

PV247 – Development I

Introduction to ASP.NET and related technologies

- **ASP.NET Basics**

- What is ASP.NET?
- What is a request?
- How does ASP.NET deal with stateless http?
- ASP.NET request/page/control life cycle
- REST Services

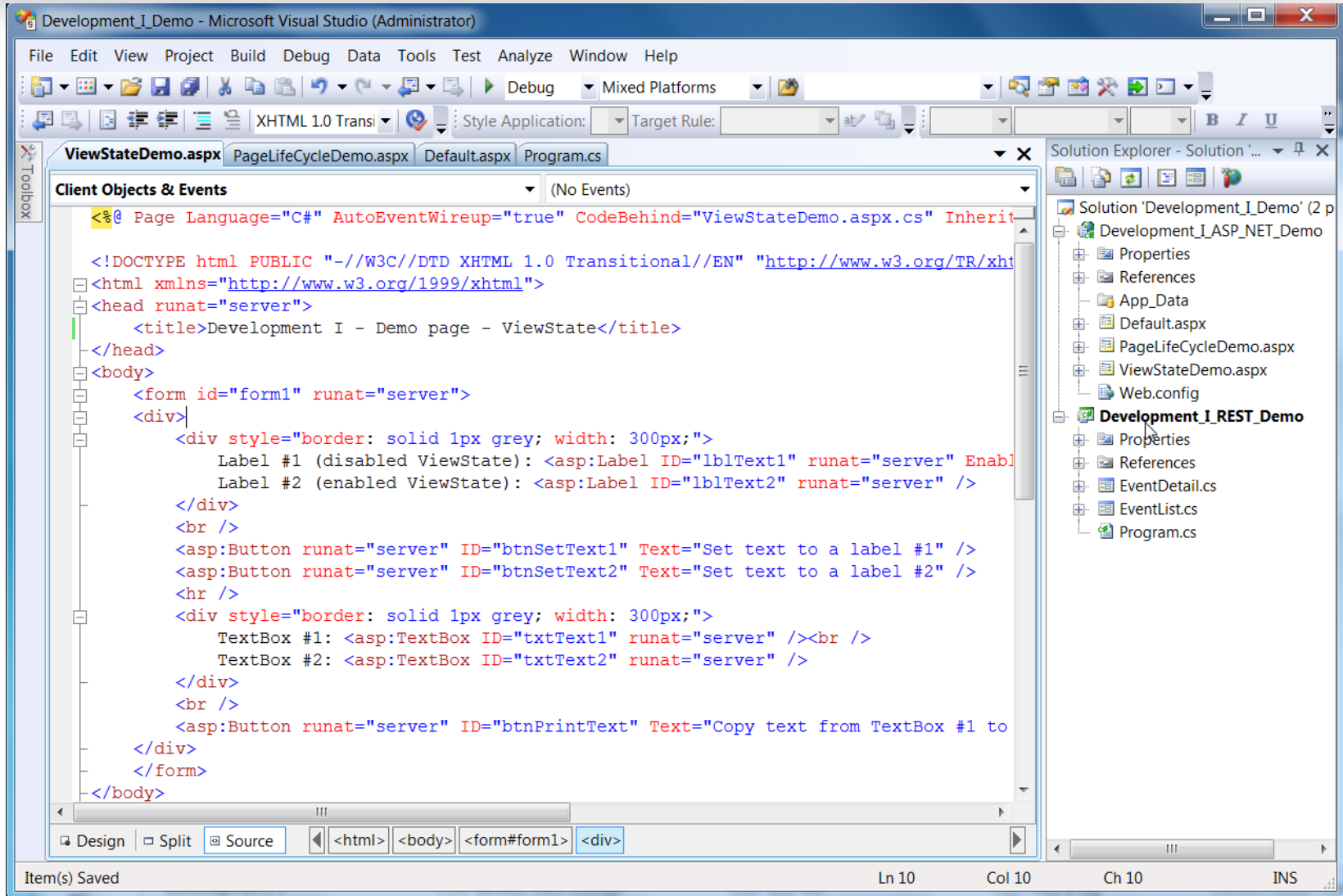
- **Kentico CMS Platform Basics**

- CMS.IO namespace

What is ASP.NET?

- **Active Server Pages .NET**
- Platform for creating dynamic web applications
- You can use any .NET language as a code-behind
- Development of ASP.NET WebForms applications can be similar to the development of WinForms applications. Similar, not same!

