Web-based Apps in .NET

Objectives

“Real-world applications are typically multi-tier, distributed designs involving many components — the web server being perhaps the most important component in today's applications...”

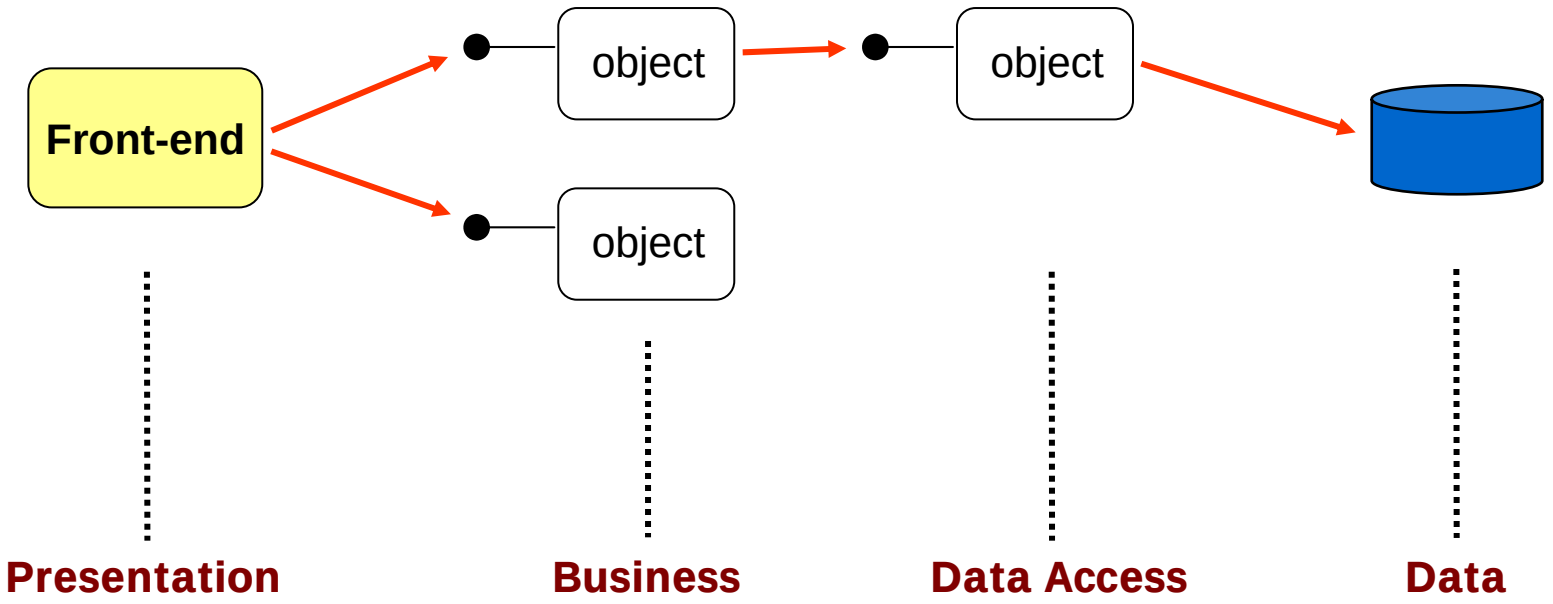
- **Web-based applications**
- **IIS**
- **WebForms**

Part 1

- **Web-based applications…**

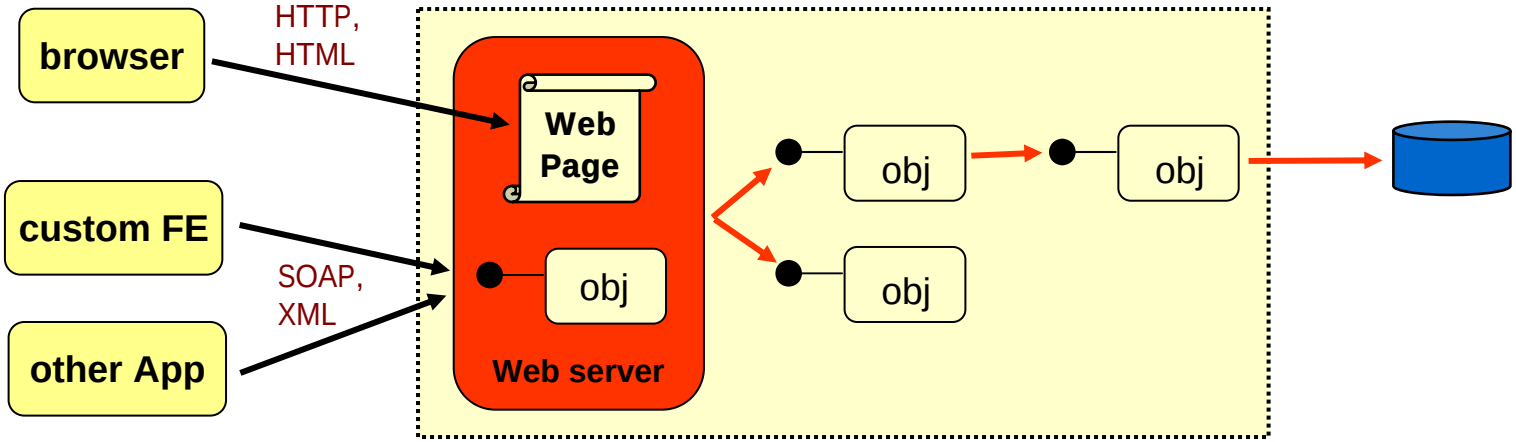
Application design

- **Many applications are designed with N levels or "tiers"**
  - good separation of concerns
  - enables reuse of back-end tiers across varying FEs



Web-based applications

- **Web-based apps offer many advantages:**
  - extend reach of application to people AND platform
  - based on open, non-proprietary technologies
- **Two types:**
  - ***WebForms***: GUI-based app displayed in browser
  - ***WebServices***: object-based app returning raw XML

