポストバックが発生
  - ページ内で必要とされるすべてのViewStateをサーバに送信、復帰処理を行わなければならない
  - 画面のちらつきが発生
- ＜軽い＞ ポストバック＝クライアント・コールバック
  - XML-HTTPによって実現されるコールバック処理
    - クライアントサイド・スクリプトからのサーバ・リクエストをクライアントサイド・スクリプトの関数（コールバック関数）で受け取る
  - XML-HTTPをサポートするブラウザが必要
    - 一般的には、Internet Explorer 5.0以降

クライアントサイド・スクリプト連携
クライアント・コールバック（解説1）

クライアント・コールバックの実装（「.aspx」ファイル）

```
<%@ Implements Interface="System.Web.UI.ICallbackEventHandler" %>

<script language="javascript">
function callBackResult(result,context){
    document.getElementById('txtResult').value=result;
}
</script>

<form runat="server">
    <asp:TextBox ID="txtId" runat="server" />
    <asp:Button ID="btnRef" runat="server" Text="参照" />
    <asp:TextBox ID="txtResult" runat="server" />
</form>
```

コールバック関数

クライアントサイド・スクリプト連携
クライアント・コールバック（解説2）

クライアント・コールバックの実装（「.aspx.vb」ファイル）

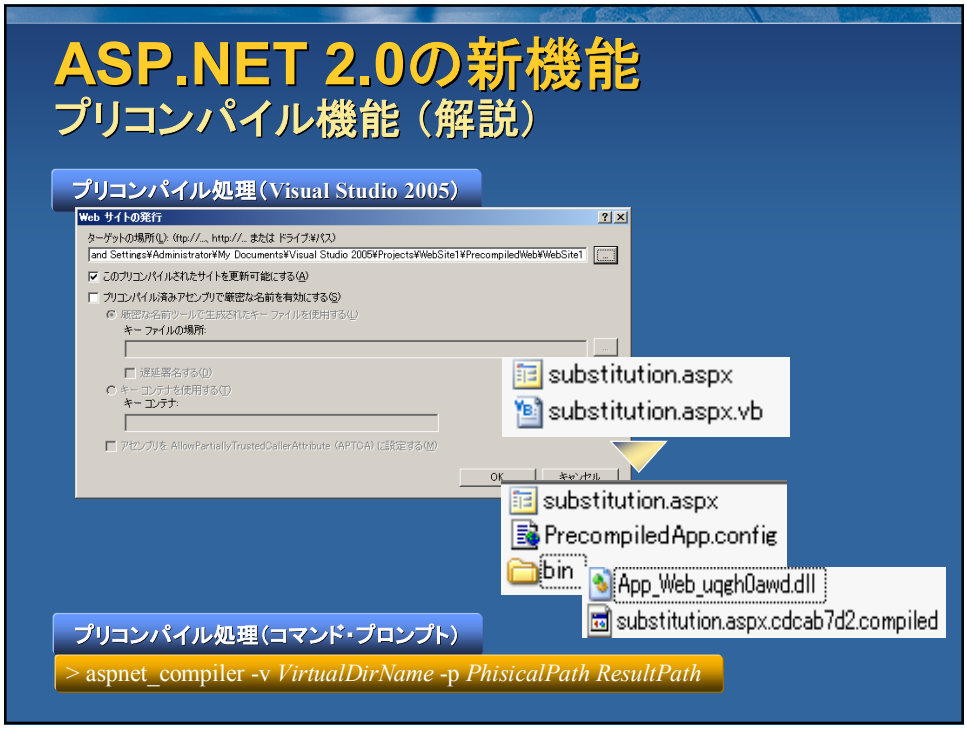
```
Protected Sub Page_Load(sender As Object, e As System.EventArgs)
  Dim sref As String = Page.ClientScript.GetCallbackEventReference(Me,
"document.getElementById('txtId').value", "callBackResult", "null")
  btnRef.Attributes.Add("onclick", sref & ";return false;")
End Sub

Function RaiseCallbackEvent(arg As String) As String _
  Implements ICallbackEventHandler.RaiseCallbackEvent
  Select Case arg
    Case “A001” : Return “ASP.NET vs. Struts フレームワーク徹底比較"
    Case “A002” : Return “ASP.NET2.0が変えるWebアプリ開発の世界"
    Case “A003” : Return “ASP→ASP.NET移行テクニック"
    Case Else : Return "不明"
  End Select
End Function
```

GetCallbackEventReference(
ICallbackEventHandlerインターフェイスを実装したコントロール,
RaiseCallBackメソッドの引数を表すクライアントサイド・スクリプト,
コールバック関数の名前,
コールバック関数に引き渡すコンテキスト値)

ASP.NET 2.0の新機能
プリコンパイル機能

- 従来の動的コンパイル機能の問題点
  - 第3者によるソースコードの解読／改変が可能
  - 初回リクエスト時の処理パフォーマンスが低下
  - コンパイル・エラーが実行時まで発覚しない可能性がある
- 事前コンパイル機能
  - あらかじめすべてのリソースをコンパイル提供
    - 動的コンパイルの問題点を解消
  - 事前コンパイルの方法
    - Visual Studio 2005より[Webの公開]
    - aspnet_compilerコマンド（コマンド・プロンプト）
  - 生成されるファイル
    - PrecompiledApp.config、App_Web_*.dll、*.compiled