Example of ASP.NET page

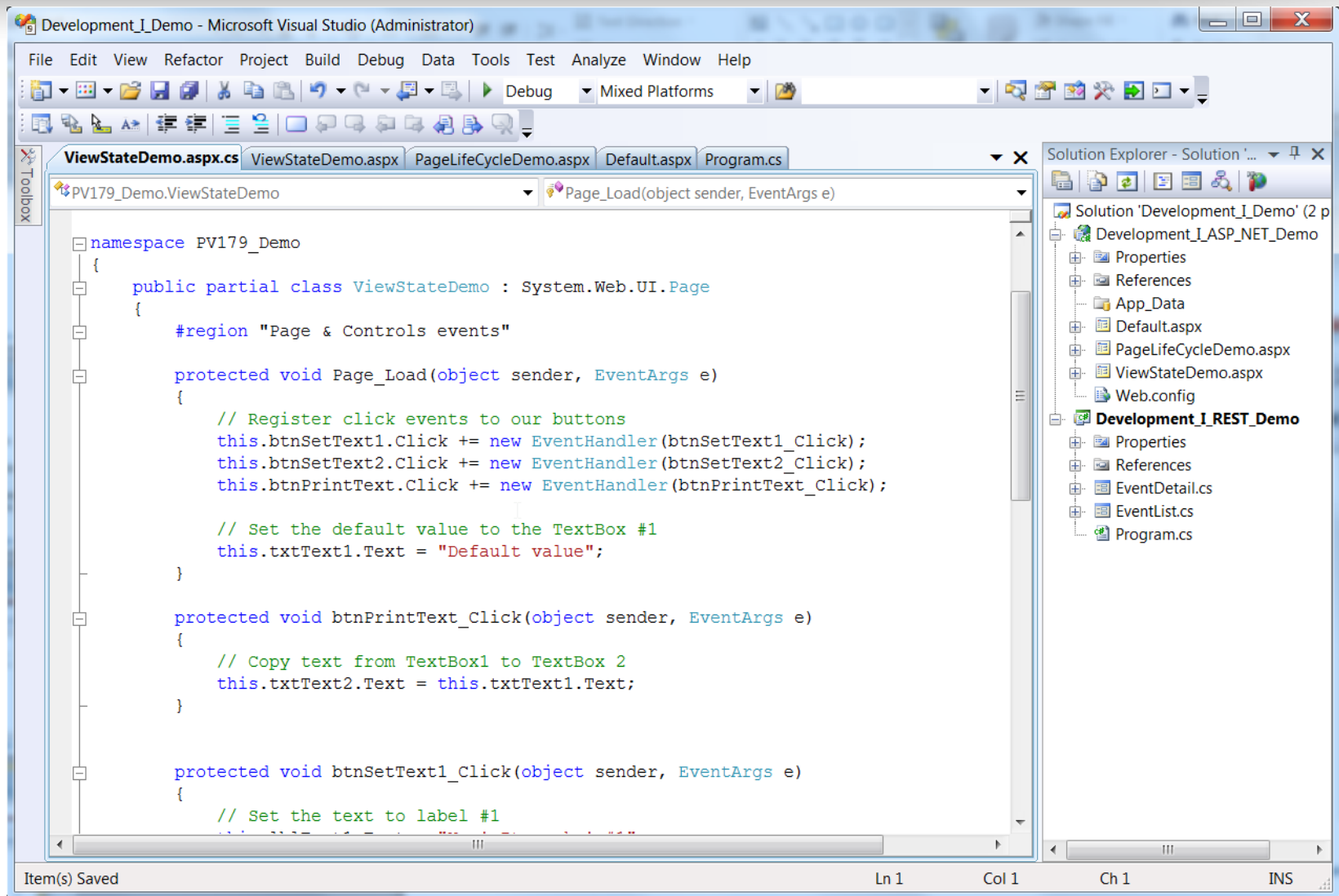


The screenshot displays the Microsoft Visual Studio (Administrator) interface. The main window shows the source code of an ASP.NET page, `ViewStateDemo.aspx`. The code is written in HTML and includes ASP.NET server controls. The page structure is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ViewStateDemo.aspx.cs" Inherits="..." %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1" [?]>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Development I - Demo page - ViewState</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <div style="border: solid 1px grey; width: 300px;">
        Label #1 (disabled ViewState): <asp:Label ID="lblText1" runat="server" Enabled="false" />
        Label #2 (enabled ViewState): <asp:Label ID="lblText2" runat="server" />
      </div>
      <br />
      <asp:Button runat="server" ID="btnSetText1" Text="Set text to a label #1" />
      <asp:Button runat="server" ID="btnSetText2" Text="Set text to a label #2" />
      <hr />
      <div style="border: solid 1px grey; width: 300px;">
        TextBox #1: <asp:TextBox ID="txtText1" runat="server" /><br />
        TextBox #2: <asp:TextBox ID="txtText2" runat="server" />
      </div>
      <br />
      <asp:Button runat="server" ID="btnPrintText" Text="Copy text from TextBox #1 to" />
    </div>
  </form>
</body>
```

The Solution Explorer on the right shows the project structure for 'Development_I_Demo' (2 p), which includes files like `Properties`, `References`, `App_Data`, `Default.aspx`, `PageLifeCycleDemo.aspx`, `ViewStateDemo.aspx`, and `Web.config`. Below the code editor, the breadcrumb navigation shows the current location: `<html> <body> <form#form1> <div>`. The status bar at the bottom indicates 'Item(s) Saved', 'Ln 10', 'Col 10', 'Ch 10', and 'INS'.

ASP.NET page – code behind



```
namespace PV179_Demo
{
    public partial class ViewStateDemo : System.Web.UI.Page
    {
        #region "Page & Controls events"

        protected void Page_Load(object sender, EventArgs e)
        {
            // Register click events to our buttons
            this.btnSetText1.Click += new EventHandler(btnSetText1_Click);
            this.btnSetText2.Click += new EventHandler(btnSetText2_Click);
            this.btnPrintText.Click += new EventHandler(btnPrintText_Click);

            // Set the default value to the TextBox #1