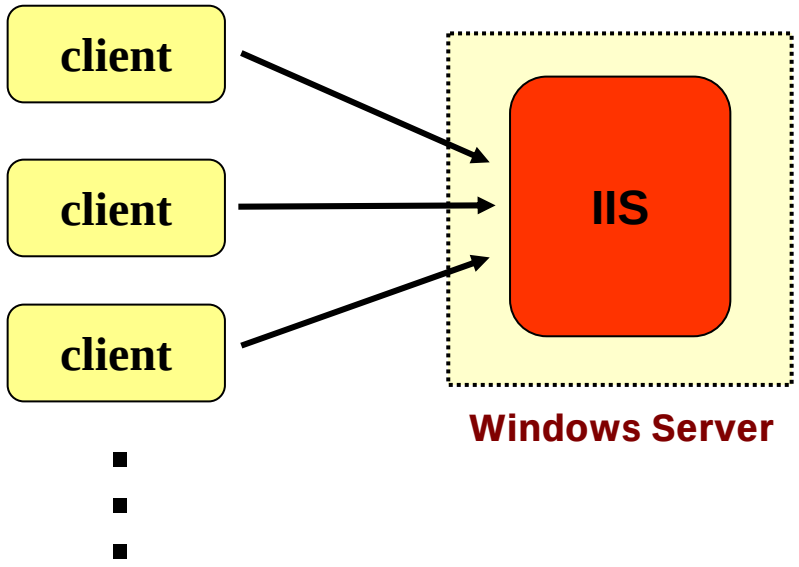


Part 2

- **IIS…**

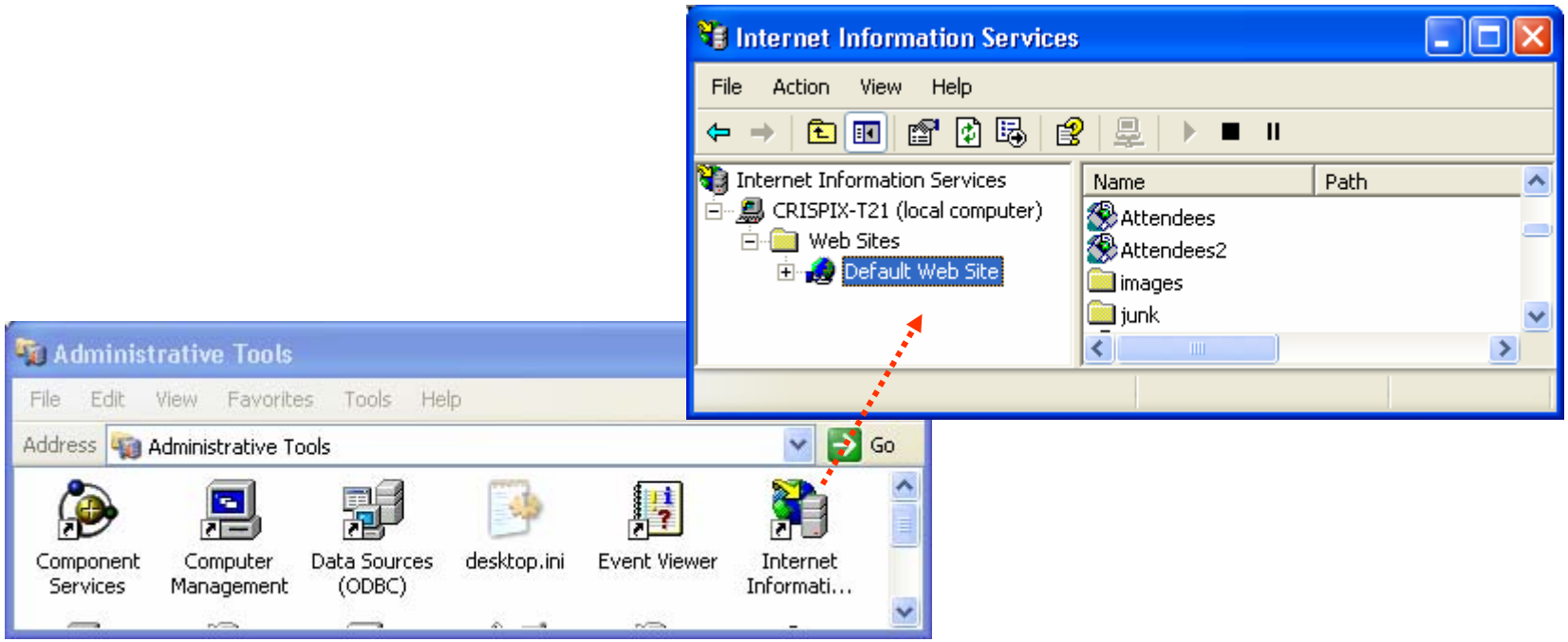
Internet Information Services (IIS)

- **IIS is Microsoft's Web Server**
  - runs as a separate process "inetinfo.exe"
  - requires a server-like OS: Windows NT, 2000, XP Pro
  - multi-threaded to service thousands of requests…



Configuring IIS

- **Configured manually via:**
  - control panel, admin tools, Internet Information Services