ASP.NET 2.0の新機能
プリコンパイル機能（解説）



ASP.NET 2.0の新機能
新しいコードビハインド・モデル

- 従来のコードビハインド・モデル
  - 「.aspx」ファイルで利用しているサーバ・コントロールと同型同名のフィールドを明示的に定義
  - Visual Studio .NET 2003では定義を自動生成
    - #Region～#End Regionで不可視に
- Partial Class / Partial Type
  - 1つのクラスを複数のソースコードで記述可能
  - 従来のコードビハインド・モデルの問題点を解消
  - CompileWith属性（@Page ディレクティヴ）
  - Partial（partial）キーワード

ASP.NET 2.0の新機能
新しいコードビハインド・モデル（解説）

従来のコードビハインド・モデル（「.aspx」ファイル）

```
<%@ Page Src="sample.aspx.vb" Inherits="Sample_aspx"... %>
<asp:Label id="lbl" runat="Server" />
```

従来のコードビハインド・モデル（「.aspx.vb」ファイル）

```
Public Class Sample_aspx : Inherits Page
    Public lbl As Label
    ...中略...
End Class
```

新しいコードビハインド・モデル（「.aspx」ファイル）

```
<%@ Page CompileWith="sample.aspx.vb" ClassName="Sample_aspx"... %>
<asp:Label id="lbl" runat="Server" />
```

従来のコードビハインド・モデル（「.aspx.vb」ファイル）

```
Partial Class Sample_aspx
    ...中略...
End Class
```

ASP.NET 2.0の新機能
クロスページ・ポストバック

- 従来のポストバックの問題点
  - 自分自身へのポストバックが基本
  - 他ページにポストするには、自分で受けたポストバックをServer.Transferメソッドなどで転送する必要

従来のクロスページ・ポストバック

```
Sub Button_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Server.Transfer("ポストバック先のURL")
End Sub
```

- クロスページ・ポストバック
  - PostbackUrlプロパティでポストバック先を指定
    - Button、TextBoxなどイベントの発生元で指定可能
  - ポスト元のビューステートは維持

ASP.NET 2.0の新機能
クロスページ・ポストバック（解説）

ポストバック元のページ（before.aspx）

```
<asp:TextBox ID="txtNam" runat="server“ />
<asp:Button ID="btnSubmit" runat="server” Text="送信"
          PostBackUrl="~/after.aspx" />
```

ポストバック先のページ（after.aspx）

```
Sub Page_Load(sender As Object, e As EventArgs)
    Dim txtNam As TextBox = _
          CType(Page.PreviousPage.FindControl("txtNam"), TextBox)
    lblNam.Text = txtNam.Text
End Sub
```

PreviousPageプロパティでポストバック元のページを取得

ASP.NET 2.0の新機能
Membership API

- メンバ情報を操作するためのAPIを提供
  - データソースを意識しないプログラミングが可能
  - Membershipクラス
    - ユーザ情報の操作（CreateUser、UpdateUser、DeleteUser）
    - ユーザ情報の取得（FindUsersByEmail、FindUsersByName、GetUserNameByEmail、GetAllUsers、GetUser）
    - ユーザ情報の検証（ValidateUser）など
  - MembershipUserクラス
    - UserName、Email、IsLockedOut、LastLoginDate、LastLockoutDate、LastPasswordChangedDateなど

ASP.NET 2.0の新機能
Membership API（解説1）

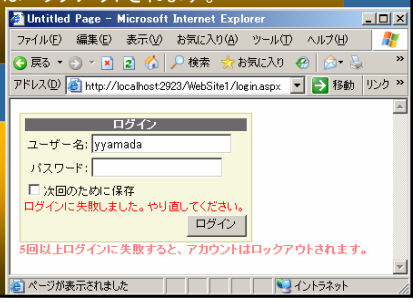
- ロックアウト処理
  - 複数回ログインに失敗すると、アカウントをロックアウト

ログインエラー時の処理

```
Sub Login1_LoginError(sender As Object, e As System.EventArgs)
    Dim usr As MembershipUser = Membership.GetUser(Login1.UserName)
    If usr.IsLockedOut Then
        Label1.Text = Membership.Provider.MaxInvalidPasswordAttempts & _
            "回以上ログインに失敗すると、アカウントはロックアウトされます。"
    End If
End Sub
```

メンバシップ・プロバイダの設定

```
<membership
  defaultProvider="SqlProvider" ...>
  <providers>
    <add name="SqlProvider" ...
      maxInvalidPasswordAttempts
            ="リトライ可能回数" />
  </providers></membership>
```



ASP.NET 2.0の新機能
Membership API（解説2）

ロックアウトの解除

```
Dim usr As MembershipUser = Membership.GetUser(txtUsr.Text)
usr.UnlockUser()
```

アカウント情報の更新

```
Dim usr As MembershipUser = Membership.GetUser()
usr.Email = CType(email.Text, String)
usr.PasswordQuestion = CType(pq.Text, String)
usr.Comment = CType(comment.Text, String)
Membership.UpdateUser(usr)
```

アカウント情報の更新

```
Membership.DeleteUser(txtUsr.Text)
```

Appendix

- 関連リソース
  - ASP.NET Quickstart Tutorial
    http://beta.asp.net/QUICKSTART/aspnet/default.aspx
  - ASP.NET 2.0が変えるWebアプリ開発の世界
    http://www.atmarkit.co.jp/fdotnet/asp2review/index/index.html
  - サーバサイド技術の学び舎 – WINGS
    http://www.wings.msn.to/
- 関連セッション
  - 【VS 1】.NET Framework 2.0における新機能
  - 【VS 2】Visual Studio 2005による
    Webアプリケーション開発【基礎編】
  - 【VS 4】Visual Studio 2005における開発言語の強化
    【Visual Basic 編】
  - 【VS 7】Visual Studio 2005における開発言語の強化
    【C# 編】

Microsoft®
Your potential. Our passion.™