            this.txtText1.Text = "Default value";
        }

        protected void btnPrintText_Click(object sender, EventArgs e)
        {
            // Copy text from TextBox1 to TextBox 2
            this.txtText2.Text = this.txtText1.Text;
        }

        protected void btnSetText1_Click(object sender, EventArgs e)
        {
            // Set the text to label #1

```

The Solution Explorer on the right shows the project structure:

- Solution 'Development_I_Demo' (2 p)
 - Development_I_ASP_NET_Demo
 - Properties
 - References
 - App_Data
 - Default.aspx
 - PageLifeCycleDemo.aspx
 - ViewStateDemo.aspx
 - Web.config
 - Development_I_REST_Demo
 - Properties
 - References
 - EventDetail.cs
 - EventList.cs
 - Program.cs

What is a request?

The screenshot shows the Fiddler HTTP Debugging Proxy interface. The top menu includes File, Edit, Rules, Tools, View, and Help. Below the menu is a toolbar with various actions like Comment, Reissue, Remove, Resume All, Streaming, AutoDecode, Process Filter, Find, Save, Launch IE, Clear Cache, Encoder, Tearoff, and MSDN Search. The main window is divided into several panes:

- Web Sessions:** A table listing captured sessions. The selected session (number 16) is highlighted.
- Headers:** A pane showing the request headers for the selected session.
- Raw:** A pane showing the raw HTTP request and response data.
- Transformers:** A pane showing the request body content.

Web Sessions Table:

#	Result	Proto...	Host
1	200	HTTP	www.fiddler2.cor
2	302	HTTP	go.microsoft.co
3	200	HTTP	services.commu.
4	302	HTTP	go.microsoft.co
5	200	HTTP	services.commu.
6	302	HTTP	go.microsoft.co
7	301	HTTP	rss.msnbc.msn.c
8	302	HTTP	pheedo-rdr.msn.
9	200	HTTP	pheedo.msnbc..
10	401	HTTP	intr
11	200	HTTP	intr
12	200	HTTP	intr
13	200	HTTP	intr
14	200	HTTP	intr
15	302	HTTP	www.google.co
16	200	HTTP	www.google.c
17	200	HTTP	ssl.gstatic.cor
18	200	HTTP	www.google.c
19	200	HTTP	www.google.c
20	200	HTTP	www.google.c
21	200	HTTP	www.google.c
22	200	HTTP	www.google.c
23	200	HTTP	www.google.c
24	200	HTTP	www.google.c
25	200	HTTP	www.google.c
26	200	HTTP	www.google.c
27	200	HTTP	ssl.gstatic.cor
28	200	HTTP	www.google.c

Request Headers (Session 16):

```
GET http://www.google.cz/ HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: cs-CZ
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727;
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
Host: www.google.cz
```

Response Headers (Session 16):