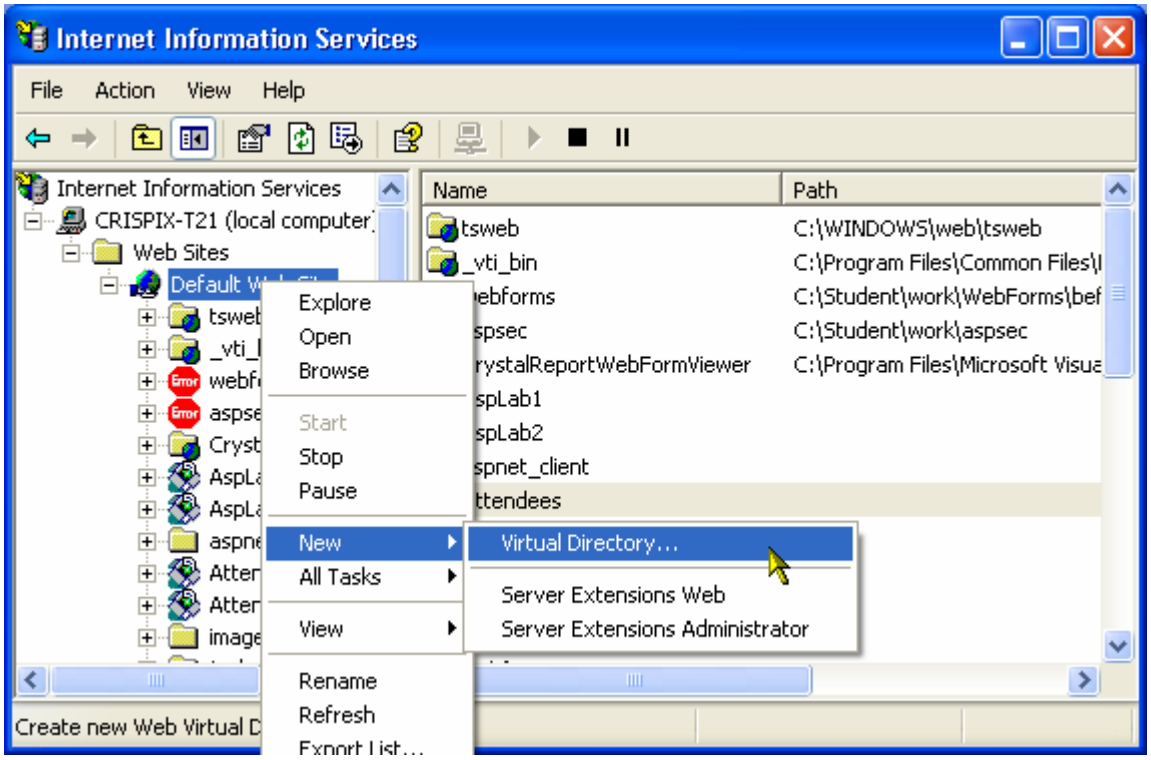
A web site

- **IIS deals with web sites**
- **A web site = a web application**

- **How IIS works:**
  - each web site has its own physical directory on hard disk
  - each web site is assigned a virtual name for that directory
  - users surf to web site based on virtual name
- **Example:**
  - web site lives in **`C:\Inetpub\wwwroot\WebSite`**
  - web site's virtual name is "AAAPainting"
  - IIS maps virtual to physical…

Creating a virtual directory

- **When in doubt, right-click :-)**
  - right-click on "Default Web Site", New, Virtual Directory…

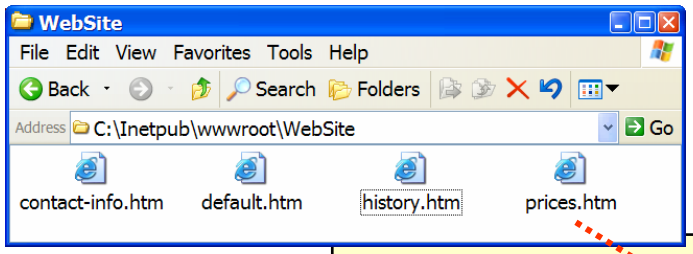


Types of web sites

- **From IIS's perspective, 2 types:**
  - static
  - dynamic

Static web sites

- **A static web site has the following characteristics:**
  1. all web pages are pre-created and sitting on hard disk
  2. web server returns pages without any processing

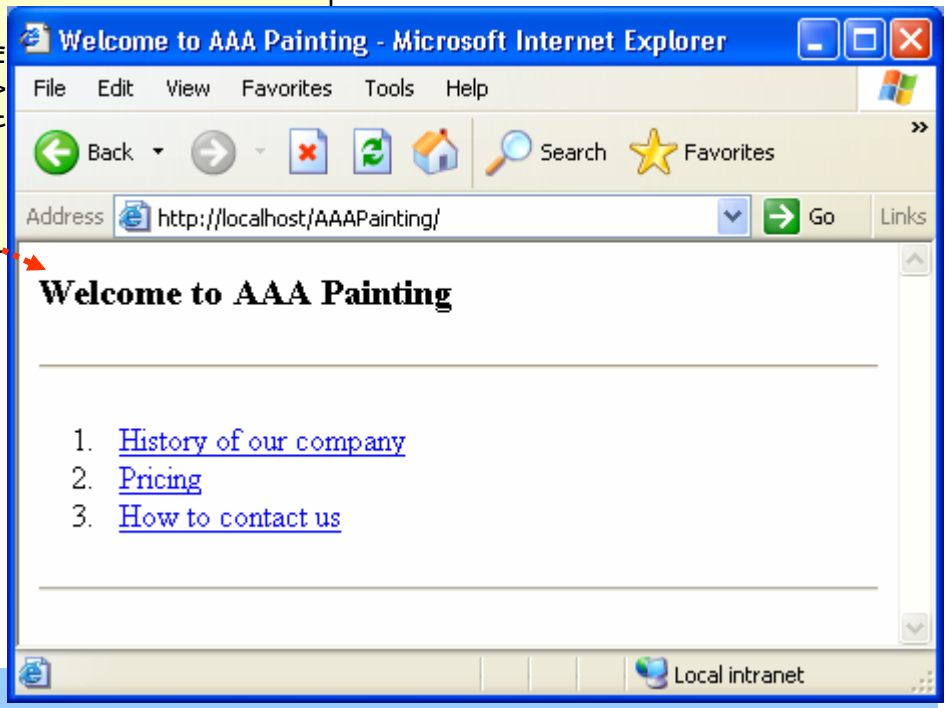



```
<title>Welcome to AAA Painting</title>

<html>
<h3>Welcome to AAA Painting</h3>
<HR>
  <ol>
    <li> <A HREF="history.htm">History of our company</A>
    <li> <A HREF="prices.htm">Pricing</A>
    <li> <A HREF="contact-info.htm">How to contact us</A>
  </ol>
<HR>
</html>
```

HTML

- **Static web sites are typically pure HTML**
  - pages may also contain script code that runs on client-side

```
<title>Welcome to AAA Painting</title>

<html>
<h3>Welcome to AAA Painting</h3>
<HR>
  <ol>
    <li> <A HREF="history.htm">History of
    <li> <A HREF="prices.htm">Pricing</A>
    <li> <A HREF="contact-info.htm">How t
  </ol>
<HR>
</html>
```



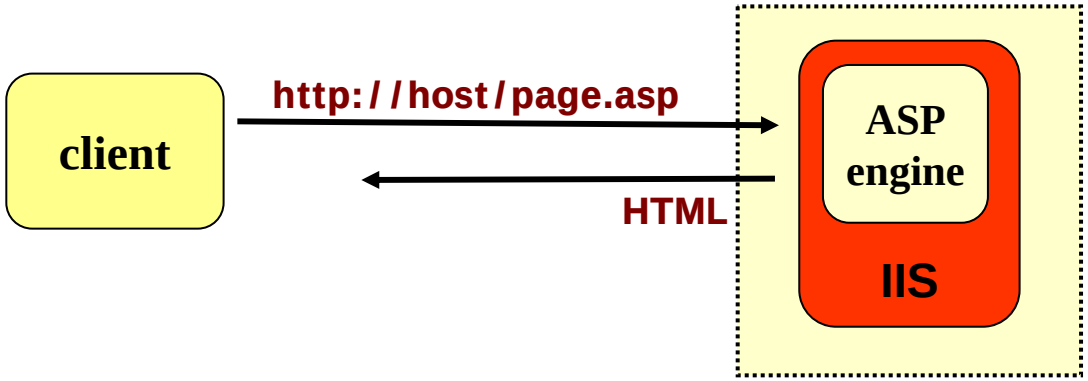
Dynamic web sites

- **A dynamic web site involves server-side processing**
- **How does IIS tell the difference?**
  - based on file extension of requested web page…

- **Example:**
  - static request: http://localhost/AAAPainting/default.htm
  - dynamic request: http://localhost/Workshop/default.asp

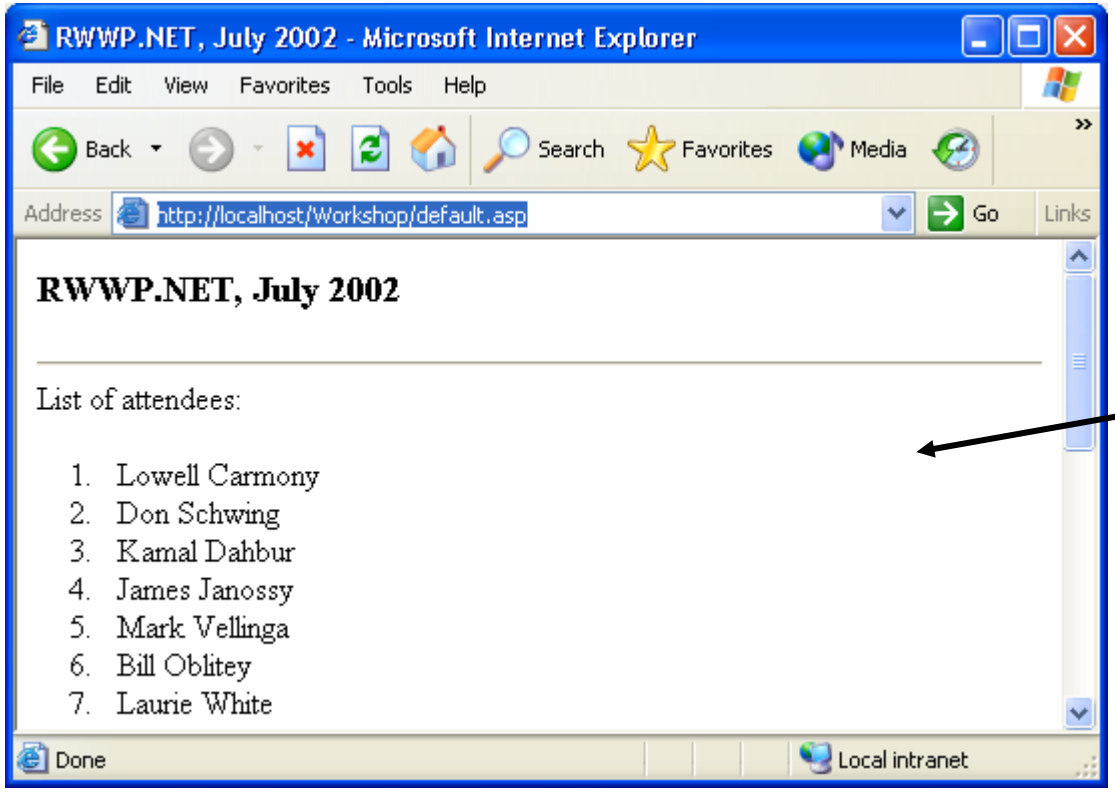
ASP

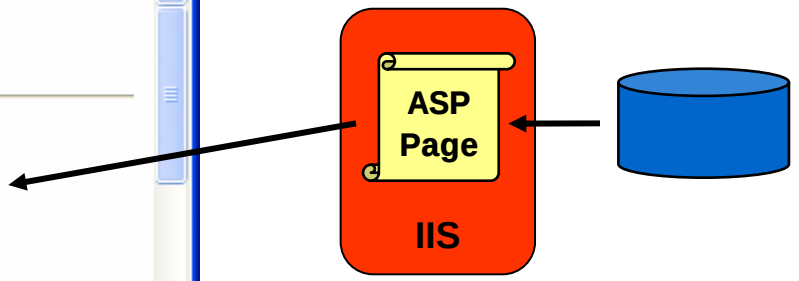
- **Active Server Pages**
- **Microsoft's server-side programming technology**
  - ASP based on scripting languages, interpreted
  - ASP.NET based on .NET languages, compiled, faster, …



Example

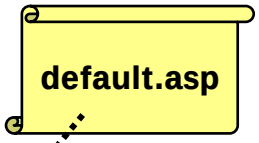
- **We want to dynamically create web page of attendee's**
  - i.e. generate the page by reading names from a database





ASP page — part 1, presentation

- **ASP page = HTML + code…**




```
<title>RWWP.NET, July 2002</title>

<html>
<h3>RWWP.NET, July 2002</h3>
<HR>
  List of attendees:
  <ol>
    <%
        Call OutputAttendees( )
    %>
  </ol>
<HR>
</html>

<SCRIPT LANGUAGE="VBScript" RunAt="Server">
   Sub OutputAttendees()
     .
     .
     .
```

inline code block

ASP page — part 2, logic


```
   Sub OutputAttendees()
      Dim dbConn, rs, sql, firstName, lastName

      sql = "Select * From Attendees"
      Set dbConn = CreateObject("ADODB.Connection")
      Set rs = CreateObject("ADODB.RecordSet")

      dbConn.Open("Provider=Microsoft.Jet.OLEDB.4.0;" + _
                  "Data Source=C:\Inetpub\wwwroot\Workshop\Attendees.mdb")
      rs.ActiveConnection = dbConn : rs.Source = sql : rs.Open

      Do While Not rs.EOF
         firstName = rs("FirstName")
         lastName = rs("LastName")
         Response.Write("<li> " + firstName + " " + lastName)
         rs.MoveNext()
      Loop

      rs.Close : dbConn.Close
    End Sub
</SCRIPT>
```

The key to web programming…

- **It's a client-server relationship**
  - client makes request
  - server does some processing…
  - client sees OUTPUT of server-side processing