```
HTTP/1.1 200 OK
Date: wed, 28 Sep 2011 13:08:55 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Set-Cookie: PREF=ID=7358ef38213c0bad:FF=0:TM=1317215335:LM=1317215335:S=pkW_I5Sr6NTaZ33d; expires=Fri, 27-S
Set-Cookie: NID=51=nH38jmmDYdV8EwV1wzgcjidRqBcAV5XAodiFF48KireHPwPC18F6mMhBfwfdqw3sph-9tatk0rjkw2Fo38IRpar
Server: gws
Content-Length: 54183
X-XSS-Protection: 1; mode=block
```

Response Body (Session 16):

```
<!doctype html><html><head><meta http-equiv="X-UA-Compatible" content="IE=8"><meta http-equiv="content-type
true";e:function(){google.fl=true};b:location.hash&&location.hash!="",bv:21,cf:"" ,pm:"p",pl:[],mc:0,sc:0.5
(function(){var a=google.j;window.onpopstate=function(){a.psc=1};for(var b=0,c=c["ad","bc","inpr","is","p
arguments])}}(c))();if(!window.chrome)window.chrome={};window.chrome.sv=1.00;
window.google.sn="webhp";var i=window.google.timers={};window.google.startTick=function(a,b){i[a]={t:start
</script><style id=gstyle>body{margin:0;overflow-y:scroll}#gog{padding:3px 8px 0}.gac_m td{line-height:17px
if(!window.google)window.google={};window.google.crm={};window.google.cri=0;window.cik=function(e,f,g,l,m,b
if(b&&b.substr(0,6)!="&sig2=")b="&sig2="+b;c.src=["/url?sa=t", "", "&cd=",a(m),h?"&authuser="+a(h):", goog
"&si=2": "", "&ved=",a(n),e?"&url="+a(e.replace(/#.*\/,""))).replace(/\+/g,"%2B"): "" ,&ei=", "zxyDTuykI8rV0Qwyy
(function(){try{var e=true,j=false;var m=window.gbar=window.gbar||{};function_tvh(a,b){var c=parseInt(a,10
var p={},ca={},q=[],fa=function(a,b){q.push([a,b])};ia=function(a,b){p[a]=b};ja=function(a){return a in p};
b:b};if(c)for(var d in c)a[d]=c[d];try{z(a)}catch(f)};n("mdc",p);n("mdi",ca);n("bnc",q);n("qgc",y);n("qr
```

What is a request?

- Main thing you need to remember about HTTP protocol:

HTTP is stateless protocol!

- But we need state in dynamic web applications!

How does ASP.NET deal with stateless http?

- The answer is ... **ViewState!**
- It is a technique used by an ASP.NET Web page to persist changes to the state of a Web Form across postbacks (HTTP POST to the same page that the form is on).
- **Use ViewState carefully** and only when it's really needed! It's helpful technique, but it might become too greedy and can cause the application to be less effective.

ViewState – How is it send within requests?

Label #1 (disabled ViewState):
Label #2 (enabled ViewState): Yay! It works! #2

Set text to a label #1 Set text to a label #2

TextBox #1: Default value
TextBox #2: Default value

Copy text from TextBox #1 to TextBox #2

```
Source of: http://localhost:53053/ViewStateDemo.aspx - Mozilla Firefox
File Edit View Help