Part 3

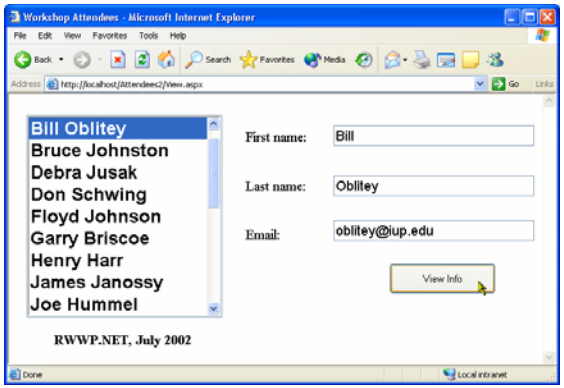
- **WebForms…**

Traditional form-based web apps

- **HTML already supports the creation of form-based apps**


```
<HTML>
 <HEAD>
   <title>Login</title>
 </HEAD>

 <BODY>
 <h2>Please login:</h2>
 <form method="get" action="Main.htm" id="Login">
    Username: <INPUT type="text" id="Name"> <BR>
    Password: <INPUT type="text" id="pwd">  <BR> <BR>
    <INPUT type="submit" value="Login">
 </form>
 </BODY>
</HTML>
```

WebForms

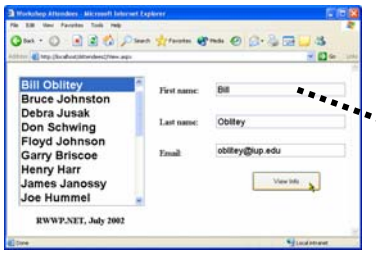


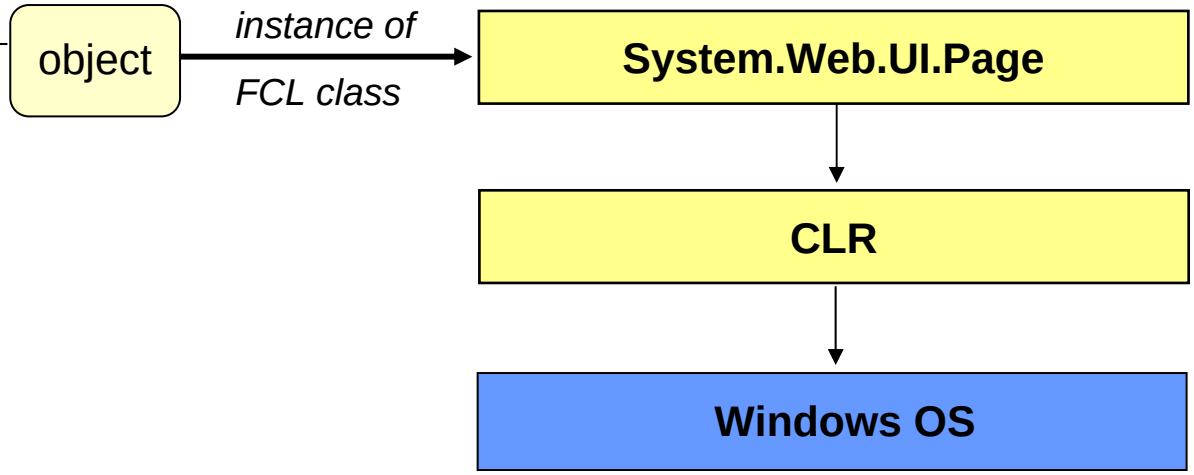
- **Web-based, form-based .NET apps**

- **Based on many technologies:**
  - IIS
  - ASP.NET (extension to IIS)
  - .NET Framework SDK (CLR, FCL, tools)
  - Visual Studio .NET (drag-and-drop creation)

Abstraction

- **Like WinForms, WebForms are based on classes in FCL**
  - separates WebForm app from underlying platform



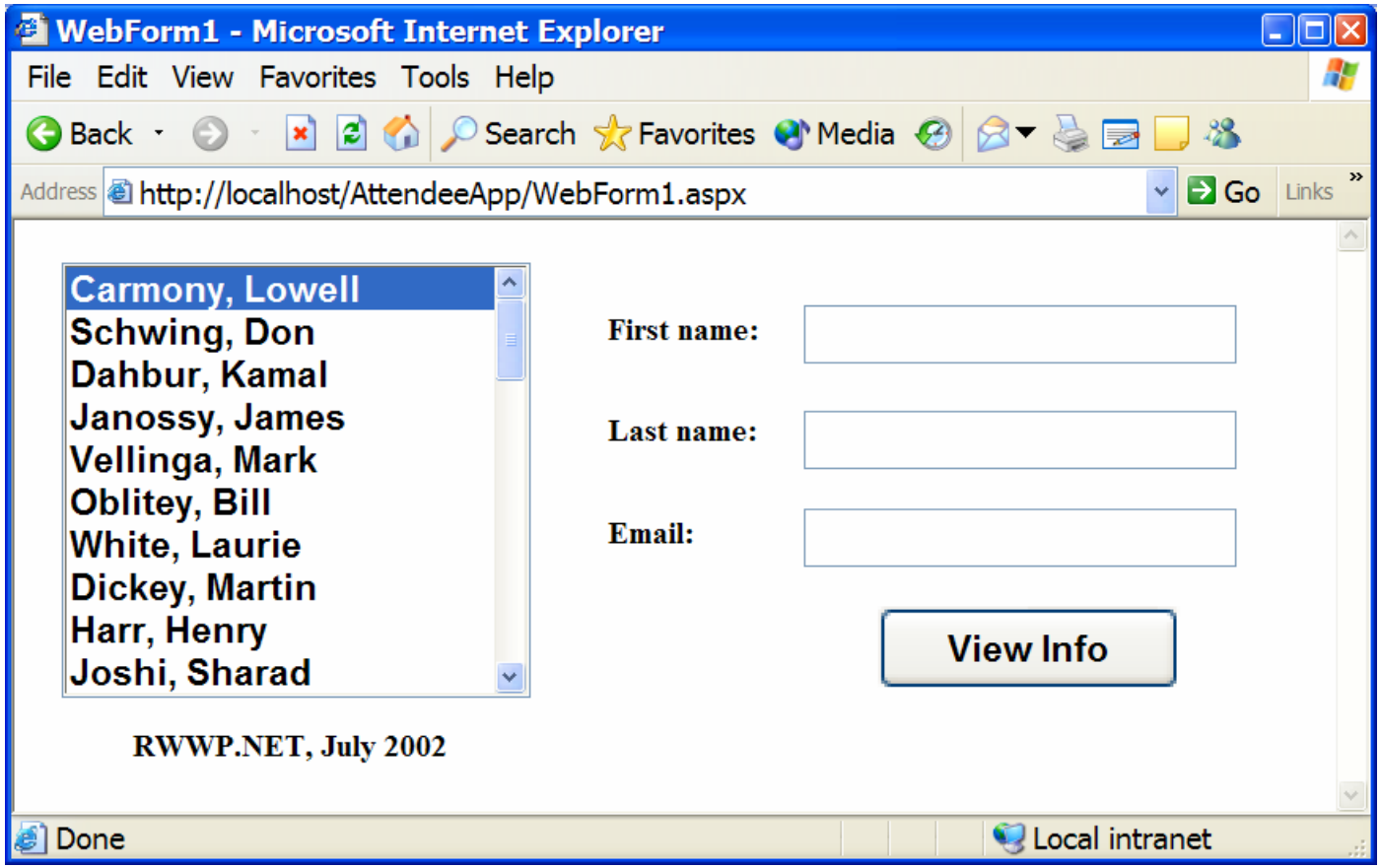


Creating a WebForm app

- **Good news: much like creating a WinForm app!**
  1. create appropriate project in Visual Studio
  2. design form(s) via drag-and-drop of controls
  3. program events

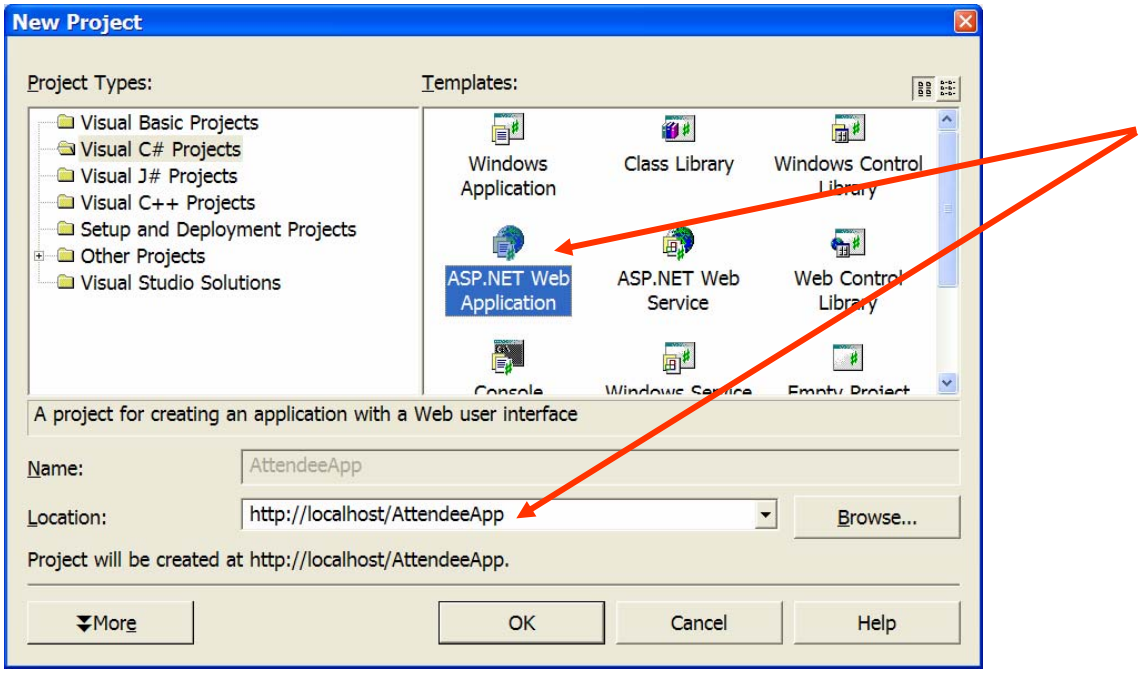
Example

- **A simple WebForms app to view attendee info from DB**



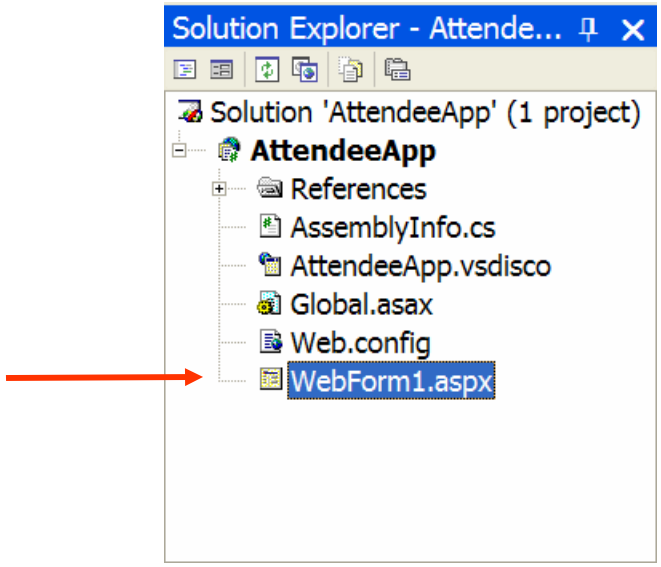
(1) Create ASP.NET Web App project

- **Location = name of web site = "http://localhost/AttendeeApp"**
  - virtual directory: AttendeeApp
  - physical directory: C:\Inetpub\wwwroot\AttendeeApp