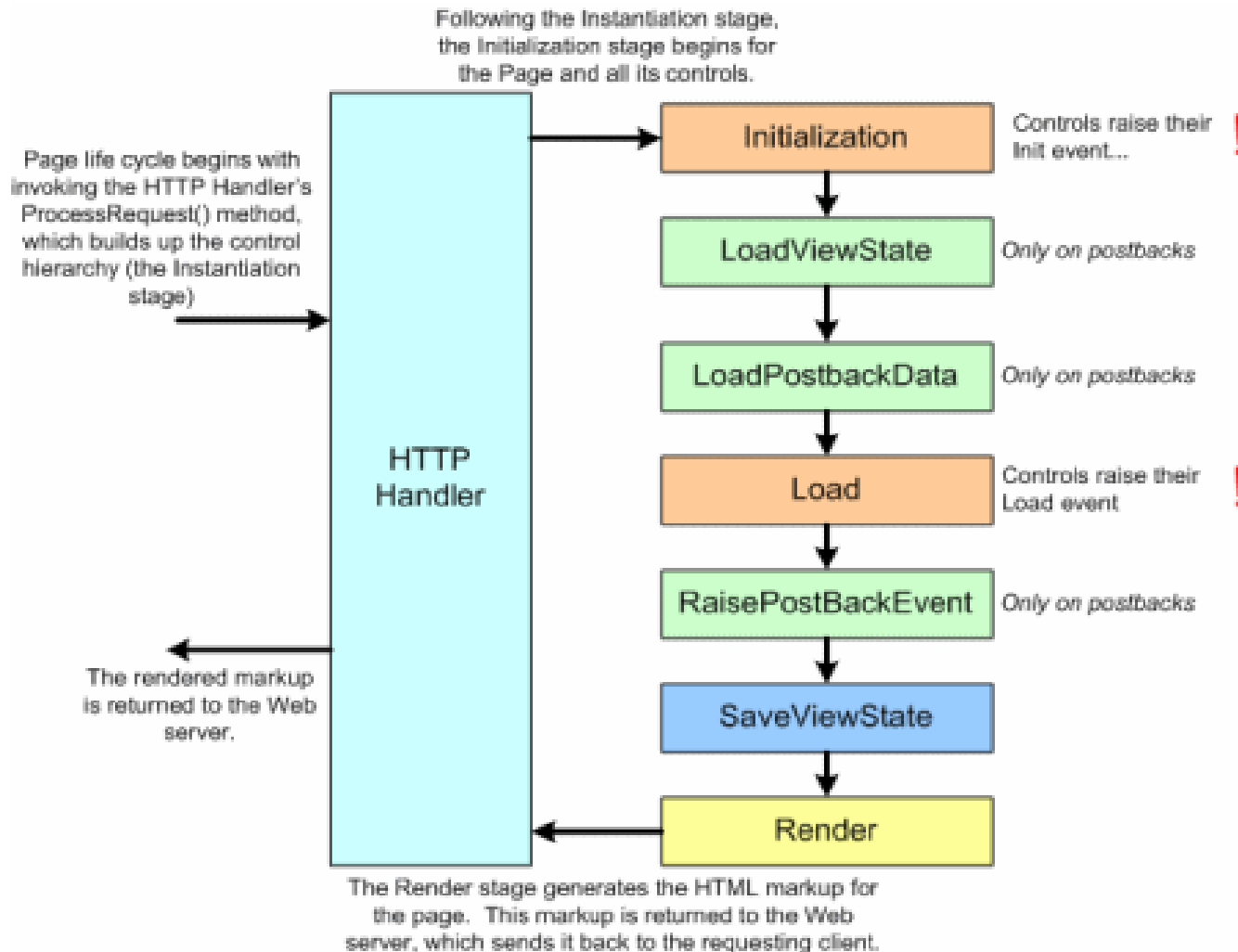
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org
/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
    PV179 Demo page - ViewState
</title></head>
<body>
    <form name="form1" method="post" action="ViewStateDemo.aspx" id="form1">
    <div>
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTE2NzgyMTcyNTEPZBYCAgMPZBYCAgMPDxYCHgRUZXh0BRFYXkhIE10IHdvcmtzISAjMmRkZlIt
Ehb7Ze3GsCiwWgDZkEOL2bTPf" />
</div>