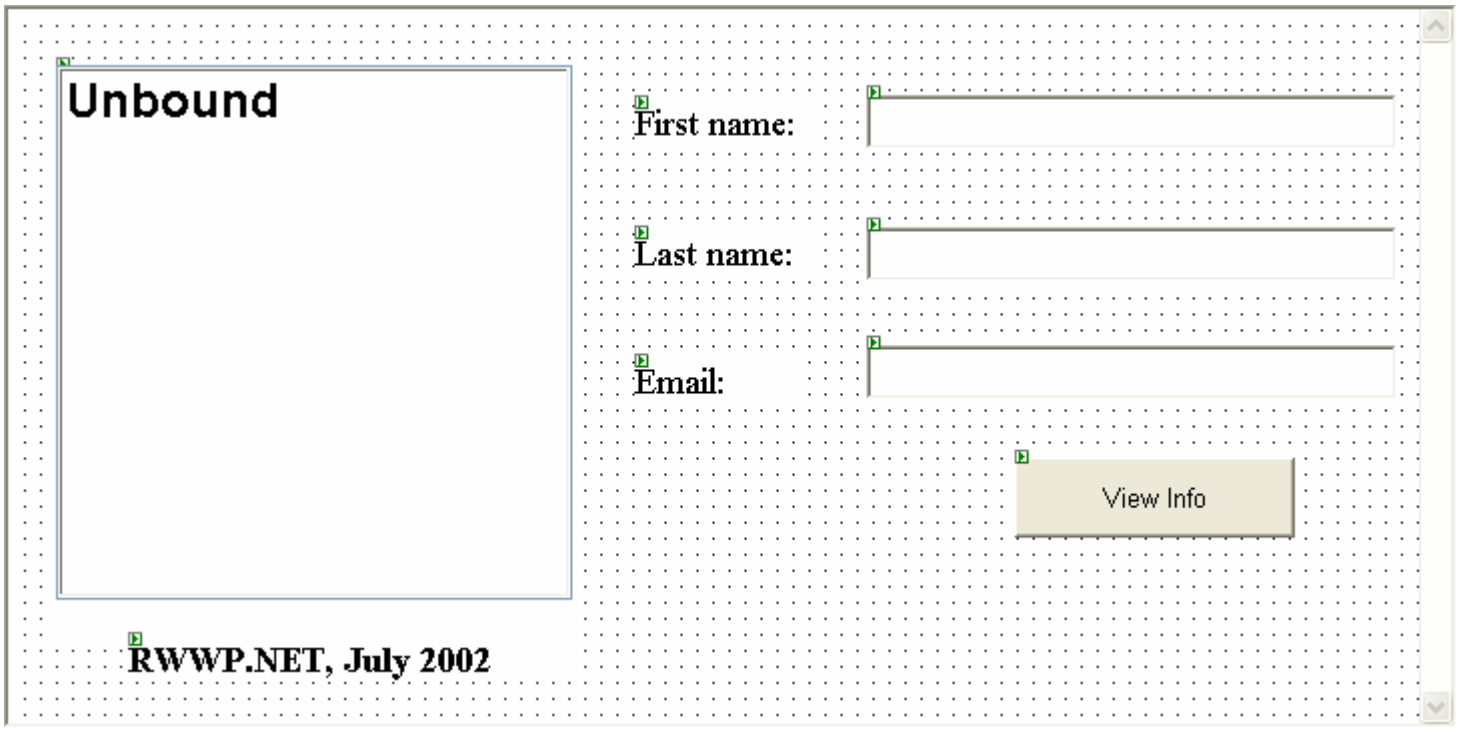
(2) Project layout

- **VS .NET configures IIS for you**
- **VS .NET creates web site for you**
  - contains 1 form-based web page
  - named WebForm1.aspx by default
  - ignore other files for now…



(3) Design WebForm

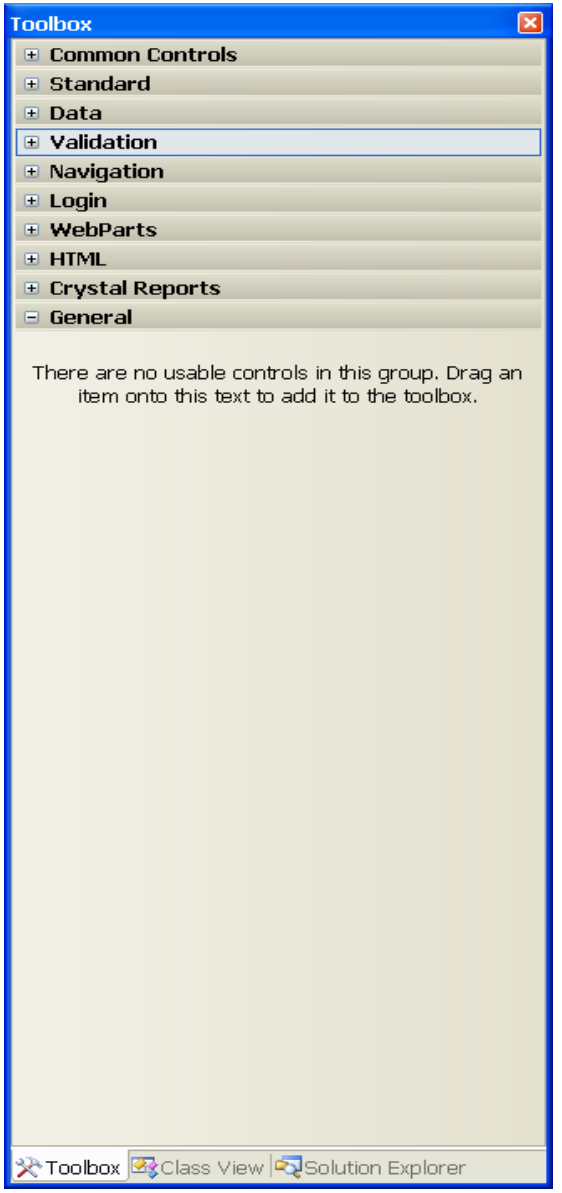
- **Just like a WinForm**
  - drag-and-drop from toolbox…



Web controls vs. HTML controls

- **Toolbox contains 2 types of controls:**
  - those under HTML
  - others
- **Both generate pure HTML on client**
  - though sometimes with JavaScript!

- **Web controls:**
  - work like WinForm counterparts
- **HTML controls:**
  - mimic standard HTML controls



(4) Implement events

- **WebForms are event-driven, as you would expect:**
  - on Page_Load, fill list box from database
  - on cmdViewInfo_Click, display info about selected attendee

- **In each case, standard C# database programming…**

```
private void Page_Load(...)
{
  IDbConnection  dbConn = null;
  try {
    dbConn = new OleDbConnection(sConnection);
    dbConn.Open();
    .
    .
    .
```

(5) Run!

- **You can run from within VS**
- **You can debug from within VS**

- **How does it work?**
  - starts up a session of IE
  - attaches debugger to IIS
  - displays .aspx page marked as "Start Page" in your project
    - right-click on .aspx page you want to start with
    - select "Set as Start Page"

(6) Reminder — client-server relationship!