    <div>
        <div style="border: solid 1px grey; width: 300px;">
            Label #1 (disabled ViewState): <span id="lblText1"></span><br />
            Label #2 (enabled ViewState): <span id="lblText2">Yay! It works! #2</span>
        </div>
        <br />
        <input type="submit" name="btnSetText1" value="Set text to a label #1"
id="btnSetText1" />
        <input type="submit" name="btnSetText2" value="Set text to a label #2"
id="btnSetText2" />
        <br />
        <div style="border: solid 1px grey; width: 300px;">
            TextBox #1: <input name="txtText1" type="text" value="Default value"
id="txtText1" /><br />
            TextBox #2: <input name="txtText2" type="text" value="Default value"
id="txtText2" />
        </div>
        <br />
        <input type="submit" name="btnPrintText" value="Copy text from TextBox #1 to
TextBox #2" id="btnPrintText" />
    </div>
</form>
</body>
</html>
```

ASP.NET page/control life cycle

- To be able to work with ASP.NET pages and controls properly you need to understand the life cycle of these elements.
- Most important phases of page/control life cycle are:
 - PreInit
 - Init
 - Load
 - PreRender
 - Render

ASP.NET page/control life cycle



Source: <http://i.msdn.microsoft.com/dynimg/IC152667.gif>

Where to get more information?

- Where to start:

<http://msdn.microsoft.com/en-us/library/ywdtth2f%28v=vs.71%29.aspx>

- More about page life cycle:

<http://msdn.microsoft.com/en-us/library/ms178472.aspx>

- More details about how the ViewState works:

<http://msdn.microsoft.com/en-us/library/ms972976.aspx>

Let's have a REST!

- **Representational state transfer (REST)** is a style of a service architecture
- Instead of complex technologies (RPC, SOAP, etc.) you use simple HTTP requests which are supported by all the clients!
- Commonly used formats of REST responses are XML, JSON, AtomPub

Let's have a REST!

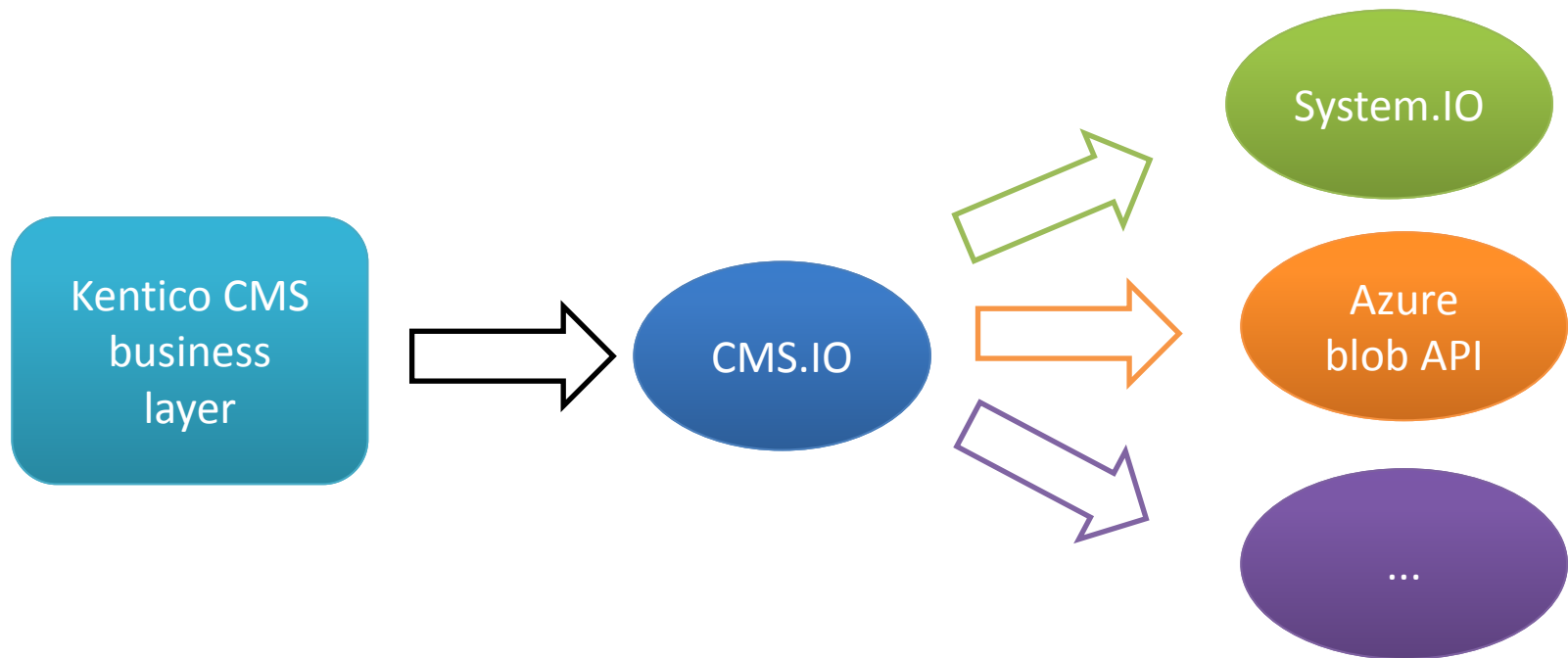
- Conforming to the REST constraints is referred to as being "**RESTful**," service
 - Access the objects within the system defining clear structure of URLs
 - The structure of URLs should be self-explaining and self-navigating
- Kentico CMS supports RESTful service

How to request a REST Service

- Create request
- Wait for response
- That's all!

- DEMO

Layer between Kentico CMS and System.IO



- Allows the system to use various IO providers (local file system, Azure Blob, Amazon S3, ...)
- To implement a provider you inherit from the base class and override the IO methods
 - Directory
 - File
 - FileStream
 - etc.