- **The server contains lots of code**
  - see physical directory…
- **But the client sees only HTML!**
  - "View Source" in browser…

ASP.NET programming model

- **On the surface WebForms appear like WinForms**
- **But the programming model is different underneath**
  - due to ASP.NET
  - due to client-server paradigm

#1: Traditional dialog doesn't work

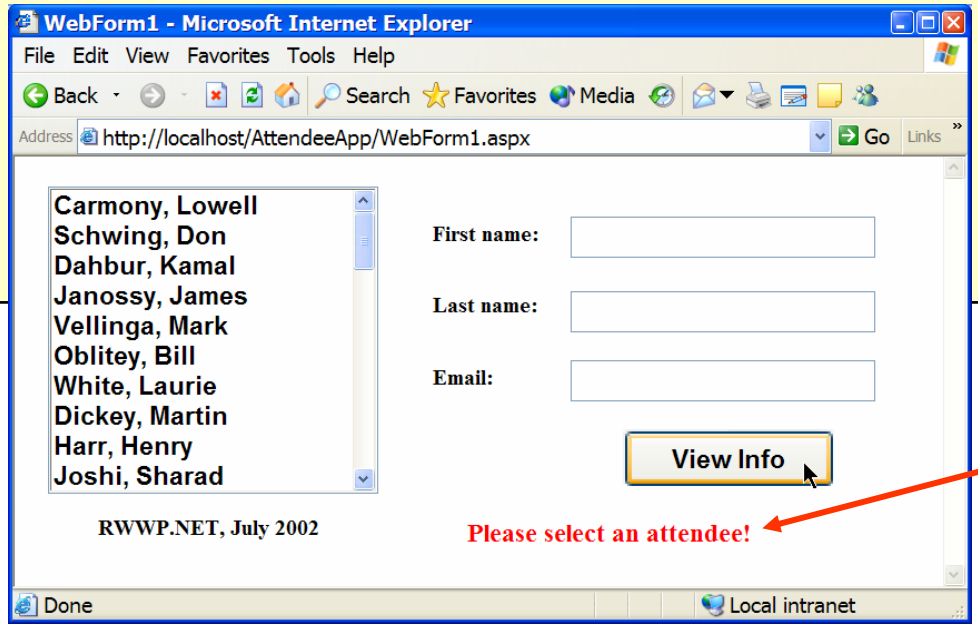
- **For example, these do not work:**
  - MessageBox.Show( )
  - form1.Show( )
- **Why not?**
  - think about where form would appear…

- **Solutions:**
  - if you want to tell user something, display via label on page
  - if you want to show another page, redirect browser

Web-based dialogs

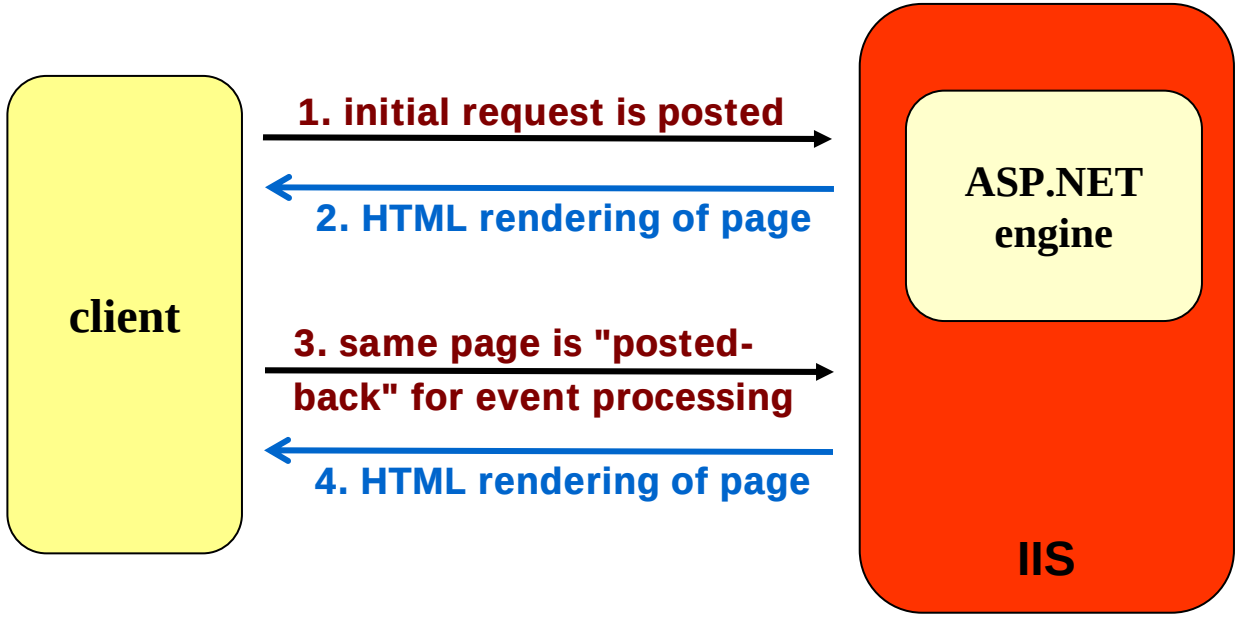
- **To display a message to user:**

```
private void cmdViewInfo_Click(...)
{
   if (this.ListBox1.SelectedItem == null) {
     // nothing selected
     this.lblErrorMsg.Text = "Please select an attendee!";
     return;
   }

   .
   .
   .
```



#2: Fewer events

- **There are fewer events to program in WebForms**
  - primarily Change and Click events only
- **Why?**
  - because each event represents 1 round-trip to server for processing
  - and thus event processing represents a very expensive activity



#3: AutoPostBack

- **In fact, some events aren't posted right away…**
  - instead event is "queued" until page is eventually posted back
- **To force immediate postback?**
  - set control's `AutoPostBack` property (if it has one)

- **Example:**
  - list box doesn't postback when you click on an item
  - instead, event is queued until later (e.g. button is clicked)

#4: Postbacks

- **There is a distinction made between:**
  - first request that is posted to server for page X by client C
  - subsequent "postbacks" of page X to client C

- **Example:**
  - Page_Load event triggers every time
  - but typically only need to initialize form the first time!

```
private void Page_Load(...)
{
   if (this.IsPostBack)  // no need to reload list box!
     return;

   .
   . // first request, load list box from DB
   .
```

#5: Statelessness

- **Web apps are stateless**
- **Each page request is a self-contained operation:**
  - connection is opened
  - request is posted
  - result is returned
  - connection is closed

- **Implications? By default:**
  - no session state (i.e. no data for client C across pages)
  - no global state (i.e. no data across all clients & pages)
  - postback state *is* maintained for you by .NET
    - e.g. contents of list box

Summary

- **Web-based applications are commonplace**
- **Web-based applications are often mission-critical**
- **Two options:**
  - WebForms: form-based
  - WebServices: object-based

- **WebForms make Web-based apps far easier to build…**

References

- **Books:**
  - F. Onion, "Essential ASP.NET"
  - D. Sussman et al., "Beginning ASP.NET 1.0 with C#"

- **Web sites:**
  - http://www.asp.net/
  - http://www.asp.net/webmatrix/